2017 International Conference on Security, Pattern Analysis,
and Cybernetics (SPAC)

Research and Implementation of Parallel Lane Detection Algorithm Based on GPU

Ying Xu
College of Computer Science
Chongqing University
Chongqing , China
xuy@cqu.edu.cn

Bin Fang
College of Computer Science
Chongqing University
Chongqing , China
fb@cqu.edu.cn

Xuegang Wu
College of Communication
Chongqing University
Chongqing , China
xgwu@cqu.edu.cn

Weibin Yang
College of Automation
Chongqing University
Chongqing,China
ywb@cqu.edu.cn

Abstract—**Graphic Processing Unit (GPU) with the powerful computing ability is widely used for Parallel Computing. This paper raised a parallel Lane Detection Algorithm based on GPU acceleration, which could reduce the computing time for processing large amounts of data and solve large-scale complex problems. We implemented Median filter, Differential excitation and Hough transform on compute unified device architecture (CUDA). This algorithm took the advantages of GPU in parallel computation, memory management and reasonably allocated the computational resources and the corresponding computational tasks to the host and device in the Lane Detection. In this paper, different size of the image are processed and the experiment result proved that with the amount of data increases, the GPU acceleration will get good results.**

Keywords—Lane detection, Differential excitation, GPU, Hough transform, CUDA

I. INTRODUCTION

Lane detection based on machine vision is to extract lane markings from images which are taken from cameras mounted on intelligent vehicles. As a key research topic in the field of intelligent transportation, it has always been a hot topic of scholars [1-4]. In the past few decades, many vision-based algorithms have been proposed for lane detection [5-9]. [9] propose a novel robust lane detection method based on differential excitation. They extract the region of interest (ROI) by considering human visual attention, then enhance texture information and remove the noise effectively through differential excitation. They employ the Weber's law to obtain the binary images. Furthermore, they select the points that satisfy the proposed rules as voting points. Finally, they get lane markings through Hough transform. This method works well when lane markings are blurred and vestigial, But it has poor real-time performance. In order to improve the speed of lane line detection, [10] adopt a parallel accelerated processing technology to improve the speed of detecting line for Hough transform. They divide the image into two interesting regions assigned to two CPUs respectively for processing. Then, the two independent detecting results are combined. Experimental results indicate that the detecting speed is only increased by 39. 1% in the proposed method.

GPU is specialized microprocessor that accelerate 3D or 2D graphics operations. Recent GPU, which has many processing units connected with a global memory, can be used for general purpose parallel computation. CUDA (Compute Unified Device Architecture) [11] is an architecture for general purpose parallel computation on GPU. Using CUDA, we can develop parallel algorithms to be implemented in GPU. Therefore, many studies have been devoted to implement parallel algorithms using CUDA [12,13].

In this paper, we introduce the method of GPU-based parallel lane detection algorithm. we make full use of the characteristics and advantages of the GPU in parallel computation, combing Median filter, Differential excitation, Hough transform with the CUDA parallel programming model to accelerate these algorithm.so that the system can better meet the performance requirements of practical applications. The proposed method includes four steps: parallel fast Median filter based on CUDA, parallel differential extraction based on CUDA and parallel improved Hough transform based on CUDA. The flow of the algorithm is shown below Fig.1.

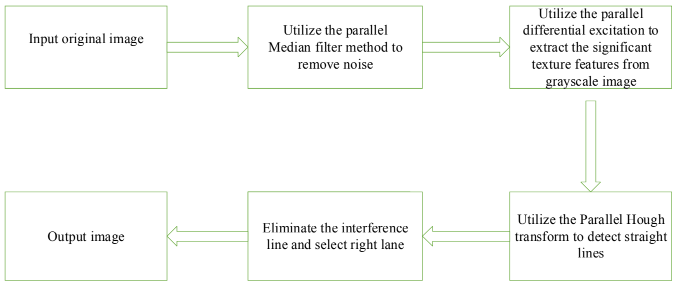
Input original image
Utilize the parallel Median filter method to remove noise
Utilize the parallel differential excitation to extract the significant texture features from grayscale image
Utilize the Parallel Hough transform to detect straight lines
Eliminate the interference line and select right lane
Output image

Fig. 1. Block diagram of the proposed algorithm

The main contribution of this paper is to present a new parallel lane detection algorithm based on GPU. Firstly, we utilize the parallel median filter method to remove noise from original image and use the parallel differential excitation to extract the significant texture features from grayscale image. Then we reduce redundant points by selecting voting points from binary image, which decreases the computation burden of the subsequent processing. Finally, we use the parallel Hough transform to extract lane markings. The experiments demonstrate that the proposed algorithm has better accuracy and less computation time against the typical methods.

The reminder of this paper is organized as follows: In Section II, the proposed method is elaborated with the main steps described in Sections II.A to II.C. The relevant experiment results are presented and discussed in Section III. Finally, conclusions are drawn in Section IV.

II. THE PROPOSED PARALLEL ALGORITHM

In this section, we describe our parallel method for lane detection based on GPU in details. Given an input image, we first obtain its ROI and transform it to a grayscale image and we deal with the grayscale image with parallel fast Median filter algorithm based on GPU in Section A. Then, we transform grayscale image to a binary image by parallel differential extraction based on a shared memory in Section B. Finally, we extract the lane markings with the parallel Hough transform method in section C.

A. *Fast Median Filter*

Acquired by the camera mounted on a moving vehicle, the images usually contain some useless information except lane markings, such as background and noise [9]. To improve the detection speed, the proposed method furthest enhances lane markings and removes the interference information by image preprocessing. In this paper, image preprocessing includes ROI cropping, graying and noise reduction in order. To realize median filtering algorithm efficiently on CUDA platform, uses a special fast median filtering algorithm [14]. The processing of fast median filtering is shown below Fig. 2.

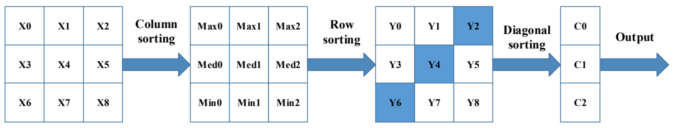

Fig. 2. The processing of fast median filtering

$$Y2 = \min[\max 0, \max 1, \max 2] \tag{1}$$

$$Y4 = med[mid0, mid1, mid2] \tag{2}$$

$$Y6 = \max[\min 0, \min 1, \min 2] \tag{3}$$

$$C1 = med[Y2, Y4, Y6] \tag{4}$$

The method is very suitable for parallel computing on the real-time processor and GPU architecture can be more energy efficient than CPU architecture for certain workloads [15]，A great improvement has been made on the selected algorithm according to the features of GPU computing. The improved algorithm uses texture memory to store data sources, shared memory and registers to store operations result. And by the method that threads in the same block can share sorted results, the number needed to compare in the computing process is rapidly reduced. it can be effectively applied in real-time image processing. Fig.3 shows the Overview of the parallel median filter algorithm:

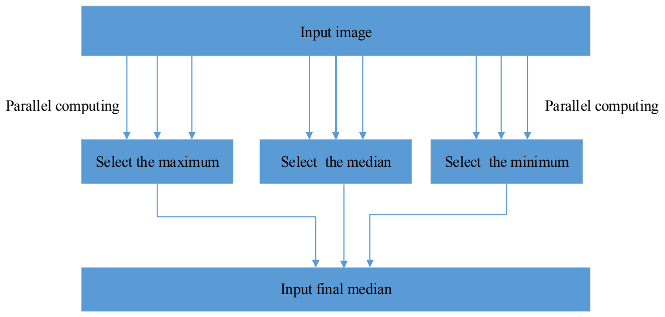

Fig. 3. Overview of the parallel median filter algorithm

B. *Parallel Differential excitation*

Weber's law reveals the relationship between physical quantity and mental quantity [16]. Weber's law is useful in image processing and the differential excitation is derived from Weber's Law [17].the differential excitation result of the image is computed as follow:

$$k_1 = I \otimes f_1 \tag{5}$$

$$k_2 = I \otimes f_2 \tag{6}$$

Where I represents the grayscale input image, $\otimes$ represents convolution operation, $f1$ and f_2 are two different masks, The mask f_1 is a 3×3 local window for computing a pixel's sum of the relative intensity differences against its 8 neighbors. The mask f_2 is used to get the intensity of the image. k_1 denotes the sum of the differences for each pixel and k_2 is the intensity of each pixel in the image .the differential excitation result is gotten as the following equation:

$$\xi = \arctan(\frac{k_1}{k_2}), (\xi \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)) \tag{7}$$

After image preprocessing, original images have been converted to grayscale images. The difference is obvious between lane markings and background in grayscale images, which can be identified by human eyes. Since the differential excitation reflects the degree of grayscale change in local window, we uses the differential excitation model to extract the significant texture features from grayscale images, but it has higher computation complexity of single CPU. To solve this problem, this paper proposes a parallel differential excitation algorithm with GPU based on CUDA by analyzing the parallelism of differential excitation. The main algorithm running on the main GPU is responsible for set Block and Grid size according to the hardware configuration, and transfers the grayscale image to graphics card for parallel computing.

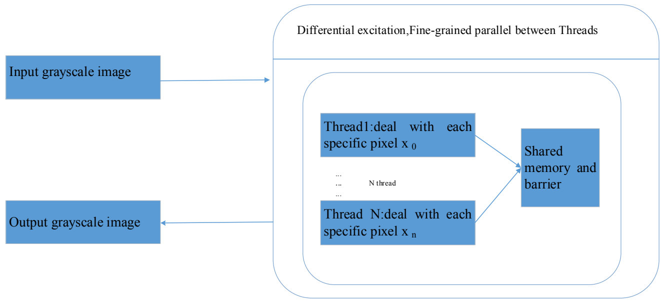
Differential excitation,Fine-grained parallel between Threads
Input grayscale image
Output grayscale image
Thread1:deal with each specific pixel x 0
N thread
Thread N:deal with each specific pixel x n
Shared memory and barrier

Fig. 4. Flow of parallel differential excitation Algorithm

As shown in Fig. 4, we set Block and Grid size according to the GPU hardware and image size, and its value is computed as:

$$GRID\_X = \frac{WIDTH + BLOCK\_X - 2}{BLOCK\_X - 2} \tag{8}$$

$$GRID\_Y = \frac{HEIGHT + BLOCK\_Y - 2}{BLOCK\_Y} \tag{9}$$

Where $WIDTH$, $HEIGHT$ are the height and width of the image respectively, $GRID\_X$ is the x coordinate of a block and $GRID\_Y$ is the y coordinate of a block, $BLOCK\_X$, is the x coordinate of a thread and $BLOCK\_Y$ is the y coordinate of a thread , The coordinate of a thread tells which point it will be operating on .For our parallel algorithm, each GPU thread block is set to have a thread dimension of 16 by 16.

AS the Fig.4 shown, Parallel Differential excitation Algorithm was implemented in the CUDA by parallel processing. We divide a big data into several small parts within the size of $BLOCK\_X$ and $BLOCK\_Y$, and each part will be assigned to a threads for processing. For example, thread N deal with the pixel X_n , and then the final results can be reorganized after merging the processed data.

C. Parallel Hough transform

On account of the limited field of vision, the lane shape can be regarded as straight line, and Hough transform is less sensitive to noises when it is used to detect straight lines [18]. Hence, we employ Hough transform in extracting lane markings. The basic idea of Hough transform is the duality between points and lines, which means that the slope and intercept of a straight line in the image space are corresponding to a point in the parameter space. Since the slope of the vertical line is infinite, Hough transform coverts all points in the image to sinusoids in polar coordinate parameter space by the following equation:

$$\rho = x\cos\theta + y\sin\theta \tag{10}$$

Where (x, y) is a point in the image space, ρ and θ are the magnitude and the angle respectively. The magnitude shows the probability of line existence in the image. the range of ρ and θ in this paper is respectively obtained as:

$$\rho_{\min} = 0, \rho_{\max} = (Width^2 + Height^2)^{1/2} \tag{11}$$

$$\theta_{\min} = 5^\circ, \theta_{\max} = 90^\circ - \arctan\left(\frac{Height}{2Width}\right) \tag{12}$$

Where $Height$ represents the height of image, $Width$ represents the width of image. If a candidate line is beyond the range, we regard it as interference information.

The Hough Transform is a digital image processing method for the detection of shapes which has multiple uses today [18]. A disadvantage of this method is its sequential computational complexity, particularly when a single processor is used. The main objective of this work is to modify and implement the computationally intensive and time consuming process of Hough Transform for line detection on GPU. In this paper, we use the powerful calculation function of GPU to the most time-consuming part of Hough transform to calculate. The task of intensive computations in Hough Transform process have been distributed among different cores of GPU. This process of Hough transform being data parallel process, it is suitable for execution on GPU and can produce excellent speedup in execution process. The flow of the algorithm is shown below Fig. 5.

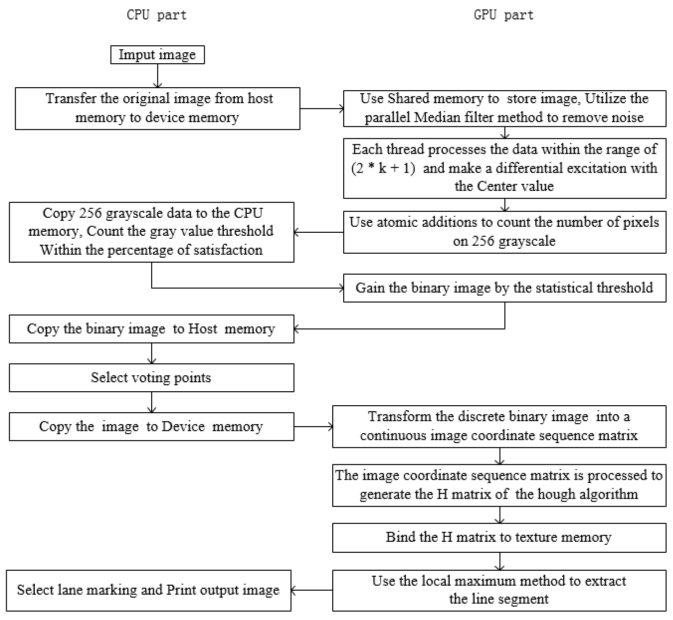
CPU part
GPU part
Imput image
Transfer the original image from host memory to device memory
Use Shared memory to store image, Utilize the parallel Median filter method to remove noise
Each thread processes the data within the range of (2 * k + 1) and make a differential excitation with the Center value
Copy 256 grayscale data to the CPU memory, Count the gray value threshold Within the percentage of satisfaction
Use atomic additions to count the number of pixels on 256 grayscale
Gain the binary image by the statistical threshold
Copy the binary image to Host memory
Select voting points
Copy the image to Device memory
Transform the discrete binary image into a continuous image coordinate sequence matrix
The image coordinate sequence matrix is processed to generate the H matrix of the hough algorithm
Bind the H matrix to texture memory
Use the local maximum method to extract the line segment
Select lane marking and Print output image

Fig. 5. Design Process of Hough transform Algorithm Based on CUDA

For straightforward mapping strategy, since workloads for different pixels in image domain are independent with each other, they can be mapped onto different threads. From the programming perspective, it can be simply realized by unrolling the outmost two loops of the sequential code and adjusting some necessary dependencies. The Design Process of Hough transform Algorithm Based on CUDA is shown in Fig.

5.Four major steps are involved. First, the original image is transferred from host memory to device memory. Then the GPU starts the transformation process. After the transformation is completed, GPU threads will search for intensive points in transformed domain and output them as straight line in a new image. In the end, the new image will be transferred back to CPU and print out new image with right lane.

In order to detect multiple maximum values in one time, the traditional serial algorithm is to divide the accumulator matrix into many blocks. In this paper, we use the local maximum method of calculating the maximum of four neighborhoods to replace the traditional method. When calculating the maximum of the four neighborhoods, each value in the accumulator matrix can be processed in parallel to improve the processing speed.

The way to calculate a four neighborhood is as follows: In the accumulator matrix, if the value is greater than the value in the four neighborhoods and is greater than the straight line threshold, it is considered a straight line.

$$\begin{cases} A(X_m, Y_m) > A(X_{m-1}, Y_m) \\ A(X_m, Y_m) > A(X_{m+1}, Y_m) \\ A(X_m, Y_m) > A(X_m, Y_{m-1}) \\ A(X_m, Y_m) > A(X_m, Y_{m+1}) \end{cases} \tag{13}$$

Where (X, Y) is a point in the matrix A, Where A represents the accumulator matrix of Hough transform.

A lane marking may be detected as several different straight lines, To avoid this problem, through sufficient experimental verification, the angle difference of each two detected lane markings can't be less than 8º[9]. When we find a candidate straight line, we compute the angle differences of the line against each detected lines. If anyone of the differences is less than 8º, we take the candidate line as interference line and directly remove the candidate line.

III. Experimental Results and Performance Analyses

In this section, the experimental results of several typical road conditions are displayed and explained. The proposed algorithm is tested on an open test database, which is together published by HIPS (Hefei Institutes of Physical Science, Chinese Academy of Sciences) and IAIR (Institute of Artificial Intelligence and Robotics, Xi'an Jiao tong University). And the database is captured by the autonomous vehicles in real traffic environment. We select 1233 image frames from the database as a test sample set with fixed interval. Besides, we conduct the experiments on Visual Studio 2013 ,OpenCV2.4.9 and Cuda7.5. The type of Graphics Card is NVIDIA GeForce GTX 660Ti .

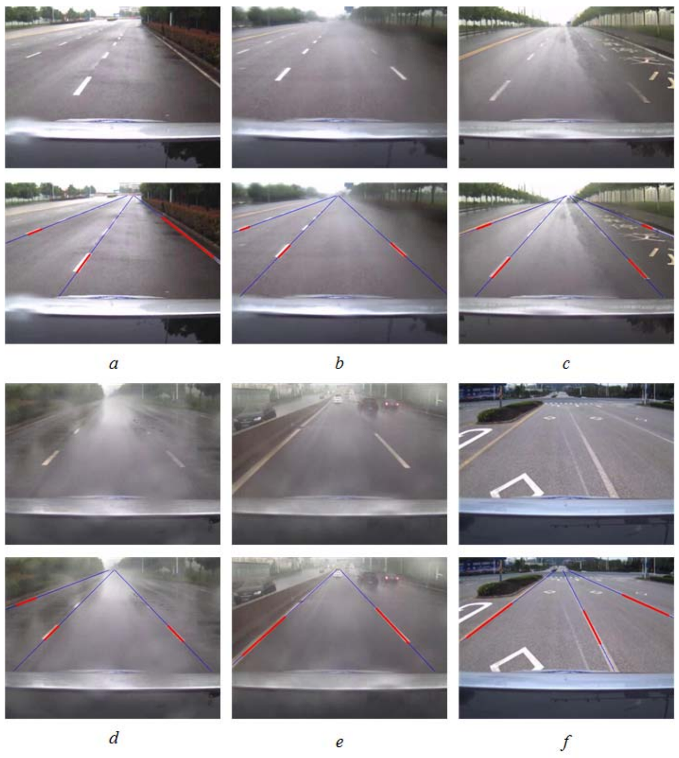

Fig. 6. Some experimental results on road images. The top images in *a-f* are original images and blue lines in the bottom images are detected lane markings.

TABLE I. PERFORMANCE OF FAST MEDIAN FILTER ON GPU

Dimension in pixels	*CPU(ms)*	*GPU(ms)*	*Speedup*
640*480	12.96	22.93	0.57
1280*960	39.62	24.79	1.59
1920*1440	80.07	28.84	2.78
3200*2400	201.56	38.00	5.30

TABLE II. PERFORMANCE OF DIFFERENTIAL EXCITATION ON GPU

Dimension in pixels	*CPU(ms)*	*GPU(ms)*	*Speedup*
640*480	18.13	17.63	1.02
1280*960	59.04	19.21	3.07
1920*1440	112.19	26.33	4.26
3200*2400	288.91	36.54	7.91

TABLE III. PERFORMANCE OF HOUGH TRANSFORM ON GPU

Dimension in pixels	*CPU(ms)*	*GPU(ms)*	*Speedup*
640*480	153.6	104.63	1.47
1280*960	599.00	111.64	5.37
1920*1440	1366.56	144.06	9.49
3200*2400	3589.21	162.11	22.14

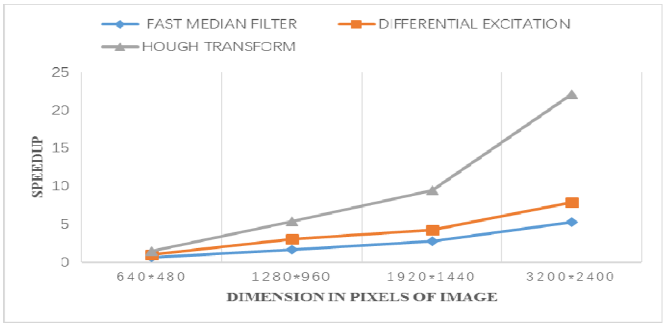

Fig. 7. Speedup of the GPU methods for three parallel algorithms.

The real-time performance is an important index to measure comprehensive performance of an algorithm. the lane in the image are detected. Also, the result of the GPU implementation is equivalent to that of the CPU implementation. we compares the computing time of our parallel algorithm to the traditional algorithm of three parts in table I ,table II and table III . According to the three tables , the computing time of every kernel of the GPU implementation is shorter than that of the CPU implementation. We have achieved 35.35 times speed-up with the size of image is 3200 by 2400 in total. Fig. 7 shows the speedup under different size of images in our three parallel algorithms . Also, we can see from Fig. 7 that the speedup, under same setting, grows at a super-linear rate with respect to the size of the data.

Experimental results from Table I to Table III and Fig.7 show that compared with the CPU-based acceleration algorithm, the proposed CUDA-based algorithm greatly speeds up the three parallel algorithms. But from Table I, we can find that the acceleration rate is less than 1, because when the data size is too small, the number of Blocks allocated on each multiprocessor is not enough and the GPU computing resources cannot be fully utilized, and when the data volume increases to a certain extent, The number of blocks allocated on each multiprocessor is also saturated, GPU computing resources have also been fully utilized, then if you continue increasing the amount of data, GPU computing time will grow with the size of data by the same growth rate. In GPU computing system, big Data can get a better speedup.

IV. Conclusions

In this paper, we introduce a parallel lane detection algorithm based on GPU for urban roads in some challenging scenarios The experimental results confirm that the proposed algorithm gains higher accuracy and robustness comparing to the typical methods. Furthermore, the proposed algorithm can meet the real time requirement well. Also, we have implemented it with a modern GPU system, NVIDIA GeForce GTX 660Ti. with the image size increasing, the speed advantage of the parallel algorithm is greater than the serial algorithm, and it obtains more than 35.35 times speedup measured in total.

Acknowledgment

This paper is supported by National Natural Science Foundation of China (61472053, 91420102, 61703062), China Postdoctoral Science Foundation (2014M550456) and Chongqing Postdoctoral Special Funding Project (Xm2014087), and Chongqing University Postgraduates' Innovation Project (No. CYS15019).

References

[1] Y. Sebsadji, J. P. Tarel, P. Foucher, and P. Charbonnier, “Robust road marking extraction in urban environments using stereo images,” in Intelligent Vehicles Symposium (IV), 2010 IEEE, 2010, pp. 394–400.

[2] J. Deng and Y. Han, “A Real-time System of Lane Detection and Tracking Based on Optimized RANSAC B-spline Fitting,” in Proceedings of the 2013 Research in Adaptive and Convergent Systems, 2013, pp. 157–164.

[3] Zhenyu He, Xin Li, Dachen Tao, Xinge You, and Yuan Yan Tang, Connected Component Model for Multi-Object Tracking, IEEE Transactions on Image Processing, 2016, vol.25, no.8, pp.3698-3711.

[4] Zhenyu He, Shuangyan Yi,Yiu-Ming Cheung, Xinge You, and Yuan Yan Tang, Robust Object Tracking via Key Patch Sparse Representation, IEEE Transactions on Cybernetics, no.47 pp.354-364, 2017.

[5] F. M. Sebdani and H. Pourghassem, “A robust and real-time road line extraction algorithm using hough transform in intelligent transportation system application,” in Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on, 2012, vol. 3, pp. 256–260.

[6] G. Liu, S. Li, and W. Liu, “Lane detection algorithm based on local feature extraction,” in Chinese Automation Congress (CAC), 2013, 2013, pp. 59–64.

[7] A. Borkar, M. Hayes, and M. T. Smith, “A Novel Lane Detection System With Efficient Ground Truth Generation,” IEEE Trans. Intell. Transp. Syst., vol. 13, no. 1, pp. 365–374, 2012.

[8] Ju Han Yoo, Seong-Whan Lee, Sung-Kee Park, Dong Hwan Kim, A Robust Lane Detection Method Based on Vanishing Point Estimation Using the Relevance of Line Segments, IEEE Transactions on Intelligent Transportation Systems, vol.99, pp.1-13, 2017.

[9] Zhao P, Fang B, Yang W B. A robust lane detection algorithm based on differential excitation[C]//Machine Learning and Cybernetics (ICMLC), 2016 International Conference on. IEEE, 2016: 1015-1020.

[10] Wang S, Xu B, Su D. Lane Line Parallel Detection Based on Hough Transform Hough [J]. Electronic Technology, 2015, (12):96-99.

[11] Z. Yang, Y. Zhu, and Y. Pu, “Parallel Image Processing Based on CUDA,” International Conference on Computer Science and Software Engineering, 2008.

[12] Suh, Jung W, Youngmin Kim, “Accelerating MATLAB with GPU Computing: A Primer with Examples”, Newnes, Boston, 2013.

[13] Tian W, Xu F, Wang H Y. Fast scale invariant feature transform algorithm based on CUDA ［J］. Computer Engineering，2010，36(8):219 －221.

[14] Wang Y X, He Y Y. Fast median filter algorithm based on FPGA[J]. Computer Application Research, 2009,26 (1): 224 ~ 226.

[15] Power, Jason, Yinan Li, Mark D. Hill, Jignesh M. Patel, and David A. Wood, “Implications of emerging 3D GPU architecture on the scan primitive”, ACM SIGMOD Record, Vol. 44, pp. 18-23, 2015.

[16] A. K. Jain, Fundamentals of digital image processing. Prentice-Hall, Inc., 1989.

[17] J. Chen, S. Shan, C. He, G. Zhao, M. Pietikainen, X. Chen, and W. Gao, “WLD: A Robust Local Image Descriptor,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 9, pp. 1705–1720, 2010.

[18] G. Xu, L. Yang, X. Liu, and R. Li, “Research of Road Extraction Based on Hough Transformation and Morphology,” in Computer Science Service System (CSSS), 2012 International Conference on, 2012, pp. 2261–2264.