Jurnal Polimesin
Department of Mechanical Engineering
State Polytechnic of Lhokseumawe
http://e-jurnal.pnl.ac.id/polimesin
e-ISSN : 2549-1999 No. : 4 Month : August
p-ISSN : 1693-5462 Volume : 23 Year : 2025

Processing dates: received on 2025-03-30, reviewed on 2025-06-18, accepted on 2025-07-06 and online availability on 2025-08-31

Development of a self-driving RC car with a lane-keeping system using a pure pursuit controller

Aulia Rahman[1]*, M. Jurej Alhamdi[1], Kahlil Muchtar[1], Yudha Nurdin[1], Roslidar Roslidar[1], Safrizal Razali[1], Riki Effendi[2]
[1]Department of Electrical Engineering and Computer Engineering, University of Syiah Kuala, Banda Aceh 23111, Indonesia
[2]Department of Mechanical Engineering, University of Muhammadiyah Jakarta, Jakarta 10510, Indonesia
*Corresponding Author: aurahmn@usk.ac.id

Abstract

The development of autonomous vehicles is crucial for enhancing driving safety, comfort, and efficiency. This research presents the design of a self-driving Remote Controlled (RC) car at a 1:10 scale, equipped with a lane-keeping system and a pure pursuit controller. The primary objective is to evaluate the effectiveness of integrating computer vision techniques with trajectory tracking control to maintain lane stability. Lane detection was achieved using a sliding windows algorithm, while polynomial fitting estimated the lane centerline. A stereo camera provided spatial perception, capturing images that were processed to determine the steering angle needed to minimize deviation between the lookahead point and the viewpoint of the vehicle. Experimental results show that the system-maintained lane position with minimal deviation, achieving an average steering angle of 90.44° on straight paths, 65.4° on right turns, and 113.1° on left turns. These results demonstrate the feasibility of combining vision-based lane detection with a pure pursuit controller to improve path-tracking accuracy and stability in autonomous vehicles.

Keywords:
Self-driving RC car, pure pursuit controller, lane-keeping system

1 Introduction

The development of autonomous vehicle technology aims to increase driving safety, driving comfort, and its economy, as well as reduce the traffic accident rate [1]. An autonomous mobile robot must perform many complex information processing tasks in real time. Brooks [2] proposes a robust layered control system for a mobile robot as opposed to building a control system into a series of functional units. Many modern mobile robots implement this architecture to address challenges, such as the Defense Advanced Research Projects Agency (DARPA) Urban Challenge [3], where intelligent autonomous vehicles were able not only to travel significant distances in off-road terrain but also to operate in urban scenarios.

Although autonomous vehicles promise a safe, comfortable, and efficient driving experience, the are still unsolved problems and challenges to be addressed. From high-level system architecture, security, communications, control, as well as core function algorithms, including localization, mapping, perception, planning, and human-machine interfaces, need to be improved and considered over time [4]. Baden et al. [5] discussed the typical path tracking methods for autonomous vehicles, such as motion planning and control techniques, including pure pursuit control, rear/front wheel-based feedback, feedback linearization, control Lyapunov design, and linear/nonlinear Model Predictive Control (MPC).

The rapid progress toward developing new perception, control, and motion planning techniques has driven the advancement of autonomous vehicle development. One key algorithm essential for autonomous vehicles is path tracking, which aims for the vehicle to follow a reference path accurately while ensuring vehicle stability and satisfying control performance. This path tracking method can be divided into model-based and geometric approaches. Many model-based path tracking methods rely on the kinematic or dynamic characteristics of the vehicle to predict the error between the vehicle's actual motion and the path. Popular methods include the Linear Quadratic Regulator (LQR) as in [6], [7], [8], [9], [10], and [11], and MPC as in [12], [13], and [14]. Despite their good accuracy, these methods are sensitive to the accuracy of localization, the curvature of the path or trajectory, and parameter changes. They also require substantial computational resources and longer calculation times to find feasible solutions [15]. Although sensitive to parameter variations, the ability of MPC to predict future system states and handle constraints explicitly makes it a promising technique to address these challenges [16], [17], [18], [19]. In addition to path tracking, MPC is also used for controlling speed [20] and yaw stability [21].

A comprehensive review of control strategies for path tracking in autonomous vehicles is discussed in [22], [23], and [24], especially focusing on lateral control as seen in [25], [26], [27], and [28]. It summarizes the implementations and disadvantages of various control techniques, while also highlighting the challenges and unresolved problems in path tracking. The pursuit of efficient path tracking control using preview path tracking with experimental validation is examined in [29]. The integration of path tracking control with vehicle stability using MPC and Proportional Integral Derivative (PID), discussed in [30] and [31], as well as MPC and LQR [32], is also covered. Recent advancements in artificial intelligence and deep learning algorithms are now driving the implementation of lateral control in autonomous vehicles, particularly in challenging driving scenarios. They utilize images as input to the vehicle and produce the lateral commands, as demonstrated in [33], [34]. The trend in artificial intelligence techniques used in autonomous vehicles can be seen in [35], which proposes adaptive path tracking with a back propagation neural network to control the coefficient of the pure pursuit algorithm. Semantic segmentation is also employed to plan paths for small autonomous vehicles discussed in [36]. Other recent path tracking methods include adaptive path tracking [37], optimal control [38], and neuro-adaptive approaches [39]. The application of this path tracking control in the industrial sector, such as in warehouse robots delivering goods between stations, is explored in [40]. The implementation of path tracking control in autonomous racing cars is discussed in [41], [8], with an emphasis on its importance in autonomous racing [42]. Horri et al. [43] used waypoint navigation combined with lane-keeping assistance to operate an autonomous vehicle in a city environment. Additionally, MPC parking is examined in [44].

Geometric-based path tracking control uses the geometric relationship between the vehicle and the reference path, making it less affected by path smoothness and localization errors. The two main geometric-based algorithms are the pure pursuit [45] and the Stanley [46]. This method has proven itself to be highly capable and robust for path tracking. It has been implemented a vehicle called Stanley which won the DARPA Grand Challenge as well as high-speed autonomous racing using an F1/10 platform [47], [48]. The algorithm requires the odometry information of the car and a global racing line and works by generating a relative heading to a waypoint within the race line, which is a distance l_d away from the car on the race line. Instead of using odometry, this research utilized a computer vision technique and a stereo camera to establish the look-ahead distance.

2 Research methods

2.1 Remote-controlled (RC) car model

Most RC car models are scaled-down versions of their real-life counterparts, with the scale expressed as a ratio of the real size of the car represented by the model and the size of the model, which is normally written as 1:n or 1/n where n = 5, 8, 10, 16, 18, 24. The smaller the RC car, the larger the n in that ratio.

The hardware platform used in this study is a 1/10 th scale RC car since it was widely used as a research platform for autonomous cars [48]. This size was preferred because it can accommodate various autonomous control inputs, sensors, electronics, and a small-sized computing module, which enables autonomous decision making [47]. Before integrating all components of the RC car, the researcher sketches the layout to consider the maximum payload and center of gravity of the car in order to ensure the car runs properly. The sketch of the car designed is shown in Fig. 1 is a front and side view of the car. The perspective view of the 3D model of the car is shown in Fig. 2.

Fig. 1. Sketch of the autonomous vehicle used in this research

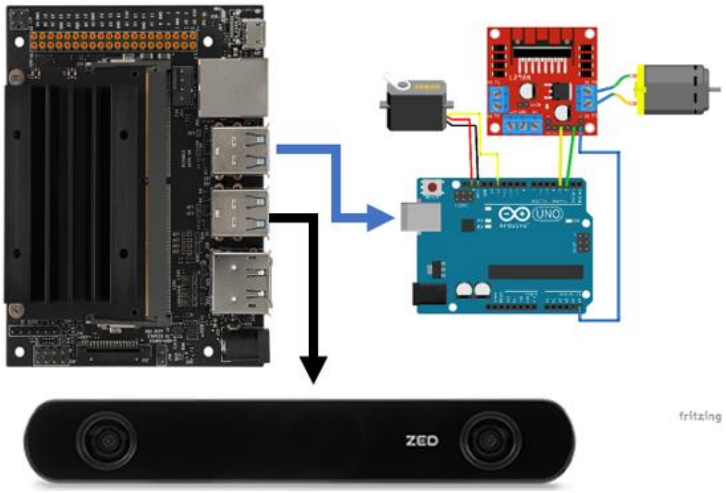


Fig. 2. Schematic of the electronic components of the RC car

The main components of the RC car can be categorized as sensors, actuators, and embedded systems. The main components are shown in Fig. 3. For the sensor of the car to gather the images of the track researcher used only a stereo camera. The collected images are passed to the control algorithm to detect the lines and compute the steering angles. The camera used is a stereo camera called the ZED 2 camera from Stereo Labs. It is 175 mm wide and 30 mm tall. The video output supports 720p video mode with 60/30/15 frames per second with a resolution of 2560x720 pixels. The researcher connects this camera using a USB 3.0 port to the Jetson Nano. The schematic of all the main electronic components of the car is shown in Fig. 2.

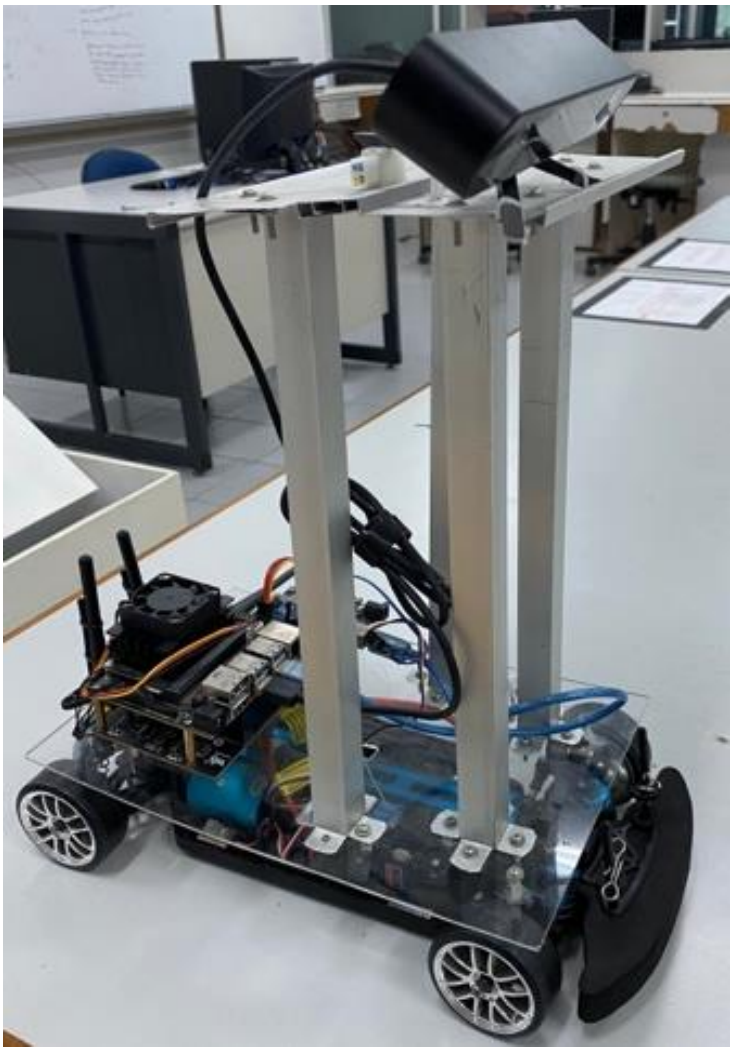


Fig. 3. The final assembly of the RC car

The electric motor is one of the most important parts of the car. In this research, the researcher used a Commercial Off The Shelf (COTS) component from the original RC car, a 3300KV brushless motor. It is equipped with a Hobbywing 45 Amp fan-cooled brushless electronic speed controller. Researchers modify only the power supply of the car, which is a four Panasonic NCR 18650 Li-ion battery, 3400 mAh, 3.7V, 30A, with a flat top. One servo motor is used to control the steering wheels of the car, which also comes with the car.

Jetson Nano is an embedded system that run the operating system that runs the necessary software, such as Open CV, and serves as a platform to run Python code. The Jetson Nano is a quad-core ARM CPU (1.43 GHz), 128-core dedicated GPU, and 4 GB of shared LPDDR4 RAM. It is connected to Arduino Uno, which sends commands to the motor and servo. The complete autonomous vehicle is shown in Fig. 3.

2.2 Road course

The arena used to test the RC car consists of a path marked with two white tapes, so it is like a road. The outer edges are approximately 300 cm long and 200 cm wide, which makes the arena look like an oval shape. The distance between the outer edge and the inner edge is 40 cm. The path or the road forms a closed loop which consists of straight and curved segments. The material for the base of the road is made up of a light textile carpet. The overall road course is shown in Fig. 4. Since the path is a closed loop that only allows turning in one direction, it is necessary to manually change the car's heading to enable it to turn in the opposite direction.

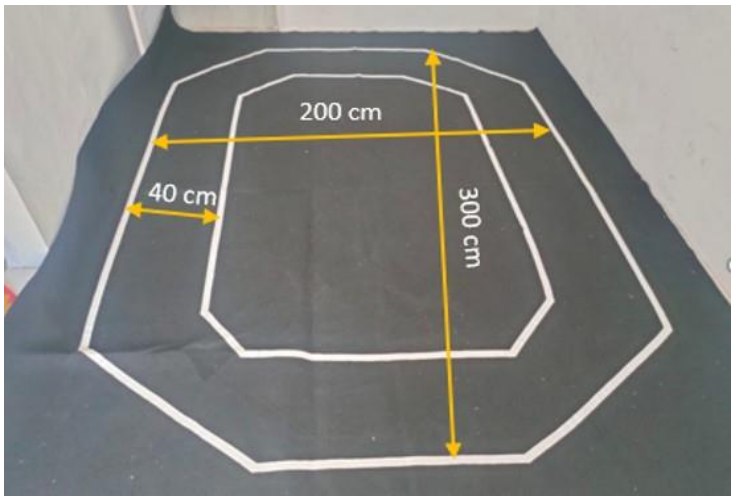


Fig. 4. The road course is utilized to test the RC car

2.3 Lane-keeping system

The lane detection system employs a stereo camera called ZED 2 from Stereo Labs to capture track images with a resolution of 672 × 376 pixels. The image will be examined by the image processing pipeline to extract the lane and then sent to the pure pursuit controller, which will control the steering angle of the front wheel to follow the given path. The image processing procedures for extracting the lanes are illustrated in Fig. 5.

Fig. 5. Image processing steps to detect the lanes of the track

The lane detection system pipeline uses six steps for the car to detect the lines, which are discussed in the following:

2.3.1 Bird's eye view

The first step to detect this line is to modify the perspective image or transform the perspective of the car's viewpoint from forward to upward. So that the detected line projection can be seen clearly and always in focus. The red dot in the illustration represents the original image's origin, and the yellow dot represents the reference point to which the perspective will be shifted.

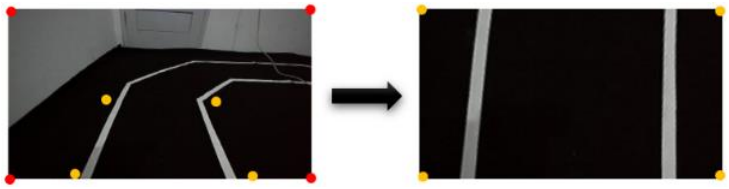
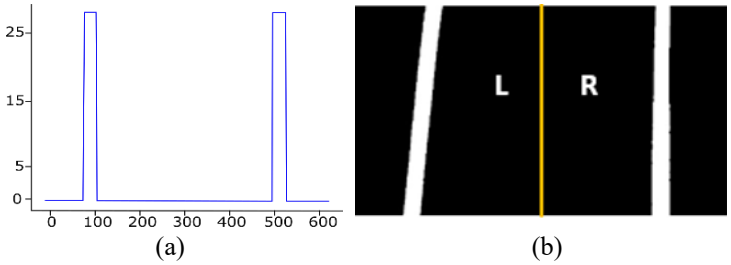


Fig. 6. Original image perspective view and bird-eye view transformation

2.3.2 Binary image

The second step in this image processing is to modify or delete color components by separating the track with lines and applying the image threshold range to the image. In this study, the image threshold was set to 60 - 255. If the value is between 60 and 255, the pixel value in the image will be set to 255 (white); otherwise, it will be 0 (black). The binary image of the track with its histogram can be seen in Fig. 7.



(a) (b)

Fig. 7. (a) The binary image of the track, and (b) the histogram of the binary image of the track

2.3.3 Histogram

The image's histogram will be calculated next. The image histogram is computed by totaling the pixel values in each image column. Before computing the histogram, the image will be separated into two equal parts, the right and left sides denoted by L and R. By taking the maximum value on the right and left image histograms, this histogram will be utilized to establish the starting point of the sliding windows that will identify line pixels. The red circle in the figure represents the greatest value for each segment of the histogram, which will be used as the starting point value for the sliding windows to detect line pixels. The image's maximum point value is located at 94.538.

2.3.4 Sliding windows

The sliding windows algorithm is used next in image processing to find path lines. This method operates similarly to convolution. The pixel position will be preserved if there is a pixel in the window with a value of 255, which is the outcome of the binary image process. This will then serve as a reference point for visualising the line and calculating the path's centre value, which will be used for car steering control with the pure pursuit algorithm. In this study, ten sliding windows were used for each line. The dimensions of the sliding windows utilised are indicated below.

2.3.5 Polynomial fitting

The following stage in image processing to detect path lines is to use polynomial fits. The sliding window pixel position values will be inserted into the equation $y = ax2 + bx + c$, with the coefficients a, b, and c values derived from the polynomial fitting to the power of 2. This polynomial fitting involves a Python function called numpy.polyfit (x, y, 2). Where x is the value of the x pixel position of the sliding window detected line, and y is the value of the y pixel position of the sliding window detected line. The sample value of the detected line pixels is shown in Fig. 8.

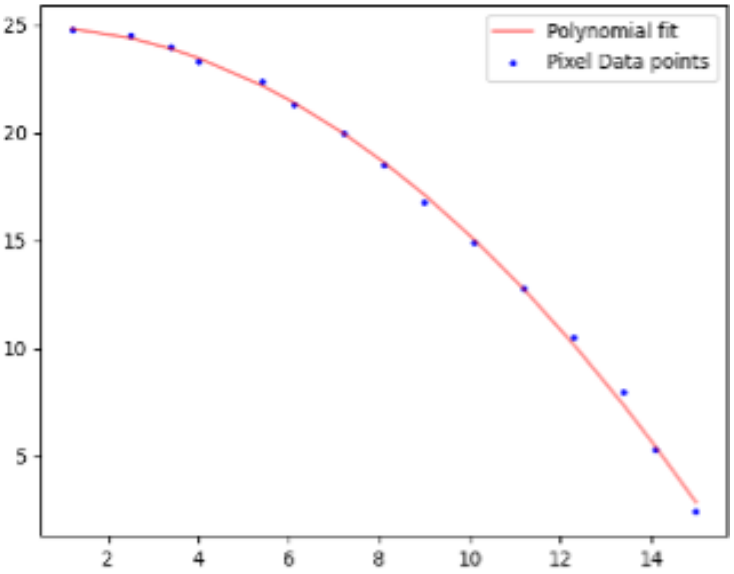


Fig. 8. Plot of polynomial fitting

2.3.6 Lanes visualization

Drawing the line visualisation procedure is the final step of line detection. This line visualisation procedure makes use of the OpenCV library's cv2.polylines function. The point visualised by this function is the polynomial fitting result point.

The last step in detecting path lines is using the sliding window algorithm. This algorithm works the same as convolution: if there is a pixel line or a pixel with a value of 255, which is the line value resulting from the binary image process contained in the window, then the pixel position will be stored and become a reference point for visualising the line. The output was the midline value of the path that will be used for car steering control with the pure pursuit algorithm. The number of sliding windows used in this study is 10 for each line, with a size of 37 pixels in height and 70 pixels in width. The lane visualized in a red line is shown in Fig. 9.

Fig. 9.Line visualization

2.4 Pure pursuit controller

The pure pursuit algorithm was originally developed as a method for calculating the arc necessary to get a robot back onto a path [45]. The algorithm is a geometric lateral control method that can be easily implemented in several applications, including autonomous robots. Given a car or robot with a reference path, the pure pursuit controller algorithm calculates and determines a forward viewpoint or reference point on the reference path with the forward sight distance and the vehicle's current actual position. Then it calculated the vehicle's steering angle, which is used as the actual steering angle value for the car to maneuver on the road. The controller is tasked with pursuing a point on the reference trajectory that lies in front of the vehicle or the lookahead point.

Fig. 10 shows that 2 important points in this algorithm, namely the vehicle position and the look-ahead point. The coordinates with the point (x, y) are the coordinates of the RC car, and the coordinates with (xlook ahead, ylook ahead) are the coordinates of the point of view ahead, or in other words, the destination point. The distance between the car and the reference point is denoted as l_d, while α represents the angle between the car's heading and the radius of curvature at the reference point. The goal of the pure pursuit controller is to get a steering angle that can make the car maneuver precisely to the desired reference point.

The distance of this reference point or look-ahead point from the car, denoted as l_d, is a constant distance with l_d = 600 mm. It was measured from the back wheel of the car to the look-ahead point.

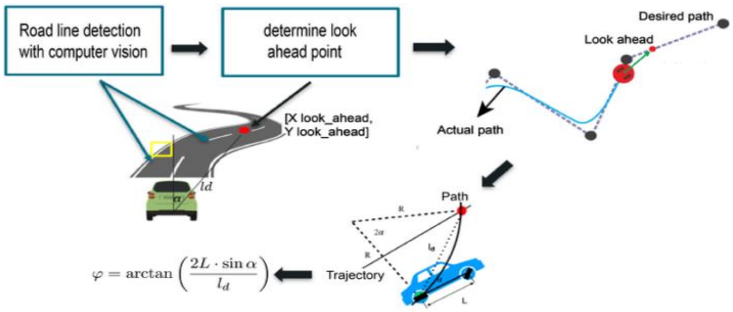


Fig. 10. Schematic of pure pursuit controller algorithm

The difference between the look-ahead point and the car viewpoint is defined as error, which is used to calculate the steering angle of the car. Using the notation from Fig. 11, the error is calculated using Eq. (1).

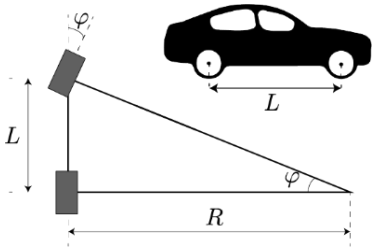


Fig. 11 Ackerman bicycle model

$$\alpha = \arctan\left(\left|\frac{x_2 - x_m}{y_2 - y_1}\right|\right) \quad (1)$$

The definition of the car viewpoint, look-ahead point, and the distance of the car to the look-ahead point l_d. It can also be seen that the difference between the car viewpoint and look-ahead point, denoted by error, which needs to be minimized by the pure pursuit controller, as shown in Fig. 12. By utilizing the Ackerman bicycle model, the car's steering angle can be calculated using Eqs. (2) to (5).

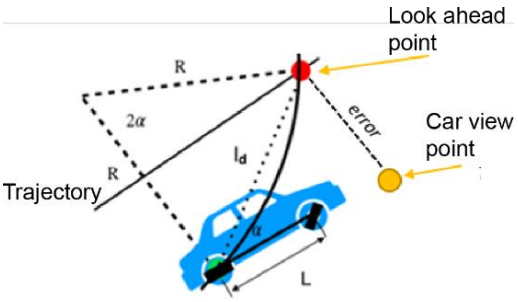


Fig. 12. The car viewpoint, the look-ahead point, and the distance of the car to the look-ahead point (l_d)

$$\frac{l_d}{\sin(2\alpha)} = \frac{R}{\sin\left(\frac{\pi}{2} - \alpha\right)} \quad (2)$$

$$\frac{l_d}{2\ \sin(\alpha)\cos(\alpha)} = \frac{R}{\cos(\alpha)} \quad (3)$$

$$\frac{l_d}{\sin(\alpha)} = 2R \quad (4)$$

$$\kappa = \frac{2\sin(\alpha)}{l_d} \quad (5)$$

Another important component of the pure pursuit algorithm is the error angle denoted by α. It is crucial in determining the steering direction of the vehicle to follow a predefined path. This angle is formed by the positional difference between the vehicle's current viewpoint and the look-ahead point, which serves as the reference for the next movement direction. Geometrically, α is the angle between the straight line connecting the vehicle's position to the look-ahead point and the vehicle's heading. This angle is calculated using basic trigonometric principles, specifically the Pythagorean theorem, to determine the distance and angle based on the vehicle's coordinates and the look-ahead point's coordinates. The illustration of the error angle calculation is shown in Fig. 13.

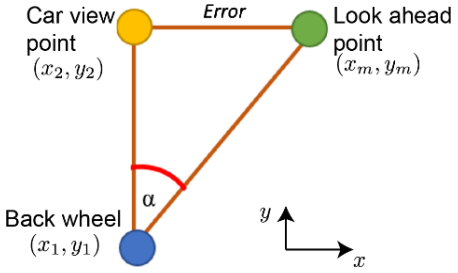


Fig. 13. The calculation of error

where κ *is* defined as the curvature of the circular arc. Applying two degrees of freedom of the Ackerman model, the steering angle is calculated using Eq. (6).

$$\delta = \arctan(\kappa L) \quad (6)$$

Where L is the wheelbase. Using Eqs. (6) and (5), the desired steering angle can be calculated by Eq. (7).

$$\delta l_d(t) = \arctan\left(\frac{2L\,sin(\alpha(t))}{l_d}\right) \quad (7)$$

3 Results and discussion

The researcher tested the car with a very simple track that is made to test the lane-keeping behaviour, and the RC car successfully navigated the track with minimal errors. The testing track is shown in Fig. 14.

Fig. 14. The RC car on the real testing track with the lane-keeping system

Fig. 15 shows the histogram of detected lines when the car turns right. This histogram is used as the start point for the sliding windows algorithm. The processes for turning left.

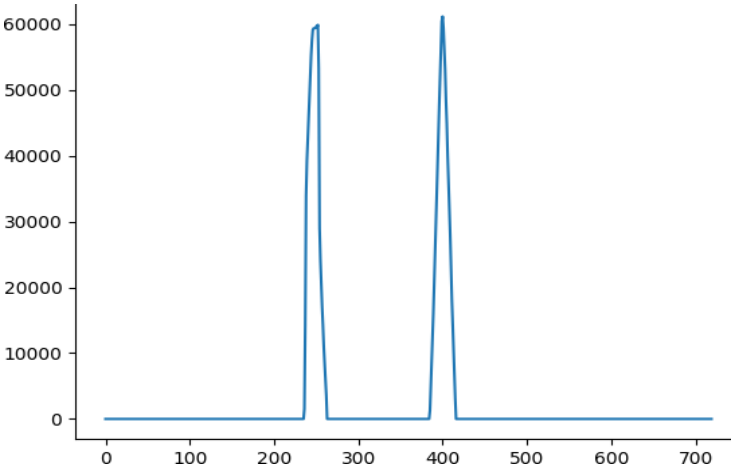


Fig. 15. Histogram result for the turning right scenario

The result of the sliding windows is used to estimate the curve of the turning right path. It consists of a stack of nine sliding windows of 37px by 70px, which detect the path segment location. The result of determining the center point of the path (look-ahead point). The first part of the pure pursuit control process is to determine the center point of the path or look ahead point, which is used as a reference point for the car to follow the path and in the calculation of the car's steering degree or value. This point is calculated by utilizing the information from the line detection results from the polynomial fitting process. Polynomial fitting produces many line pixel coordinates (x, y) taken at one point each on the left line and right line with the y value being the midpoint of all detected line pixels on the y axis (pts left y, pts right y) and the x value being the midpoint of all detected line pixels on the x axis on the left line and right line (pts right, pts left) which used sliding windows method as show in Fig. 16.

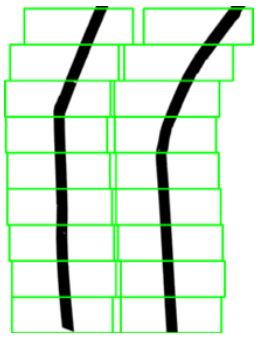


Fig. 16. The sliding windows of detecting the turning right scenario

Fig. 17 shows the detected line and the center point or a look-ahead point as the result.

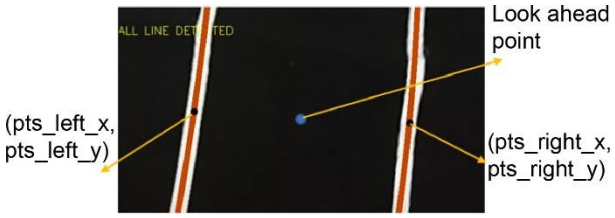


Fig. 17. The detected lines and their center point

Fig. 18 shows the final complete estimation of the path illustrated with the green line, the computation of the middle of the path, and the look-ahead point of the car. This is used to compute the steering angle to drive the car following the middle path.

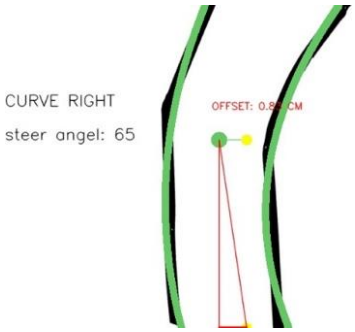


Fig. 18. The steering angle computation is performed after the estimation of the curve path

The summary of all turning scenarios of the car is shown in Fig. 19. As we can see, the estimated curves follow the path precisely. It shows the estimated curve of the paths in a green line for turning right or left, as well as for the straight path. The graph also shows the steering angle and the error between the car viewpoint and lookahead point.

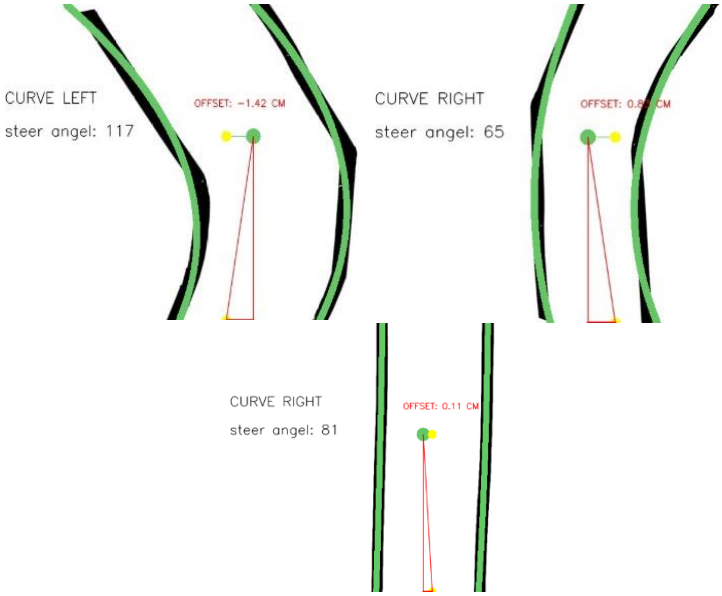


Fig. 19. The visual of steering angle calculation shows the car turning left, turning right, and moving straight

From Table 1, it can be seen that on a straight path, the average steering angle is 90.44º, on the right turn lane, the average steering angle value is 65.4º, and on the left turn lane, the average steering angle value is 113.1º. Our RC car was able to track the path correctly and keep the car at the center of the path.

Table 1. Error angle and parameters of the pure pursuit controller component

Error angle	l_d (m)	L (m)	Turn	Steering angle
0.68	0.6	0.4	R	90
37.45	0.6	0.4	L	123.03
35.43	0.6	0.4	R	58.29
30.87	0.6	0.4	R	54.89
25.74	0.6	0.4	R	75.92
32	0.6	0.4	L	109.24
45.15	0.6	0.4	L	127.38
24.34	0.6	0.4	L	102.79

4 Conclusions

This research demonstrated the development of a self-driving RC car with a lane-keeping system using a pure pursuit controller integrated with computer vision. The system utilized off-the-shelf and modified components to construct a 1:10 scale prototype capable of detecting lane boundaries and maintaining stable trajectory tracking. The sliding windows algorithm effectively identified lane markers, while polynomial fitting provided curve estimation of the track. These features enabled the pure pursuit controller to compute steering angles that kept the vehicle centered. Experimental evaluation confirmed that the system achieved consistent lane-keeping performance, with average steering angles of 90.44° on straight segments, 65.4° on right turns, and 113.1° on left turns. These results highlight the robustness of the approach in handling varying road geometries. The integration of computer vision and pursuit-based control offers a practical solution for improving path-following accuracy. Future work will focus on refining control algorithms and testing the system in more complex driving scenarios.

References

[1] I. Kotseruba and J. K. Tsotsos, "Attention for Vision Based Assistive and Automated Driving: A Review of Algorithms and Datasets," IEEE Transactions on Intelligent Transportation Systems, vol. 23, pp. 19907– 19928, Nov. 2022.

[2] R. Brooks, "A robust layered control system for a mobile robot," IEEE Journal on Robotics and Automation, vol. 2, no. 1, pp. 14–23, 1986.

[3] M. Buehler, K. Iagnemma, and S. Singh, The DARPA Urban Challenge: Autonomous Vehicles in City Traffic. Springer Tracts in Advanced Robotics, Springer Berlin Heidelberg, 2009.

[4] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," IEEE access, vol. 8, pp. 58443–58469, 2020.

[5] B. Paden, M. Cˇ a´p, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self driving urban vehicles," IEEE Transactions on Intelligent Vehicles, vol. 1, no. 1, pp. 33–55, 2016.

[6] Z. Wang, K. Sun, S. Ma, L. Sun, W. Gao, and Z. Dong, "Improved linear quadratic regulator lateral path tracking approach based on a real time updated algorithm with fuzzy control and cosine similarity for autonomous vehicles," Electronics, vol. 11, no. 22, p. 3703, 2022.

[7] E. Alcala, V. Puig, J. Quevedo, T. Escobet, and R. Comasolivas, "Autonomous vehicle control using a kinematic lyapunov based technique with LQR LMI tuning," Control Engineering Practice, vol. 73, pp. 1–12, 2018.

[8] X. Fan, J. Wang, H. Wang, L. Yang, and C. Xia, "LQR trajectory tracking control of unmanned wheeled tractor based on improved quantum genetic algorithm," Machines, vol. 11, no. 1, 2023.

[9] C. Hu, R. Wang, F. Yan, and N. Chen, "Should the desired heading in path following of autonomous vehicles be the tangent direction of the desired path?," IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 6, pp. 3084–3094, 2015.

[10] J. Chen, W. Zhan, and M. Tomizuka, "Autonomous driving motion planning with constrained iterative LQR," IEEE Transactions on Intelligent Vehicles, vol. 4, no. 2, pp. 244–254, 2019.

[11] K. Lee, S. Jeon, H. Kim, and D. Kum, "Optimal Path Tracking Control of Autonomous Vehicle: Adaptive Full State Linear Quadratic Gaussian (LQG) Control," IEEE Access, vol. 7, pp. 109120–109133, 2019.

[12] P. Stano, U. Montanaro, D. Tavernini, M. Tufo, G. Fiengo, L. Novella, and A. Sorniotti, "Model predictive path tracking control for automated road vehicles: A review," Annual Reviews in Control, 2022.

[13] S. Cheng, L. Li, X. Chen, J. Wu, and H. d. Wang, "Model predictive control based path tracking controller of autonomous vehicle considering parametric uncertainties and velocity varying," IEEE Transactions on Industrial Electronics, vol. 68, no. 9, pp. 8698–8707, 2021.

[14] C. E. Beal and J. C. Gerdes, "Model Predictive Control for vehicle stabilization at the limits of handling," IEEE Transactions on Control Systems Technology, vol. 21, no. 4, pp. 1258–1269, 2013.

[15] N. Guo, X. Zhang, Y. Zou, B. Lenzo, and T. Zhang, "A computationally efficient path following control strategy of autonomous electric vehicles with yaw motion stabilization," IEEE Transactions on Transportation Electrification, vol. 6, no. 2, pp. 728–739, 2020.

[16] K. Berntorp, R. Quirynen, T. Uno, and S. D. Cairano, "Trajectory tracking for autonomous vehicles on varying road surfaces by friction adaptive nonlinear Model Predictive Control," Vehicle System Dynamics, vol. 58, no. 5, pp. 705–725, 2020.

[17] P. Falcone, H. Eric Tseng, F. Borrelli, J. Asgari, and D. Hrovat, "MPC based yaw and lateral stabilisation via active front steering and braking," Vehicle System Dynamics, vol. 46, no. S1, pp. 611–628, 2008.

[18] Y. Xu, W. Tang, B. Chen, L. Qiu, and R. Yang, "A Model Predictive Control with preview follower theory algorithm for trajectory tracking control in autonomous vehicles," Symmetry, vol. 13, no. 3, p. 381, 2021.

[19] Z. He, L. Nie, Z. Yin, and S. Huang, "A two layer controller for lateral path tracking control of autonomous vehicles," Sensors, vol. 20, no. 13, p. 3689, 2020.

[20] H. Zhou, J. Gao, and H. Liu, "Vehicle speed preview control with road curvature information for safety and comfort promotion," Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, vol. 235, no. 6, pp. 1527–1538, 2021.

[21] B. Ren, H. Chen, H. Zhao, and L. Yuan, "MPC based yaw stability control in in wheel motored ev via active front steering and motor torque distribution," Mechatronics, vol. 38, pp. 103–114, 2016.

[22] Q. Yao, Y. Tian, Q. Wang, and S. Wang, "Control strategies on path tracking for autonomous vehicle: State of the art and future challenges," IEEE Access, vol. 8, pp. 161211–161222, 2020.

[23] N. S. Abu, W. M. Bukhari, M. Adli, S. N. Omar, and S. A. Sohaimeh, "A comprehensive overview of classical and modern route planning algorithms for self driving mobile robots," Journal of Robotics and Control (JRC), vol. 3, no. 5, pp. 666–678, 2022.

[24] N. Hassan and A. Saleem, "Analysis of Trajectory Tracking Control Algorithms for Wheeled Mobile Robots," in 2021 IEEE Industrial Electronics and Applications Conference (IEACon), (Penang, Malaysia), pp. 236–241, IEEE, Nov. 2021.

[25] A. Biswas, M. A. O. Reon, P. Das, Z. Tasneem, S. M. Muyeen, S. K. Das, F. R. Badal, S. K. Sarker, M. M. Hassan, S. H. Abhi, M. R. Islam, M. F. Ali, M. H. Ahamed, and M. M. Islam, "State of the Art Review on Recent Advancements on Lateral Control of Autonomous Vehicles," IEEE Access, vol. 10, pp. 114759–114786, 2022.

[26] Z. Zhu, X. Tang, Y. Qin, Y. Huang, and E. Hashemi, "A survey of lateral stability criterion and control application for autonomous vehicles," IEEE Transactions on Intelligent Transportation Systems, pp. 1–18, 2023.

[27] Y. Cao, W. ShangGuan, B. Cai, L. Chai, and W. Qiu, "Predictive trajectory planning for on road autonomous vehicles based on a spatiotemporal risk field," IEEE Intelligent Transportation Systems Magazine, vol. 15, no. 1, pp. 400–420, 2023.

[28] J. Jiang and A. Astolfi, "Lateral control of an autonomous vehicle," IEEE Transactions on Intelligent Vehicles, vol. 3, no. 2, pp. 228–237, 2018.
[29] S. Xu and H. Peng, "Design, analysis, and experiments of preview path tracking control for autonomous vehicles," IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 1, pp. 48–58, 2020.
[30] S. Feraco, A. Bonfitto, N. Amati, and A. Tonoli, "Combined lane keeping and longitudinal speed control for autonomous driving," in International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. 59216, p. V003T01A018, American Society of Mechanical Engineers, 2019.
[31] M. Samuel, M. Mohamad, M. Hussein, and S. M. Saad, "Lane keeping maneuvers using Proportional Integral Derivative (PID) and model predictive control (MPC)," Journal of Robotics and Control (JRC), vol. 2, no. 2, pp. 78–82, 2021.
[32] X. Shi, H. Wang, L. Chen, X. Sun, C. Yang, and Y. Cai, "Robust path tracking control of distributed driving six wheel steering commercial vehicle based on coupled active disturbance rejection," IEEE Transactions on Vehicular Technology, pp. 1–13, 2023.
[33] H.g. Kim, J. Myoung, S. Lee, K.m. Park, and D. Shin, "Design of ai powered hybrid control algorithm of robot vehicle for enhanced driving performance," IEEE Embedded Systems Letters, pp. 1–1, 2023.
[34] D. Li, D. Zhao, Q. Zhang, and Y. Chen, "Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]," IEEE Computational Intelligence Magazine, vol. 14, pp. 83–98, May 2019.
[35] L. Liu, M. Xue, N. Guo, Z. Wang, Y. Wang, and Q. Tang, "Investigating the path tracking algorithm based on bp neural network," Sensors, vol. 23, no. 9, 2023.
[36] H.L. Tran and T.V. Dang, "An ultra fast semantic segmentation model for amr's path planning," Journal of Robotics and Control (JRC), vol. 4, no. 3, pp. 424–430, 2023.
[37] X. Zhou, Z. Wang, H. Shen, and J. Wang, "Robust adaptive path tracking control of autonomous ground vehicles with considerations of steering system backlash," IEEE Transactions on Intelligent Vehicles, vol. 7, no. 2, pp. 315–325, 2022.
[38] T. Li, H. Ren, and C. Li, "Intelligent electric vehicle trajectory tracking control algorithm based on weight coefficient adaptive optimal control," Transactions of the Institute of Measurement and Control, p. 01423312221141591, 2023.
[39] X. Zhou, H. Shen, Z. Wang, H. Ahn, and J. Wang, "Driver centric lane keeping assistance system design: A noncertainty equivalent neuro adaptive control approach," IEEE/ASME Transactions on Mechatronics, 2023.
[40] A. Ubaidillah and H. Sukri, "Application of odometry and dijkstra algorithm as navigation and shortest path determination system of warehouse mobile robot," Journal of Robotics and Control (JRC), vol. 4, no. 3, pp. 413–423, 2023.
[41] E. Alcala´, V. Puig, J. Quevedo, and U. Rosolia, "Autonomous racing using linear parameter varying Model Predictive Control (LPV MPC)," Control Engineering Practice, vol. 95, p. 104270, 2020.
[42] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing," IEEE Open Journal of Intelligent Transportation Systems, vol. 3, pp. 458–488, 2022.
[43] N. Horri, O. Haas, S. Wang, M. Foo, and M. S. Fernandez, "Mode switching control using lane keeping assist and waypoints tracking for autonomous driving in a city environment," Transportation Research Record, vol. 2676, no. 3, pp. 712–727, 2022.
[44] D. J. Kim, Y. W. Jeong, and C. C. Chung, "Lateral Vehicle Trajectory Planning Using a Model Predictive Control Scheme for an Automated Perpendicular Parking System," IEEE Transactions on Industrial Electronics, vol. 70, pp. 1820–1829, Feb. 2023.
[45] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Tech. Rep. CMU RI TR 92 01, Carnegie Mellon University, Pittsburgh, PA, January 1992.
[46] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel,P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.E. Jen drossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, Stanley: The Robot That Won the DARPA Grand Challenge, pp. 1–43. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.
[47] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open source evaluation environment for continuous control and reinforcement learning," in Proceedings of the NeurIPS 2019 Competition and Demonstration Track (H. J. Escalante and R. Hadsell, eds.), vol. 123 of Proceedings of Machine Learning Research, pp. 77–89, PMLR, 08–14 Dec 2020.
[48] V. S. Babu and M. Behl, "f1tenth.dev an open source ros based f1/10 autonomous racing simulator," in 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE), pp. 1614– 1620,2020.