Motion Planning for Autonomous Driving

Anoushka Baidya
Robotics Engineering
Worcester Polytechnic Institute
abaidya@wpi.edu

Jinesh Rajasekhar
Robotics Engineering
Worcester Polytechnic Institute
jrajasekhar@wpi.edu

Loahit Krishnamurthy
Robotics Engineering
Worcester Polytechnic Institute
lkrishnamurthy@wpi.edu

***Abstract*—Autonomous vehicles have the potential to revolutionize the transportation industry by providing safe and efficient mobility solutions. The goal of this project is to develop a Local Motion Planning framework for Autonomous Driving in urban environments capable of generating smooth, feasible and efficient paths to enable autonomous vehicles to navigate through obstacles and reach their desired destinations. The implemented local Planner will be tested in the CARLA simulator. The results should show that the proposed framework is able to generate smooth and safe paths in both static and dynamic environments.**

***Index Terms*—Collision avoidance, Motion Planning, RRT, RRT* algorithm**

I. Introduction

Autonomous vehicles (AVs) have recently gained significant attention as a possible solution to transportation-related problems such as accidents. With the increase of AVs, it is crucial to develop efficient algorithms for path planning, which is a crucial aspect of the decision-making process in autonomous vehicles. Path planning is finding a feasible path for an autonomous vehicle to reach its desired destination while avoiding obstacles and ensuring safety.

In recent years, there have been significant advances in path planning for autonomous vehicles. Researchers have developed various algorithms and techniques for path planning, including graph-based methods, sampling-based methods, and reinforcement learning methods. These methods have been designed to overcome challenges in the path planning process, such as real-time performance, reliability, and scalability.

The development of path-planning algorithms for autonomous vehicles is still a challenging task due to the complexity of the problem and the constraints of the autonomous vehicle's environment, where the vehicle must react to changing situations and adjust its path accordingly. This project aims to design path-planning algorithms for autonomous vehicles and examine the challenges and future prospects in this area. The focus is placed on implementing collision avoidance with surrounding traffic and infrastructure, followed by implementing a path planning algorithm and assessing their strengths and weaknesses.

II. Vehicle Modelling

Developing a robust vehicle model is a critical aspect of controlling autonomous vehicles. Such models must consider both the kinematics, which pertains to the position and velocity of the vehicle and the dynamics, which concern the forces and torques acting on the vehicle. One approach to modeling vehicle motion is to consider the geometric constraints that define its motion or to consider all the forces and moments acting on the vehicle.

The Bicycle Model is a commonly used framework for modeling the kinematics and dynamics of autonomous vehicles. In this paper, we focus on a framework for low-speed autonomous vehicles and therefore model our vehicle motion based on the kinematic model. Our vehicle model is subject to non-holonomic constraints, which dictate that the robot can move forward and turn while rolling but cannot move sideways directly.

To model the differential constraints on our vehicle, we use the Bicycle Model. In this model, the front wheels represent the front right and left wheels, while the rear wheels represent the rear right and left wheels. The selection of reference points can affect the results of the kinematics equation. We are assuming that the velocity v points in the same direction as each wheel, which is referred to as the no-slip condition, enabling us to calculate the vehicle's forward speed based on its wheels' rotation speed.

The below equations represent the model [10], where the car's position is represented by its x and y coordinates, and the heading angle is represented by theta. The velocity of the car is represented by v, the steering angle of the front wheel is represented by δ, the heading angle of the car is denoted by θ and the wheelbase of the car is represented by L.

Using the rear axle reference point, the complete kinematic bicycle model is

$$\begin{aligned} \dot{x}_r &= v_r \cos(\theta) \\ \dot{y}_r &= v_r \sin(\theta) \\ \dot{\theta} &= \frac{v_r}{L} \tan(\delta) \end{aligned} \tag{1}$$

where $v_r$is the velocity of the rear wheel

For the front axle reference point, the complete kinematic bicycle model for the differential Constraint is given by

$$\begin{aligned} \dot{x}_f &= v_f \cos(\theta + \delta) \\ \dot{y}_f &= v_f \sin(\theta + \delta) \\ \dot{\theta} &= \frac{v_f}{L} \sin(\delta) \end{aligned} \tag{2}$$

where $v_f$is the velocity of the front wheel

By considering these models and constraints, we can create a comprehensive vehicle model that accurately reflects the motion of an autonomous vehicle.

III. METHOD

A. Trajectory Planning

In autonomous driving, the motion planning component is responsible for generating a collision-free trajectory that satisfies the vehicle's dynamic and kinematic constraints. This is achieved by taking into account information about the static and dynamic obstacles around the vehicle. Typically, the output of the motion planner is passed to the local feedback control layer, which generates input signals to regulate the vehicle to follow the desired trajectory.

A constant velocity minimum acceleration reference trajectory is a trajectory that is commonly used in autonomous vehicle navigation to generate smooth and comfortable driving behavior. The idea behind this reference trajectory is to maintain a constant velocity while minimizing the vehicle's acceleration and jerk (rate of change of acceleration). This helps reduce the discomfort and jarring motions that passengers can experience in a vehicle constantly accelerating and decelerating.

To generate this trajectory, the desired velocity of the vehicle is first determined, and the initial and final positions of the trajectory are defined. A third-order polynomial function is then used to generate a smooth curve that connects the initial and final positions with a constant velocity. The coefficients of the polynomial function are set such that the velocity remains constant, and the acceleration and jerk are minimized while enforcing constraints on the initial and final positions, velocities, accelerations, and jerks. Once the coefficients of the polynomial function are determined, the trajectory can be tracked by the autonomous vehicle using a feedback control loop. This loop adjusts the steering and throttle inputs to follow the trajectory as closely as possible.

The cubic trajectory is given by the following [11]: The equation for x is given by:

$$x = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

where a_0, a_1, a_2, and a_3 are constants and x is the sate variable.

Considering four states, a trajectory is generated by using interpolation between different waypoints at the selected time instant. This interpolation is done between the initial states $S_0 = (x_0, \dot{x}_0, y_0, \dot{y}_0)$ and the final states $S_f = (x_f, \dot{x}_f, y_f, \dot{y}_f)$ where x and its time derivatives is the longitudinal position and velocity while y is the lateral equivalents.

$$\begin{bmatrix} x_0 \\ \dot{x}_0 \\ x_f \\ \dot{x}_f \end{bmatrix} \tag{3}$$

The longitudinal and lateral trajectories are calculated separately, assuming two different polynomials for the positions of the form

$$\begin{bmatrix} 1 & T_i & T_i^2 & T_i^3 \\ 1 & T_f & T_f^2 & T_f^3 \\ 0 & 1 & 2T_f & 3T_f^2 \\ 0 & 0 & 2 & 6T_f \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} x_0 \\ \dot{x}_0 \\ x_f \\ \dot{x}_f \end{bmatrix} \tag{4}$$

for longitudinal trajectory and similarly for the lateral trajectory

$$\begin{bmatrix} 1 & T_i & T_i^2 & T_i^3 \\ 1 & T_f & T_f^2 & T_f^3 \\ 0 & 1 & 2T_f & 3T_f^2 \\ 0 & 0 & 2 & 6T_f \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} y_0 \\ \dot{y}_0 \\ y_f \\ \dot{y}_f \end{bmatrix} \tag{5}$$

In this case, the initial and final time to cover the distance between each waypoint is given by

$$\mathrm{T}_i = 0 \text{ and } T_f \quad = \frac{d}{v} \tag{6}$$

Where d is the distance between the waypoints and v is the velocity of the vehicle.

Overall, the constant velocity minimum acceleration reference trajectory is a useful tool for generating smooth and comfortable driving behavior in autonomous vehicles, improving ride quality for passengers, and reducing wear and tear on vehicle components.

B. Feedback Controller Design

In autonomous vehicles, the feedback controller is crucial in stabilizing the vehicle's path or trajectory against uncertainties and modeling errors. To address the control problem in autonomous vehicles is divided into two parts: longitudinal controller and lateral controller.

The longitudinal controller regulates the vehicle's speed along a fixed path, which can be achieved with reasonable accuracy using control techniques. The controller senses the vehicle's speed and adjusts the throttle and brake commands to match the desired speed set.

On the other hand, the lateral controller corrects any accumulated error and tracks changes in the path direction by selecting the steering angle. At any given point, it calculates the error between the vehicle's current position and the reference trajectory's nearest point and steers the car toward that point to minimize the error.

For our controller model, we have chosen the PID (Proportional-Integral-Derivative) controller for both the longitudinal and lateral control of the car [12]. The PID controller is a widely used feedback control algorithm known for its robustness, flexibility, and efficiency. Its use in longitudinal and lateral control offers several advantages, such as accuracy, stability, and adaptability.

The PID controller equation is given by:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de}{dt}$$

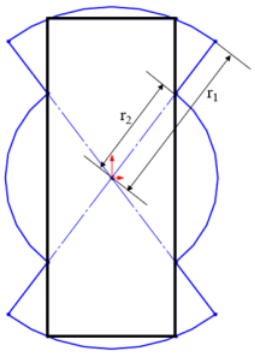

Fig. 1. Bounding Area

Where $u(t)$ is the control input at time t, $e(t)$ is the error at time t, and K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively.

Overall, the PID controller is a reliable and effective control algorithm that can significantly improve the performance, efficiency, and safety of autonomous vehicles. Its simplicity and flexibility make it a popular choice for various longitudinal and lateral control applications.

C. Collision Avoidance

The intuitive way for a collision check is to consider a circular bounding area for both the ego vehicle and the obstacle. But in the case of cars driving in narrow lanes, this method won't work. To check accurately for collision, a collision-checking algorithm was to be developed that more accurately considers the actual shape of obstacles and the ego vehicle itself.

It uses two different radius coaxial circles to bond the area. The bounding area geometry is shown in Fig.1. The bounding area considers the orientation of the ego and the obstacle vehicle to select the respective collision circle radius. The radii of both circles can be increased to provide more buffer for safety. It also can't be increased to an extent where it's impossible to overtake a vehicle, so proper tuning is needed for the two radii.

Fig.2 demonstrates how the collision avoidance algorithm works. At first, it calculates the angles between the vehicles and the line connecting the centers of the vehicles. The collision radius is detected based on which the range in angle lies. In Fig.2, the collision radius for the ego vehicle would be r_1 while the obstacle vehicle's collision radius would be r_2. Adding the two radii will give the collision distance between two vehicles. If the actual distance between the centers is smaller than the collision distance, collision is detected.

D. Motion planning in static environment

We use RRT* for motion planning. RRT* (Rapidly-Exploring Random Trees*) is a commonly used algorithm in motion planning, which helps to find the shortest path from a start state to a goal state in high-dimensional configuration space.

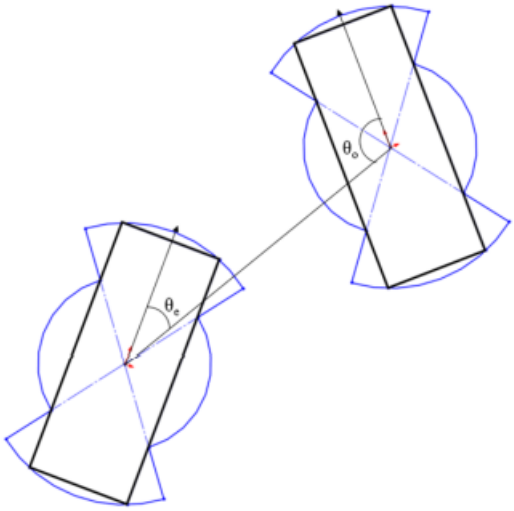

Fig. 2. Geometry of Bounding Area between vehicles

The algorithm works by constructing a tree of states, where the start state is the tree's root, and the goal state is a node on the tree. The tree is gradually grown by randomly selecting a state in the configuration space and adding it to the tree if it is valid. The tree is then optimized to find the shortest path between the start and goal states. Fig.3 shows the pseudo-code for RRT* algorithm.

The main advantages of RRT* are its simplicity, fast convergence, and ability to expand toward unexplored regions of the configuration space. It balances accuracy and time complexity well, making it an effective and efficient solution for motion planning problems.

IV. Progress Achieved

- Configured the environment - We have successfully set up the environment in Carla and configured environment variables to ensure smooth implementation. Additionally, we have also set up the ego vehicle for our project.
- Modeled the Autonomous Vehicle - To simulate the motion of the ego vehicle, we have employed the simplified bicycle model.
- Implemented Controllers - We have implemented a PID controller for both lateral and longitudinal control, which ensures that the vehicle follows a desired trajectory.
- Collision Avoidance - Implemented Collision detection for stationary vehicles.
- Implemented Motion Planning Algorithms - Our project incorporates a motion planner that enables the vehicle to navigate from the starting point to the destination while avoiding any static obstacles.

V. Future Targets To Achieve

- Implement Velocity Obstacles algorithm to deal with dynamic obstacles.
- Integrate Velocity-obstacles based collision avoidance with the RRT* planner.

*RRT** algorithm

$T \leftarrow InitializeTree();$

for $k = 1$ *to* K *do*

  $x_{rand} \leftarrow RANDOM\_STATE();$

  $x_{near} \leftarrow NEAREST\_NEIGHBOR(x_{rand}, T);$

  $x_{new} \leftarrow STEER(x_{rand}, x_{near})$

    $u \leftarrow SELECT\_INPUT(x_{rand}, x_{near});$

    $x_{new} \leftarrow NEW\_STATE(x_{near}, u, \Delta t);$

    if $obstaclefree(x_{new})$ *then*

    return x_{new};

$x_{near\_neighbor} \leftarrow findnear\_neighbor(T, x_{new}, r);$

if $obstaclefree(x_{new}, T, r)$ *then*

  $T \leftarrow Chooseparent(x_{new}, x_{near\_neighbor}, T);$

    for each $x_{near\_neighbor}$ $calculate(dist(x_{new}, x_{near\_neighbor}) + \text{cost}(x_{near\_neighbor}, x_{init}))$

    $x_{near\_neighbor_{min\,cost}} \leftarrow \min(dist(x_{new}, x_{near\_neighbor}) + \text{cost}(x_{near\_neighbor}, x_{init}));$

    $x_{new\_parent} \leftarrow x_{near\_neighbor_{min\,cost}}$;

    return x_{new_parent};

  $T \leftarrow rewire(T, x_{new}, x_{near\_neighbor});$

    for each $x_{near\_neighbor}$ $calculate(dist(x_{near\_neighbor}, x_{new}) + \text{cost}(x_{new}, x_{init}))$

    $x_{new_{min\,cost}} \leftarrow \min(dist(x_{near\_neighbor}, x_{new}) + \text{cost}(x_{new}, x_{init}));$

    $x_{near\_neighbor_{newparent}} \leftarrow x_{new_{min\,cost}}$;

    return $x_{near\_neighbor_{newparent}}$;

Fig. 3. RRT* Algorithm

- Test the above algorithms in commonly encountered scenarios in urban driving such as :
  - Overtaking a moving vehicle
  - Avoiding a collision with a vehicle approaching head-on
  - Avoiding a collision with a vehicle approaching from the side at an intersection

VI. Division of Labour and Planned Schedule

TABLE I
PROPOSED SCHEDULE OF THE MILESTONES TO COMPLETE THIS PROJECT

Week	Milestone
1-2	Familiarization with Carla Simulation
3-4	Environment creation with static and dynamic obstacles
4-6	Implementing Dynamic Collision Avoidance
6-8	Implementation of RRT* algorithm
8-9	Integration of algorithms with simulation environments
10-11	Testing and validation
12	Buffer Time
13-14	Code report and documentation

VII. Preliminary Results

Considering the timeline from Table I, which we have made for the project proposal, we are on track. We successfully created a simulation environment in the Carla simulator and integrated it with PyGame to give a better visualization. We created a test case with three static vehicles; the ego vehicle could outmaneuver them. The trajectory of the ego vehicle can be found in Fig.5. The simulation video is found here: **Video**.

Fig. 4. Collision Avoidance by Ego vehicle: **Video**

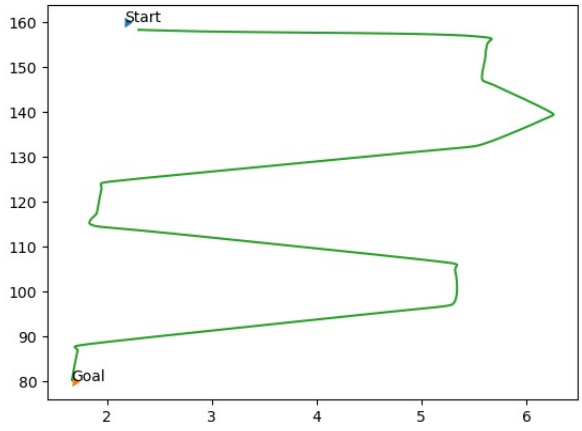

Fig. 5. Trajectory of Ego vehicle

Inferring from the Fig.5 and the Fig.4, on the alternates sides of the lane, obstacle vehicles were preset, and the ego vehicle was able to outmaneuver them using RRT* and collision avoidance.

VIII. Challenges

A. Carla Installation

Getting to understand Carla was time-consuming since we were not familiar with the software initially. Also, Carla requires several dependencies to be installed before it can be used. Installing these dependencies required additional troubleshooting.

B. Tuning PID Controllers

PID controllers are a critical component of motion planning and are used to control the steering, throttle, and braking of a vehicle. Tuning PID controllers involves adjusting the proportional, integral, and derivative gains to achieve optimal performance. PID controllers had to be tuned to ensure that the vehicle moves smoothly and accurately along the desired

trajectory while also responding quickly to changes in the environment. Finding the right balance between stability and responsiveness was challenging and required multiple iterations of tuning.

C. Bounding Radius for Collision Avoidance

Collision avoidance is a critical component of motion planning, and finding the right bounding radius is essential to ensure that the vehicle can navigate safely in its environment. The bounding radius for collision avoidance can have a significant impact on the performance of your vehicle. We had to balance the need for safety with the need for optimal performance to ensure that the motion planning system is effective.

ACKNOWLEDGMENT

We are deeply grateful to Dr. Flickinger for offering us the chance to work on such a remarkable project. His extensive knowledge and experience in the field will be of immense value to us as we navigate the challenges of this project. We look forward to his continuous support and valuable insights that will shape the direction and success of our work.

REFERENCES

[1] Peralta, F.; Arzamendia, M.; Gregor, D.; Reina, D.G.; Toral, S. A Comparison of Local Path Planning Techniques of Autonomous Surface Vehicles for Monitoring Applications: The Ypacarai Lake Case-study. Sensors 2020, 20, 1488. https://doi.org/10.3390/s20051488

[2] Hart, P., Nilsson, N., Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100–107. https://doi.org/10.1109/tssc.1968.300136

[3] Y. K. Hwang and N. Ahuja, "A potential field approach to path planning," in IEEE Transactions on Robotics and Automation, vol. 8, no. 1, pp. 23-32, Feb. 1992, doi: 10.1109/70.127236.

[4] Sertac Karaman, Emilio Frazzoli.;Sampling-based Algorithms for Optimal Motion Planning. https://doi.org/10.48550/arXiv.1105.1186.

[5] Janson L, Schmerling E, Clark A, Pavone M. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. The International Journal of Robotics Research. 2015;34(7):883-921. doi:10.1177/0278364915577958

[6] S. Taheri, "An investigation and design of slip control braking systems integrated with four wheel steering," Ph.D. dissertation, Clemson University, 1990.

[7] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo and S. Thrun, "Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing," 2007 American Control Conference, New York, NY, USA, 2007, pp. 2296-2301, doi: 10.1109/ACC.2007.4282788.

[8] D. Wilkie, J. van den Berg and D. Manocha, "Generalized velocity obstacles," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp.5573-5578, doi:10.1109/IROS.2009.5354175.

[9] Dosovitskiy, Alexey et al. "CARLA: An Open Urban Driving Simulator." ArXiv abs/1711.03938 (2017).

[10] B. aden M. Ca . Z. Yong, D. Yershov and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," in IEEE Transactions on Intelligent Vehicles, vol. 1, no. 1, pp. 33-55, March 2016

[11] B. Nordell. Trajectory planning for autonomous vehicles and cooperative driving. Examensarbete Inom Elektroteknik, Advanced Niva, 30 HP Stockholm, Sverige 2016, 2016.

[12] Samak, Chinmay and Samak, Tanmay and Kandhasamy, Sivanathan, "Control Strategies for Autonomous Vehicles" in Chapter 02 of the book Autonomous Driving and Advanced Driver-Assistance Systems (ADAS)

APPENDIX

- Simulation Setup - Loahit, Jinesh
- Controller Implementation - Anoushka
- Trajectory Generation - Anoushka
- Collision Avoidance - Jinesh
- RRT* Algorithm - Loahit
- Report & Presentation- Anoushka, Loahit, Jinesh