Local Path Planning

CSCE Summer Intern Project for Undergraduate Students

Path Planners
Syed Mustafa

Supervisor: Dr. Song
Department of Computer Science
Texas A&M University

July 4, 2023

Contents

Figure 1: CARLA simulator

Abstract

This report goes over my attempt at the configuration generation of the vehicle to follow a certain path. Using CARLA simulator I modeled the car as a 2D-Bicycle and applied the path tracking algorithm of Pure-Pursuit.

1 Non Holonomic

A non-holonomic vehicle is a type of vehicle that has constraints on its motion, particularly in terms of its velocity and steering capabilities. Unlike holonomic vehicles, which can move in any direction and change their orientation independently, non-holonomic vehicles have limitations on their movement due to mechanical or physical constraints. For example, a car is a non-holonomic vehicle because it can only move forward or backward and change its direction by turning its wheels. It cannot move sideways or rotate in place without additional maneuvers. The mechanical model used to emulate a vehicle include:

1. 4 wheel Ackerman Steering model
2. Bicycle Model

2 Bicycle Model

For this project, I decided to go with the bicycle model because it was a simplified version of the 4-wheel Ackerman steering model. Figure 2 shows the 2D bicycle Model. L is the wheelbase. θ is the heading angle and δ is the steering angle. In a typical bicycle model, a reference point needs to be chosen. This can be at the center of gravity, the rear axle, or the front axle. Figure 3 shows the different reference points. For the project, the rear axle is chosen as the reference point.
The current state of the bicycle model is given by:

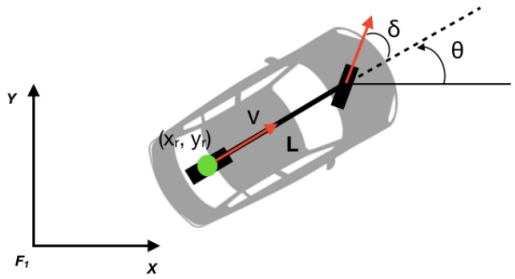


Figure 2: CARLA simulator

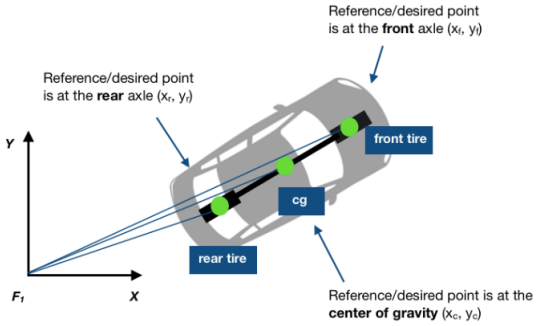


Figure 3: CARLA simulator

$S = \begin{bmatrix} x \\ y \\ \theta \\ \delta \end{bmatrix} \ni (x, y)$ are the coordinates the reference point, θ is the vehicles heading angle and δ is the steering angle.

The inputs of the model are $[v_f, \phi]$ where v_f is the forward velocity and ϕ is the steering rate. To compute the next state of the model we must first compute the first derivative of the state S given be

$$\dot{S} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} v_f \cos(\theta) \\ v_f \sin(\theta) \\ v_f \frac{\tan(\delta)}{L} \\ \phi \end{bmatrix}$$

By looking at Figure 4 we can see that the state change is derived through simple trigonometric derivation.

3 Pure Pursuit Controller

Pure pursuit is a simple control algorithm that tracks the vehicle path. It tracks the path of a vehicle using only path the geometry of the vehicle's kinematics and the reference path. The control algorithm uses a look-ahead distance l_d. It is, as the name would suggest, the distance the vehicle looks ahead to its target point (the point vehicle wants to track). The algorithm computes the steering angle δ using the l_d. Figure 5 shows the pure pursuit paradigm.
The equations are as follows:

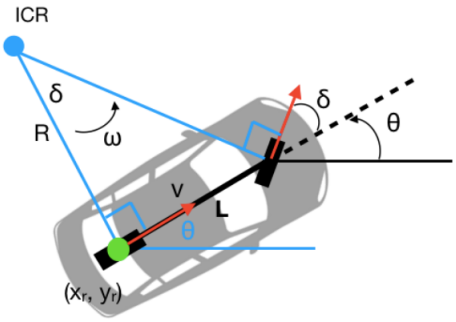


Figure 4: CARLA simulator

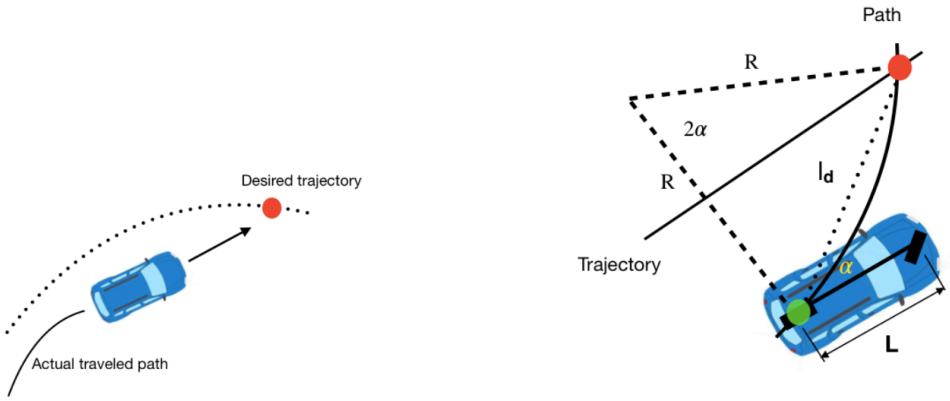


Figure 5: The Pure Pursuit Paradigm

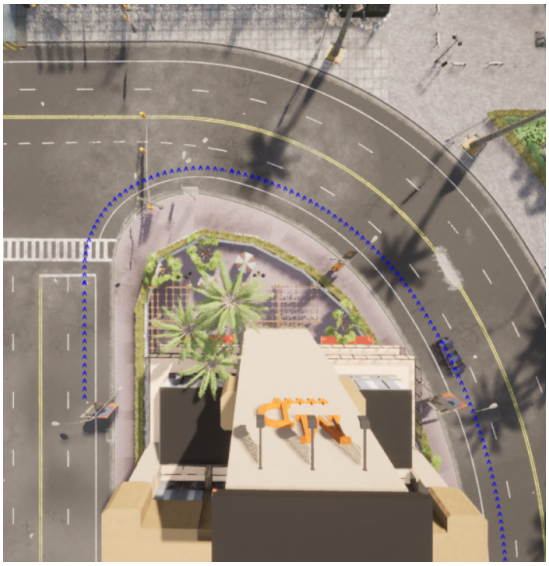
1. $\delta = \arctan(\frac{2L\sin\alpha}{l_d})$
2. $\alpha = \arctan(\frac{t_y - y}{t_x - x}) - \psi \ni (t_x, t_y)$ is the target coordinate, (x, y) is the current coordinate and ψ (yaw) is the angle between the inertial frame and horizontal frame about the z-axis.
3. The pure pursuit is a simple control algorithm, i.e. it does not take dynamics into account. Thus if l_d is tuned for smaller velocities then the vehicle would behave very aggressively for larger velocities.
4. A solution for this problem is basing the look ahead distance on velocity, i.e. $l_d = K_{dd} * v_f \ni v_f$ is the forward velocity and K_{dd} is a constant.

The algorithm can be written as the following steps:

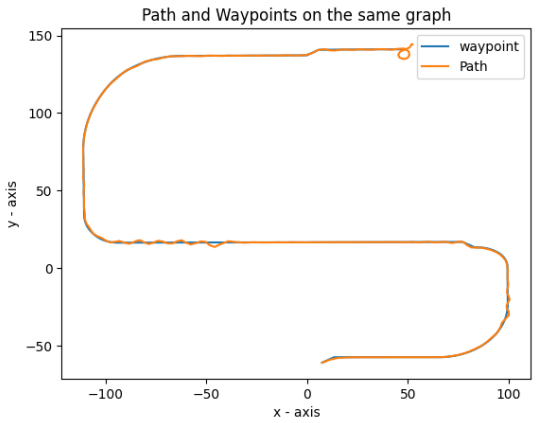
1. Get the current state of the vehicle (including the yaw angle).
2. Get the target point.
3. Compute l_d using the formula $l_d = K_{dd} * v_f$.
4. Compute α using the formula $\alpha = \arctan(\frac{t_y - y}{t_x - x}) - \psi$.
5. Compute the steering angle, using the above-stated formula.
6. Apply the steering angle to the control of the vehicle.

4 Simulation Result

The code is present on GITHUB https://github.com/MMustafa01/Local_Path_Planning/blob/main/CARLA_Codes/Control_pure_persuit.ipynb.
What I noticed from the simulations is that for the most part, the tracking algorithm follows the given path. However, in some instances, the car starts oscillating off the path. I think the reason for this is that when the vehicle speed increases so does the lookahead distance. Due, to this the car starts oscillating.
Next step: Right now the path was given through the global planner that CARLA provided. The next step would be to implement a planner that would give a path from the start location to the end location.



(a) Simulation on Carla



(b) The desired path (blue) vs the actual path (orange)

Figure 6: The simulation Results