khazanah informatika
Jurnal Ilmu Komputer dan Informatika

Detection of Highway Lane using Color Filtering and Line Determination

Iwan Muhammad Erwin[*], **Dicky Rianto Prajitno** , **Esa Prakasa**
Computer Vision Research Group, Research Center for Informatics
National Research and Innovation Agency (BRIN), Bandung, Indonesia
*iwan018@brin.go.id, masime2000@gmail.com

Abstract - Traffic accidents are generally caused by human error as a driver. The main cause is that the vehicle shifts away from the driving lane without the driver realizing it. Usually because the driver is sleepy or drunk. Therefore, it is necessary to have a system that functions to assist the driver's navigation to stay on the correct driving path, such as a driver assistance system (DAS). In this system, the driving lane detector is the main part. This system serves to assist the driver's navigation to stay on the correct driving path. Vehicles are installed with cameras to record video towards the road ahead. Computers are also installed for image processing, identifying left and right road lines and forming ego-lane. This paper offers an image processing-based method for recognizing driving lane and presenting visualizations in real time. This method has been tested using a data set, that video driving on Indonesian highway on the Cipali and Palikanci sections using dashboard camera. The test results obtained an accuracy of 99.25%.

Keywords: lane detector, camera, image processing, Indonesian highway, real time, accuracy

Article info: *submitted September 27, 2021, revised December 26, 2021, accepted January 17, 2022*

1. Introduction

The line on the road surface is helpful as a guide for the driver to direct his vehicle. This line has various forms, namely dash lines and solid lines. Its application on the road can be in the form of single lines and double lines. The colors used for the road markings are generally white and yellow, following the Regulation of the Minister of Transportation of the Republic of Indonesia No. 34 of 2014 concerning Road Marking. In this regulation, it is also explained that the dash line-markings function as lane dividers, traffic directors, this line have a minimum width of 10 cm. These dash lines have the same length, 3 meters for designations with a vehicle design speed of less than 60 Km/h with a distance between lines of 5 meters. As for the design speed of vehicles more than 60 Km/h, the line length is 5 meters, and the distance between lines is 8 meters. The solid line functions as a barrier, and divides the path and prohibit crossing this line. The thickness of the solid line as a roadside barrier is at least 10 cm and 15 cm for highway. Identifying and tracking road lines is key to developing algorithms for driver guidance systems or driverless vehicles.

The driver assistance system (DAS) is constantly evolving. The development of DAS was started from Lane Keeping Assistance (LKA), Lane Departure Warning System (LDWS), Lane Centering Assist (LCA), to Advanced Driver Assistance System (ADAS). In recent years essential ADAS functions such as LKA, LDWS and LCA have become common in the middle-class cars [1]. The utilization of the assistance system is expected to reduce accidents events.

LKA is a system where the vehicle will maintain its position to remain in the driving lane. LDWS is a system where an alarm will sound to warn the driver that the vehicle moves away from the driving lane. LCA is a system that helps the vehicle to stay in the middle of the driving lane. Suppose the vehicle is somewhat shifted to the right. In that case, the system will automatically adjust the steering wheel slightly to the left so that the vehicle position remains in the middle of the driving lane. The three methods above will turn off if the driver turns on the turn signal sign turning left or right, usually used to change lanes.

While ADAS has a broader meaning because it includes control of the Anti-lock Braking System (ABS), cruise control, blind-spot monitoring, auto parking control, and communications for navigation. ADAS generally uses multiple sensors, including LIDAR, radar, camera, image processing and computer vision.

A good highway must have a clear road line. With the condition of excellent and clear road line markings and long roads, drivers can use this to activate the ADAS system on their vehicles. It will help the driver rest for along this road.

Research on lane detection has been carried out, among others: Weiwei Chen et al. conducted a review of lane detection in image processing and semantics and concluded that research on lane detection is still a challenge [2]. Li Yong Ma et al. introduced adaptive threshold segmentation of lane gradient images by combining the Canny edge detection technique and the ONSU algorithm [3]. Peerawat et al. used Randomize Hough Transform to detect lane driving and claimed that the computation time was shorter than the standard Hough Transform [4]. Jinsheng Xiao et al. used an extended Kalman filter to detect lanes and claimed that the recognition rate was better than the RANSAC method [5]. Trun-Thrien Tran et al. used the HSI color system to help detect lane driving and showed that the results were much better than RGB color space [6]. Kyoungtaek Choi et al. introduced ego path detection using a method based on the endpoint of the highway path by comparing it with positional data obtained from GPS [7]. Chin-Pang Huan et al. introduced an adaptive road mask to detect lanes by determining the vanishing point and recalculating the ROI [8]. A. F. Cela et al. proposed an algorithm to detect road lanes based on unsupervised and adaptive classifiers using HSV images [9]. J. G. Kuk et al. presented a computationally efficient and robust path detection algorithm based on the Hough transform [10]. Chao Li et al. succeeded in detecting multi-lane using a Kalman filter and a monocular camera [11]. VioLET, introduced by J. C. McCall and M. M. Trivedi, is a system that can detect and track driving lanes [12]. P. V. Date and V. Gaikwad combine image processing with fuzzy logic to determine the lane detection and departure warning system [13].

Implementing a driving lane detector using image processing is highly dependent on the road environment. Roads must have clear and uniform road lines. Based on the regulations in Indonesia, there are several types of road lines, namely dash lines, and solid lines The lines have to be colored in either white or yellow. Problems that often occur are exposure to light on the road, shadows on the road, faded road lines due to weather, peeling road lines, and bad weather [14]. Some methods can improve the image quality, namely Retinex [15], [16]. Using Retinex, the problems can be minimised. However, there will be other problems, for example, the computational process that is increasingly complex and time-consuming.

The paper consists of four sections. The first section introduces research background and presents previous works related with this paper. The second section discuss the method used in this research. The method is constructed by a sequence of processing steps, namely data video acquisition, ROI determination, blur correction, HSL-grayscale color conversion, the edge and line detections, and ended by drawing lines to the video frames. The algorithm results are described in the third section. The fourth section concludes and summaries all facts found in the research. The author acknowledgment is given in the end of the paper.

1. Methods

Most lanes are designed to be relatively easy not only to encourage order but also to make it easier for human drivers to drive vehicles at a consistent speed. Therefore, the first approach is to detect straight lines using edge detection and feature extraction techniques. We will use OpenCV, an open-source library of computer vision algorithms.

The method can be explained as follows: applying ROI (Region of Interest), blur correction using Gaussian Blur function, changing the color system from RGB (Red Green Blue) to HSL (Hue Saturation Lightness), applying an HSL filter to a color that matches the color of the road line, changing to a grayscale image, applying the canny function, applying the houghLine function, forming left and right lines, forming an area driving lane, overlay with origin image. Repeat this process to the next image. The proposed method can be described as a flowchart and codeblock, as shown in Figure 1 and 2, respectively.

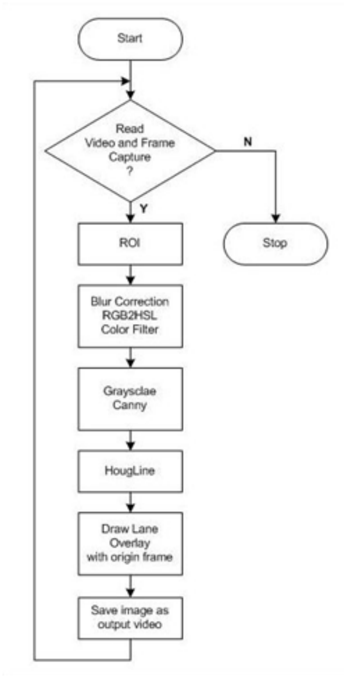

Start
Read Video and Frame Capture ?
N
Y
ROI
Stop
Blur Correction RGB2HSL Color Filter
Graysclae Canny
HougLine
Draw Lane Overlay with origin frame
Save image as output video

Figure 1.Flowchart of method

```
while openedCamera == True do
    $I_{RGB}$ ← capture(video)
    $I_{RGB}$` ← ROI($I_{RGB}$)
    $I_{RGB}$`` ← GaussianBlur($I_{RGB}$`, filterSize=5)
    $I_{HSL}$ ← colorConversionRGBtoHLS ($I_{RGB}$``)
    $I_w$ ← intensityMask($I_{HSL}$, $wTh_{Low} \leq I_{HSL} \leq wTh_{Up}$)        //white line threshold
    $I_y$ ← intensityMask($I_{HSL}$, $yTh_{Low} \leq I_{HSL} \leq yTh_{Up}$)        //yellow line threshold
    $I_{mask}$ ← $I_w \cup I_y$
    $I_{canny}$ ← edgeDetection($I_{mask}$, filterType=Canny)
    lines ← lineDetection($I_{canny}$, filterType=Hough)
    $I_{lines}$ ← drawLines($I_{RGB}$, lines)
end
```

Figure 2. Codeblock of method

a. Read Video and Frame Capture

The first step, read the video and capture a single video frame (an image), cv2.VideoCapture(videoFile). The captured image is 1280 ×720 pixels in size. There are 30 frames in 1 second video duration. An example of image capture results can be seen in Figure 3.

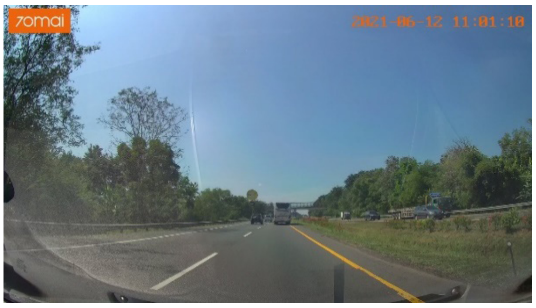


Figure 3. Display a frame from video capture

b. The ROI Determination

The next step is to determine the ROI, which is an interesting area to focus on. Within this area, several objects must be observed, while objects outside the ROI can be ignored. The shape of the ROI can be adjusted to the needs of the observations. It can be a square, triangle, circle, trapezoid or other polylines. Areas outside the ROI can be masked, for example, made black using the zeros_like and bitwise_and function. The image pixels in ROI will be preserved as its origin, while outside, the ROI will be made black. Using ROI makes the computational load smaller and faster. This article uses a trapezoid ROI. The ROI and masking in the image can be seen in Figure 4.

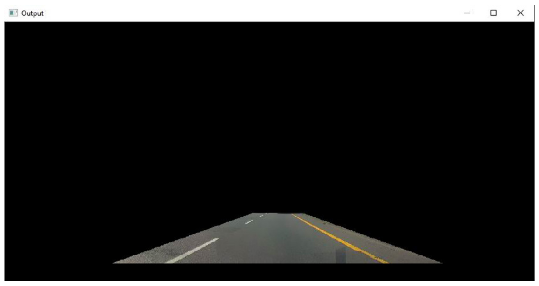


Figure 4. Display a frame with ROI and mask

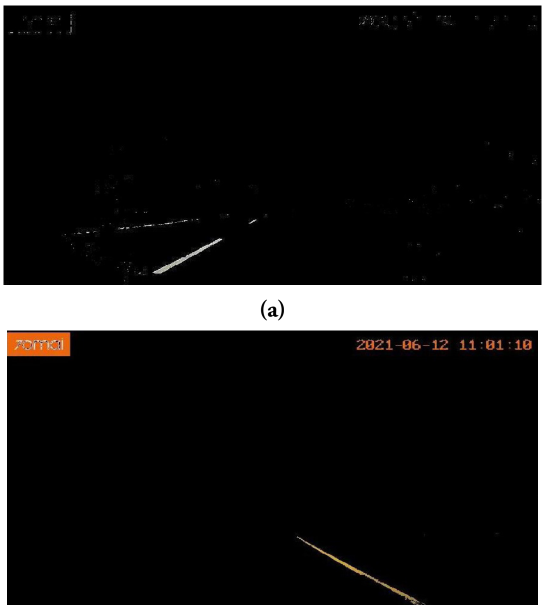
c. Blur Correction, Convert to HSL and Color Filter

The next step is blur correction and changing the RGB color system to HSL. The RGB color system can be written as rgb[red, green, blue], which combines the three basic colors. These three basic color parameters can be adjusted in intensity with integer numbers 0 to 255. Thus rgb[255, 0, 0] represents red, rgb[0, 255, 0] is green, and rgb[0, 0, 255] is blue. The RGB colour system makes it very difficult to represent a color due to differences in light exposure and color saturation.

The HSL color system can be described as a geometric cylinder. Hue values from 0-360 degrees, with a value of 0 being red. Increasing the value of the hue to 60, then it turns yellow. Hue equals 120 is green, 180 is cyan, 240 is blue, 300 is magenta, and back to 360 is red. Saturation and lightning values are between 0-100%, where pure color occurs at S=100% and L=50%. With a value of L = 0, all colors will be black, and with a value of L=100%, all colors will be white. Writing format as hsl[hue, saturation, lightness], for example hsl[0, 100%, 50%] displays pure red color. While pure blue is written as hsl[270, 100%, 50%].

Where the lower limit value of yellow is yel_low and the upper limit value is yel_up. Likewise with white color, namely white_low and white_up. A program has been created to determine the range of white and yellow colors that are passed. The white lower and white upper values can be set by sliding the slide bar. Then repeat using program to determine the lower and upper limits of yellow. The algorithm extracts the white lane by applying the intensity intervals as follows. The interval values of H, S, and L, are defined at 60 to 255, 130 to 255, and 15 to 45, respectively. The algorithm can find the yellow lane by only accepting the H, S, L values within the range of 90 to 255. The values outside this range will be either minimised to 0 or maximised to 255. The result is as in Figure 5.

In this case, the HSL range values for white are determined as follows: lower threshold [60, 130, 15] and upper threshold (255, 255, 45). In yellow, lower threshold [90, 90, 90] and upper threshold (255, 255, 255]. Combining white and yellow filters produces the image shown in Figure 6.



(b)

Figure 5. Applying algorithm to extract (a) white and (b) yellow

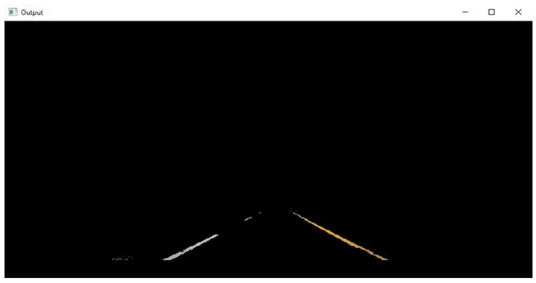


Figure 6. Frame view with ROI and white and yellow color filters

d. Grayscale and Canny Operation

The next step, grayscale and canny operations are performed. Canny edge detector is a method used to find edges in an image. Canny edge detection algorithm consists of several steps as follows: noise reduction, gradient calculation, non-maximum suppression, double threshold and edge tracking by hysteresis. In openCV library, there is a function to convert the image to grayscale and a Canny edge detector function. The result is shown in Figure 7.

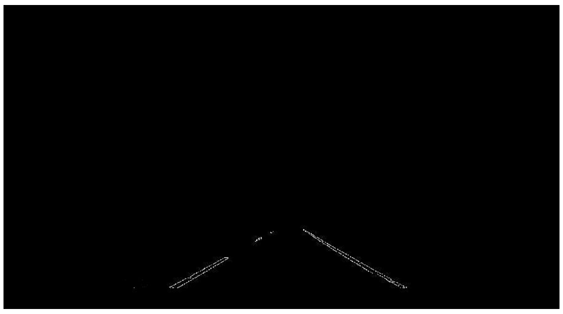


Figure 7. Grayscale and canny operation results

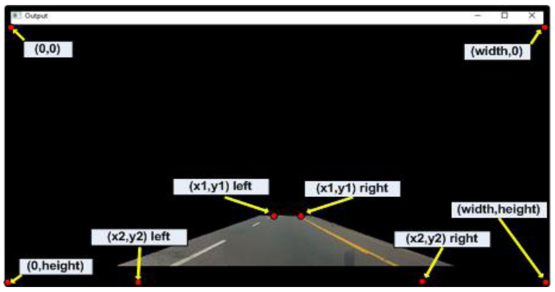


Figure 8. Position of X1, Y1 and X2, Y2 , left and right

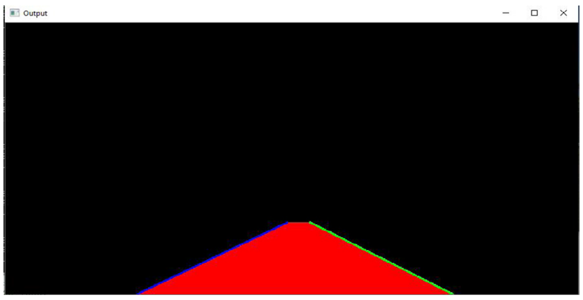


Figure 8. Closed Polyline as driving lane

e. Hough Transform and Draw Lines

From the Canny image, a Hough Transform operation is performed to detect the lines in this image. Hough transform produces multiple lines. For example, a line from point A(x_a,y_a) to point B(x_b,y_b), then the gradient or slope can be calculated by equation (1). Referring to the middle point of ROI, then from the slope value it can be seen that the line is on the left side if the slope is negative. The line is on the right if the slope is positive. y-intercept can be calculated by equation (2).

$$slope = \frac{(y_a - y_b)}{(x_a - x_b)} \quad (1)$$

$$yintercept = y_b - (slope * x_b) \quad (2)$$

The many lines generated by the hough transform can be calculated the average slope (avgslope). Calculations are made on the left and right sides of the line. Similarly, it is necessary to calculate the average y-intercept.
The positions of x_1 and x_2 can be seen from the illustration in Figure 7. From the average slope value and the values of the y_1 and y_2 coordinates determined, the importance of x_1 and x_2 can be calculated by the equation (3) and (4).

$$x_1 = (y_1 - yintercept)/avgslope \quad (3)$$

$$x_2 = (y_2 - yintercept)/avgslope \quad (4)$$

where y_1 is the same as the top y in the ROI plane, and y_2 is image height, as shown in Figure 8.

By knowing four points, namely (x1,y1) left, (x2,y2) left, (x1,y1) right and (x2,y2) right, it forms a closed polyline. The result is shown in Figure 9 and then overlaid with the original image. The result is as shown in Figure 10. And so on, it is processed again for the next video frame.

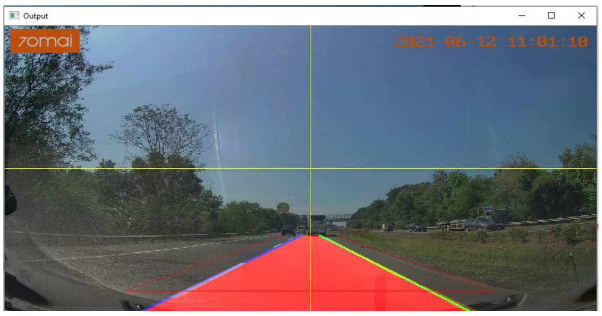


Figure 9. The final result is a driving lane on the video frame

We use video recordings of driving trips that show the left and right road lines to test our method. We use a dashboard camera installed in the cabin, in front of the center mirror, as in Figure 11.

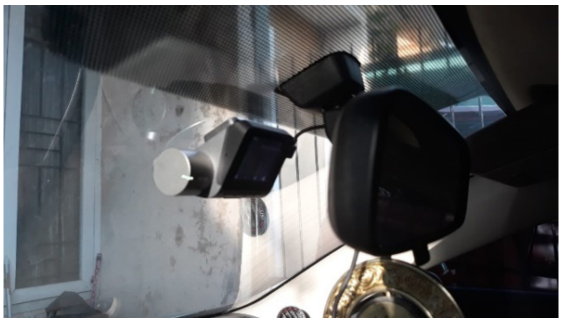


Figure 10. In-vehicle camera installation

Camera settings for recording 3 minutes video clip in mp4 format at a resolution of 1280x720 pixels. The camera frame rate is set at 30 fps. Dashcam video recording on the Cipali highway from KM 166 produced 3 videos, Cipali_1, Cipali_2 and Cipali_3. The road texture in this section is a mixture of concrete and asphalt-coated concrete roads. The Palikanci highway section from km 207, produced 3 videos, Palikanci_1, Palikanci_2 and Palikanci_3. The Palikanci highway section has a hot mix asphalt texture. Video capture during the day from 10.31 am to 11.10 am. Execution of program code with 6 video inputs above produces 6 output videos in avi format.

3. Result and Discussion

To calculate the accuracy of the lane detector, we use the Intersection over Union (IoU) method, as depicted in Figure 12. By comparing the ground truth image and the predicted image. Ground truth can be explained as a reality desired to be predicted by a model (system), through direct measurement and observation of objects. In this case, the ground truth can be created manually, that is, an area between the left and right road lines on the ego-lane, with the upper point is the same as the ROI upper point, and the lower point is the same the image lower point. At the same time, the detection result is the overlay generated by the model as lane detected. Furthermore, the ground truth and overlay images are changed to binary images (black and white), as shown in Figure 13. By counting the number of pixels in intersection and union, the accuracy can be calculated. The IoU value is computed by applying Equation 5.

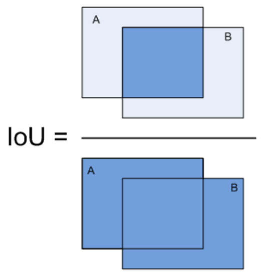


Figure 11. The IoU concept to calculate accuracy.

$$IoU = \frac{A \cap B}{A \cup B} \tag{5}$$

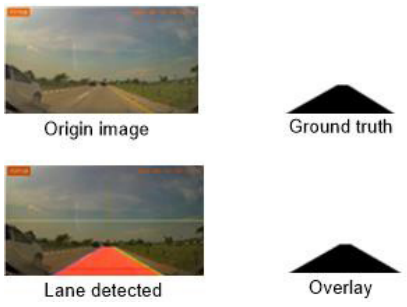
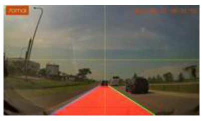
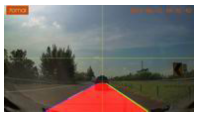


Figure 12. Binary image of ground truth and lane detected

From six output videos is sampled ten images respectively. In some cases, the image cannot be used as a sample, for example, when the vehicle changes lanes. Using out_cipali_1.avi, we sampled ten images, as shown in Table 1.

Table 1. Image sampling from video out_cipali_1.avi

No	File name	image
1	cip1_1.jpg	
2	cip1_2.jpg	
3	cip1_3.jpg	
4	cip1_4.jpg	
5	cip1_5.jpg	
6	cip1_6.jpg	
7	cip1_7.jpg	
8	cip1_8.jpg	
9	cip1_9.jpg	
10	cip1_10.jpg	

The sampled image is calculated for ground truth and its overlay by converting it into a binary image. Using the IoU concept, the calculation results are presented in Table 2.

Table 2. IoU results at out_cipali_1.avi sampling

File (jpg)	*Pixels*				*Acc. (%)*
	Ground truth	*Overlay*	*I*	*U*	
cip1_1	2533317	2515659	2513643	2535804	99.13
cip1_2.	2540178	2513103	2510268	2543418	98.70
cip1_3	2528508	2526459	2521524	2533950	99.51
cip1_4	2539005	2541966	2533461	2548011	99.43
cip1_5	2534217	2542419	2532423	2544753	99.52
cip1_6	2531979	2540454	2530188	2542938	99.50
cip1_7	2538501	2547894	2537082	2550033	99,49
cip1_8	2535918	2540868	2533554	2544027	99,59
cip1_9	2531421	2535081	2528382	2538804	99.59
cip1_10	2531328	2542950	2529486	2545323	99.38
				Avg. Acc.	99.38

The same process is carried out on the next output video, there are out_cipali2.avi, out_cipali3.avi, out_palikanci1.avi, out_palikanci2.avi and out_palikanci3.avi the results are presented in Table 3.

Table 3. accuracy calculation results on sampling result on six video outputs

Output video	*Acc.*
out_cipali_1.avi	99.38 %
out_cipali_2.avi	99.00 %
out_cipali_3.avi	99.00 %
out_palikanci_1.avi	99.45 %
out_palikanci_2.avi	99.35 %
out_palikanci_3.avi	99.30 %
Average accuracy	**99.25 %**

The computation time is depicted on Figure 14. The average processing time is 18.01 millisecond/frame. The algorithm is executed by using a personal computer. The computer specification are as follows: 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80 GHz processor, RAM 8 GB, and HDD 500 GB. The algorithm can complete the process at a rate of 18.01 milliseconds/frame. Meanwhile, the camera speed is set at 30 frames/second. By applying this speed, a new frame will be generated after 33,33 milliseconds. Since the processing time (0.018) is less than generated frame duration (0.033), then the process will be completed before a new frame appears. This condition can ensure the processing of lines detection can be completed in real-time.

When compared with the results obtained by other authors, using Adaptive Threshold Segmentation of Lane Gradient Image obtained accuracy 91,7% to 97,4% [3]; using Randomized Hough Transform produce accuracy 93.6 % to 95.2% [4]. Comparison of accuracy can be seen in Table 4.

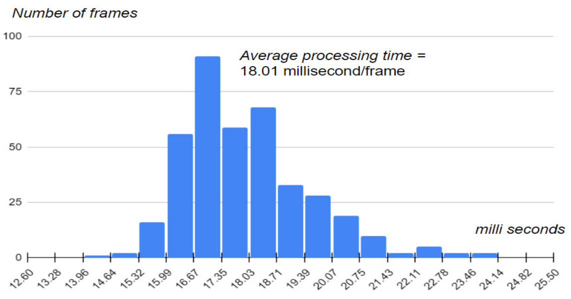


Figure 13. Processing time

Table 4. Comparison of lane detection methods

No	Author, Year	Data	Method	Accuracy
1	Li-Yong *et al.* [3], 2018	The Washington City lane data sets, RGB, Dashboard view	Hough transform and Kalman filter	91.7 -97.4 %
2	Peerawat *et al.* [4], 2018	RGB, Dashboard view	Randomized Hough transform	93.6 – 95.2 %
3	Our method	RGB, Dashboard view	Color filtering and Hough transform	99.25 %

4. Conclusion

Driving lane detection is the primary and most important part of the driver guidance system or autonomous vehicle system. This paper describes how to build a driving lane detection system for highway in Indonesia. Based on the test results and accuracy calculations, our method can detect driving lanes. The average value of accuracy is 99.25%. The problem occurs determines the lower and upper thresholds of the white and yellow filters. Sometimes there are sections of highway that do not fit this filter. Some sections of highway require slightly different filter values, possibly due to faded road markings. Also the influence of weather and light exposure. Thus, in the next research is how to make this white and yellow filter adapt to the above conditions, combined with adaptive ROI.

Acknowledgement

This research was supported by the National Research and Innovation Agency (BRIN) by providing research equipment. The work is part of the National Priority Program on Electrical Vehicle at the Engineering Science Research Organization, BRIN.

Reference

[1] O. Törő, T. Bécsi, and S. Aradi, "Design of Lane Keeping Algorithm of Autonomous Vehicle", *Period. Polytech. Transp. Eng.*, vol. 44, no. 1, pp. 60–68, Jan. 2016.

[2] Weiwei Chen, Weixing Wang, Kevin Wang, Zhaoying Li, Huan Li, Sheng Liu, "Lane departure warning systems and lane line detection methods based on image processing and semantic segmentation: A review", *Journal of Traffic and Transportation Engineering (English Edition)*, Volume 7, Issue 6, 2020, pp 748-774, 2020.

[3] M. Li-Yong, H. Chun-Sheng, H. Yu-Qing, L. Yun-Jing and Y. Pei-Lun, "A Lane Detection Technique Based on Adaptive Threshold Segmentation of Lane Gradient Image," *2018 4th Annual International Conference on Network and Information Systems for Computers (ICNISC)*, pp. 182-186, 2018.

[4] Mongkonyong Peerawat, Nuthong Chaiwat, Siddhichai Supakorn and Yamakita Masaki, "Lane detection using Randomized Hough Transform", *IOP Conference Series: Materials Science and Engineering*, vol 297, 2018.

[5] Xiao J., Luo L., Yao Y., Zou W., and Klette R., "Lane Detection Based on Road Module and Extended Kalman Filter", *Lecture Notes in Computer Science (LNCS)*, vol 10749, pp 382-395, 2018.

[6] Tran TT., Bae CS., Kim YN., Cho HM., and Cho SB. "An Adaptive Method for Lane Marking Detection Based on HSI Color Model", *Communications in Computer and Information Science (CCIS)*, vol 93, pp 304-311, 2010.

[7] Kyoungtaek Choi, Jae Kyu Suhr and Ho Gi Jung, "In-Lane Localization and Ego-Lane Identification Method Based on Highway Lane Endpoint", *Hindawi Journal of Advanced Transportation*, Volume 2020, 2020.

[8] Chin-Pan Huang, Chaur-Heh Hsieh and Jin-Long Li, "A Robust Lane Detection Method Using Adaptive Road Mask", *The Proceedings of the International Conference on Digital Information Processing, Data Mining, and Wireless Communication,* Dubai, UAE, pp 154-158, 2015.

[9] A. F. Cela, L. M. Bergasa, F. L. Sánchez and M. A. Herrera, "Lanes Detection Based on Unsupervised and Adaptive Classifier," *2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks*, pp. 228-233, 2013.

[10] J. G. Kuk, J. H. An, H. Ki and N. I. Cho, "Fast lane detection & tracking based on Hough transform with reduced memory requirement," *13th International IEEE Conference on Intelligent Transportation Systems*, pp. 1344-1349, 2010.

[11] Chao Li, Hongzhe Liu, Yongrong Zheng and Hanyu Xuan, "Multi-lane Detection Based on RMFP For Self-Driving in urban traffic scenes", *Proceedings of the 2016 4th International Conference on Sensors, Mechatronics and Automation (ICSMA 2016)*, pp. 692-703, Dec. 2016.

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

[13] P. V. Date and V. Gaikwad, "Vision based lane detection and departure warning system," *2017 International Conference on Signal Processing and Communication (ICSPC)*, pp. 240-245, 2017.

[14] S. Shirke and C. Rajabhushanam, "A study of lane detection techniques and lane departure system," *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, 2017, pp. 1-4, 2017.

[15] Analysa M. Gonzales, Artyom M. Grigoryan, "Fast Retinex for color image enhancement: methods and algorithms," *Proc. SPIE 9411, Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications 2015*, 94110F, March 2015.

[16] Ling Tang, Shunling Chen, Weijun Liu, Yonghong Li,"Improved Retinex Image Enhancement Algorithm," *Procedia Environmental Sciences*,Volume 11, Part A, pp. 208-212, 2011.