

A Road Marking Extraction Method Using GPGPU

Dajun Ding¹, Jongsu Yoo¹, Jekyo Jung¹, Kwon Soon¹

¹Daegu Gyeongbuk Institute of Science and Technology, IT Converge Department
Daegu, Republic of Korea
{dajunding, ejong1, jekyo,soonyk}@dgist.ac.kr

Abstract. In driving assistance system (DAS), road marking's data can provide important assistance for driving safety. As the input image usually includes unnecessary information, lane detection system usually needs to remove most unnecessary data except for the lane markings. In this paper, a road marking extraction method is proposed to separate the painted lane lines using a special designed band-pass filter. Then we tried the acceleration method based on the integral image as well as the GPGPU. Road markings can be segmented automatically and adaptively in every frame. The proposed method is applied to various video images from black box, and is verified to be robust.

1 Introduction

The extraction of road markings is an essential step for several vision-based systems in intelligent transportation applications as for example Advanced Driver Assistance Systems (ADAS) [1].

As an important part of ADAS, lane detection can be used to get the position and orientation of the vehicle with lane data, and a region including roads is necessary to warn a driver of lane departure. Most road marking detection algorithms consist of the following three steps: Road marking feature extraction; Geometrical model estimation; Parameters' tracking of the model. This article focuses on the first step: road marking feature extraction. In fact, the feature extraction result has the direct relationship with the following processes. If the segmented lane lines are too redundant, the efficiency and accuracy will have an obvious bad influence.

This structure of this paper is described as follows. Section II presents related work of road marking extraction algorithms. In Section III we will describe our band-pass filter to achieve our goal. Some acceleration methods such as integral image will also be described in this section. In Section IV, we mainly described the expediting operation using GPUGU. And we also compare the efficiency between CPU and GPU. Finally, Section V gives concluding remarks and perspectives.

2 Related work

Despite the fact that road feature extraction is usually the first step of any lane detection system, it is often partially described in papers describing lane detection, even if numerous variants were proposed.

Different extractors have been proposed over the years to estimate the position of road markings parts in images. These extractors are based on the characteristics of the road markings, either geometric, photometric or both [1].

2.1 Threshold method

Being painted in white or bright yellow and usually locating on a relative dark road background, lane marking's segmentation can be done using threshold by a simple minded approach. The easiest threshold method uses a fixed value T_f to separate the pixels which grayscale values are smaller or bigger than T_f . It seems clear that this method shouldn't be robust to illumination. As for that, some adaptive threshold algorithms are applied on the lane detection field. For example, using OTSU algorithm to get the maximum probability of separate the foreground and background [2], or using local data of the image intensity within a small neighbor region of the current pixel. Some threshold based detection result is shown in Fig.1. Nevertheless, we need to care about one point: even though the lane markings are of more bright gray value, this issue is not vice versa. After threshold, lots redundant connected components will be gathered as the foreground, which will take following steps more processing time to remove them.

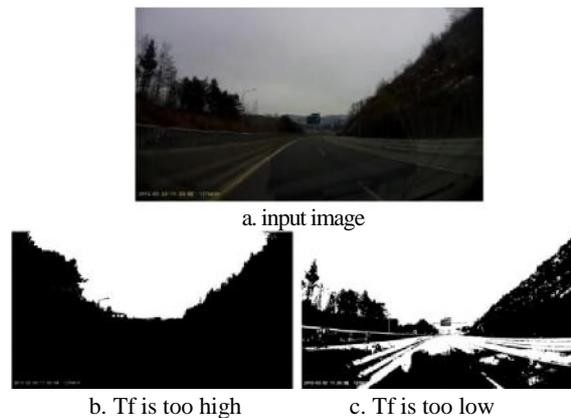


Fig.1. Using Threshold Method

2.2 Gradient analysis method

Except for the grayscale information, lane markings' geometrical data are also important features to track. Having the nature structure, road markings are usually

drawn by long and straight lines, especially for these solid lane lines. As for the dotted lane lines, the marking shape in the near field also has the similar shape. For that reason, some researchers decided to detect the edges from the grayscale image directly using some algorithms like Canny [3] or Sobel. A further gradient analysis may be followed to remove fault edges. Gradient analysis method is robust to the changing of the illumination. However this method suffers from an analogous problem of the threshold method, it will get too much information. The edge gradient data from grayscale image are even much more than the threshold results in which edges also need to be detected. On the other hand, turning gray image into edges will also lose the color information.

2.3 Specific designed filters

To combine grayscale data and gradient data in a single process, some specific filters are designed to extra the lane lines. Among them, one of the most famous 2D filters is the steerable filter [6].

$$\begin{aligned}
 (1) \quad & \frac{\partial}{\partial x} \left(\frac{1}{\sigma} e^{-\frac{x^2}{2\sigma^2}} \right) = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} \\
 (2) \quad & \frac{\partial}{\partial y} \left(\frac{1}{\sigma} e^{-\frac{y^2}{2\sigma^2}} \right) = -\frac{y}{\sigma^2} e^{-\frac{y^2}{2\sigma^2}} \\
 (3) \quad & \frac{\partial}{\partial x} \left(\frac{1}{\sigma} e^{-\frac{y^2}{2\sigma^2}} \right) = 0 \\
 (4) \quad & \frac{\partial}{\partial y} \left(\frac{1}{\sigma} e^{-\frac{x^2}{2\sigma^2}} \right) = 0
 \end{aligned}$$

Steerable filters have a number of desirable properties that make them excellent for a lane position detection application. Formula (1-3) described the partial derivative result at pixel(x,y). It has been shown that the response of any rotation θ of the Gxx filter can be computed using (4). The linearity of convolution enables to compute the maximum and minimum response of the filter. And a good response at point(x,y) for scale σ and rotation θ is chosen to be displayed as lane marking pixels. Steer filter should be combined with the IPM (Inverse Perspective Mapping) algorithm for calculation. These filters are optimal for one marking width. But they are efficient enough for tackling a large width range.

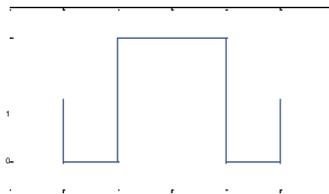


Fig.2. The shape of the top-hat filter

To fit the lane markings' dark-light-dark nature structure, a top-hat filter [5] which performs a convolution with a shape like Fig.2 is shown here. This simple one-dimensional kernel's scale should be changed according to different rows. And it doesn't need the calculation like IPM.

As this filtering method is relatively robust and can be easily accelerated using the integral image or the cumulative function, we decide to use a similar shape kernel for the lane markings' extraction. Since the operation in our method and the top-hat filter is so alike, we will introduce more characters of it in section III.

3 Road Marking Extraction Method

The first step of lane marking extraction in our method is road ROI (region of interest) detection. Determination of the road ROI is important especially in the lane detection because road environment is usually complex and the unnecessary information becomes noise. Isolating the ROI from other region before the computation can reduce the processing time and result in good performance [6].

In order to detect the start and end border of the ROI, the method we used combined the vanishing point detection and straight line segments analysis [6]. The detection result is illuminated in Fig.3, and the road ROI is highlighted using a red border.

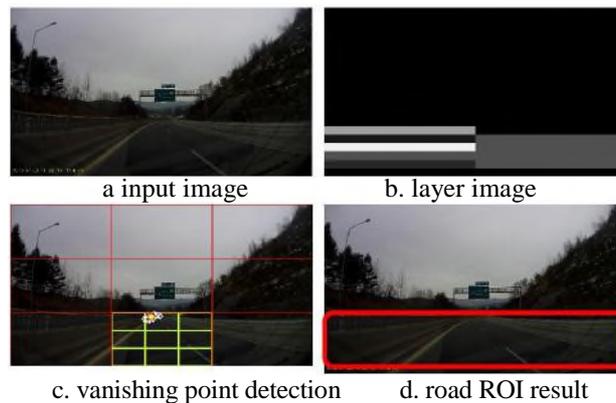


Fig.3. Road ROI Detection

After cutting the road ROI, the main processing of the road markings' extraction will be operated using the following one-dimensional band-pass filter. The main shape of this kernel is shown in Fig.4. As we can see, the form of our filter is similar to the original top-hat one. To fit the application in different scale image, the filter kernel's width should be related to the image size and the currently operated pixel's row coordinate.

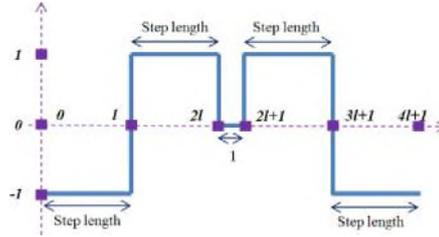


Fig.4. The shape of the one-dimensional band-pass kernel

There are some differences of this kernel and the original top-hat one. To get a center point for the convolution, the kernel width is set be an odd number. To make sure the positive and negative step length be the same, the value 1 parts and value -1 parts of the kernel has the same length. To fit the demand of parallel computing, the width of the filter kernel is more discrete.

The step length range of the kernel is changed from 1 to maximum value according to the image size. In our method, the maximum filter width is set as two times wider than the current row's marking's width, and the step length used a half value of the road width. The reason why we use some stipulation like this is we want the convolution has a positive value just when the kernel center got the marking's border and until the center is out of the road marking's region. Fig.5 shows how this kernel can be applied in one row. If the kernel got a position like Fig.5.a, the convolution response will be negative. Until the kernel's center arrived at the marking's left border. The filtering should have a maximum response when the kernel's center just locates in the middle of the road marking in situations like Fig.5.c. Positive response will last until the kernel is out of the right border like Fig.5.d.

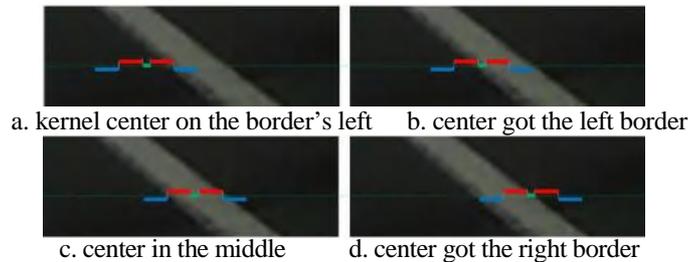


Fig.5. Convolution using the one-dimensional kernel

We tested the relationship between the video resolution and the step length in our experiment which is taken in a regular Korea highway environment. If the captured video resolution is 1080P (1920*1080), the nearest or the widest lane markings are from 45 to 50 pixels in the captured image, so the maximum length step here is about 23. If we use the 720p (1280*720) video, the widest marking is about 30-33, and the maximum step is about 15. So we decided to use a simple linear relationship in (7). The total size of the kernel is depicted in (5-7).

$$\left(\quad \right) * \quad + \quad (5)$$

non-marking parts are removed. Majority foreground part is the lane markings. Median filter removed lots point noise in Fig.6.c.

4 Algorithm acceleration using GPGPU

GPGPU (general purpose computation on GPUs) enables real time processing for an algorithm requiring huge computational cost. GPUs can only process independent vertices and fragments, but can process many of them in parallel. This is especially effective when the programmer wants to process many vertices or fragments in the same way. In this sense, GPUs are stream processors – processors that can operate in parallel by running one kernel on many records in a stream at once. In this section, we will describe the architectures on the GPU for road marking detection.

The main detection in our method has three parts: 1.Cumulative function calculation, 2.Getting the filter result in every point, 3.Median filtering. When we do the second and third steps, every point's handling has no relationship of other handled points. As for that, they are quite appropriate for parallel computing. For the cumulative or the integral image's calculation, in one line the summed value has a direct connection to the former point. So the parallelism is not as good as the other two steps. Even we separate the whole image cumulative function work into a work of line by line, the calculation time is still much larger than the CPU one. In our experiment, the cumulative function takes about 3.4ms and 2.0ms using one 1080P and 720P image, which are both much slower than the CPU's 2.1 and 0.9ms. In this case, for the first step, we use CPU's integral image's result. For the following the other two steps, algorithm will be worked on GPU. Under this circumstance, our method is a kind of heterogeneous computing.

For the experiment tool, we use OpenCL for testing. It is the currently dominant open general-purpose GPU computing language. And more experiment platforms information is shown in Table.1. Fig.7 illuminates the calculation results using GPGPU.

We used two different resolution videos for experiments, which are 1280*720 and 1920*1080 separately. The calculation time is shown in Tabel.2-3.

Table.1 Platform List

Platform	CPU	GPU	CPU Software	GPU Software
Type	Intel Core i7-4770	Nvidia GTX 760	Visual Studio	OpenCL
Parameter	3.40 GHz	2GB	2010	1.2

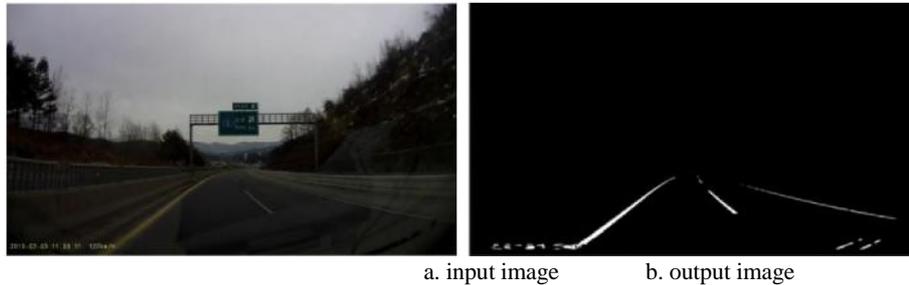


Fig.7. Using the band-pass kernel on GPU

Table.2 calculation time Using 720p video
*GPU's cumulative function part uses CPU's data

Video	Video1 1280*720		
	CPU original method	CPU integral method	GPU
Cumulative function	/	0.85	*0.85
Data uploading	/	/	0.28
Band-pass filter	6.98	2.31	0.22
Median filter	1.04	1.01	0.46
Data downloading	/	/	0.30
Total time	8.02	4.17	2.11

Table.3 calculation time Using 1080p video
*GPU's cumulative function part uses CPU's data

Video	Video2 1920*1080		
	CPU original method	CPU integral method	GPU
Cumulative function	/	2.07	*2.07
Data uploading	/	/	0.46
Band-pass filter	18.71	5.29	0.33
Median filter	2.14	2.10	1.05
Data downloading	/	/	0.50
Total time	20.85	9.46	4.41

From the data analysis in Table.2-3, we have some conclusions as follows: 1. Computing steps in our work nearly happened at the same time, the calculation speed for the parallel computing is very fast. 2. It takes a relative large time for data transformation from memory to GPU. 3. With the increasing of the image size, the GPU acceleration multiple also has an improvement.

Since the main time is used in the data transformation step, so both the data uploading and downloading only need to be done for one time in a single frame. In fact, in the following steps of the lane detection, we can also use Hough transform or RANSAC algorithms which are very suitable for parallelism. With a simple thinking, the acceleration ratio using GPU can be much bigger if we finish the majority work on the GPU instead of CPU. For more GPGPU acceleration comparison results, the magnification data are shown in Table.4.

Table.4 Acceleration multiple

Video	Video1 (1280*720)		Video2 (1920*1080)	
	CPU integral method	GPU	CPU integral method	GPU
Band-pass Filter	3.0	31.7	3.5	56.7
Median Filter	1	2.3	1	2.0
Total time	1.9	3.8	2.2	4.7

5 Conclusion

In this paper, we proposed a band-pass filter based methods to extract the road painting marking from a captured video. The kernel we used is created discretely for the demand of parallel computing. The original filter convolution is speeded up by using the cumulative function and GPGPU. In the end, we compared the detection time of different methods. As shown in our experiment results, most background information can be removed, and nearly all the road marking's information can be extracted efficiently and robustly.

Acknowledgement. This work was supported by the DGIST R&D Program of the Ministry of Science, ICT and Future Planning (14-IT-02, 14-HRLC-01).

References

- 1.Veit, T., et al.: Evaluation of road marking feature extraction. Intelligent Transportation Systems, ITSC, IEEE, pp. 174-181, 2008
- 2.Xu, H., Li, H.: Study on a robust approach of lane departure warning algorithm. Signal Processing Systems (ICSPS), IEEE, 2: V2-201-V2-204, 2010
- 3.Z. Xu, Li, Z. F.: A Robust Lane Detection Method in the Different Scenarios. International Conference on Mechatronics and Automation (ICMA), pp.1358-1363, Chengdu, China, Aug. 2012
- 4.McCall, J. C., et al: Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation", Intelligent Transportation Systems, IEEE Transactions, 7(1): 20-37, 2006
- 5.Huang, A. S., Moore, D., Antone, M. et al.: Finding multiple lanes in urban road networks with vision and lidar. Autonomous Robots, 26(2-3): 103-122, 2009
- 6.Ding, D., Lee, C., Lee, K. Y.: An Adaptive Road ROI Determination Algorithm for Lane Detection. TENCON 2013, International conference on, IEEE, 2013