Lane detection image processing algorithm based on FPGA for intelligent vehicle

Huiqin Zhan
School of Automation Engineering
University of Electronic Science and Technology of China
Chengdu, China
zhanhq@uestc.edu.cn

Lin Chen
School of Automation Engineering
University of Electronic Science and Technology of China
Chengdu, China
775466751@qq.com

***Abstract*—Self-driving cars are of great significance in reducing traffic accidents, traffic congestion, energy conservation and environmental protection, etc. Real-time lane detection is the basic function of autonomous driving perception system. Lane detection based on image processing is one of the main methods in lane detection at present, and there are many kinds of detection algorithms. However, the problem is that the algorithm model is complex and the computation amount is large, most of which can only be realized on the CPU+GPU platform, resulting in low cost performance, high power consumption and large volume, which is not suitable for vehicle-mounted requirements. In order to meet the needs of real-time lane detection performance, power consumption and flexibility requirements. In this paper, based on FPGA development platform, lane line detection algorithm based on image processing and deep learning is designed to achieve the fast lane line detection effect of structured roads, speed up to 104 FPS above. And in view of the road scene shadow, the method proposed in this paper can provide better detection result.**

Keywords—FPGA，lane detection，real-time performance，image processing

I. INTRODUCTION

With the coming of the new industrial revolution represented by artificial intelligence technology, the automobile industry has also ushered in the 4.0 era, and its development strategy target is the fully automatic driving vehicle. The key to the realization of autonomous driving is to have scene sensing system, high precision map and positioning service, planning decision and motion control system. Lane detection is an indispensable basic function of intelligent vehicles in the process of driving on structured roads[1]. The basic setting of smart car's fast driving requires the accuracy and real-time performance of lane detection, and it also needs to adapt to various scenarios, such as weather changes and tree shadows on the road.

Research directions of lane detection at home and abroad mainly include: implementing complex algorithms based on high-end machines such as PC, and implementing real-time algorithms based on embedded platform[2]. The lane detection algorithm realized on PC platform can directly use high-performance CPU or GPU to compare the detection effects of different algorithms quickly and conveniently. At the same time, complex algorithms can be used to improve the detection effect, but the power consumption and real-time performance of CPU and GPU cannot be ignored in the on-board environment[3][4].

The development of real-time lane detection system based on embedded platform is also an important research direction. The hardware resources on embedded platform are relatively scarce and it is difficult to implement complex algorithms. Therefore, lane detection on embedded platform needs to be considered from the aspects of detection accuracy and real-time performance[5][6].

In this paper, the proposed method combines the digital image processing algorithm and deep learning algorithm, using Xilinx ZYNQ7035 for SoC hardware development platform. HLS development tool is used to realize the design of hardware circuit based on FPGA[7].

Data preprocessing is realized in VIVADO environment, and data transmission path in the system is designed and built, so as to finally realize lane detection of structured road. Under the clock frequency of 100MHz, the detection speed reaches over 104fps, which completely meets the real-time requirements.

II. THE OVERALL DESIGN SCHEME OF LANE DETECTION

The whole lane detection system is mainly divided into three parts: image acquisition and preprocessing, lane line detection, lane line type recognition, as shown in figure 1.

A. Data pre-processing

Road information is collected by camera and input into FPGA in the form of image data via AXI protocol. This part includes data format conversion and transmission interface conversion, and the function of this part is to convert data into RGB24 format.

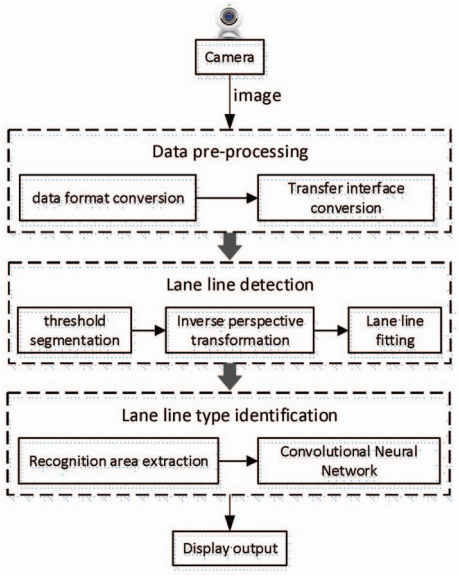


FIG. 1. Overall block diagram of lane detection

This work was supported the National Key R&D Program Project of China (2017YFB0102603)

B. Lane line detection

Lane line detection is realized by image processing algorithm, including threshold segmentation, inverse perspective transformation and lane line quadratic curve fitting. The final output detection results include the curvature radius of the current lane, the bending direction of the lane, the direction and distance of the vehicle deviating from the lane center, etc. At the same time, the lane line coordinates are output to facilitate the dynamic interception of the identification area in the lane line type identification module.

C. Lane line type identification

In the process of driving, vehicles need to pay attention to the type of lane lines. Different types of lane lines have different meanings. This design adopts the method of deep learning to realize lane line type recognition, and finally displays the type on the output image.

III. IMAGE DATA PREPROCESSING

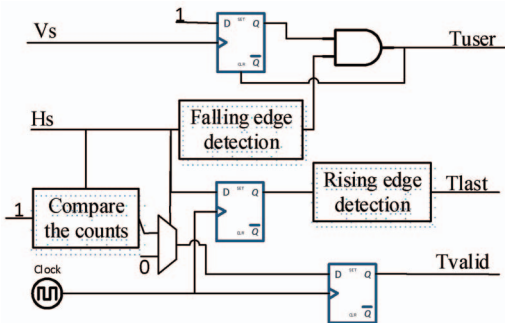
A. Image data format conversion

This design uses the ov7725 camera which supports the output of RGB565, YUV and other formats to collect image data. Since the data format required for subsequent processing is the 24-bit parallel transmission RGB24 format, format conversion is required. The format conversion module is realized in the FPGA, because of the camera is 8 bits to the output, so a RGB565 data was transmitted through two clock cycles, and so on each transmission two bytes on a format conversion processing, RGB three channel respectively in low zero padding form 8 bits of data, the final parallel output in 24 RGB24 format. Since the output pixel rate of the camera is 25MB/s, the 100MHz system clock set in this paper can process the input data in time and make the car drive faster.

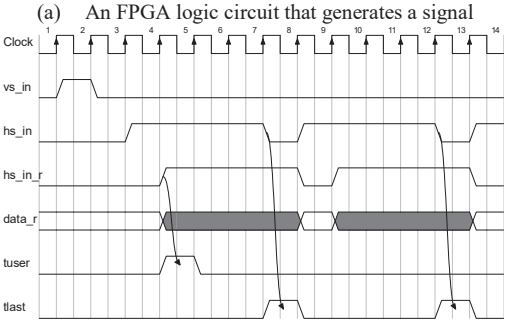
B. Transfer interface conversion

The output interface of data format conversion module is common parallel interface, while most IP provided by FPGA of Xilinx company is AXI4 interface. Therefore, this paper designs the transmission interface conversion module, and converts the output interface of data format conversion module into AXI4 interface, which is convenient to realize the fast and efficient transmission of camera data.

According to AXI4 - Stream protocol, convert AXI4 - Stream interface is mainly produce the right tuser tlast and



(a) An FPGA logic circuit that generates a signal



(b) signal timing

FIG. 2. The logic circuit that generates critical signals and signal timing

tvalid signal, tuser signal represents the start of a frame transmission, tlast signal represents the each row of the last valid data, and the tvalid signal means the data is valid. In order to save hardware resources, this paper does not use the IP core provided by the government, but uses Verilog for logic design. The three important signals are generated by the input

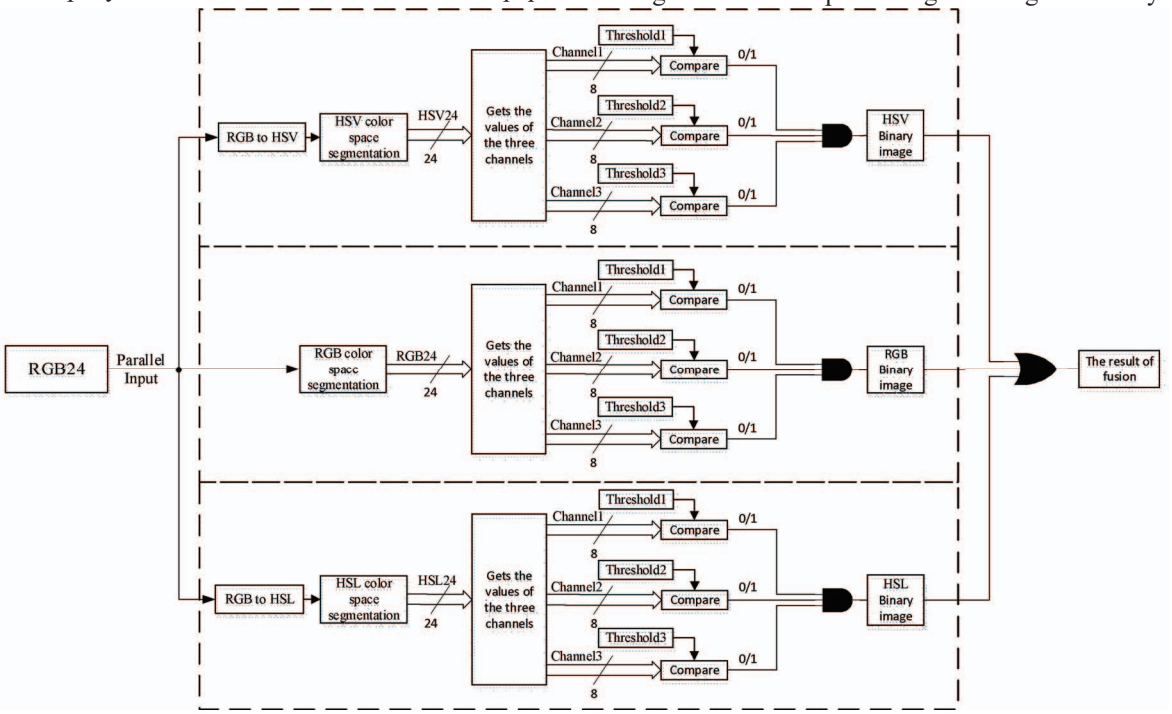


FIG. 3. Threshold segmentation algorithm and fusion

line signal field signals and effective data signals, as shown in figure 2(a), and the schematic diagram of signal waveform is shown in figure 2(b).

IV. Lane Line Detection Algorithm and Implementation

This design uses the Xilinx HLS development tool and the image processing algorithm to realize lane line detection, and finally outputs information such as the direction and the radius of curvature of the current lane, the direction and distance of the vehicle from the center of the lane, which is conducive to safe driving.

A. Threshold segmentation

Due to the obvious color characteristics of lane lines relative to the lane background, this design divides lane lines by setting threshold ranges in the three color Spaces of RGB, HSL and HSV, and then integrates the three segmentation

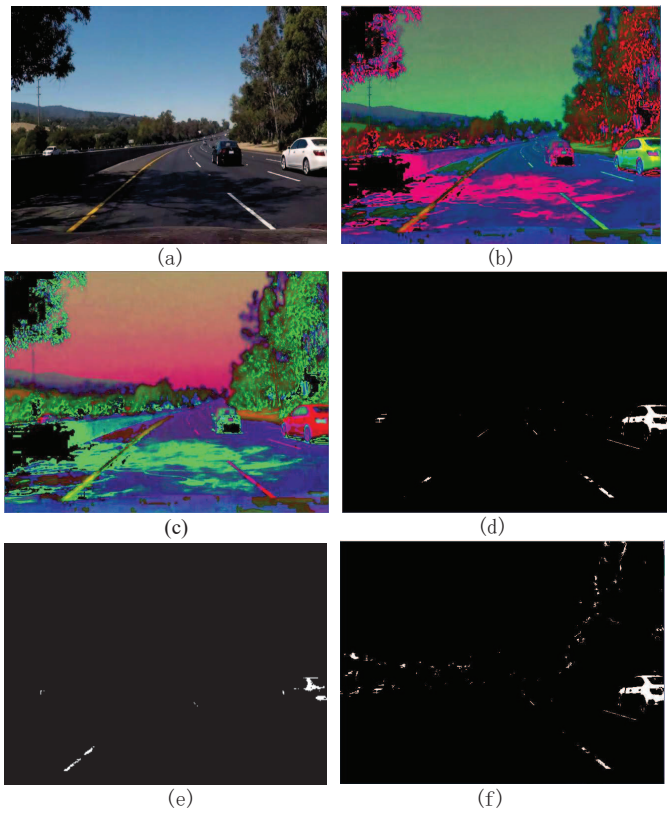


FIG. 4. Processing results in RGB, HSL, HSV color space. (a) RGB original image; (b) HSL original image; (c) HSV original image; (d) segmentation result in RGB space; (e) segmentation result in HSL space; (f) segmentation result in HSV space

results to reduce the extent of incomplete lane lines caused by single segmentation. Threshold segmentation algorithm and fusion are shown in figure 3, and the color space processing results of RGB, HSL and HSV are shown in figure 4.

The three results are superimposed according to Figure 3. The final segmentation results are shown in Figure 5:

FIG. 5. Overlay results. (a) original image; (b) final segmentation result

B. Inverse perspective transformation

Due to shadow occlusion in the original figure, lane line information as shown in figure 5(b) is incomplete. This paper

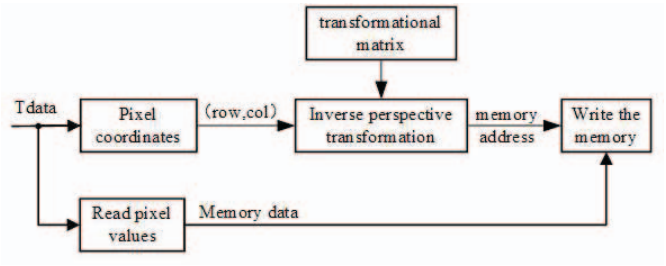


FIG. 6. Diagram of the inverse perspective transformation process

adopts the inverse perspective transformation method to restore the characteristics of parallel left and right lane lines, so as to facilitate interpolation according to this feature to complete lane line information and then fit the complete lane line. The diagram of inverse perspective transformation is shown in figure 6.

In this paper, we use an inverse perspective transformation method that is simple to calculate and does not require the internal and external parameters of the camera. The transformation equation is shown in Formula 1, where the 3*3 matrix containing parameters is the transformation matrix to be solved:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix} * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (1)$$

Where the is the coordinates of the original image, is the coordinate of the corresponding image after the transformation, and $x = \frac{x'}{w'}, y = \frac{y'}{w'}$, after the matrix operation is expanded, the transformation equation is obtained as shown in formulas 2 and 3:

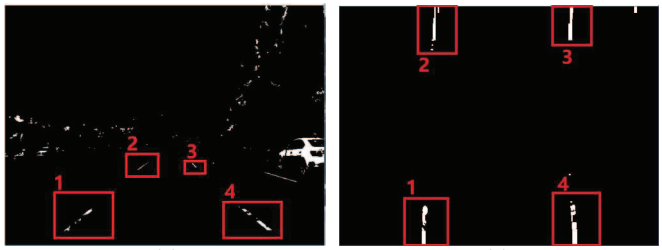
$$x = \frac{x'}{w'} = \frac{a_{11}u + a_{12}v + a_{13}}{a_{31}u + a_{32}v + 1} \quad (2)$$

$$y = \frac{y'}{w'} = \frac{a_{21}u + a_{22}v + a_{23}}{a_{31}u + a_{32}v + 1} \quad (3)$$

A total of 8 unknowns need to be solved. Therefore, the coordinates of 4 original image points before transformation and 4 corresponding points after transformation can be used to substitute the coordinates of 8 points into formulas 2 and 3 to solve the value of $a_{11} \ldots a_{32}$ parameters. Inverse perspective transformation is performed on FIG. 3, and the result is shown in FIG. 7.

The results before and after the inverse perspective transformation are shown in figure 7. The pixels in the red box no. 1-4 in figure 7 (a) are transferred to the positions in the red box no. 1-4 in figure 7 (b) after the inverse perspective transformation. As can be seen from the figure (b), in the

bird's-eye view after the transformation, the features of the left and right lane lines are restored.



(a) (b)

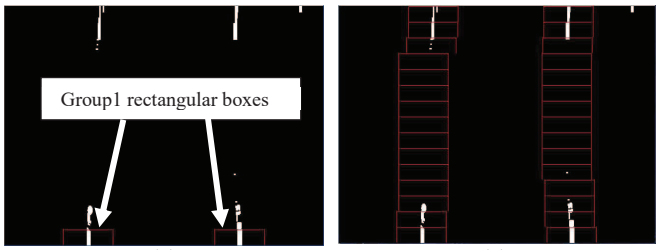
FIG. 7. Reverse perspective transformation. (a) untransformed original image; (b) inverse perspective transformation result

C. *Lane line fitting*

According to figure 7, the rough outline of the lane line can be obtained after perspective transformation. However, due to the characteristics of the lane line itself, uneven illumination and object occlusion, the lane line after transformation is incomplete. At this time, the complete lane line can be restored by means of fitting, so as to facilitate subsequent processing.

In this design, the lane line is divided into 15 parts by the sliding box, and finds the central position coordinates of the sliding rectangular box in each part, then the coefficients of the fitting formula are derived according to the least squares rule, and finally matches the lane line.

The first is the sliding box to find the lane line. This design traverses the bird's eye view to count the number of non-zero pixels in each column. After the traversal, the two largest columns are found, as the initial position of the lane line, to draw the first set of rectangular boxes with the initial position as the center, as shown in Figure 8(a).



(a) (b)

FIG. 8. Sliding box to find the lane line

Then count the number of non-zero pixels in each column of the first set of rectangular boxes to find the position with the largest number as the center position of the next set of rectangular boxes. Repeat the above steps until the full bird's eye view is processed, and the result is shown in Figure 8(b).

Finally, according to the least square rule, the linear equations for solving the coefficients of the fitted curve equation are derived, as shown in formula 4. Then, the center coordinates of the rectangular box in the above figure are substituted into the equation, which can be solved by Gaussian elimination method. The final fitted curve equation is shown in formula 5.

FIG. 9. Quadratic curve fitting results

$$\begin{bmatrix} \sum_{i=1}^{m} 1 & \sum_{i=1}^{m} x_i & \sum_{i=1}^{m} x_i^2 \\ \sum_{i=1}^{m} x_i^1 & \sum_{i=1}^{m} x_i^2 & \sum_{i=1}^{m} x_i^3 \\ \sum_{i=1}^{m} x_i^2 & \sum_{i=1}^{m} x_i^3 & \sum_{i=1}^{m} x_i^4 \end{bmatrix} * \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m} y_i \\ \sum_{i=1}^{m} y_i x_i \\ \sum_{i=1}^{m} y_i x_i^2 \end{bmatrix} \tag{4}$$

$$col = a_2 * row^2 + a_1 * row + a_0 \tag{5}$$

The resulting fitting curve equation is shown in Figure 9.

Using quadratic fitting not only provides smooth lane line information, but also helps to calculate the curvature and radius of the lane. The curvature and radius are calculated as follows:

$$k = \left|\frac{d\varphi}{ds}\right| = \left|\frac{\frac{y''}{1+y'^2}dx}{\sqrt{1+y'^2}dx}\right| = \left|\frac{y''}{\left(1+y'^2\right)^{3/2}}\right| \tag{6}$$

$$\rho = \frac{1}{k} = \left|\frac{\left(1+y'^2\right)^{3/2}}{y''}\right| \tag{7}$$

Due to equation $y = a_2x^2 + a_1x + a_0, y' = 2a_2x + a_1, y'' = 2a_2$, in order to have a certain forward-looking, this paper calculates the radius of curvature at $x = 0$.

According to the actual width of the lane and the number of pixels corresponding to the lane in the image, the actual distance corresponding to each pixel can be calculated, and further, whether the vehicle deviates from the center of the

FIG. 10. The result of lane line detection

lane and the direction and distance from the lane can be calculated. Meanwhile, the bending direction of the road ahead can be judged according to the coefficient of the curve fitting equation.

The inverse perspective transformation matrix obtained in IV.B is inverted to obtain the perspective transformation matrix, then the bird's-eye view use perspective transformation and the information such as the radius of curvature of the lane is superimposed. The result is shown in Fig. 10, and the output information is shown in Table 1.

TABLE I. LANE LINE DETECTION OUTPUT INFORMATION

Left 14cm	Indicates that the current vehicle is 14 cm to the left of the center of the lane.
Radius 578m	Indicates that the radius of curvature of the current lane is 578m.
Turn left	Indicates that the road ahead is bent to the left

V. LANE LINE TYPE IDENTIFICATION

(a) (b)

FIG. 11. Extraction of identification region. (a) Original image displaying; (b) Identification area selection

A. Lane line identification network design and training

In the process of driving, vehicles need to pay attention to the types of lane lines. Different types of lane lines have different meanings. In this paper, HLS development tool and channel deep learning method are used to identify lane line

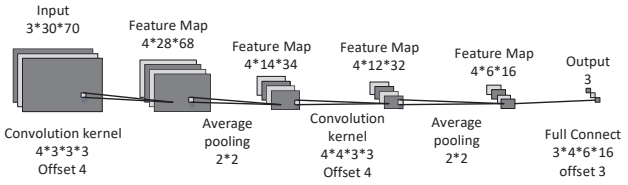


FIG. 12. Network structure design

types. The data set includes yellow solid line, white dotted line and white solid line.

According to the center position of the sliding box obtained in IV.C, an area containing the lane line is extracted

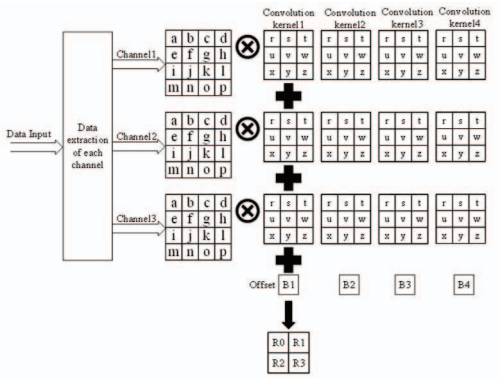
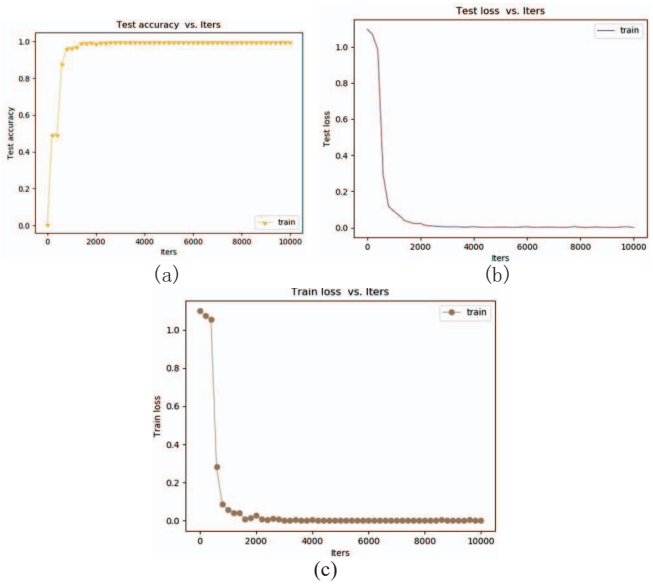


FIG. 13. Convolution operation



(a) (b) (c)

FIG. 14. Training results. (a) test accuracy; (b) test loss; (c) training loss

from the original image for the convolutional neural network to recognize the lane line type, as shown in figure 11.

This paper chooses to use the network structure provided

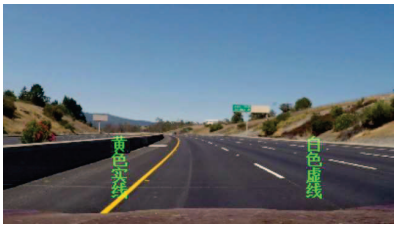


FIG. 15. Network test results

by Caffe for cifar10 data set. This paper modifies the network based on this network, and finally designs the network structure of 3-channel input and 3 classification result output, as shown in figure 12.

The convolution operation involves more repetitive steps. In this paper, all the channels of the input data are convoluted simultaneously with the same convolution kernel, so that the convolution operation time is greatly reduced. In addition, all convolution kernels can be operated together to further reduce computation time. The calculation process is shown in figure 13.

The network is trained in a Linux environment, and the training results are shown in Figure 14.

The lane line recognition result is shown in Figure 15.

In the figure, "yellow solid line" is displayed on the left lane line, indicating that the lane line type identified here is yellow solid line after inference network. Similarly, the lane line type on the right is white dotted line, which is consistent with the actual situation.

VI. ALGORITHM OPTIMIZATION DESIGN IN FPGA

Since the lane detection system designed in this paper needs to be implemented on the ZYNQ hardware platform, it is necessary to consider whether the hardware resources such as LUT, DSP, BRAM, etc. on the platform are sufficient and whether the real-time requirements can be met, but the resources and speed are antagonistic relationship. So this

article needs to optimize the program to meet the design requirements in terms of resources and real-time.

A. Parameter fixed point

In this design, floating point multiplication and addition

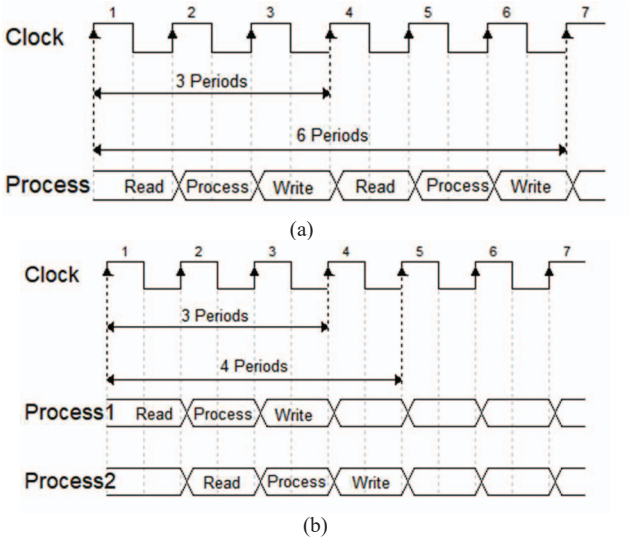


FIG. 16. Pipeline optimization. (a) Unused pipeline; (b) Use pipeline

operations are involved in many places, especially when implementing the lane line type identification network, which consumes a lot of time and is unfavorable for real-time performance. In order to save the comprehensive time, the program of the lane line type identification network part and the recognition result display part is extracted here, and the floating point number rotation and the fixed point number operation comparison experiment are separately performed. By using fixed-point numbers instead of floating-point numbers, the timing convergence effect can be obtained, and at the same time, the smaller Latency and Interval are obtained, which greatly improves the real-time performance.

B. Pipeline

Pipeline optimization is the most commonly used optimization strategy to increase throughput and reduce latency. In this paper, more pipeline operations are used in the implementation of lane detection. For example, when implementing the threshold segmentation module, the whole process is divided into three steps: reading data, processing data, and writing data. It takes 3 clock cycles to complete the an operation without the pipeline. If two operations are completed, a total of six cycles are required (as shown in Fig. 16(a)), and it takes a lot of time to complete the traversal of the entire image. After using the pipeline, it takes 3 cycles to complete the first operation, but only one cycle is required for the completion of the later operation (as shown in Figure 16(b)), which greatly reduces the delay. Pipeline processing is used in modules such as inverse perspective transformation and lane line fitting, which greatly improves the processing speed.

C. Array segmentation

Array segmentation is to store the data in one storage area into multiple storage areas, which can read the data in multiple storage areas simultaneously, reducing the time required to read data. For example, when performing convolution operations, the reasonable segmentation of the input data can greatly shorten the convolution operation time.

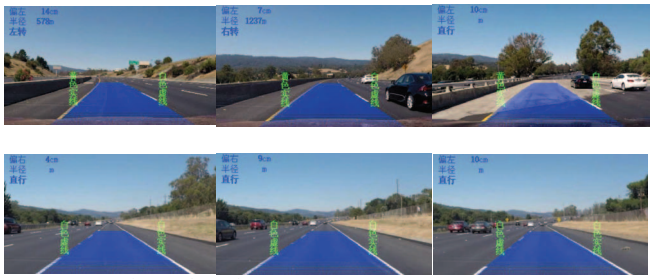


FIG. 17. Experimental results

TABLE II. PERFORMANCE COMPARISON RESULTS OF PLATFORM, ARM PLATFORM AND PC PLATFORM IN THIS PAPER

VII. THE EXPERIMENTAL RESULTS AND PERFORMANCE COMPARISON

Then, according to the characteristics of FPGA, the program is optimized by pipeline localization, and the final implementation result is shown in figure 17.

The lane line detection and lane line type identification program was transplanted to the PC i7 8750H platform and ARM Cortex A9 for testing, and the performance comparison results are shown in table 2

VIII. CONCLUSION

In this paper, HLS tool in FPGA design is used to complete the development of FPGA circuit related to lane detection, image processing algorithm is used to realize lane line detection, and convolution neural network algorithm is used to realize lane line type recognition. Then, the pipeline and fixed point method are used to optimize the program and improve the real-time performance of the system. Finally, it is implemented on the Xilinx ZYNQ7035 SoC development platform, which can output information such as lane curvature radius, vehicle deviation direction and distance from lane center, left and right lane line type, and the processing speed is over 104 frames, meeting the real-time requirements.

REFERENCES

[1] Wenjie,Song Yi Yang,Mengyin Fu,et al.Lane Detection and Classification for Forward Collision Warning System Based on Stereo Vision[J].IEEE Sensors Journal,Volume 18,Issue 12,June15,15 2018:5151-5163.

[2] Sahar Malmir,Majid Shalchian.Design and FPGA implementation of dual-stage lane detection, based on Hough transform and localized stripe features[J].Microprocessors and Microsystems,Volume 64,2019,Pages 12-22.

[3] M.Borkar,M.Hayes,M.Smith.A novel lane detection system with efficient ground truth generation,IEEE Trans.Intell.Transp.Syst.,13(1)(March 2012),pp.365-374

[4] Seokha Hwang,Youngjoo Lee.FPGA-based real-time lane detection for advanced driver assistance systems[J].2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS).

[5] Yuanjin Li, Wancheng Zhang, Nanjian Wu. A fast lane detection system based on parallel processor and FPGA implementation [J]. Journal of electronics and information technology,2010,32(12):2901-2906.

[6] P.Y.Hsiao,C.W.Yeh,S.S.Huang,L.C.Fu.A portable vision-based real-time lane departure warning system:day and night.IEEE Trans.Veh.Technol.,58(4)(May 2009),pp.2089-2094.

[7] Wu yanxia, liang kai, liu ying, cui huimin. Progress and trend of deep learning FPGA accelerator [J]. Journal of computer science, Vol. 41, 2018 Paper online publication number No.118, Online Publishing No.118 1-21

	operating frequency	Processing time of a frame （ms）	fps	time-consuming comparison （times）
This paper	100MHz	8.74664~9.53284	104.9~114.3	1
ARM platform	1GHz	417~422	2.3~2.4	47.6
PC platform	2.2GHz	15.4~18.2	55~65	1.8