Transport and Telecommunication, 2021, volume 22, no. 4, 461–470
Transport and Telecommunication Institute, Lomonosova 1, Riga, LV-1019, Latvia
DOI 10.2478/ttj-2021-0036

REAL-TIME LANE LINE TRACKING ALGORITHM TO MINI VEHICLES

***Jozsef Suto*[1]**

[1]*Department of IT Systems and Networks, University of Debrecen*
Debrecen, Hungary, 26 Kassai street, 4028
suto.jozsef@inf.unideb.hu

Autonomous navigation is important not only in autonomous cars but also in other transportation systems. In many applications, an autonomous vehicle has to follow the curvature of a real or artificial road or in other words lane lines. In those application, the key is the lane detection. In this paper, we present a real-time lane line tracking algorithm mainly designed to mini vehicles with relatively low computation capacity and single camera sensor. The proposed algorithm exploits computer vision techniques in combination with digital filtering. To demonstrate the performance of the method, experiments are conducted in an indoor, self-made test track where the effect of several external influencing factors can be observed. Experimental results show that the proposed algorithm works well independently of shadows, bends, reflection and lighting changes.

Keywords: autonomous vehicle, lane line detection, lane line tracking, Hough transformation

1. Introduction

In recent years, autonomous vehicles attracted great attention not only in automobile industries but also in academic cycles. Leading automobile (Tesla, BMW, etc.) and information technology (IT) companies (Google, Baidu, etc.) invested a huge amount of money into the autonomous vehicle research and development (Hussain and Zeadally, 2019). The application of autonomous vehicles is also appeared in many industrial fields such as in human and goods transportation (Alonso *et al.,* 2020). Actually, the rapid development of autonomous vehicles is not so surprising because their application can significantly reduce companies' expenses and eliminate several human factors such as fatigue and drunk. Therefore, much progress has been made toward this direction and several methods have been proposed until now (Hu *et al.,* 2020).

For perceiving environment, typically two sensors are used: camera and/or LIDAR. It is well reflected by the leading self-driving car developer companies such as Google and Tesla. To the autonomous navigation, Google mainly uses LIDAR while Tesla's method is primarily based on vision (Zhao *et al.,* 2018). Obviously, in application requiring high degree of safety and precision, those sensors can be used in combination, even with different weighting.

Both sensor types have its own advantages and disadvantages. Currently, three-dimensional imaging radars are the most efficient sensors. They are widely used in self-driving cars. The main drawback of industrial LIDAR sensors is their price which can be even $80.000. In spite of LIDAR, the price of cameras is low, but their quality is strongly influenced by environment conditions, especially from light (Kocic *et al.,* 2018).

Due to the high information content, low price and operating power, cameras are the primary sources of many driving assistance system's (DAS) algorithm. In simple applications, cameras or in other words visual sensors become the eyes of autonomous vehicles that capture lane (or road) scenes in front of the vehicle. In this work we used only one camera as incoming data source.

Autonomous navigation is strongly influenced by computer vision, artificial intelligence, and intelligent control technology. Obviously, the key concept of a lane line tracking algorithm is the lane boundary detection. To increase the accuracy of lane detection, many researchers try to develop more complicated algorithms (Du and Tan, 2016). However, complicated algorithms require more time to execute and cannot be used in embedded systems. In this paper we propose a lane line tracking algorithm which exploits machine vision techniques including the Hough transformation. The proposed method can run in real-time on a mini computer.

The remaining of this article is structured into 6 sections. In the next section, we describe the conventional workflow of Hough based lane line algorithms and some related works. The 2. section presents the popular “Donkey” RC car, which was our test car. The 3. section is a detailed description about the proposed algorithm. The 4. section presents the experimental conditions and results. Finally, the last section is the conclusion.

2. Related work

Vision-based lane detection techniques can be categorized into three main classes: model, feature, and deep learning based. Model-based approaches require prior knowledge and heavy computation which prevent real-time operation. Deep learning techniques are becoming more popular but they require huge data sets which are not available in many cases (Bojarski *et al.,* 2016). To overcome this problem, simulator software is available for developers and researches (Smolykov *et al.,* 2018). However, the simulated environment does not match the real one in many cases.

Feature-based algorithms rely on basic image characteristic such as edge, texture, and color information (Wu *et al.,* 2019). Those features are used to fit straight lines or curves. In most cases, straight lines are coming from the Hough transform while curve fitting is combined with other lane detection technique or even with Hough. For example, in the work of Cao *et al.* (2019) curve fitting has been performed after a perspective transformation and a sliding window line detection method. They used the Tusimple dataset as data source and they measured approximately 22ms image processing time. However, the effectiveness of perspective mapping decreases if there is an obstacle on the road or vibrations occur during vehicle motion (Yoo *et al.,* 2017).

The general Hough-based lane line identification method transforms the original image into greyscale. Grey image is filtered by a Gaussian filter to remove noise. Thereafter, edges are extracted typically with the Canny edge detection method. The edge image is the input of the traditional Hough transform which gives back polar coordinate system parameters of detected lines.

In last years, many variants of the traditional algorithm have been published. Zheng *et al.* (2018) introduced some constraint to the Hough transformation to exclude short and false line segments. Their algorithm achieved 15.3 fps on a general computer. In embedded systems the processing time would be lower which prevents real-time operation.

Liu *et al.* (2018) utilized K-means clustering to find out the correct lane lines from straight lines coming from Hough transformation. The main drawback of their method is the poor real-time performance. The authors measured only 1.5 fps on a common desktop computer.

Andrade *et al.* (2019) defined an own lane selection process which takes consider the line angle and horizontal distance of the line segment.

Yoo *et al.* (2017) proposed a voting function to vanishing point estimation. From vanishing point candidate lane lines are extracted. Finally, lane lines are selected according to a score function.

In the work of Liu *et al.* (2018) the idea was similar as in the conventional Hough based algorithms but they applied line segment detector instead of Hough transform mainly due to its image processing time.

Although, more possible solutions have been proposed in earlier studies for lane line recognition and tracking, most of them suffer from at least one of the below problems:

- Does not work in real-time on embedded systems.
- Hard or even impossible to realize because the description is not complete.
- The algorithm does not handle common environmental factors.

3. Test vehicle

To keep cost low, we used a popular RC (remote controlled) car as test vehicle. Its name is “Donkey” car (https://www.donkeycar.com/). The controller device of the vehicle is a Raspberry Pi 4 model B with 2 GB memory and quad-core ARM Cortex 1.5GHz central processing unit. It equipped with Raspbian Linux system and supports several serial communications protocols including SPI, UART and I2C. On the vehicle there is a PCA9685 16-channel 12-bit PWM (pulse width modulation) servo driver which communicates with Raspberry Pi via I2C interface. The PCA9685 is responsible for both DC (direct current) and servo motor control. With PWM signal we can set the rpm (revolutions per minute) of DC motor thereby the velocity of the vehicle. By PWM signals we can also adjust the steering angle in the servo motor.

In our vision-based lane tracking approach, 2D images captured from the Pi camera. The camera is mounted on the front of the vehicle and it connects to the Raspberry Pi via a dedicated camera port. Our

Pi camera is an 8-megapixel Module v2 with 1080p30 maximum video resolution. An illustration about the test vehicle can be seen on Figure 1.

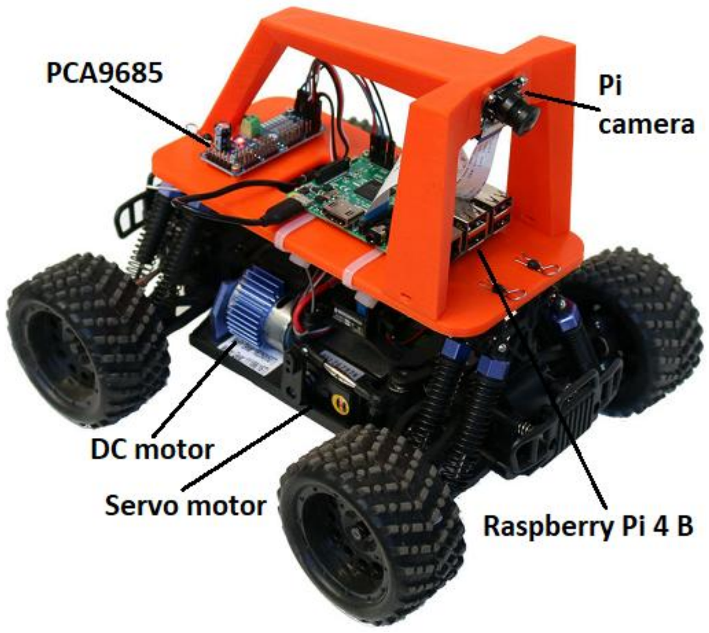

Figure 1. The used test vehicle and its main components

4. Proposed method

The workflow of our algorithm is similar to the standard Hough-based lane detection chain but introduces some significant modification in order to the lane tracking be real-time and reliable. The block diagram of the algorithm can be seen on Figure 2. In the following subsections, the components of the algorithm are presented.

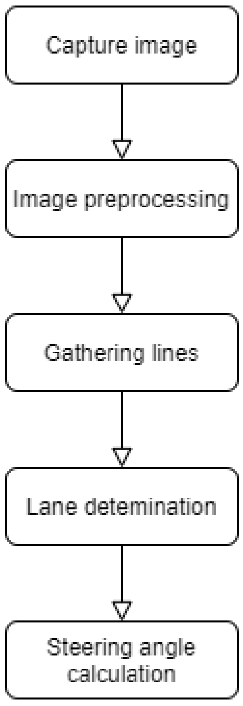

Figure 2. Block diagram of the algorithm

4.1. Image processing

In the work of Lee and Moon (2018), images are scaled down to a low resolution because they claimed that, lane detection performance is almost the same if the resolution is larger than a certain value. We have experienced this as well. The resolution of camera images was 160x120 pixels because lane lines are well visible, even in such a small resolution.

In some earlier works, the captured image is divided into more fields. For example, Deng and Wu (2018) divided the input image into three parts: near field, far field, and sky area. In some other studies, the authors used vanishing point based adaptive ROI (region of interest) determination (Lee and Moon, 2018). In our algorithm the ROI is a fixed region (160x70 pixels) in the near field of the image.

After ROI cutout, the rectangular RGB (red, green, blue) ROI is transformed into the HSV (hue, saturation, value) color space. Unlike most previous methods, we used the H layer of the HSV color space as grey scale image because in this layer yellow and white colors (typical colors of lane lines) have high contrast.

Thereafter, the grey image goes through filtering (blurring) and edge detection. The filter is a 5x5 pixels width Gaussian kernel which remove noise from the image. As edge detector, we used the Sobel operator but only in the horizontal direction. On the output image the contours of lane lines are strong and the calculation process is faster than in the case of more complex edge detector methods (e.g. Canny). Finally, a simple threshold function (1) converts the “edge image” into a binary image. In (1), $b(x,y)$ and $g(x,y)$ indicate the binary and the filtered grey image pixels. In our implementation the threshold value: $T = 100$.

$$b(x,y) = \begin{cases} 0, & f(x,y) < T \\ 1, & f(x,y) \geq T \end{cases}. \tag{1}$$

The threshold function removes all edges weaker than the threshold value. Surviving edges will be the input of “Gathering lines” component of the algorithm. An illustration about the image preprocessing steps on a sample image can be seen on Figure 3. Its (a) part is a sample image. (b) is the ROI which will be cut out of the sample image. (c) illustrates the ROI after greyscale transformation. Finally, (d) is the binary image which has been generated from the greyscale ROI.

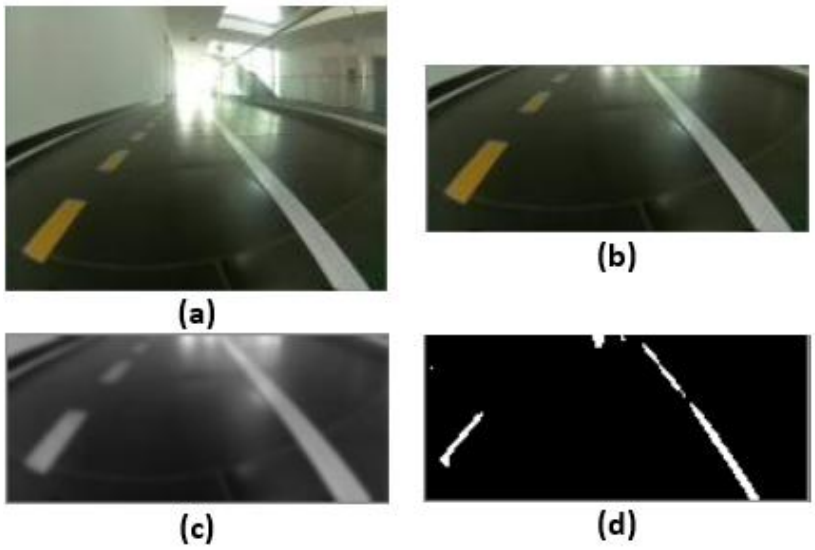

Figure 3. (a) Original image, (b) ROI, (c) filtered grey image, (d) binary image

4.2. Gathering lines

On the binary image, lines are detected by the popular probabilistic Hough transform which is a faster version of the conventional Hough transformation.

For better comprehensibility, at first, we present the used coordinate system where the image is located. The coordinate system can be seen on Figure 4 where the red dashed line is the vertical center line of the image. The vertical center line is defined by $c1 = (x_{c1}, y_{c1})$ and $c2 = (x_{c2}, y_{c2} = 0)$ points.

Under normal circumstances, we can suppose that, the vehicle located between the lanes. It means that one of the lanes there is on the left side while the other one on the right side of the image relative to the vertical center. On Figure 4 the two continuous straight lines illustrate a possible lane.

In the algorithm, lane lines (if they exist) are represented by straight lines which come from the Hough transform. Lines that come from the probabilistic Hough are determined by their two end points: $p^i_1 = (x_1, y_1)$ and $p^i_2 = (x_2, y_2)$ where i is the line index. We defined the location of those two endpoints according to the following condition: $y_1 \geq y_2$. If two points are known, the equation of a line ($y = mx + b$) can be easily calculated with (2) and (3).

$$m_{curr} = \frac{y_2 - y_1}{x_2 - x_1}. \tag{2}$$

$$b_{curr} = y_2 - m * x_2. \tag{3}$$

Now, we know both end points and the equation of all lines that come from the Hough transform.

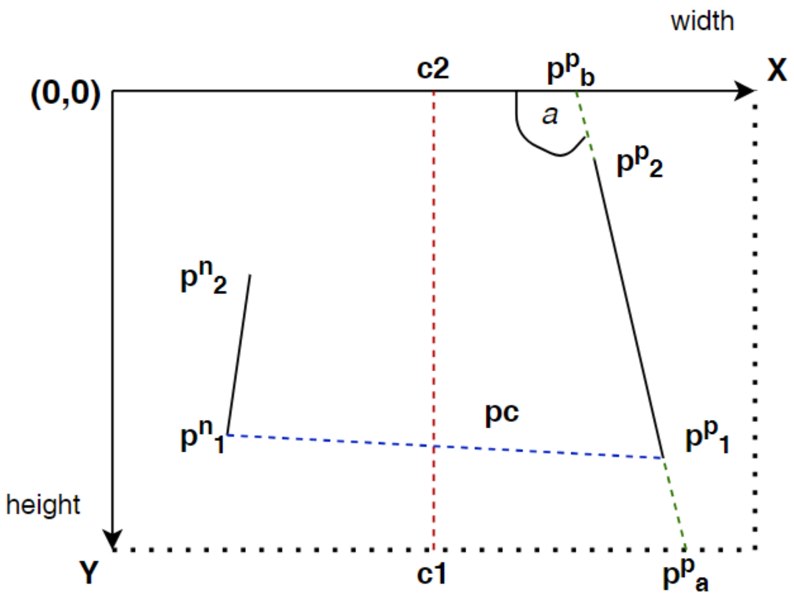

Figure 4. The coordinate system of the image where width and height refer to the image size

Lane lines have limited angle. Therefore, we defined a filter to the α angle between the line segment and the x axis: $10\mathring{} \leq \alpha \leq 80\mathring{}$. α can be calculated from the slope according to the (4) formula.

$$\alpha = \left|\frac{180}{pi} * \arctan(m)\right|. \tag{4}$$

We also supposed that, the p^i_1 point of a lane line should be located on the lower region of the image. Therefore, lane lines have to meet the following condition: $y_1 \geq 0.4 * y_{c1}$. Those lines that do not meet the above defined conditions will be ignored.

Lanes are located in a fan-shaped area centered on the estimated vanishing point (Yoo *et al.,* 2017). According to this property, the algorithm constitutes line pairs from the left and right side of the image where the left-side line has negative slope while the right-side line has positive slope. By this step, false detections can be significantly decreased because false lines will be filtered out.

To sum up, the outcome of this stage is two lists which contain lines with negative slope from the left side of the image and lines with positive slope from the right side of the image.

4.3. Lane detection

In this stage of the algorithm three cases are possible:

- There are both positive and negative lines.
- There is only one type of line (positive or negative).
- There are no any lines.

If both line types are available, the algorithm tries to find a line pair (one positive and one negative line) that assigns the road. The best line pair is chosen according to the (5) criterion where the two constant weights (0.7 and 0.3) were set according to our earlier experimental results.

$$\min\,(0.7 * (dn + dp) + 0.3 * dc). \tag{5}$$

In (5), dc is the Euclidean distance between pc and c. pc is the center point of the line which connects the two p^i_1 points of line pairs (see Figure 4). *dn* and *dp* are also two Euclidean distances between the current negative and positive lines and the earlier, weighted average lane lines (its calculation is detailed in the next section). More specifically, between $p^i_a = (x_a,\ y_a = 0)$ and $p^i_b = (x_b,\ y_b = image\ height)$ points (see Figure 4) of the current lane line and the previous lane line. In p^i_a and p^i_b the y value is given, thus we have to determine x values. Since we know the equation of lines, it can be easily calculated with (6).

$$x_i = \frac{y_i - b_i}{m_i}. \tag{6}$$

If only one line type is available, the algorithm will select that one as a possible lane line which satisfies the (7) criterion where the constant weights are the same as in (5).

$$\min\,(0.7 * dnp + 0.3 * dp1) \tag{7}$$

In (7), $dp1$ is the Euclidean distance between c and p^i_1 while *dnp* is Euclidean distance between the current negative or positive line and the earlier negative or positive lane line (similarly as before).

4.4. Steering angle calculation

In the last stage of the algorithm again three cases are possible:

- There is a line pair – lane lines
- There is only one lane line (positive or negative)
- There are no any lane lines

If a line pair is available (in normal case), the current steering angle (θ_{curr}) calculation is based on the (8) formula. In the formula, the constant weights are also derived from earlier experimental results.

$$\theta_{curr} = 0.8 * \theta_{vp} + 0.2 * \theta_d. \tag{8}$$

Due to the perspective effect of projection from the 3D real environment to a 2D image, parallel lines (lane lines) meet at a single point inside or outside of the image (Moon *et al.,* 2018). This is the intersection point or in other words the vanishing point ($vp = (x_{vp},\ y_{vp})$). *vp* can be determined by the intersection of lines formula (9) where *j* is another line index.

$$x_{vp} = \frac{b_j - b_i}{m_i - m_j}.$$

$$y_{vp} = m_i * x + b_i. \tag{9}$$

In (8), θ_{vp} is an angle between the ($c1$, $c2$) and ($c1$, *vp*) lines (10). An illustration about it can be seen on Figure 5. The sign of θ_{vp} depends on the x coordinates of *vp* and *c2*:

$$\theta_{vp} = \begin{cases} -\frac{180}{pi} * \arccos\left(\frac{ab}{\|a\|\|b\|}\right), x_{vp} > x_{c2} \\ \frac{180}{pi} * \arccos\left(\frac{ab}{\|a\|\|b\|}\right), x_{vp} \le x_{c2} \end{cases}.$$

$$a = (x_{vp} - x_{c1})(y_{vp} - y_{c1}). \tag{10}$$

$$b = (x_{c2} - x_{c1})(y_{c2} - y_{c1}).$$

The algorithm also takes consider the distance between the $c1$ point and the lane lines. In (8), θ_d is calculated with the (11) formula where *dpx* and *dnx* is the horizontal distance (x axis) between $c1$ and p^i_a depending on the line type (see Figure 5).

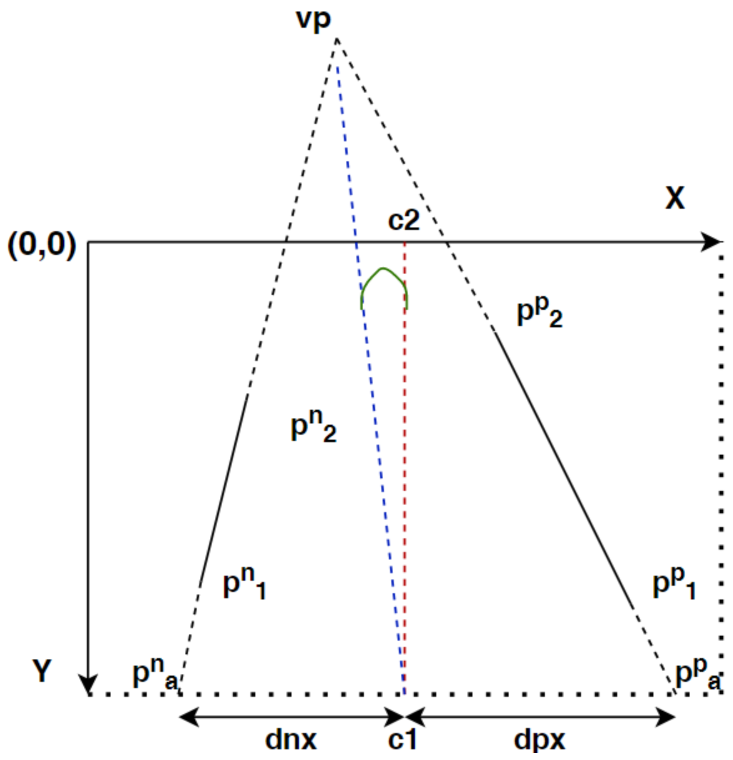

Figure 5. Illustration to angle calculation

$$\theta_d = \begin{cases} -60 * \frac{dpx-dnx}{dpx+dnx}, & dpx > dnx \\ 60 * \frac{dnx-dpx}{dpx+dnx}, & dpx \le dnx \end{cases}. \tag{11}$$

If only one line is available, the current steering angle calculation is based on the (12) formula where θ_{t-1} is the previous steering angle while θ_Δ is based on the difference between the current (dnx_t, dpx_t) and previous dnx_{t-1} or dpx_{t-1} distances (depending on the line type).

$$\theta_t = \theta_{t-1} + \theta_\Delta. \tag{12}$$

During the calculation of θ_Δ the program compares the current and previous distances. If the current distance is higher, then the vehicle should move away from the line, otherwise the vehicle should turn toward the line. Formally, θ_Δ can be calculated with (13) if the line has negative slope or (14) if the line has positive slope. In (13) and (14) d indicates dnx or dpx depending on the line type.

$$\theta_\Delta = \begin{cases} 60 * \frac{d_t - d_{t-1}}{d_{t-1}}, d_t > d_{t-1} \\ -60 * \frac{d_{t-1} - d_t}{d_{t-1}}, d_t \le d_{t-1} \end{cases}. \tag{13}$$

$$\theta_\Delta = \begin{cases} -60 * \frac{d_t - d_{t-1}}{d_{t-1}}, d_t > d_{t-1} \\ 60 * \frac{d_{t-1} - d_t}{d_{t-1}}, d_t \le d_{t-1} \end{cases}. \tag{14}$$

The algorithm continuously keeps track the previous steering angle and lane lines. Therefore, at the end of the last stage, it updates the steering angle and lane lines' parameters (15), (16). In our implementation $w_1 = 0.1$ and $w_2 = 0.05$.

$$\theta_t = w_1 * \theta_{curr} + (1 - w_1) * \theta_{t-1}. \tag{15}$$

$$m_t^i = w_2 * m_{curr}^i + (1 - w_2) * m_{t-1}^i.$$

$$b_t^i = w_2 * b_{curr}^i + (1 - w_2) * b_{t-1}^i. \tag{16}$$

At the end of the algorithm the θ_t angle will control the movement of servo motor (vehicle).

5. Experimental results

The proposed lane line tracking algorithm is implemented on a Raspberry Pi model 4 B mini-computer which is the controller device on our Donkey car (an alternative can be the Nvidia Jetson Nano). The programming environment was Python taking advantage of the OpenCV library.

In many earlier works, researches used videos from databases as information source. Unlike them, our primary goal was to test the algorithm in real environment, on a self-made test track. The test trach has been constructed in the first-floor corridor of our faculty. A picture about the test track can be seen on Figure 6.

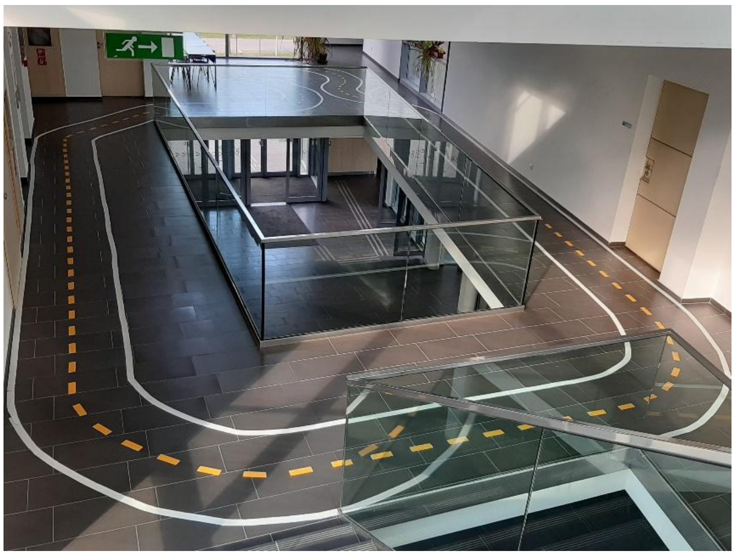

Figure 6. The test track

As can be seen on the above figure, the track is bordered with large windows and walls. It means that the lighting conditions are strongly alternating at different locations of the track. In additions, tiles are reflected and the bright gaps between tiles cause unwanted edges in the camera image.

Instead of presenting several test images about the operation of the algorithm, we shared two videos on the popular YouTube video sharing portal. On the first one, the movement of the vehicle is visible from the external environment. Figure 7 shows a sample image from the video. The full video is available on the following link: https://www.youtube.com/watch?v=H8w6vUymk_s

The second video includes the original camera image and another image generated by the algorithm. On the image, the green line is the heading direction, red line(s) are the current lane line(s) while blue lines are the previous lane lines. Figure 8 presents a sample image from the video. The video is available on: https://www.youtube.com/watch?v=t1Wi3suHSzs.

During the tests, the car moved with low velocity. The image processing speed was between 25-30 fps (frame per second).

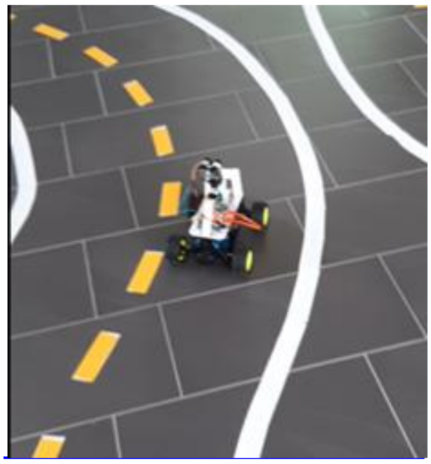

Figure 7. Sample image from the first video

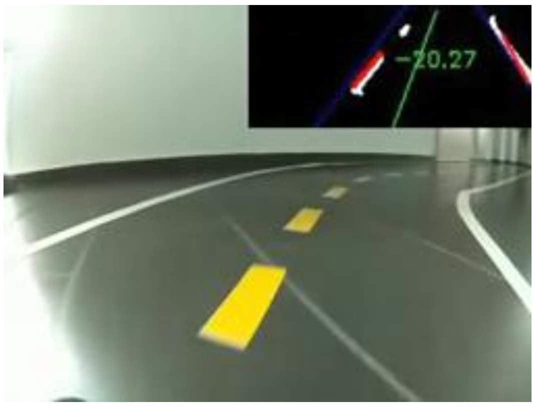

Figure 8. Sample image from the second video

6. Conclusions

According to the experimental results we can conclude that, the proposed lane tracking algorithm is working well on a self-made test track. It means that, the vehicle appropriately follows lane lines. Our test track includes several external factors that can be found in outdoor environment (shadows, illumination changes, reflection, different bends). Therefore, we assume that, it is a good test environment. Despite environmental factors, experimental results show that the proposed algorithm provides accurate lane maneuvering and fulfills the real-time operation requirement on a mini-computer.

Although, the algorithm is protected to a certain degree against poor road conditions (e.g. eroded lane marks) and light reflections, but cannot handle every possible case (e.g. missing lanes). In order to test the robustness of the proposed algorithm, we have to test it outdoor under different weather conditions (probably on another vehicle).

To sum up, the proposed lane tracking algorithm is well usable in real-time indoors where the vehicle must follow a predefined path.

Acknowledgements

This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

References

1. Alonso, E., Arpon, C., Gonzalez, M., Fernandez, R.A., Nieto, M. (2020) Economic impact of autonomous vehicles in Spain. *European Transport Research Review,* 12, 59.
2. Andrade, D.C., Bueno, F., Franco, F.R., Silva, A., Neme, J.H.Z., Margraf, E., Omoto, W.T., Farinelli, F.A., Tusset, A.M., Okida, S., Santos, M.M.D., Ventura, A., Carvalho, S., Amaral, R.S. (2019) A novel strategy for road lane detection and tracking based on a vehicle's forward monocular camera. *IEEE Transactions on Intelligent Transportation Systems,* 20(4), 1497-1507.
3. Bojarski, M., Testa, D.D., Dworakowsi, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K. (2016) *End to end learning for self-driving cars*. arXiv:1604.07316.
4. Cao, J., Song, C., Song, S., Xiao, F., Peng, S. (2019) Lane detection algorithm for intelligent vehicles in complex road conditions and dynamic environments. *Sensors,* 19(14), 3166.
5. Deng G, Wu, Y. (2018) Double lane line edge detection method based on constraint condition Hough transform. In: *17'th International Symposium on Distributed Computing and Applications for Business Engineering and Science*. Wuxi, 19 Oct., 107-110.
6. Du, X., Tan, K.K. (2016) Vision-based approach towards lane line detection and vehicle localization. *Machine Vision and Applications* 27 (2), 175-191.
7. Hu, J., Xiong, S., Zha, J., Fu, C. (2020) Lane detection and trajectory tracking control of autonomous vehicle based on model predictive control. *International Journal of Autonomous Technology,* 21(2), 285-295.
8. Hussain, R., Zeadally, S. (2019) Autonomous cars: research results, issues, and future challenges. *IEEE Communications Survey & Tutorial,* 21(2), 1275-1313.

9. Kocic, J., Jovicic, N., Drndarevic, V. (2018) Sensors and sensor fusion in autonomous vehicles. In: *Proceedings of the 26'th Telecommunications Forum*, Serbia, 20-21 November, 420-425.
10. Lee, C., Moon, J.H. (2018) Robust lane detection and tracking for real-time applications. *IEEE Transactions on Intelligent Transportation Systems,* 19(12), 4043-4048.
11. Liu, J., Lou, L., Huang, D., Zheng, Y., Xia, W. (2018) Lane detection based on straignt line model and K-means clustering. In: *7'th Data Driven Control and Learning Systems Conference*, Enshi, 25-27 May, 527-532.
12. Liu, S., Lu, L., Zhong, X., Zeng, J. (2018) Effective road lane detection and tracking method using line segment detector. In: *Proceedings of the 37th Chines Control Conference*. Wuhan, 25-27 July, 5222-5227.
13. Moon, Y.Y., Geem, Z.V., Han, G.T. (2018) Vanishing point detection for self-driving car using harmony search algorithm. *Swarm and Evolutionary Computation,* 41(8), 111-119.
14. Smolykov, M.V., Florov, A.I., Volkov, V.N., Stelmashchuk, I.V. (2018) Self-driving car steering angle prediction based on deep neural network. An example of CarND udacity simulator. In: *IEEE 12th International Conference on Application of Information and Communication Technologies*, Almaty, 17-19 Oct. 2018, 1-5.
15. Wu, C.B., Wang, L.H., Wang, K.C. (2019) Ultra-low complexity block-based lane detection and departure warning system. *IEEE Transactions on Circuits and Systems for Video Technology,* 29(2), 582-593.
16. Yoo, J.H., Lee, S.W., Park, S.K., Kim, D.H. (2017) A robust lane detection method based on vanishing point estimation using the relevance of line segments. *IEEE Transactions on Intelligent trasportation Systems,* 18(2), 3254-3266.
17. Zhao, J., Liang, B., Chen, Q. (2018) The key technology towad the self-driving car. *International Journal of Intelligent Unmanned Systems,* 6(1), 2-20.
18. Zheng F., Luo, S., Song, K., Yan, C.W., Wang, M.C. (2018) Improved lane line detection algorithm based on Hough transform. *Pattern Recognition and Image Analysis,* 28(2), 254-260.