*2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*
*Amity University, Noida, India. Oct 13-14, 2022*

2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO) | 978-1-6654-7433-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRITO56286.2022.9964917

# Parallel Parking and Parking Assist System for Autonomous and Semi-Autonomous Vehicles

Poojitha Cheedalla
*Department of Computer Science and Engineering affiliated to JNTUH*
*Gokaraju Rangaraju Institute of Engineering and Technology affiliated to JNTUH*
Hyderabad, India
cheedalla.poojitha1999@gmail.com.

Madhavi Karanam, Professor
*Department of Computer Science and Engineering affiliated to JNTUH*
*Gokaraju Rangaraju Institute of Engineering and Technology affiliated to JNTUH*
Hyderabad, India
bmadhaviranjan@yahoo.com

***Abstract*— Modern autonomous vehicles are designed to provide convenience for people in their daily lives. A level 5 Advanced Driver Assistance System (ADAS) does not require any human attention and they are free from Geo-fencing. There is research going on and is not yet fully implemented. So this system can only be classified as fully autonomous driving level 1 to 4. Here Level-1 cars involves the human drivers and it is lowest level of automation. And in Level-2 there is no self-driving but instead the human sits in the car and this level of vehicles controls both steering, accelerating/decelerating. In Level-3 includes human interaction, if the system designed fails, designed on the basis of environmental detection. Level-4 vehicles are Self-driving, there are speed limits already designed within the systems. Assistive parking is the first application of autonomous vehicles that is fully autonomous when considering both risks and environments. In this research, we propose parallel parking assistance for autonomous and semi-autonomous vehicles. The plan is based on the kinematic model of the vehicle which is used as a parameter for a Model Predictive Controller (MPC) to control the vehicle autonomously and use the A* algorithm is a Informed Search Algorithm or a Best-First Search Algorithm it aims to determine the shortest path possible to reach the specified goal node, and here this algorithm is used to find the path planning and parking. Using this system, the vehicle is retrieved from the parking spot and re-parked by reversing along the obtained path. We propose methods of planning independent of a vehicle's initial position or orientation. The vehicle parks by following reference trajectories generated by the system.**

***Keywords—Parallel Parking, Advanced Driver Assistance system(ADAS), Model Predictive control(MPC).***

## I. INTRODUCTION

People can expect comfort and convenience from autonomous vehicles. There are many advanced driver assistance systems (ADAS) present in most vehicles on the road today. Due to safety and legal considerations, however, there are only a few applications utilizing fully automated driving technology. Hence, autonomous parking may be one of the first fully autonomous applications within a few years because of its relatively low risk and well-known environment. Additionally, parking has become a major challenge in metropolitan cities as the number of vehicles has increased. Furthermore, parking slots and garage spaces can be reduced without taking human factors into account. One of the most prominent benefits of autonomous vehicles is autonomous parking [1].

The last three decades have seen several proposals for automatic parking systems. The first type of strategy is to plan a parking trajectory. It is imperative to keep in mind that when designing a reference trajectory, which may be followed by the vehicle while moving and parking into the berths, it is pertinent to consider several key points. In order to successfully implement this kind of strategy, there are two major difficulties that are typically encountered. A vehicle cannot steer itself freely because it is a moving robot. In trajectory planning, it is difficult to adequately account for dynamic constraints. There is a tendency to address dynamic constraints indirectly through reference trajectories curvatures, which are usually not explicit. Unfortunately, as a result of the errors, it may not be able to implement the planned controls. Second, a feedback controller can be added that reshapes the steering actions and makes sure that the vehicle follows this reference trajectory approximately as well as the reference trajectory, based on the gap between the reference parking trajectory and the actual parking trajectory. A proper feedback controller is not an easy task to develop and implement, due to the design and implementation costs involved. Furthermore, there are two stages to implementing this strategy.

As discussed earlier, judging from [2] autonomous vehicles have four research issues relating to their functionalities. Localization, perception, design logic and motion planning and control are the research areas these vehicles cover. There are already existing methods for addressing autonomous parking issues in most of the research domains, except for motion planning and vehicle control. It is still necessary to overcome more stringent requirements when dealing with motion planning and vehicle control techniques in parking scenarios. This is necessary in order to make appropriate optimizations. This paper focuses on developing an autonomous parking system in a simulation environment using the matplotlib library. Specifically, this paper looks at the combination of control systems based on autonomous steering systems using MPC controller and path planning using A* algorithm to accommodate all types of parking situations, specifically the frequent parallel-parking and perpendicular-parking scenarios, as well as the rigid parking scenarios with obstacles.

Further this paper is presented as follows: In Section-II, we presented a little brief about the other experimentations done by other researchers in the field: Literature Survey. In section-III, we explain the background information about the systems like formulations, algorithms, tools used, etc. In section-IV, we described the methodology and

implementation of the proposed system. In section-V we discussed about the results and simulation environment. In section VI we concluded the paper.

## II. Literature Survey

Jyun-Hao Jhang et al., designed a system to overcome autonomous parking problems by combining a vehicle controller that is focused on parking and has a sampling-based motion planner. Path Planning and parking for various parking scenarios is made feasible with a motion planner that utilizes smooth-feedback Bi-RRT*. The steering and speed controls are combined to reduce the negative consequences, and the MPC calculation and the vehicle control are carried out simultaneously in the proposed controller in advance [3].

Jiyuan Tan et al., proposed a different automatic parking system to solve APP problem which have a three-step guidance control strategy which has five dynamic variables controlling its trajectory: X0, Y0, δf, V, and θ. It should be able to accommodate a maximum vertical distance from the berth in relation to its length and width. In this novel method, the desired control actions are evaluated directly rather than first identifying the desired trajectory. The steps are mentioned as follows:

Step 1: From the initial state, drive straight to reach the starting position by making a slight angle change between the lateral and longitudinal axis of the vehicle [4].

Step 2: Drive the vehicle backward until the steering angle from the lateral axis to the longitudinal axis of the vehicle reaches the preselected critical angle position, starting from the starting position with a completely right turn (setting the steering angle) with a velocity of v [4].

Step 3: Once the vehicle reaches the final position in the berth (angle from X-axis to longitudinal axis of vehicle is approximately 0) the vehicle will be driven backward with full turns to the left (setting steering angle).

Jerome T et al., proposed a fuzzy Neuro control system-based APP for vehicles. In this study, the authors demonstrate how to maneuver an autonomous car-like mobile robot into parallel parking using sensors. Specifically, the project seeks to parallel park a car-like mobile robot following a backward maneuver in a 5-degree polynomial reference path [5]. In this model they used simple kinematics of a vehicle is used. Later using 5-degree polynomial paths as the training data, a subtractive clustering algorithm is employed to determine the fuzzy controller. In turn, the controller is trained by using a neuro-fuzzy inference system with adaptive inference and true-false training [5]. A car-shaped mobile robot (CLMR) is being equipped with eight ultrasonic sensors strategically placed to avoid radial imprecision, which measure the angle of inclination of the car [5]. A specific accelerometer is also being used to measure the angle of the car.

Bai Li et al., proposed a spatial-temporal decomposition method for APP motion optimization to generate reliable initial guesses and to facilitate the NLP-solving process [6]. The proposed initialization strategy was compared with current competitors in a series of comparative simulations, and it has been concluded that the proposed motion planner has the potential to meet the needs of online planning missions. They were able to do this by using the kinematics of a vehicle for defining the optimal control problem. Then by applying the Interior-Point Method (IPM) we can approximate a nonlinear based system using the nonlinear based system for approximations. When there are a number of active constraints involved, IPM is efficient when dealing with large-scale problems. By incorporating logarithmic barrier weights into the optimization objective, IPM computes solutions for a sequence of barrier problems. There are so many scenarios to handle in the proposed plan that it generates a parking motion every time in tens of seconds, which indicates that it is not suitable to handle on-site or online motion planning. By making the motion planner unified, a wide range of scenarios can be handled instead of just a few special cases. Even though the present study employs only objective knowledge, reasonable parking motions can still be generated [6].

A fuzzy inference-based auxiliary parking system is proposed by Ozkul et al. To facilitate parallel parking efficiently and accurately, the system is based on fuzzy inference. In real time, images are captured through the input provided by two low-resolution digital cameras located outside of the vehicle. The necessary information about the parking position and distance can be extracted from the scene after online CPU filtering, edge detection, line extraction, and image processing are applied. These algorithms, in conjunction with fuzzy logic, can produce accurate results. By providing motorists with concrete operating information, the parking system can advise them on how to maneuver and control the speed of their vehicles. In contrast to automated processes, this method provides drivers with operational recommendations [7].

## III. Background System

### A. A* Path Planning

In this section, we are going to examine a path planning problem, in particular the problem of finding the shortest path route between two points in a graph G = (V, E) where V is the set of nodes representing the locations of the environment, and E is the list of valid ways to move between the nodes. In order to simplify the definition of G, we define an alternate graph form where G = (V, N) [8].

**Algorithm 1** A* Search

**Input:** Graph $\mathcal{G}$, movement cost $c$, start $v_s$, and goal $v_g$
**Output:** Shortest path $P$

1: Initialize $\mathcal{O} \leftarrow v_s, \mathcal{C} \leftarrow \emptyset, \texttt{Parent}(v_s) \leftarrow \emptyset$.
2: **while** $v_g \notin \mathcal{C}$ **do**
3: Select $v^* \in \mathcal{O}$ based on Eq. (1).
4: Update $O \leftarrow O \setminus v^*, \mathcal{C} \leftarrow \mathcal{C} \cup v^*$.
5: Extract $\mathcal{V}_{\text{nbr}} \subset \mathcal{V}$ based on Eq. (2).
6: **for each** $v' \in \mathcal{V}_{\text{nbr}}$ **do**
7: Update $\mathcal{O} \leftarrow \mathcal{O} \cup v', \texttt{Parent}(v') \leftarrow v^*$.
8: **end for**
9: **end while**
10: $P \leftarrow \texttt{Backtrack}(\texttt{Parent}, v_g)$.

Fig. 1. A* path finding algortithm [8]

This work makes use of an A* search algorithm. When constructing a shortest path, all nodes are explored and then the most promising node is selected from a list of candidates, after which the neighboring nodes expanding the candidate list until the goal is reached and a shortest path is found. It should be noted here that the parameters for selecting nodes

in the Algorithm 1 (Line 3) are determined by using the following criteria:

$$\vartheta^* = arg\min\big(g(v) + h(v)\big), v\epsilon 0 \quad (1)$$

A node selection is performed using O ⊂ V, which is an open list of potential nodes. In search algorithms, the actual cost of accumulating c(v') for nodes v' is updated incrementally during each step, representing the total cost of the current best path from $v_s$ to v. Conversely, h(v) is a heuristic function used in grid world to estimate the total cost between v and $v_g$, called the straight line distance. Similarly to Line 4, the closed list, C ⊆ V, is used to store all of the selected nodes.

According to Algorithm 1, Line 5, we expand Vnbr ⊂ V to all neighbors of v *:

$$V_{nbr} = \{v'[v' \epsilon N(v^*) \wedge v' \notin O\, v' \notin C\} \quad (2)$$

To propose new selection candidates in the next iteration, $V_{nbr}$ is then added to O in Line 7. After vg is selected in Line 3 and stored in C, Backtrack is used to obtain P by tracing the parent nodes in the path from $v_g$ to $v_s$.

### B. Vehicle Kinematics

As shown in Figure 1, the vehicle model is defined along with the parameters relevant to this study. The kinematics of four wheels can be represented by the bicycle model. The details of the bicycle model are in [32] and [33]. In addition, the Ackermann steering geometry is used since the vehicle will be traveling at low speeds. Accordingly, the position of the vehicle (x, y) in the world coordinate corresponds to the centroid of the rear of the vehicle. Yaw angle θ is defined as the angle between the reference angle of the vehicle and the x-axis of the world coordinate system. The scalar alone the vehicle axis is used to define the vehicle speed, v. As far as steering kinematics are concerned, steering angle δ, is defined as the angle between the heading of the vehicle and the steering wheel orientation. Based on the dimensions of the vehicle, $R_{turn}$ is the turning radius. In addition, the wheelbase, WB the height, and the width all contribute to $R_{turn}$ [9].

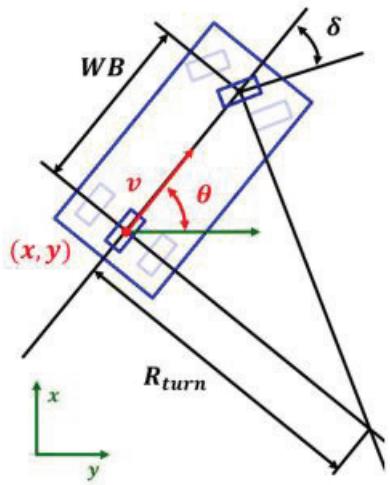


Fig. 2. Kinematic model of vehicle [9]

### C. MPC Controller

An MPC model is optimized iteratively over finite horizons. By utilizing a numerical minimization algorithm, an optimal control strategy is computed (for a relatively short time horizon in the future) based on the current plant state. This paper explores trajectories emerging from the current state and finds (through the solution of Euler-Lagrange equations) a cost-minimizing control strategy until the end of time. A control strategy consisting of only the first step is implemented, and then the state of the plant is sampled again, followed by repeated calculations, resulting in a new predicted state path and new control. Due to the continuous shifting of the prediction horizon, MPC is also known as receding horizon control.

In almost all cases, MPCs are formulated in state spaces. Assume a linear discrete-time description of the system in state space.

$$x(K+1) = Ax(k) + Bu(k), x(0) = x_0 \quad (3)$$

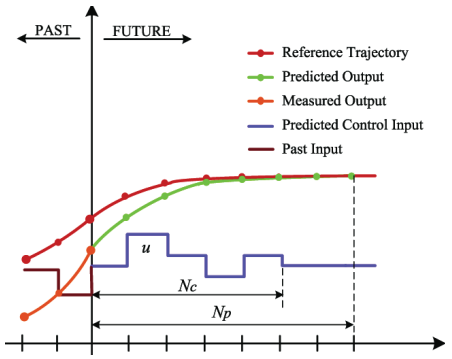


Fig. 3. Discrete MPC controller

### D. Linear Steering and Speed Kinematic Model

Using the referred papers [34] and [33] as guidelines for the proposed controller, they aim to control speed and steering simultaneously. A summary is given below:

$$z_{k+1} = Az_k + Bu_k + C \quad (4)$$

According to the following definition, the vehicle state, $z_k$ and the control input, $u_k$, are:

$$z_k = \begin{bmatrix} x_k \\ y_k \\ v_k \\ \theta_k \end{bmatrix}, u_k = \begin{bmatrix} a_k \\ \delta_k \end{bmatrix}, \quad (5)$$

Here is a description of state-space matrices A, B, and C:

$$A = \begin{bmatrix} 1 & 0 & \cos(\bar{\theta})\, dt & -\bar{v}\sin(\bar{\theta})\, dt \\ 0 & 1 & \sin(\bar{\theta})\, dt & \bar{v}\cos(\bar{\theta})\, dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{\tan\bar{\delta}}{L} & 1 \end{bmatrix}, \quad (6)$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ dt & 0 \\ 0 & \frac{\bar{v}}{L\cos^2(\bar{\delta})} dt \end{bmatrix}, \quad (7)$$

$$C = \begin{bmatrix} \bar{v}\sin(\bar{\theta})\bar{\theta}\,dt \\ -\bar{v}\cos(\bar{\theta})\bar{\theta}\,dt \\ 0 \\ \frac{\bar{v}\times\bar{\delta}}{L\cos^2(\bar{\delta})} \end{bmatrix} \quad (8)$$

This linear state space model tries to regulate the position (x, y), speed (v), and heading () of a vehicle using acceleration (a) and steering angle () as control inputs. The state space matrices A, B, and C are created by fusing the control input and prediction state in a discrete-time fashion. As a result, the discrete-time model and control for the linear state space model used for autonomous cars [9].

*E. B-Spline curve*

Unlike Bezier curves, B-spline curves have many advantages and overcome many of the shortcomings of Bezier curves. A B-spline curve is expressed as follows [10]:

$$Q_{i,n}(u) = \sum_{k=0}^{n}[P_{i+k}\ F_{k,n}(u)] \quad (9)$$

$$F_{k,n}(u) = \frac{1}{n!}\sum_{j=0}^{n-k}[(-1)^j\ C_{n+1}^{j}(u+n-k-j)^n \quad (10)$$

When a real value xi is given, then $x_0 \le x_1 \le \cdots \le x_{m-1}$, i= 1,2,…,m-n; k=0,1,2,…,n; called knots, represents the parametric curve of B-splines of degree n.

Using formula (14) we calculate the path planning function for a B-spline:

$$F_{i,3}(u) = \frac{1}{6}\begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix}[G] \quad (11)$$

$$G = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \quad (12)$$

$$P = \begin{bmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{bmatrix}, \text{ ι= 1,2,.. m −3} \quad (13)$$

*F. Parking Trajectory*

There are many variables and complexes involved in automatic parking. Thus, the present paper proposes a novel method that includes both vertical and horizontal coordinates of parking starting points, and offers a new method of calculating parking starting point area that is two dimensional. Fig.4 shows the radius and central angle of the arc near the end of the parking by the double-arc planning method. Arcs from a starting point and azimuth are planned from the azimuth and radius of the parking by this method. The turning radius R1 should be set to the minimum turning radius that is required for the vehicle. The radius, centre angle, and centre speed of the parking arc are all determined by R2 which indicates the radius and centre speed. The starting point of parking can be determined based on some constraints. There is a distance of d1 between the trajectory and the lane line. If we compare the trajectory to the angle of parking, we find the closest distance to be d2. A distance of d3 to the left front corner of the vehicle indicates the nearest location to the lane line. d4 represents the closest distance between the vehicle's right front corner and the lane line [10]. The vehicle must meet the following requirements in order to avoid a collision:

$$\begin{cases} d_1 > \frac{L_k}{2};\ d_2 > \frac{L_k}{2}; d_3 > 0; d_4 > 0 \\ \theta_1 < \frac{p_i}{2};\ \theta_1 + \theta_2 < p_i \end{cases} \quad (14)$$

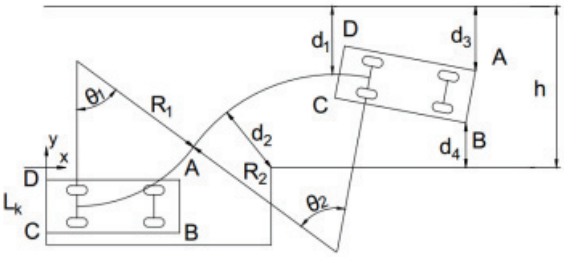


Fig. 4. B- Spline Curve [10]

## IV. METHODOLOGY

In the proposed vehicle controller, the vehicle is controlled as shown by a designated parking route to resolve problems associated with autonomous parking. According to the proposed vehicle controller, the vehicle speed and steering should be controllable by the suggested motion planner simultaneously in autonomous parking in order to guarantee that the outcomes of the suggested motion planner are precisely tracked and successfully carried out. Furthermore, the solution to these practical problems is to implement a vehicle controller that helps overcome some of the problems encountered in real-world applications, including the need to simultaneously control speed and steering, the delay between a steering request and a steering reaction, as well as the computation delay. As a result of the development of a model predictive speed and steering control, a main structure of the proposed vehicle controller has been developed [3].

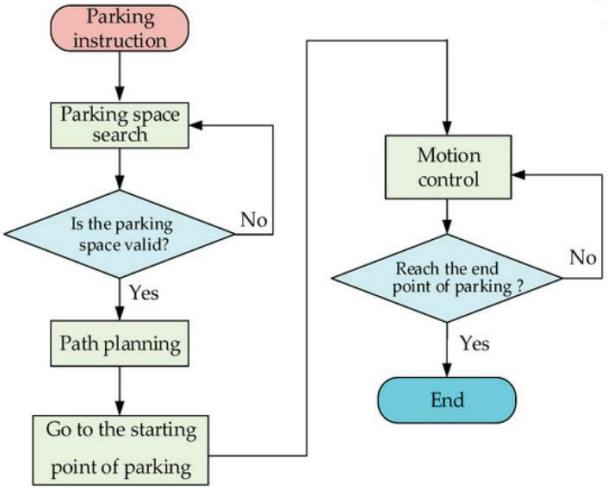


Fig. 5. *FlowChart*

The figure below shows the steps involved in autonomous parallel parking.

The driver issue parking instructions after the vehicle arrives at its destination. This function enables a vehicle to determine whether a parking space found within a specified distance of the destination is suitable for parking and eases the search process until finding a parking spot that satisfies the requirements for parking.

Having located a parking space, a secure and collision-free parking lane is planned by the autonomous parking system, and the vehicle travels to the beginning of the carpark.

The automated parking system manages the car's movement based on the intended path using signals for front wheel angle and vehicle speed. The driver gets back in their car after the car exits the parking area without a hitch or other incident.

### A. Environment

To be able to visualize the performance and working of the proposed system we designed an simulation environment to get a visual render using the OpenCV library. When env.render(x,y,angle) is used, the controller process can be placed in the coordinate system of grid map [11]. To simulate the model of a vehicle certain parameters are considered to create a vehicle model as shown in fig 6. Also there is a need to simulate the parking environment with few static vehicles and empty places to see how accurate the model works.

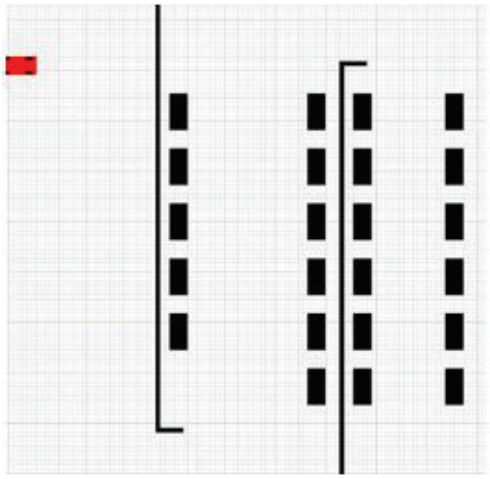

Fig. 6. Vehicle model in a simulated grid map

### B. Path Planning and Trajectory Generation

To drive the autonomous vehicle to the parking space A* algorithm generates the shortest possible path. Traditional A* algorithms may generate paths that are very close to obstacles [12]. The path will pose a high risk of collision with obstacles to a an autonomous car if it is selected as the planning path. In order to plan a path effectively, it is essential to maintain a safe distance from barriers. In path planning, the expansion distance is used to keep a safe distance between the obstacles. In autonomous car path planning, rasterized maps are used. Expanding around obstacles, the primary unit for expansion distance is a grid of measurement. The Algorithm showed in fig.1 and (1), (2) provides the reference trajectory to the system to direct the vehicle model to the starting side parking slot.

Once the car reaches the parking starting point with the help of b-spline algorithmic equations are mentioned in (9) - (14). A path is smoothed and scaled into an environment of 1000*1000 after being initially found in a discrete 100*100 space. The vehicle model then travels along these points [11].

### C. Path Tracking

The kinematic model and dynamic model of the vehicle is used to simulate the vehicle model in order to know and simulate how exactly the vehicle act while implementing the control system. The parameters are explained in section - III-B. Using these parameters and equations of motion the state space matrices are formulated in (4) to (8).

Using the MPC controller, vehicle steering and speed are controlled based on the model. It is possible to use a linearized model with the MPC controller. A kinematic model is linearized around the operating point in this case, and the optimization is then performed.

This controller has a prediction horizon of 0.2 s, therefore it measures 5 steps ahead. Taking advantage of the system's performance capabilities and computational complexity, sampling time and horizon were selected according to the optimization problem [13].

The reference path is generated by projecting the car's current location onto the track/centerline of the road in order to follow it. Referencing is done before the car moves. Reference distances are determined by desired velocity.

### D. Parallel Parking

The algorithm must choose 4 rules that apply to parking positions based on this part. The following is one of those rules. The fig.4 illustrating the specification for path design.

After finding a path to the park position, the algorithm will calculate the arrival angle. An agent chooses two coordinates for the ensure1 and ensure2 points based on the arriving angle ($x_0$, $y_0$). In other words, these are the points where parking will begin and end [14].

Thereafter, the parking path is mapped using two circle equations, as shown below, from starting point ($x_0$, $y_0$) to end position ($x_1$, $y_1$). A perfect parking alignment is achieved by adding two straight lines to the path. Using ensure coordinates, MPC controls the motion of the agent through paths and parking lots.

## V. Results

In this chapter, we simulate motion planner comparisons in parking scenarios. In advance, simulation tests are conducted to determine the effects of each technical element in the proposed vehicle controller. Simulations and experiments are based on the same parameters. The simulated model of car from starting position and parking is shown in below fig.7.

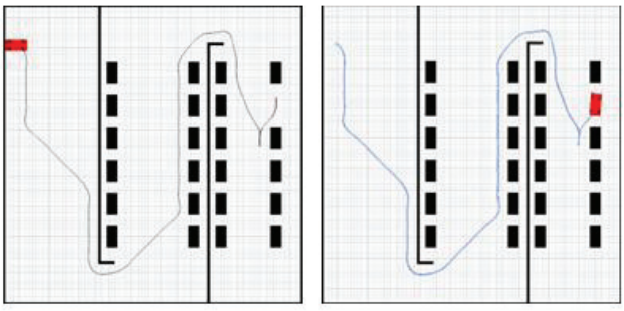

Fig. 7. Parallel Parking Images at Starting and Ending Points

The first result in Fig.7 shows the orientation of the vehicle using the MPC controller in parallel parking. It is important to keep in mind that the car initially traveled at a 55 degree angle clockwise from the horizon, then this angle was corrected to be almost horizontal.

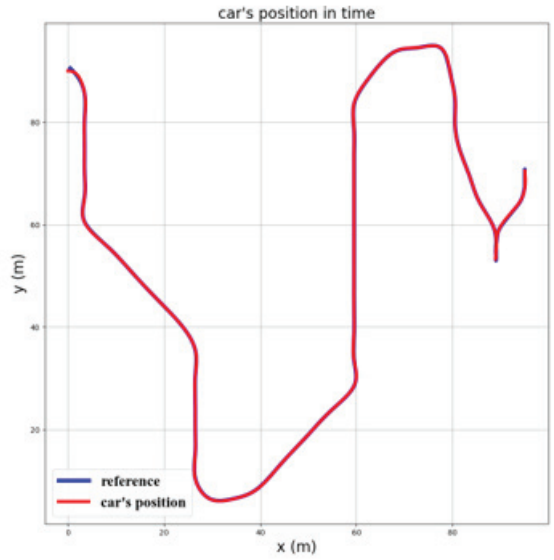


Fig. 8. Simulation of Parallel Parking

An A* reference trajectory and the desired trajectory are compared in the second result. Figure 8 illustrates the same motion in both routes, although the end of the route showed more compliance. As a result, it was eventually found that the car would be parked the optimum point is located at the same y coordinate.

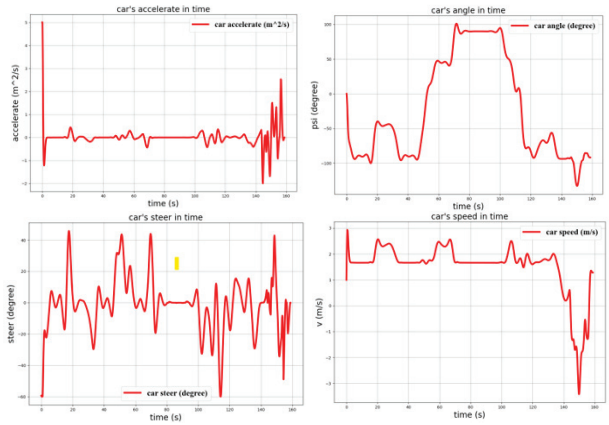


Fig. 9. Simulation graphs of acceleration, velocity, steering angle and yaw angle

In fig.9 depicts the acceleration, velocity change over the time period, steering angle of the vehicle and predicted yaw angle graphs.

## VI. Conclusion

A novel technique for automatic parking is proposed in this paper. The new method, which differs from existing methods, describes control actions directly, as opposed to seeking the desired trajectory first. There have also been attempts to split up the whole series of control actions into three simple steps that can be characterized explicitly and implemented easily. It is indeed true that the three-step control strategy is directly imitated from a human driving process.

With the dynamic parameters of the vehicle known, it is possible to detect the precise values of the guidance control parameters. The results of testing show that this method of guiding works well as long as there is enough available parking space. As well, this study also demonstrates how we can create better automatic parking systems.

## References


[1] R. Hussain and S. Zeadally, "Autonomous Cars: Research Results, Issues, and Future Challenges," in IEEE Communications Surveys & Tutorials, vol. 21, no. 2, pp. 1275-1313, Secondquarter 2019, doi: 10.1109/COMST.2018.2869360.

[2] K. Jo, J. Kim, D. Kim, C. Jang and M. Sunwoo, "Development of Autonomous Car—Part I: Distributed System Architecture and Development Process," in IEEE Transactions on Industrial Electronics, vol. 61, no. 12, pp. 7131-7140, Dec. 2014, doi: 10.1109/TIE.2014.

[3] J. -H. Jhang and F. -L. Lian, "An Autonomous Parking System of Optimally Integrating Bidirectional Rapidly-Exploring Random Trees* and Parking-Oriented Model Predictive Control," in IEEE Access, vol. 8, pp. 163502-163523, 2020, doi: 10.1109/ACCESS.2020.30.

[4] J. Tan, C. Xu, L. Li, F. Wang, D. Cao and L. Li, "Guidance control for parallel parking tasks," in IEEE/CAA Journal of Automatica Sinica, vol. 7, no. 1, pp. 301-306, January 2020, doi: 10.1109/JAS.2019.1911855.

[5] J. T. Marasigan, I. M. B. Saberon, D. P. B. San Jose, P. A. T. Sevilla and A. A. Bandala, "Autonomous parallel parking of four wheeled vehicles utilizing adoptive Fuzzy-Neuro control system," 2014 IEEE REGION 10 SYMPOSIUM, 2014, pp. 640-644, doi: 10.1109/.

[6] Li, Bai & Zhang, Youmin & Shao, Zhijiang. (2016). Spatio-temporal decomposition: a knowledge-based initialization strategy for parallel parking motion optimization. Knowledge-Based Systems. 1-18. 10.1016/j.knosys.2016.06.008.

[7] Ozkul, Tarik, Mohammed Moqbel, and Suhail BA Al-Dhafri. "Development of a hierarchical driver aid for parallel parking using fuzzy biomimetic approach." Journal of computing and information technology 18.1 (2010): 31-44.

[8] Yonetani, Ryo, et al. "Path planning using neural a* search." International Conference on Machine Learning. PMLR, 2021.

[9] Jhang, Jyun-Hao, and Feng-Li Lian. "An autonomous parking system of optimally integrating bidirectional rapidly-exploring random trees* and parking-oriented model predictive control." IEEE Access 8 (2020): 163502-163523.

[10] Ye, Mao, Xin Ji, and Yexin Zhao. "A trajectory planning method based on b-spline algorithm for automatic parking systems." Proceedings of the 3rd International Conference on Computer Engineering, Information Science & Application Technology (ICCIA 2019).

[11] A. Heydarian, "Towards Data Science," 27 July 2021. [Online]. Available: https://towardsdatascience.com/automatic-parallel-parking-system-including-path-planning-path-tracking-and-parallel-parking-in-a-ece780b2e8e0.

[12] Wang, Huanwei, et al. "The EBS-A* algorithm: An improved A* algorithm for path planning." PloS one 17.2 (2022): e0263841.

[13] Yu, Leiyan, et al. "Path Planning Optimization for Driverless Vehicle in Parallel Parking Integrating Radial Basis Function Neural Network." Applied Sciences 11.17 (2021): 8178.

[14] Tan, Jiyuan, et al. "Guidance control for parallel parking tasks." IEEE/CAA Journal of Automatica Sinica 7.1 (2019): 301-306.