# Real-Time Lane Departure and Front Collision Warning System on an FPGA

Jin Zhao, Bingqian Xie and Xinming Huang
Department of Electrical and Computer Engineering
Worcester Polytechnic Institute, Worcester, MA 01609, USA

***Abstract*—Driving safety is a top priority when designing intelligent automobiles. In this paper, we present an FPGA-based real-time system for lane departure warning (LDW) and front collision warning (FCW) , which requires intensive computation and memory access for high resolution video input from a front-view camera. Our system first converts gray scale video input into desired binary video images by applying Sobel filter, Otsu's threshold binarization and dilation. Hough transform is then applied to detect lanes for LDW. The front vehicles are detected for FCW by searching shadow area on the images. Our experimental results demonstrate the FPGA design is fully functional and it can process about 160 frames per second for 720P video input, which exceeds the real-time processing requirement.**

***Index Terms*—Lane departure warning, Front Collision Warning, FPGA, Real-time**

## I. Introduction

Each year millions of traffic accidents occurred around the world cause loss of lives and properties. Improving road safety through advanced computing and sensor technologies has drawn lots of interests from the auto industry. Advanced Driver assistance system (ADAS) thus gains great popularity, which helps drivers to properly handle different driving condition and giving warnings if any danger is insight. Typical driver assistance system includes, but is not limited to, lane departure warning, traffic sign detection [1], obstacle detection [2], pedestrian detection [3], etc. Many accidents were caused by careless or drowsy drivers, who failed to notice that their cars were drifting to neighboring lanes or were too close to the car in the front. Therefore, lane departure warning (LDW) system and front collision warning (FCW) system are designed to prevent this type of accidents.

Different methods have been proposed for lane keeping and front vehicle detection. Marzotto et al. [4] proposed a RANSAC-like approach to minimize computational and memory cost. Risack et al. [5] used a lane state model for lane detection and evaluated lane keeping performance as a parameter called time to line crossing (TLC). McCall et al. [6] proposed a 'video-based lane estimation and tracking' (VioLET) system which uses steerable filters to detect the lane marks. Others used Hough transform for feature based lane detection. Wang et al. [7] combined Hough transform and Otsu's threshold method together to improve the performance of lane detection and implemented it on a Kintex FPGA platform. Lee et al. [8] removed redundant operations and proposed an optimized Hough transform design which uses fewer logic gates and less number of cycles when implemented on an FPGA board. Lin et al. [9] integrated lane departure warning system and front collision warning system on a DSP board.

The key requirements of a driver assistance system are rapid processing time and low power consumption. We consider FPGA as the most appropriate platform for such a task. Owing to the parallel architecture, an FPGA can perform high-speed video processing such that it can issue warnings timely and provide drivers more time to response. Besides, the cost and power consumption of modern FPGAs, particularly small size FPGAs, are considerably efficient.

In this contribution, we present a monocular vision lane departure warning and front collision warning system jointly implemented on a single FPGA device. Our experimental results demonstrate that the FPGA-based design is fully functional and it achieves the real-time performance of 160 frame-per-second (fps) for high resolution 720P video.

This paper is organized as follows. Section 2 presents our approaches to lane departure warning and front collision warning systems. Section 3 introduces the hardware architecture design, especially for the Otsu's threshold and Hough transform and their implementations on FPGA. Section 4 shows the experimental platform and results. Finally, this work is concluded in Section 5.

## II. Approaches to LDW and FCW

The proposed LDW and FCW system takes video input from a single camera mounted on a car windshield. Fig. 1 shows the flow chart of the proposed LDW and FCW systems. The first step is to apply a Sobel filter on the gray scale image frame to sharpen the edges after an RGB to gray scale conversion. This operation helps highlight the lane marks and front vehicle boundary while eliminating dirty noise on the road. The operator of Sobel filter is

$$G = G_x^2 + G_y^2 \tag{1}$$

where the $G_x$ is Sobel kernel in $x$- direction as in (2) and similarly $G_y$ in $y$- direction.

$$G_x = \begin{bmatrix} 0.125 & 0 & -0.125 \\ 0.25 & 0 & -0.25 \\ 0.125 & 0 & -0.125 \end{bmatrix} \tag{2}$$

Next, an important task is to convert the gray scale video frame to a "perfect" binary image which is used by both

978-1-4799-6233-4/14/$31.00 ©2014 IEEE

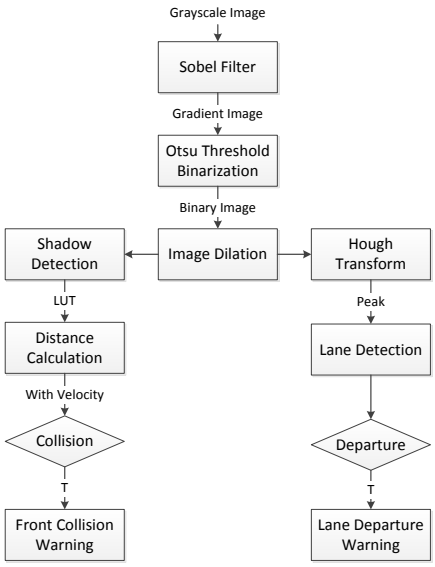


Figure 1. Flow chart of the proposed LDW and FCW systems

LDW and FCW systems. A "perfect" binary image means high signal-to-noise ratio, preserving desired information while abandoning useless data to produce more accurate results and to ease computational burden. For LDW and FCW systems, the desired information is lane marks of the road and vehicles in the front. Therefore, we cut off the top part of every image and crop the area of interest to the bottom 320 rows of a 720P picture. All subsequent calculations are conducted on this region of interest (ROI). When converting from gray scale to binary image, a fixed threshold is often not effective since the illumination variance has apparent influence on the binarization. Instead, we choose Otsu's threshold, a dynamic and adaptive method, to obtain binary image that is insensitive to background changes. The first step of Otsu's method is to calculate probability distribution of each gray level as in (3).

$$p_i = n_i/N \tag{3}$$

where $p_i$ and $n_i$ are probability and pixel number of gray level *i*. *N* is the total pixel number of all possible levels (from 0 to 255 for gray scale). The second is to step through all possible level t to calculate $\omega^*(t)$ and $\mu^*(t)$ as in (4) and (5).

$$\omega^*(t) = \sum_{i=0}^{t} p_i \tag{4}$$

$$\mu^*(t) = \sum_{i=0}^{t} i p_i \tag{5}$$

The final step is to calculate between-class variance $(\sigma_B^*(t))^2$ as in (6).

$$(\sigma_B^*(t))^2 = \frac{[\mu_T^* \omega^*(t) - \mu^*(t)]^2}{\omega^*(t) \times [1 - \omega^*(t)]} \tag{6}$$

where $\mu_T^*$ is $\mu^*(255)$ in this case. For easy implementation on an FPGA, (6) is re-written as (7)

$$\sigma_B^2(t) = \frac{[\mu_T/N \times \omega(t) - \mu(t)]^2}{\omega(t) \times [N - \omega(t)]} \tag{7}$$

where $\omega(t) = \omega^*(t) \times N$ and $\mu(t) = \mu^*(t) \times N$. Otsu's method selects level $t^*$ as binarization threshold, which has largest $\sigma_B^2(t^*)$. Fig. 2 illustrates effects of Sobel filter and Otsu's threshold binarization.

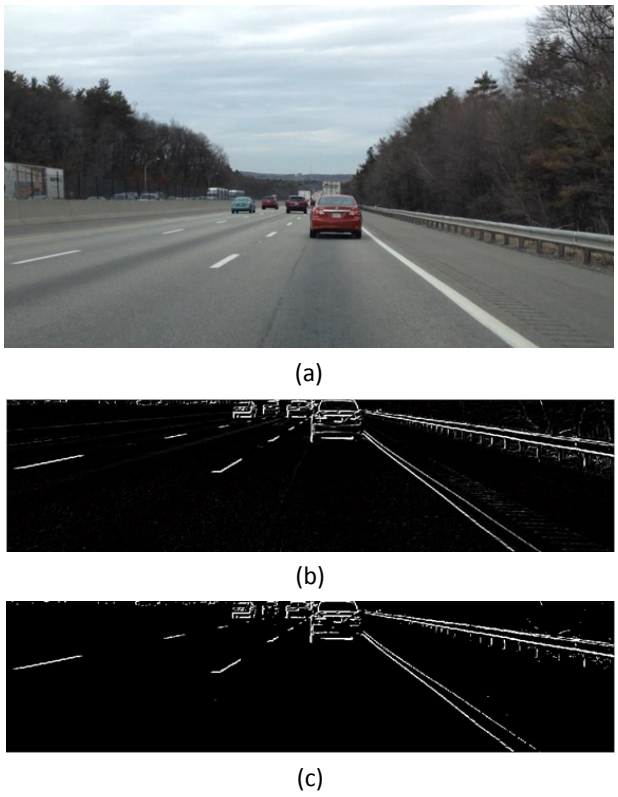


Figure 2. Illustration of Sobel filter and Otsu's threshold binarization. (a) is the original image captured from camera. (b) and (c) are ROI after Sobel filtering and binarization, respectively

Image dilation is a basic morphology operation. It usually uses a structuring element for probing and expanding shapes in a binary image. In this design, we apply image dilation to smooth toothed edges introduced by Sobel filter, which can assist the subsequent operations to perform better.

Furthermore, Hough transform (HT) is widely used as a proficient way of finding lines and curves in binary image. Since most lines on a road are almost straight in pictures, we mainly discuss the way of finding straight lines in binary image using Hough transform. Every point (*x*,*y*) in 2D space can be represented in polar coordinate system as

$$\rho = x\cos(\theta) + y\sin(\theta) \tag{8}$$

Fig. 3 illustrates 2D space to Hough space mapping. If point A and B are on the same straight line, there exists a pair of $\rho$

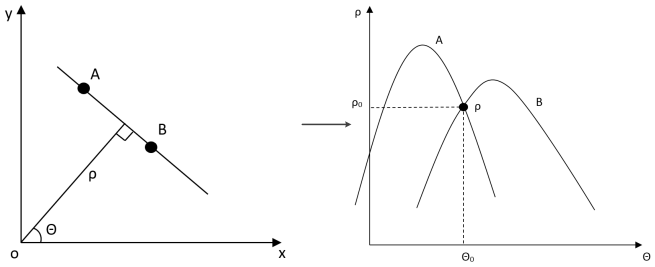


Figure 3. Hough transform from 2D space to Hough space

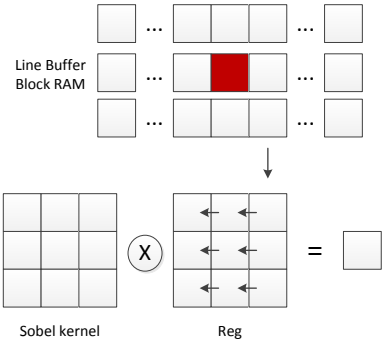


Figure 4. Sobel filter architecture

and $\theta$ that satisfy both equation, which means two curves in polar coordinate have an intersection point. As more points on the same line are added to the polar system, there should be one shared intersection point between these curves. In Hough transform algorithm, it keeps track of all intersection points between curves and the intersections with big voting imply that there is a straight line in 2D space.

By analyzing Hough transform for lane detection, we optimize the transform during the following two steps: 1) dividing the image into a left part and a right part, perform HT with positive $\theta$ for the left part and negative $\theta$ for the right part due to the slope nature of the lanes; 2) further shrink the range of $\theta$ by ignoring those lines that are almost horizontal with $\theta$ near $\pm 90$. Empirically, the range of $\theta$ is set to $[1, 65]$ for the left part and $[-65, -1]$ for the right part. We choose the most voted $(\rho, \theta)$ pairs of left part and right part as detected left lane and right lane, respectively. To avoid occasional errors on a single frame, the detected lanes are compared with results from preceded 4 frames. If they are outliers, the system takes weighted average of results of preceded 4 frames as the current frame's lanes. If either the left lane or the right lane is excessively upright (i.e. $abs\,(\theta) < 30$), the system issues lane departure warning.

For the front collision warning system, we identify the front vehicle by detecting the shadow area within a triangular region in front of the car on the binary image. From the position of the front vehicle, the actual distance is obtained using an empirical look-up-table (LUT). The system generates a collision warning signal when the distance is smaller than suggested safe distance at the vehicle traveling speed.

## III. Hardware Implementation

In this section, we elaborate the hardware architecture for LDW and FCW systems, especially for the design implementations of Sobel filter, Otsu's threshold binarization, Hough transform and front vehicle detection.

Fig. 4 is the architecture of Sobel filter. The Sobel filter module takes 3 Block RAMs (BRAMs) to store the current row of input pixels and buffer 2 rows right above it. Two registers connected to Block RAM output buffer the read data for 2 clock cycles (cc) for each row. With such an architecture, we get every pixel together with its 8 neighbors at the same cc. The Sobel kernel is then applied on the $3 \times 3$ image block. The system sums the $3 \times 3$ result to get Sobel response in $x$- and $y$- directions. The sum of square of Sobel filter response in two directions is set as gradient value of the center pixel (the red color in Fig. 4). The image dilation module has similar architecture as Sobel filter. The only difference is that it uses a structuring element of $3 \times 3$ 1's rather than Sobel kernel on the block and produces a dilated binary image.

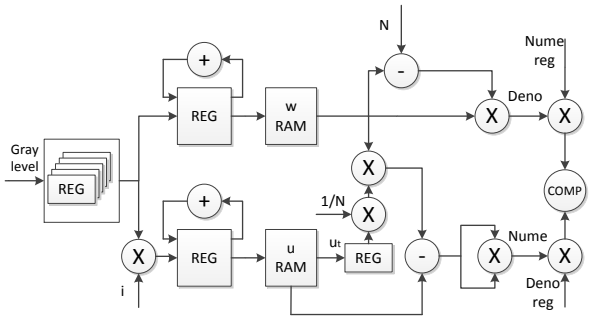


Figure 5. Datapath for Otsu's threshold calculation

We implement the Otsu's method by closely following (7). The gradient pixel intensity is 8-bit gray level as an input of this block. Hence we use 256 registers to count the total number of every level (i.e. $n_i$) in ROI of current video frame. The following step is to accumulate $n_i$ and $i \times n_i$ to get $\omega$ and $\mu$, which are stored in two separate RAMs. [10] directly computes $\sigma_B^2$ as in (6), but the division is expensive in term of processing speed and hardware resource use. [7] takes advantage of extra ROMs to convert division to subtraction in log domain. After careful consideration, we find that this part is to find the intensity level that maximizes $\sigma_B^2$ rather than get the exact value of $\sigma_B^2$. Therefore, we design a datapath to find the Otsu's threshold as in Fig. 5. As $t$ varies from 0 to 255, $\omega(t)$ and $\mu(t)$ are read from the RAMs to calculate the numerator and the denominator of $\sigma_B^2(t)$. Two registers, Numerator_reg and Denominator_reg buffer the corresponding numbers of biggest $\sigma_B^2\,(i)$ for $i < t$. Just two multipliers and one comparator are deployed to check if $\sigma_B^2(t)$ is larger

than the largest $\sigma_B^2(i)$ for $i < t$. If so, the Numerator_reg and Denominator_reg are updated accordingly. With such an architecture, we find the Otsu's threshold that is employed to convert the gradient gray-level image to black and white image.

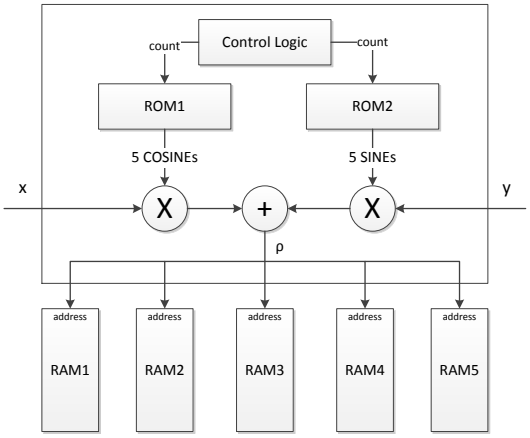


Figure 6. Hough transform architecture

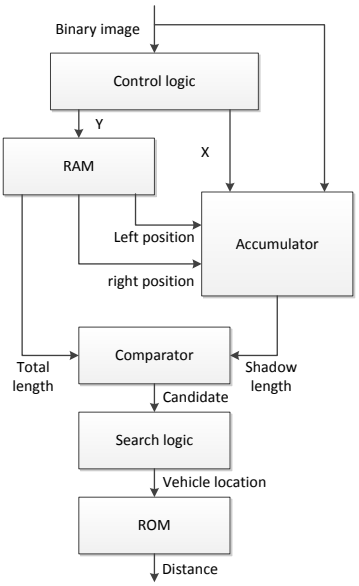


Figure 7. Flow chart of front vehicle detection

The Hough transform is the most computationally expensive block in the entire design. Here we just discuss Hough transform for the left part since the right part has almost identical architecture. As mentioned in Section II, we let $\theta$ vary across 1 to 65 to find the left and right lanes by seeking most voted $(\rho, \theta)$ pairs. It is reasonable to assume less than 10% pixels are edge pixels in a road scene. Thus, in order to achieve real-time Hough transform, we propose a pipeline and dedicated architecture as in Fig. 6. ROM1 and ROM2 store cosine and sine values for $\theta$ from 1 to 65 (totally 13 groups with every 5 successive ones as a group). When the HT module receives position information of edge pixels, it reads out cosine and sine values of $\theta$ and performs Hough transform. Control logic generates a signal *count* to indicate which 5 $\theta's$ are being processed. Once they receives the $\rho$, each block reads its corresponding value from RAMs, adds it by 1, and then write it back. After all edge pixels go through the Hough transform module, the system scans all RAM locations and finds most voted $(\rho, \theta)$ pair as the left lane. If the detected lanes are excessively upright ($\theta < 30$), a lane departure warning will be released at the very first time.

Fig. 7 shows the algorithm flow chart for detecting front vehicles. The detection is based on shadow information in a triangular region formed by the left and right lanes. When a binary image enters, the control logic generates position information X and Y. For every row, the RAM, as a LUT, provides left and right boundary positions of a targeted triangle region based on vertical information Y. The Accumulator block counts the white pixels within the triangle area for each row. If the number of filled pixels is larger than a fixed ratio to the triangle width of that row, it is a candidate for front vehicle detection. The candidate row, which are closest to the bottom of an image, is considered as the front vehicle's position. Finally, the system converts the row number to the physical distance on the ground through a LUT implemented using a ROM. If the vehicle traveling speed is known, a front collision warning can be issued when the front vehicle is closer than the safe distance.

## IV. Experimental Results

The proposed design is implemented in Verilog HDL. Xilinx ISE tool is used for design synthesis, translation, mapping and placement and routing (PAR). In order to validate the entire design, we conduct an experiment on a KC705 FPGA platform. The video streams from a camera mounted on a vehicle windshield are sent to an on-board Kintex-7 FPGA via FMC module. The FPGA performs lane departure and front collision warning examination on every video frame and exports the results to a monitor for display. The input video is 720P resolution at the rate of 60 fps. We take the video clock as system clock of our design, which is 74.25MHz in this case.

Table I
Device utilization summary

| Resources | Used | Available | Utilization |
|---|---|---|---|
| Number of Slices Registers: | 8,405 | 407,600 | 2% |
| Number of Slice LUTs: | 10,081 | 203,800 | 4% |
| Number of Block RAMs: | 60 | 445 | 13% |
| Number of DSP48s | 31 | 840 | 3% |

The resources utilization of the LDW and FCW systems on an XC7K325TFPGA is shown in Table 1. Although we verify

the design on a 7-series FPGA, it can also be implemented on a much smaller device for lower cost and less power consumption, e.g. Spartan-6 FPGA. The reported maximum frequency is 206 MHz. Thus if the system runs at maximum frequency, it can process 720P video at 160 fps, which is corresponding to the pixel clock frequency at 206 MHz. The design is validated by road tests using the FPGA-based hardware system. Fig. 8 illustrates some results of our LDW and FCW systems. The detected lanes are marked as green and the detected front vehicle is marked as blue.

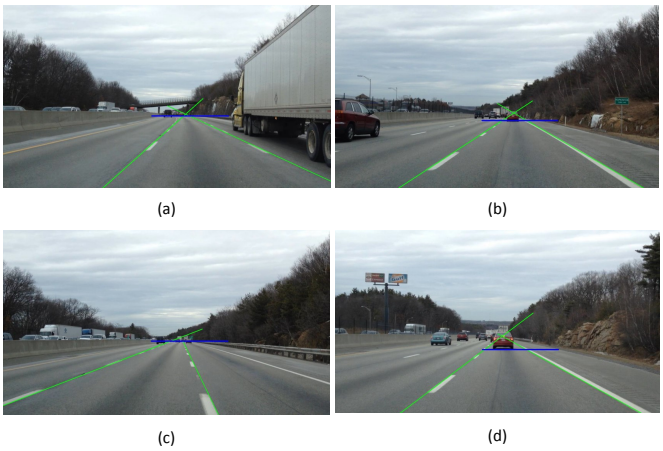


Figure 8. Results of our LDW and FCW systems

## V. Conclusion

In this paper, we present an FPGA-based design for a driver assistance system including both lane departure warning and front collision warning functions. The proposed design is implemented and validated on a KC705 FPGA platform, but it can fit into a much smaller FPGA. A set of design optimizations are applied to speedup the image processing and reduce resource utilization. Our road tests demonstrate that the design is not only working properly but also exceeds the targeted real-time performance of 60 fps. Therefore the FPGA-based system is readily available for practical applications.

## VI. Acknowledgments

The authors would like to thank The MathWorks, Inc. for support this research work.

## References

[1] J. Zhao, S. Zhu, and X. Huang, "Real-time traffic sign detection using surf features on fpga," in *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*. IEEE, 2013, pp. 1–6.

[2] A. Broggi, C. Caraffi, R. Fedriga, and P. Grisleri, "Obstacle detection with stereo vision for off-road vehicle navigation," in *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, 2005, pp. 65–65.

[3] D. M. Gavrila, "Pedestrian detection from a moving vehicle," in *Computer Vision, ECCV 2000*. Springer, 2000, pp. 37–49.

[4] R. Marzotto, P. Zoratti, D. Bagni, A. Colombari, and V. Murino, "A real-time versatile roadway path extraction and tracking on an fpga platform," *Computer Vision and Image Understanding*, vol. 114, no. 11, pp. 1164–1179, 2010.

[5] R. Risack, N. Mohler, and W. Enkelmann, "A video-based lane keeping assistant," in *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*. IEEE, 2000, pp. 356–361.

[6] J. C. McCall and M. M. Trivedi, "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 7, no. 1, pp. 20–37, 2006.

[7] W. Wang and X. Huang, "An fpga co-processor for adaptive lane departure warning system," in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 1380–1383.

[8] S. Lee, H. Son, and K. Min, "Implementation of lane detection system using optimized hough transform circuit," in *Circuits and Systems (APCCAS), 2010 IEEE Asia Pacific Conference on*. IEEE, 2010, pp. 406–409.

[9] H.-Y. Lin, L.-Q. Chen, Y.-H. Lin, and M.-S. Yu, "Lane departure and front collision warning using a single camera," in *IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2012)*, 2012.

[10] W. Jianlai, Y. Chunling, Z. Min, and W. Changhui, "Implementation of otsu's thresholding process based on fpga," in *Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on*. IEEE, 2009, pp. 479–483.