UNIVERSITY OF OKLAHOMA GRADUATE COLLEGE

Reinforcement Learning-based Adaptive Prediction Horizon for Frenet Frame Autonomous Vehicle Path Planning

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

In partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

DAVID YUN

Norman, Oklahoma

2025

Reinforcement Learning-based Adaptive Prediction Horizon for Frenet Frame Autonomous Vehicle Path Planning

A THESIS APPROVED FOR THE

SCHOOL OF AEROSPACE AND MECHANICAL ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Bin Xu, Chair

Dr. James David Baldwin

Dr. Dong Zhang

Table of Contents

List of Tables

List of Figures

Abstract

Autonomous vehicles (AV) are gaining attention as technology advances, and the demand is on the rise. With its popularity, it is important to prioritize the safety of the people. The main concern with autonomous driving is the safety and ability to stay on course. This research presents a framework for autonomous vehicle trajectory planning by using three methods. The first was using a fixed prediction horizon. The next method was using an adaptive prediction horizon. The final method and the new approach are using Reinforcement Learning (RL) with Proximal Policy Optimization (PPO) to pick the best prediction horizon value in the Frenet Frame Method (FFM). Our new approach investigates an adaptive trajectory planning framework for autonomous driving that takes the fixed path planning code and is enhanced to adaptively adjust the pre-horizon based on the distance to the obstacle in the path. This approach to the problem allows the prediction horizon to no longer be fixed. It allows the vehicle to detect the barriers and use the horizon value associated with the distance to the vehicle. This modification enables the planner to respond more intelligently to environmental constraints by shortening the horizon in proximity to obstacles and lengthening it in open road segments, thereby improving safety and trajectory smoothness. The next step was to implement RL. For this segment, the PPO algorithm from the Stable Baseline3 RL library was implemented within a custom gym environment. In this environment, the agent consists of discrete candidate prediction horizons, and it learns to select the optimal horizon by minimizing the reward function that minimizes tracking error, steering angle, and computation time. Given the action of different prediction horizons, PPO would select the optimal value for each criterion. The results showed the PPO was able to select the best value based on the reward. Showcasing that our research showed better results than the fixed and adaptive prediction horizons.

1 Introduction

The rapid advancement of technology and increasing urbanization have created a pressing need for innovative solutions to modern transportation challenges. AVs can help revolutionize society and have become increasingly popular due to the potential increase in safety and efficiency [1], [2]. One of the main challenges in the AV industry is local path planning, where the vehicle must generate an optimal trajectory that ensures safety, efficiency, and the comfort of the driver while avoiding obstacles [3], [4]. Path planning trajectory will be the focus of the paper, and the different methods for achieving the vehicle's path. In many studies, the path of the vehicle was conducted on straight highways and urban roads using available datasets [5], [6]. In this research, experiments have been done, and we generated our path data using the modified 2015 Nissan Leaf S. The Nissan Leaf was modified to make it autonomous. Relying on the GPS to create the waypoints of the path the vehicle took.

Autonomous cars are still in development; therefore, many experiments need to be conducted. There will be new situations that the car will encounter in a new environment, adverse weather, avoiding obstacles, traffic, and lanes [7], [8]. In this research, the problem is that many popular path planning methods, such as sampling-based approaches and optimization methods like the Model Predictive Control, face significant difficulties, especially where the roads have a lot of obstacles [9]. The core objective of these approaches is to determine an optimal trajectory that ensures passenger comfort, prevents collisions, and safely guides the vehicle toward its target location. Simultaneously, the underlying planning framework prioritizes lane-centering or adherence to a designated reference path, carefully accounting for both static obstacles encountered along the route.

Many trajectory planning methods have been proposed over recent years, from graph-based search algorithms to advanced RL techniques proposed. Polynomial-based methods, such as quintic polynomial trajectory planning, have been studied due to their simplicity and computation efficiency. The papers leveraged quintic polynomial planning to generate smooth lane changing trajectories, while B-spline and Bézier spline-based methods offer flexibility and comfort [10], [11]. However, these methods often struggle in an environment where there are a lot of obstacles in the path. Optimization methods such as Model Predictive Control (MPC) gained popularity due to their ability to consider the vehicle dynamics and constraints[12]. MPC can be used in trajectory planning and obstacle avoidance. It demonstrated effectiveness for precise and safe maneuvers in the autonomous driving context. RL has also been incorporated into MPC [13]. However, MPC can be very complex and time-consuming, especially when precise dynamics and constraints are added to the vehicles. RL can adapt and choose the best action in real time. There are many different approaches to Deep RL, such as Proximal Policy Optimization (PPO), Deep Deterministic Policy Gradient (DDPG), and Soft Actor-Critic Planning and decision making. The results for these methods showed the effectiveness in various autonomous driving tasks, such as predicting behavior, avoiding obstacles, and decision making [14], [15]. The RL methods offer robust, good results; however, they require extensive training and data.

Frenet Frame Method (FFM) have become one of the popular approaches for trajectory planning in autonomous driving. FFM has become popular due to its simplicity and effectiveness in decoupling the vehicle's longitudinal and lateral movements relative to a predefined reference path. Werling et al. [16] proposed the foundation of the trajectory generation in the Frenet Frame. It allowed a simplified representation of motion by clearly separating the lateral and longitudinal movements. In this context, any point along the reference path is represented using the Frenet

coordinate system, defined by tangential and normal vectors, denoted as t_r and n_r, respectively. Based on this representation, the two polynomials *s* and *d* are formulated to independently model the vehicle's forward motion and side-to-side maneuvers. Since the paths are defined relative to the lanes, FFM can handle lane change maneuvers and lane following. FFM also avoids the obstacles more easily and has better computation efficiency. Therefore, in many methods, the FFM is chosen over the traditional Cartesian coordinate. With FFM methods, much other research has been made, such as an enhanced Frenet-based planner. Xu et al [17] presented a planner that considers road curvature and obstacles, effectively addressing challenges with complex driving scenarios. Moreover. Zieggler et al.[18] In many of the FFM, the prediction horizon chosen was fixed and did not show the results of the tracking error, steering angle, and computation time of the planned path of the vehicle [18], [19].

FFM is a well-known model for tracking control problems by which human driving is emulated. In this method, the lateral and longitudinal coordinates of the reference line are separately described based on the Frenet coordinate. Decomposing lateral and longitudinal coordinates facilitates the mathematical description of curved paths. Hence, the FFM coordinate is preferred to the Cartesian coordinate in many studies since describing curves is straightforward due to the independent description of coordinates. In this regard, any arbitrary point on the (possibly curved) reference line is described by the tangential and normal vectors in the Frenet coordinate, namely t_r and n_r. Subsequently, two polynomials, namely *s* and *d*, are constructed to independently outline the ego vehicle's longitudinal and lateral maneuvers, respectively. J. Huang et al. [5] employed FFM for path planning while avoiding static obstacles. The study in [5] does not consider dynamic obstacles, and the computation time of the proposed method is not reported. Similarly, the research in [15] finds the collision-free path while following a reference line using

FFM. Both works in [5], and [15] demonstrate the effectiveness of their methodologies through simulation environments and lack experimental validations.

FFM and MPC are popular when it comes to trajectory planning, with RL on the rise, a combined method is often proposed. In many research PPO has been chosen as the model for RL. PPO was used to choose the best trajectory when combined with MPC [20], [21]. However, these systems still rely on solving optimization problems at each timestep, which can become computationally intensive. In contrast, the Frenet coordinate system simplifies the planning, which allows for simpler trajectory generation compared to MPC. By combining the FFM with RL, the system gains both computational efficiency and adaptive intelligence. The RL agent can learn to select the optimal prediction horizon based on real-time conditions such as road curvature or vehicle speed, without the complicated optimization at each timestep. Also, since the planning occurs relative to the structured reference path, the learned policies tend to be more generalized, allowing the training to be more efficient. Therefore, in this research, the RL-PPO method was chosen for this research using the FFM.

Most of the path planning studies in the existing literature do not report the computation time, tracking error, or the steering angle. It also does not state the prediction horizon that was chosen for a specific purpose. Many of the RL algorithms have only been studied using the MPC, and not much research has been done using the FFM, and if it is done, the prediction horizon value is fixed. Hence, the present work proposes an FFM-based path planning aiming to compare the results of the static, adaptive, and RL approaches in picking the prediction horizon based on the metrics of computation time, tracking error, and steering angle. Also, our data has been collected for the roads that are straight and curved. The contributions to our work are as follows:

- Implemented and evaluated three different prediction horizon selection strategies -fixed, adaptive, and reinforcement learning- in the Frenet Frame path planner for autonomous driving
- An autonomous vehicle is used to collect GPS data points using a 2015 Nissan Leaf S electric vehicle.
- Introduced a PPO-based RL agent that selects the prediction horizon based on real-time observation such as computation time, tracking error, and steering angle.
- Designed a reward structure that prioritizes one performance metric at a time – tracking error, computation time, and steering angle – allowing the agent to specialize its behavior depending on the training objective.

The remaining sections of the thesis are structured as follows. Section 2 will be a problem description and the model of the environment. Section 3 will showcase the description of the problem and talk about the environment of the FFM, RL, and PPO. The evaluation metric will show how the computation time, tracking error, and steering angle were calculated. The experimental configuration of the AV using a 2015 Nissan Leaf S will also be discussed. In Section 3, the analytical results of the three different methods will be discussed and how they compare against each other. Section 4 will be the conclusion of this research and future work of the simulation.

2 Environmental models and methods

2.1 Problem Description

In this paper, the local path planning of an AV is presented using the FFM, and the path data was collected using data from Nissan Leaf S. The data is gathered through a mounted camera on the vehicle. AV relies on path-planning algorithms to navigate through roads, avoid obstacles, and stay on track. One of the main challenges in path planning is determining the optimal prediction horizon, based on the road. The prediction horizon defines how far into the future the vehicle should plan its trajectory. The value of the prediction horizon can significantly impact the computation time, the accuracy of the vehicle to the path, and the smoothness of the vehicle. The prediction horizon also determines the trajectory of the path planning algorithm. Therefore, the three criteria are needed to determine the optimal prediction horizon: computation time, tracking error, and steering angle. With these three criteria, different methods of selecting the prediction horizon will be used. The first method is by using a fixed horizon. The second is the adaptive approach. The final method will be using Reinforcement Learning with PPO from StableBaselines3. This study evaluates three different prediction horizon selection strategies. Comparing the overall performance in the three criteria will help demonstrate that RL-based prediction horizon selection achieves the best overall performance. The proposed approach enhances real-time decision-making in autonomous driving and allows a more adaptive approach to path planning.

2.2 Environmental Model in Path Planning

The environmental model captures key elements in autonomous driving and path planning. In this environment, the autonomous car is demonstrated in an x-y coordinate system. In Figure 1, the road geometry is shown. The x and y coordinates represent the Cartesian coordinates of each

point, while θ denotes the vehicle's heading angle. The speed, v, is the velocity and is denoted by the vector pointing in the direction of the motion of the vehicle. The dynamic equation governing the motion of both the vehicle and surrounding vehicles is given by

$$z_{k+1} = z_k + f(z_k, u_k)dt, \tag{1}$$

where the state vector at a time step k is defined as

$$z_k = [x_k, y_k, \theta_k, v_k]^T \tag{2}$$

And the function $f(z_k, u_k)$ describes the system dynamics as

$$f(z_k, u_k) = [v_k cos\theta_k, v_k sin\theta_k, \omega_k, a_k]^T \tag{3}$$

where, $u_k = [a_k, \omega_k]$ consists of the linear acceleration a_k and angular velocity ω_k while dt represents the time step [22]. Throughout the experiment, environmental information has been received. For this research, the road waypoints and lane boundaries were collected to help provide information to the car, which is essential in trajectory planning. It is used to compute the distance to the next closest waypoint and the difference between the vehicle heading and the reference heading. This model is important as later in the FFM, the Cartesian coordinates will be converted to the Frenet Frame.

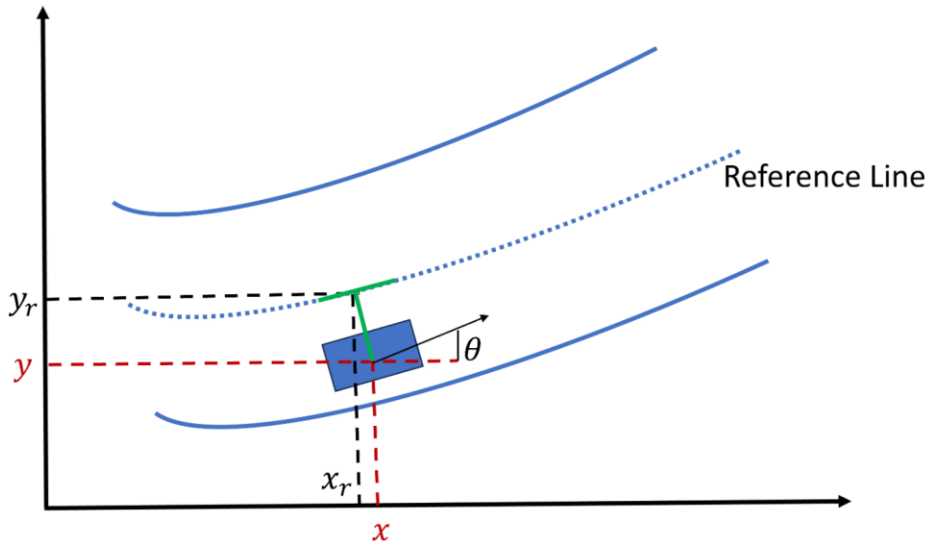

Figure 1: Reference Path and Vehicle Positioning.

2.3 Frenet Frame Method (FFM)

The FFM is widely used in autonomous vehicle path planning, and it simplifies trajectory generation by decoupling longitudinal and lateral motion [16]. Unlike Cartesian coordinates, which use x and y, the FFM represents a vehicle motion along a curved reference path using s and d. Figure 2 shows the difference between the Cartesian coordinate and the FFM coordinate. The *s* is the longitudinal displacement, and *d* is the lateral displacement. The longitudinal displacement measures how far the vehicle has traveled along the reference path, while the lateral displacement measures how far the vehicle deviates from the reference path. This makes it easier to describe complex paths, curved roads, and highway scenarios. In the FFM, the *s* and *d* are perpendicular to each other at every point in the reference line, as shown in Figure 2b. Figure 2a shows the Cartesian coordinates, and unlike the FFM, Cartesian coordinates provide an absolute position in a fixed reference system.

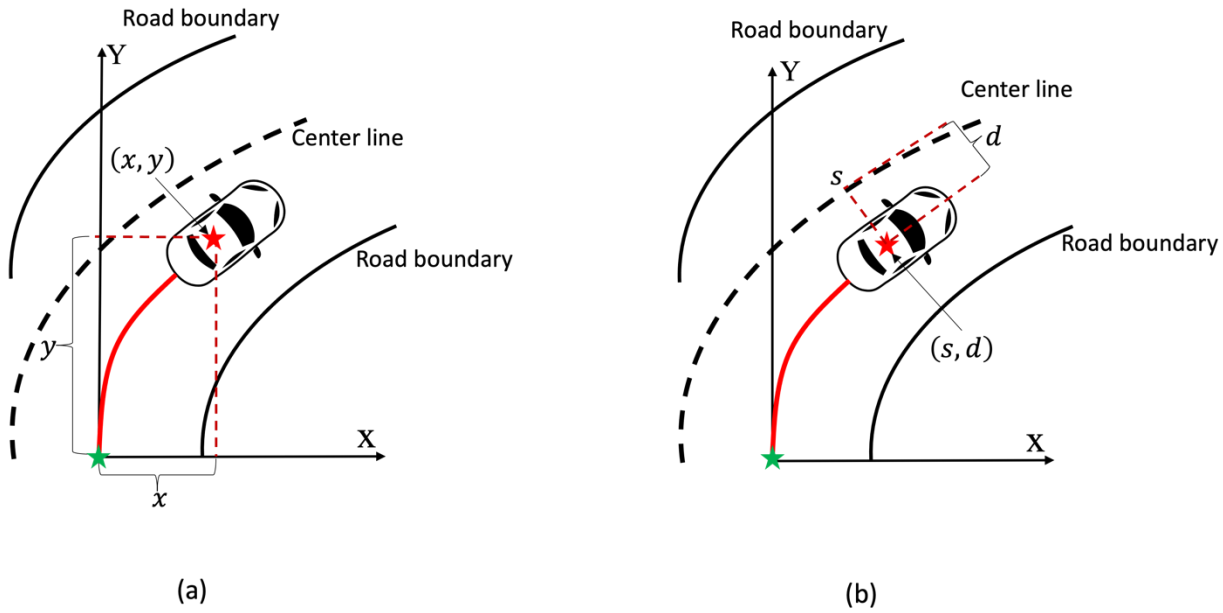

Figure 2: a) Cartesian coordinate, and b) FFM coordinate.

When getting the coordinate in the Frenet coordinates in *s* and *d,* the coordinates are then converted to the Cartesian coordinates by the following equations [22].

$$x_k = x_r - d_k \sin\theta_r, \tag{4}$$

$$y_k = y_r + d_k \cos\theta_r, \tag{5}$$

$$\theta_k = \arctan\left(\frac{d'}{1 - k_r d}\right) + \theta_r, \tag{6}$$

$$v_k = \sqrt{(\dot{s}(1 - k_r d))^2 + (\dot{s}d')^2}, \tag{7}$$

$$a_k = \ddot{s}\frac{1 - k_r d}{\cos\Delta\theta} + \frac{\dot{s}^2}{\cos\Delta\theta}\left[d'\left(k_k\frac{1 - k_r d}{\cos\Delta\theta} - k_r\right) - k_r' d - k_r d'\right], \tag{8}$$

$$k_k = (d'' + (k_r' d + k_r d') \tan \Delta\theta) \frac{\cos^2 \Delta\theta}{1 - k_r d} + k_r. \tag{9}$$

Figure 3 shows how the Frenet coordinates are converted to Cartesian and shows how an autonomous vehicle plans its motion in Frenet coordinates and then converts the chosen trajectory into Cartesian coordinates for actual movement.

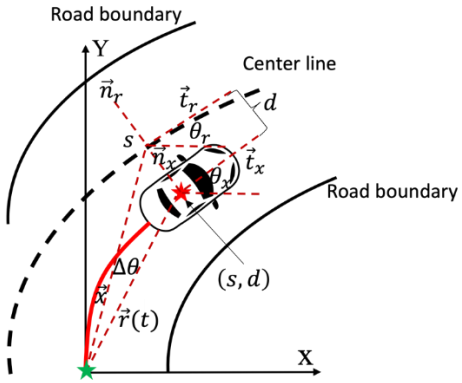

Figure 3: Frenet Frame to Cartesian coordinate conversion.

Local path planning is done in the Frenet coordinates. Local path planning focuses on short-term trajectory generation for an autonomous vehicle. In local path planning, the vehicle continuously adjusts its trajectory based on real-time sensor data, accounting for obstacles, road curvature, and speed constraints. To achieve this in the FFM, *s* and *d* can be described by the following quintic and quartic polynomials. Where equation (10) is the quintic polynomial for *d* and equation (11) is the quartic polynomial for *s* [22].

$$d(t) = c_{d_0} + c_{d_1} t + c_{d_2} t^2 + c_{d_3} t^3 + c_{d_4} t^4 + c_{d_5} t^5, \tag{10}$$

$$s(t) = c_{s_0} + c_{s_1} t + c_{s_2} t^2 + c_{s_3} t^3 + c_{s_4} t^4, \tag{11}$$

where t in this equation represents time. In the equation, we see that d and s are independently characterized by the lateral and longitudinal motions of the vehicle. During path planning, a new trajectory starts, which starts a new time. The start is denoted as t_s and ends as t_e under the assumption that t_s and t_e are known, and $t_e - t_s = T$. T is how long the trajectory took to plan, and the equation shows how it is calculated at each time step. During this time, many trajectories are made. However, only one gets chosen as the best and is selected according to the cost function. The cost function helps determine the trajectory that provides the best path for the vehicle. For each trajectory segment, multiple candidate trajectories are generated by varying the final lateral position $d(t_e)$ and final longitudinal velocity $\dot{s}(t_e)$. Therefore, in local path planning, all the trajectories will have the same initial condition at the start, however, each trajectory will lead to different endpoints. The precise form of these trajectories is determined by solving the coefficients of polynomial equations, which define the motion over time. Once these equations are computed, the vehicle's position $s(\tau)$ and $d(\tau)$can be evaluated for $\tau \in [t_s, t_e]$. By incorporating derivatives at the initial and final points, the system ensures that the generated trajectory satisfies smoothness constraints and optimizes movement based on predefined cost functions. Using the equations from (10) and (11), the following equations are made by applying the boundary conditions at the start and end times and constructing a linear system to solve for the polynomial coefficients that define the lateral and longitudinal trajectories [22].

For the lateral motion:

$$\begin{bmatrix} d(t_s) \\ \dot{d}(t_s) \\ \ddot{d}(t_s) \\ d(t_e) \\ \dot{d}(t_e) \\ \ddot{d}(t_e) \end{bmatrix} = \begin{bmatrix} 1 & t_s & t_s^2 & t_s^3 & t_s^4 & t_s^5 \\ 0 & 1 & 2t_s & 3t_s^2 & 4t_s^3 & 5t_s^4 \\ 0 & 0 & 2 & 6t_s & 12t_s^2 & 20t_s^3 \\ 1 & t_e & t_e^2 & t_e^3 & t_e^4 & t_e^5 \\ 0 & 1 & 2t_e & 3t_e^2 & 4t_e^3 & 5t_e^4 \\ 0 & 0 & 2 & 6t_e & 12t_e^2 & 20t_e^3 \end{bmatrix} \begin{bmatrix} c_{d_0} \\ c_{d_1} \\ c_{d_2} \\ c_{d_3} \\ c_{d_4} \\ c_{d_5} \end{bmatrix} \tag{12}$$

For the longitudinal motion:

$$\begin{bmatrix} s(t_s) \\ \dot{s}(t_s) \\ \ddot{s}(t_s) \\ \dot{s}(t_e) \\ \ddot{s}(t_e) \end{bmatrix} = \begin{bmatrix} 1 & t_s & t_s^2 & t_s^3 & t_s^4 \\ 0 & 1 & 2t_s & 3t_s^2 & 4t_s^3 \\ 0 & 0 & 2 & 6t_s & 12t_s^2 \\ 0 & 1 & 2t_e & 3t_e^2 & 4t_e^3 \\ 0 & 0 & 2 & 6t_e & 12t_e^2 \end{bmatrix} \begin{bmatrix} c_{s_0} \\ c_{s_1} \\ c_{s_2} \\ c_{s_3} \\ c_{s_4} \end{bmatrix} \tag{13}$$

Equations (12) and (13) shows a matrix that needs to be solved for the polynomial coefficients that define the vehicle trajectory in the FFM. Equation (12) specifically solves for the quintic polynomial coefficients that describe how the vehicle moves side-to-side or the lateral position relative to the centerline. Therefore, the boundary condition is the initial and final position, velocity, and acceleration. The vehicle has a zero lateral velocity and acceleration at the end for smoothness. Equation (12) is used to generate lane following trajectories. Equation (13) solves quartic polynomial coefficients. It defines how the vehicle moves forward along the reference line or the longitudinal position over time. In this equation, the boundary conditions are the initial position, initial velocity, final velocity, initial acceleration, and final acceleration (zero). Equations (12) and (13) provide the complete 2D path of the vehicle in the Frenet Frame. It will also show how far the vehicle will go over time and how much the vehicle will shift laterally to change or stay in the lane. Once these are computed, the equation allows the system to evaluate and optimize

the trajectory based on the cost function. The cost function chooses the best trajectory path to be chosen.

The cost function can be found by solving the equation below

$$C = C_d + C_s, \tag{14}$$

$$C_d = K_j \sum_t \ddot{d}(t)^2 + K_t T + K_d d(T)^2, \tag{15}$$

$$C_s = K_j \sum_t \ddot{s}(t)^2 + K_t T + K_d (v_d - \dot{s}(t))^2, \tag{16}$$

where C_d is the lateral cost function and C_s longitudinal cost function. K_j , K_t, and K_d are weighting parameters [22]. The cost function is designed to reduce both lateral and longitudinal jerks, minimize deviation from the reference line, and align the vehicle's speed with the target velocity. Among the candidate paths, the one with the lowest cost is chosen to be followed over the next *T* seconds.

2.4 Environmental Model in Reinforcement Learning

RL is a method of dynamic programming, where the algorithm learns to act based on the reward and punishment system [23]. It is a type of machine learning that focuses on how an agent learns to make the best decision by interacting with the environment. Reinforcement learning is used because, unlike supervised learning, which relies on input-output pairs, RL learns through trial and error. Learning slowly over time based on the punishment and rewards. At each timestep, the agent observes the current state of the environment, takes an action based on a policy, and then receives

a reward along with a new state. This is continuously repeated for a certain number of timesteps, and over time, the agent learns a policy that will maximize the cumulative reward. RL allows the agent to find new effective strategies for complex decision-making tasks automatically. Figure 4 shows the main components and how the general process goes in reinforcement learning.

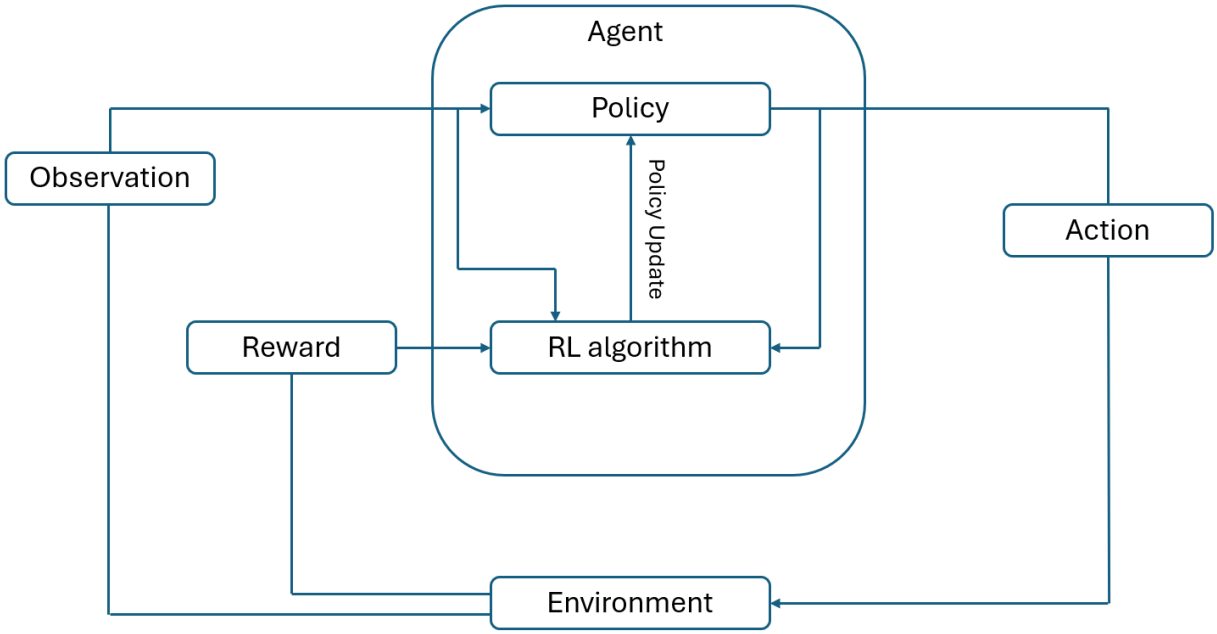

Figure 4: Important Components in RL.

The environment in RL is everything outside the agent and reacts to the agent's actions. The environment takes in the action and outputs the reward and observation back to the agent. In this research, the goal is to train the autonomous vehicle to minimize the computation time, tracking error, and steering angle by learning an optimal policy through reinforcement learning with a cost-based reward structure**.** The environment includes the simulation of the vehicle, road layout, obstacles, and all external factors the agent interacts with. This environment uses a cubic spline planner to generate the global path and continuously updates based on the agent's decisions. In the

environment, the agent will choose a prediction horizon, and the Frenet planner will use the chosen prediction horizon and generate the trajectory of the vehicles path. Therefore, the environment represents real-world driving simulations. The environment is a custom simulation framework built around the Frenet Frame code. The Frenet Frame code simulates the ego vehicle's motion. Waypoints obtained from the GPS of the Nissan Leaf define the reference path, and obstacles are placed in the path to mimic real-world driving scenarios. In the environment, there are custom rewards based on tracking errors, computation time, and steering angle that are output.

The first step in RL is the observation. The environment sends an observation to the agent. The observation is the input to the agent at each timestep, which takes in the current state of the planner's performance. In this research, the observation is a 3-dimensional vector containing tracking error, computation time, and steering angle. However, to study the effects of prioritizing each metric independently, only 1-dimension was used at a time to prioritize the specific performance. The metric is used to help perform performance metrics, allowing the agent to learn to minimize the error.

The action is defined as a set of possible choices available to an agent as it interacts with the environment. In this research, the action component is defined as a discrete set of possible prediction horizons that the agent can choose from. The available choices are values [10,11,12,13,14,15,16,17,18,19,20, 21, 22, 23, 24, 25]. The available choices, ranging from 10 to 25, represent different prediction horizon values. Shorter horizons allow the vehicle to react quickly but may result in less optimal long-term plans. Longer horizons enable better foresight but increase computation time. The PPO agent learns to select the most appropriate horizon based on the driving situation. Therefore, at each timestep, the agent will choose one of these values at

random. Since the agent is having a different action at every timestep, the reward value will be different. The policy will then choose the next action based on the rewards.

The reward function is very important in RL. They provide feedback to the agent based on the action that was performed and guide the agent towards more desirable behavior. If the selected action was not good, the reward will state that it was not good by giving a big value. Importantly, the PPO agent is trained using negative values. This is done to maximize the reward, which is equivalent to minimizing the penalties.

$$reward = -cost \tag{17}$$

where cost is

$$cost = w_1 \cdot d + w_2 \cdot |\delta| + w_3 \cdot t_{comp} \tag{18}$$

Equation (17) is used to minimize the reward and get the lowest value as its maximum. This allows PPO to choose the horizon that gives the lowest computation time, tracking error, and steering angle. In equation (18) w_1, w_2, w_3 are the weights that determine the importance of each metric (d, δ, t_{comp}). d, δ, t_{comp} is the tracking error, steering angle, and computation time, respectively. The steering angle is inside an absolute value, as the value becomes negative when the vehicle turns left and positive when turning right. Since we are looking only at the angle, the absolute value is added. Then, PPO computes the cumulative reward using equation (17). If we want to prioritize one metric, for example, tracking error, the constants for steering angle and computation time will be zero. This allows the steering angle and computation to be not considered in the rewards, instead allowing the vehicle to prioritize staying on the track.

The agent is also another component that is important in RL. In this research, the agent is the PPO itself. PPO learns a policy to select the most suitable prediction horizon given the input of the

observation. Inside the agent is the policy and the learning algorithm, which maximizes the reward together. The policy takes in the observation and creates a probability distribution of the action that will be selected. Then PPO is used to carry out the policy. PPO updates the policy using the clipped objective function to ensure stable and conservative policy improvement. The learning algorithm looks for the best trajectory that will provide the best cumulative reward over time.

In the training aspect of the RL, the PPO learns to select the best prediction horizon for Frenet path planning based on the key performance metrics as mentioned before. The training environment is kept the same, where the path and the obstacle placement are the same, only the prediction horizon will change. The training is looped where it picks a prediction horizon, runs the Frenet planner, collects the values of the metric, calculates the reward, and updates the policy based on the rewards. By iterating this step over many timesteps, the PPO agent learns to generalize the best prediction horizon based on the reference path and what metrics are being prioritized.

After the training loops, Stable_Baseline3 (SB3) has a library that logs the statistics from a training step. SB3 is a Python library that provides ready-to-use implementations of popular RL algorithms In Figure 5, the training metrics for the PPO model are shown. This is used to see the training metrics and see if the training is learning over time. The statistics are logged and can be visually seen using TensorBoard.

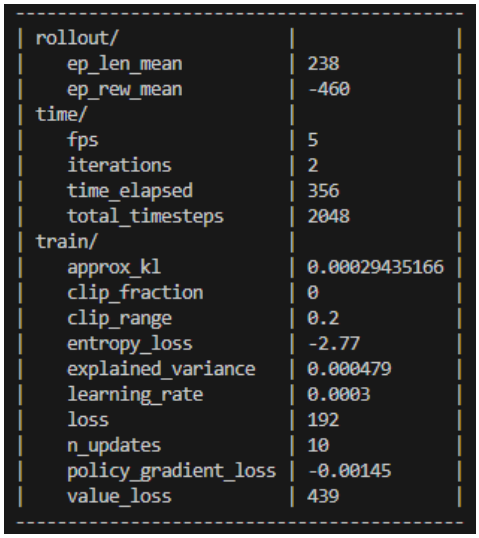

Figure 5: Training with PPO Evaluation.

This output summarizes how well the agent is learning over time — tracking performance, stability, and optimization progress. To get these statistics, RL goes through the process as seen in Figure 4. After training, a test is done to see if the RL was able to choose the optimal prediction horizon.

During testing, the PPO agent is evaluated within a separate testing environment. The testing environment is the same as the training (same road and obstacle), but does not use the training data to ensure unbiased evaluation. This means the agent can no longer update its policy during testing, instead, it applies the learned strategy to select the prediction horizon in various trajectory planning scenarios. Therefore, after evaluating the model and testing, PPO chooses the prediction horizon that had the best results during training.

2.5 Evaluation Metrics

The evaluation metric is needed to provide quantitative evidence of how well each prediction horizon strategy (Fixed, Adaptive, and PPO) performs. The performance will be measured by

calculating the computation time, tracking errors, and steering angles. First, the computation time is simply how long it takes to generate and select a valid trajectory for a given prediction horizon. Equation (19), shows the equation that is used to measure computation time

$$t_{comp} = t_{end} - t_{start} \tag{19}$$

Where t_{start} timestamp (in seconds) recorded right before the trajectory planning or optimization process begins. t_{end} is the timestamp recorded immediately after the process finishes. This metric is important for real-time performance. A shorter computation time means that the system is more responsive and capable of choosing a trajectory quickly. A longer computation time will indicate a slower response time, which is not ideal in a situation where a quick reaction is needed. Without computation time, a prediction horizon may be chosen but is too slow in practice.

The next evaluation metric is tracking error. The tracking error shows how closely the vehicle follows the reference path. At each timestep, the reference position is compared to the actual vehicle position. With the position of the vehicle obtained, the Euclidean distance can be used to calculate the tracking error. Equation (20) shows how the tracking error can be calculated

$$d = \sqrt{(x_r - x_a)^2 + (y_r - y_a)^2} \tag{20}$$

Where x_r and y_r is the reference position and x_a and y_a is the actual position of the vehicle. Lower tracking error indicates that the AV is following the reference path, which is essential for safety and accuracy. Without calculating the tracking error at each timestep, we cannot verify whether the planned trajectory results in a path that follows the reference path.

The final evaluation metric is the steering angle. The steering angle is the angle, in degrees, between the vehicle's front wheels and its longitudinal axis. It determines how sharply the vehicle turns. In the equation (21) shows Ackermann's equation

$$\delta = arctan\left(\frac{L}{R}\right) \tag{21}$$

L is the wheelbase (distance between front and rear axles [24]. The wheelbase will be a constant of 2.7 meters. This is the wheelbase obtained from the Nissan Leaf's wheelbase. Where R is the turning radius of the vehicle. Since the turning radius is not given, a new equation can be used to solve for *R*, using the equation below to calculate the curvature. The curvature is the inverse of the turning radius.

$$k = \frac{1}{R} \tag{22}$$

Since the system gives the curvature, the curvature can be used to calculate the turning radius in equation (22) and ultimately find the steering angle in equation (21) [24]. The value calculated is in radians and therefore must be converted to degrees. After that, the final steering angle can be found. This allows the steering angle to be calculated at each time step. The steering angle calculation is important as it reflects the smoothness and stability of the trajectory. Without the steering angle calculation, the safety or the physical feasibility of the vehicle cannot be measured. All these metrics are needed as they determine the best prediction horizon. In all these metrics, the minimal values are needed for making the quickest trajectory, staying in the reference path, and having minimal jerk. Minimal values are important in the fixed, adaptive, and PPO approaches. It will especially be important in PPO as the three metrics will be used as a reward for reinforcement learning. In RL, the minimal values will be considered good and will be discussed more in the next section.

2.6 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a policy gradient algorithm known for its ease of use and reliability in learning RL tasks. PPO aims to improve its policy while avoiding large updates that can crash the RL and cause instability during training [23]. More information about the policy can be found in the respective references cited. The basic policy gradient objective is defined in equation (23)

$$\hat{g} = E_t[\nabla_\theta] log \pi_\theta(a_t|s_t)\hat{A}_t \tag{23}$$

where π_θ is the stochastic policy and $\hat{A}_t$ is an estimator of the advantage function at timestep *t*. The estimator $\hat{g}$ is found by differentiating the objective as shown in equation (24).

$$L^{PG}(\theta) = E_t[\nabla_\theta] log \pi_\theta(a_t|s_t)\hat{A}_t \tag{24}$$

Adding multiple gradient steps to this objection can lead to destabilization in the policy updates. Therefore, PPO introduces equation (25) using a clipped surrogate objective, which is seen below

$$L^{CLIP}(\theta) = E_t\big[\min(r_t(\theta)\hat{A}_t\,, clip\,(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big]. \tag{25}$$

where

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{26}$$

Equation (26) is the probability ratio between the new and old policies, and epsilon is a hyperparameter that is constant. This objective prevents r_t from straying too far from 1, thus keeping updates conservative and avoiding destructive policy shifts.

PPO uses the generalization advantage estimation (GAE) and can be used to compute the advantage estimates as seen in equation (27)

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T), \tag{27}$$

Where t specifies the time index in [0, T], within a given trajectory segment (*T)*. The equation can be generalized when $\lambda = 1$. Therefore, we get a simplified equation (28)

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_t + \cdots + \cdots + \gamma\lambda^{T-t+1}\delta_{T-1}, \tag{28}$$

where

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t). \tag{29}$$

The value loss function in equation (30) is used to train the critic,

$$L_t^{VF}(\theta) = \left(V_\theta(s_t) - V_t^{target}\right)^2 \tag{30}$$

The entropy bonus is used in PPO to encourage exploration

$$S[\pi_\theta](s_t) = -\sum_a \pi_\theta(a|s_t) log \pi_\theta(a|s_t) \tag{31}$$

Equation (31) increases when the policy is more stochastic and decreases as it becomes more deterministic. It penalizes overly confident action distributions. Equation (32) combines the PPO loss function, including the clipped surrogate objective, value function loss, and the entropy bonus.

$$L_t^{PPO}(\theta) = E_t[L_t^{PPO}(\theta) - c_1 L_t^{VF} + c_2 S[\pi_\theta](s_t)] \tag{32}$$

Where $L_t^{PPO}(\theta) = (V_\theta(s_t) - R_t)^2$ is the mean squared error of the value function and c_1 and c_2 are the weighting coefficients. The training is done at each timestep, therefore the policy network (actor) is updated by maximizing the objective function in equation (33)

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}} E[L(\theta)]. \tag{33}$$

Equation (33) ensures that the policy improves concerning the clipped surrogate objective. At the same time, the critic network is updated by minimizing the mean squared error between its predicted value and the observed return, as shown in equation (34)

$$\emptyset_{k+1} = \underset{\emptyset}{\operatorname{argmin}} E[V(s_t|\emptyset) - R_t]^2. \tag{34}$$

Equations (33) and (34) allow updates to the actor and critic, which are the foundation of the PPO training loop, allowing the PPO algorithm to iteratively refine both the policy and its value estimates, making the training more stable than other algorithms.

2.7 Experimental Data Collection

In this study, the waypoints of the car were collected using the GPS data on the 2015 Nissan Leaf S. The vehicle was driven through Reeves Park in Norman, Oklahoma, and the GPS continuously recorded its longitude and latitude. The longitude and latitude data were then processed to generate a reference path as seen on Figure 6. In this research, the reference path is not a generic centerline but the actual path the vehicle took, captured through GPS data. The FFM is applied relative to this real-world path to evaluate tracking performance and replan trajectories

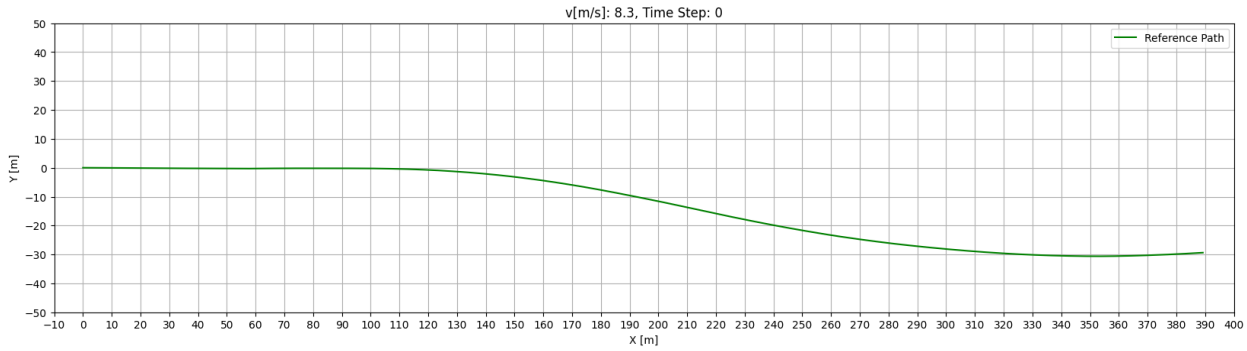

Figure 6: Reference Path

The vehicle is an all-electric car and was modified to convert the electric vehicle to an AV. The vehicle modification can be seen in Figure 7. It shows the modifications made to the vehicle to allow the electric car to drive autonomously. The modifications include the Texas Instruments AWR1642BOOST mm Wave radar, cameras, Velodyne VLP-16 Puck lidar, Xsens MTi-30 IMU, and the Swift Piski Multi GNSS GPS. To allow the vehicle to move without a driver, a drive-by-wire functionality was added. This functionality required modifications to the steering and pedals. Two computer stations were added to the trunk floor: an Alienware R13 and a Dell T3500 [22].

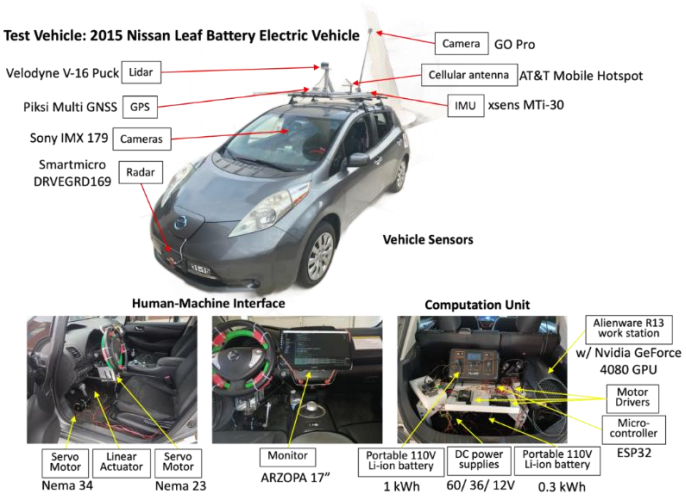

Figure 7: Modification on the 2015 Nissan Leaf S.

3 Results Analysis

In this section, we compare the results of the fixed prediction horizon, the adaptive prediction horizon, and the RL prediction horizon in the FFM. The RL prediction horizon will have additional metrics, as TensorBoard was used to analyze the training process. TensorBoard is a visualization tool that helps monitor training performance by plotting metrics such as reward, loss, value error, and policy stability over time. This allows for real-time insight into how well the PPO agent is learning and whether the training is converging effectively. In each result, the same path was taken as well as the obstacles' placement. The code was written in Python, and the plots were plotted using the Matplotlib function. The software that was used for the code and RL was in Visual Studio Code. The first discussion will be about the fixed prediction horizon and the values that resulted from the reference path from Figure 6. The reference path was taken from the GPS taken from Reeve's Park. Next, the adaptive approach was used, where the prediction horizon changes based on the distance of the vehicle to the obstacle. In this context, an obstacle refers to a known static point on the road, such as a construction barrier, parked vehicle, or designated blocking zone, defined by fixed (x,y) coordinates.

$$x > 30 \tag{35}$$

$$15 \leq x \leq 30 \tag{36}$$

$$10 \leq x < 15 \tag{37}$$

$$0 \leq x < 10 \tag{38}$$

where x is the distance of the vehicle to the obstacle. If the vehicle is between equations (35), then the prediction horizon value is 25. The same concept is done for equations (36, 37, 38), and the prediction horizon value is 20, 15, and 10, respectively. The last approach was using

reinforcement learning with PPO StableBaseline3. This is a method using RL to pick the best prediction horizon based on the rewards the agent gets due to its observation.

The path planning results of FFM were assessed based on three criteria for each simulation. The first was the average computation time. The computation is the time that records the current time at the beginning of the function execution. The function execution is the function that generates the Frenet path, converts it to the global paths, checks for collision, picks the best path, and adjusts prediction horizons based on the obstacles. This is useful as it evaluates the efficiency of the Frenet path planning algorithm and allows real-time monitoring of performance during execution. From the simulations, we know that the lower the prediction horizon, the lower the computation power needed. This is because the planner simulates fewer future states, and the optimization algorithm runs faster because it considers fewer possibilities. Next is the tracking error.

The tracking error was measured by using the equation from (20). Computed by comparing the vehicle's actual position and the reference path created from waypoints collected in the experiment. This is important as it provides how far the vehicle is from the actual path. From the simulation, the lower the prediction horizon, the less tracking error. A shorter prediction horizon means the planner only looks a short distance ahead, while a longer prediction horizon means the planner looks further ahead. With a shorter prediction horizon, the planner updates the planned trajectory more frequently. This allows the vehicle to react quickly to changes in the environment, such as obstacles and deviations from the reference path.

The last metric is the steering angle. The steering angle is how much the vehicle turns relative to its wheelbase. Therefore, the steering angle will change due to the path of the vehicle. The obstacles in the road will affect the steering angle of the vehicle as the vehicle will find a trajectory that avoids the obstacle. This is important as the lower the degree of the steering angle, the less

jerk the car will experience. This allows for the safety of the people in the vehicle and provides comfort to the people in the vehicle. With all these metrics mentioned, Table 1 shows that the PPO method performed the best when it came to measuring these metrics. In this case, the lower the result, the better, as it shows minimal error.

Table 1: Comparison of Different Methods

Average Criteria	Fixed (15)	Adaptive	PPO
Computation Time (s)	0.0883	0.0870	0.0686
Tracking Error (m)	1.0462	1.161	0.8226
Steering Angle (º)	0.1979	0.1849	0.1710

In Figure 8, the plot shows the start of the simulation. The prediction horizon in this plot has a value of 15. This value was chosen to showcase the median prediction horizon (from 10-25) that was tested. The green line shows the reference path that the vehicle is given to follow. The red line shows the predicted path the vehicle is creating in real-time. The x is to be considered as an obstacle in the path. The green triangle is the current position of the vehicle. This plot is made to help visualize the start of the vehicle and the predicted path it is planning for the vehicle. The plot shows that it senses the obstacle at 100 meters and creates a predicted path that goes around the obstacles. The initial start is at the reference path, so there will be no tracking error. There will also be no steering angle as the car is driving straight in the beginning. The speed of the car is also initialized to be 8.3 m/s at the start. The time step is 0 however the trajectory of the path is already being computed. Once there are obstacles and/or turns, the vehicle will accelerate or decelerate depending on the condition mentioned in the paper [16].

3.1 Fixed Prediction Horizon

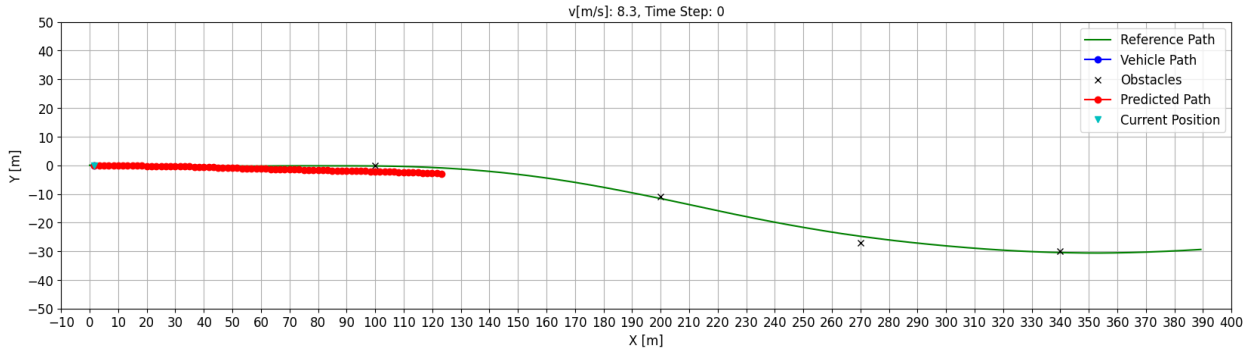

Figure 8: Fixed Prediction Horizon Start.

In Figure 9, the simulation shows the end of the simulation. At the end of the simulation, it showcases the path the vehicle took during the simulation. The plot shows how the vehicle avoided the obstacles, and the tracking error can be seen. The tracking error can be visualized by seeing the difference in the distance between the reference path and the vehicle path. Around 100 meters to 200 meters the vehicle path deviated the most from the reference path.

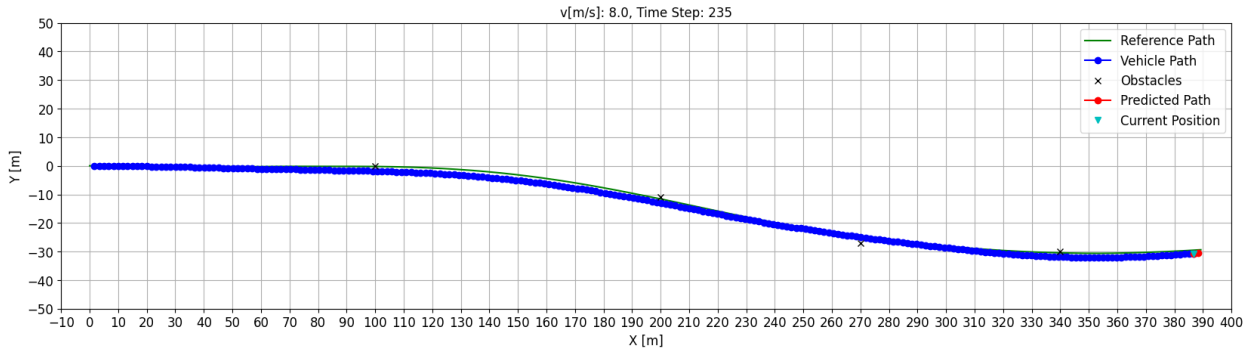

Figure 9: Fixed Prediction Horizon End.

3.2 Adaptive Prediction Horizon

In Figure 10, the adaptive method is being used. In the plot, we can see from the start that it starts planning to avoid the two obstacles that are on the track. From equations (35,36,37,38), the

prediction horizon value starts at 25 since the obstacle is more than 30 meters away. With a high prediction horizon, the planner generates a trajectory that extends up to 205 meters ahead because the vehicle is allowed to plan over a longer time window. This is farther than the plot in Figure 8. This explains why the computation time and the tracking error will increase.

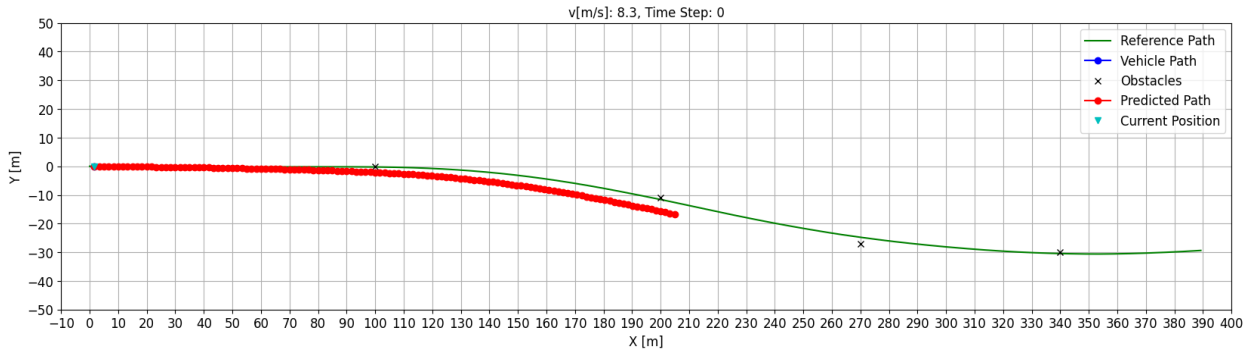

Figure 10: Adaptive Prediction Horizon Start.

Figure 11 shows the end of the adaptive simulation. From the plot, we see that the path the vehicle took is smoother and has fewer jerks. The plot also shows that it took a slightly different path than the one in Figure 9. This is due to the higher prediction horizon and saw a path that will avoid the obstacle while maintaining minimal difference from the reference path. It is important to note that since this is adaptive the prediction horizon changes depending on the vehicles distance to the obstacle. Figure 12 shows the prediction horizon changing to 10 as the distance from the vehicle to the obstacle meets the constraints in equation (36). This shows that the simulation can adaptively change when the vehicle gets closer to an obstacle.

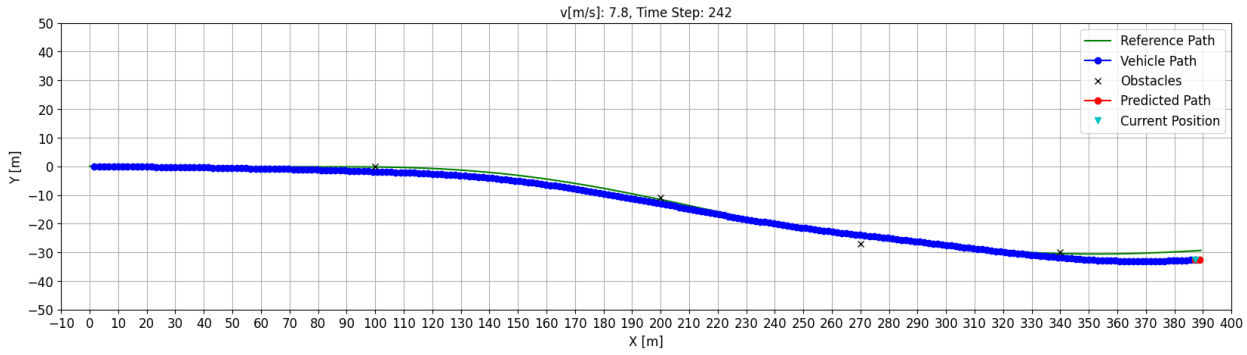

Figure 11: Adaptive Prediction Horizon End.

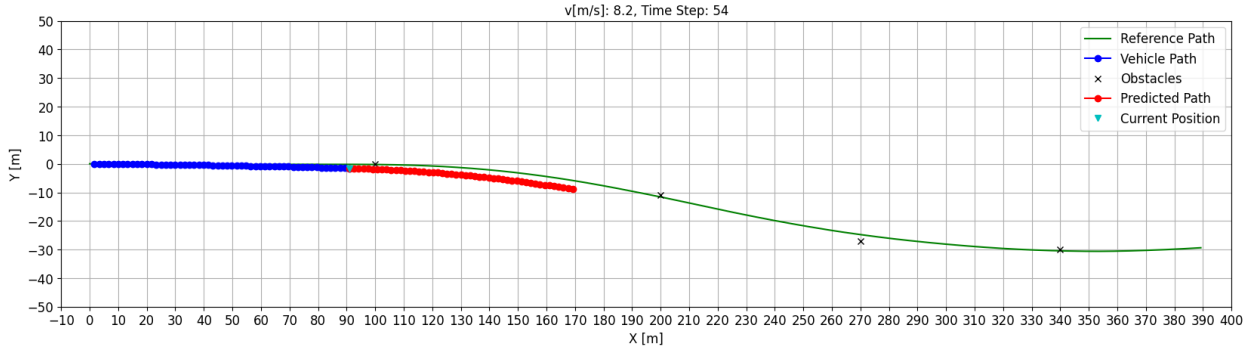

Figure 12: Adaptive Prediction Horizon Changing.

3.3 Reinforcement Learning Prediction Horizon

Next, a new approach is introduced to the fixed and adaptive approach of choosing the prediction horizon: reinforcement learning. Reinforcement learning with PPO (StableBaselines3) learns an optimal policy for path planning through trial and error in a simulated environment. Unlike fixed and adaptive approaches, PPO learns the best prediction horizon by maximizing the cumulative rewards through reinforcement learning. Instead of hardcoded rules, the vehicle learns from experience and determines which prediction horizon is best. The reward function optimizes for minimizing tracking error, computation time, and steering angle. Each reward function has its metrics: mean episodic reward (Figure 13(a)), explained variance (Figure 13(b)), value loss

(Figure 13(c)), and loss (Figure 13(d)). The plot in Figure 13(a) shows the mean episodic reward over time. The increasing trend in the plot shows that the RL agent is improving its performance and eventually stabilizes around 65,000 timesteps. Figure 13(b) is a metric that shows how well the value function explains the variance in the returns. A value close to 1 means the value function accurately predicts the expected returns. The plot shows that the value is close to one, indicating a well-learned value function. Figure 13(c) shows the value loss. The value loss measures how well the critic is learning. The critic estimates the value of a state to help guide the action. The high initial values show that at the start of the training, the model predicts inaccurate results. The steady decline means the model is improving its predictions of future rewards, stabilizing towards a lower loss. Figure 13(d) represents the total loss during training. The decreasing trend indicates convergence, meaning the model is learning effectively. The plot for the metrics for computation time is given in Figure 13. After the training is done, the model is then tested, and the prediction horizon is given as an output. A plot is plotted to show the path the vehicle took after the test has finished. Figure 14 shows the path the vehicle took. Based on the training, the RL chose a prediction horizon value of 10. This result is good as the lower the prediction horizon, the lower the computation time. The predicted path is created from the prediction horizon chosen after training.

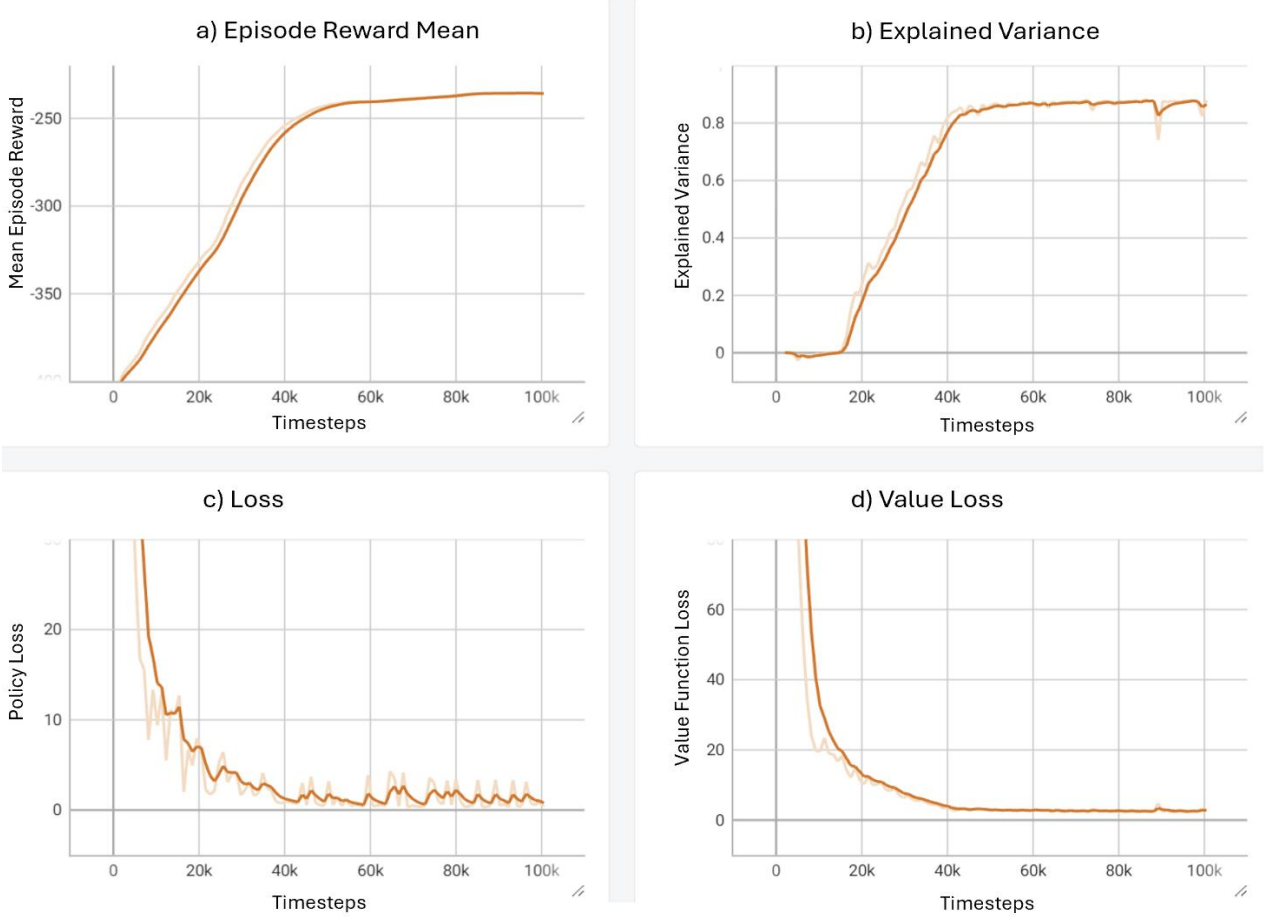

Figure 13: Reward Trend for Computation Time.

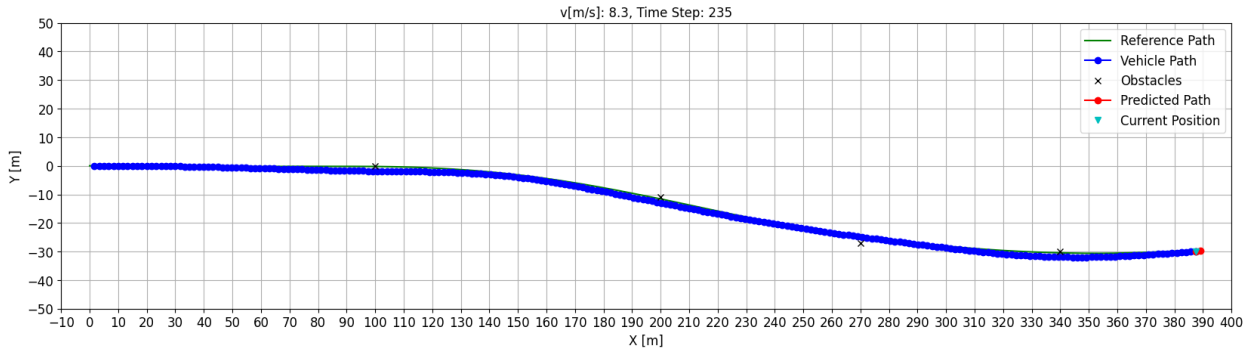

Figure 14: Path Results Prioritizing Computation Time.

The next training was done with prioritizing tracking errors. The plot for the mean episodic reward, explained variance, loss, and value loss is shown in Figures 15(a), 15(b), 15(c), and 15(d),

respectively. The plot in Figure 15(a) shows an increase in the rewards. This shows that the RL is learning and minimizing the error through time and improving its policy. The tracking error took longer to converge than the computation time, therefore, more timesteps were needed. From the plot, the RL converged around 115,000 timesteps. The explained variance plot in Figure 15(b) shows that initially, it was not able to predict the results; however, around 20,000 timesteps, it jumped to 0.4 and started to steadily increase to 0.7. For Figure 15(c), the loss value originally started very high but decreased as the training progressed. In the end, the final loss plot shows a value that is low and stable, which is a good indicator of convergence. Finally, in Figure 15(d), the plot also starts very high, but as the training progresses, the value starts decreasing. The loss stabilizes at the bottom, which means the value function is now making better estimates. This also shows that the model is converging properly. After the training is done, the model is then evaluated again. The test gave a prediction horizon value of 10, and the plot of the path the vehicle took is shown in Figure 16. As mentioned before, the tracking error is better when the lower prediction horizon value is low. Therefore, having a result of 10 for the prediction value is good, as that is the expected value. Next, the metric for steering angles will be analyzed.

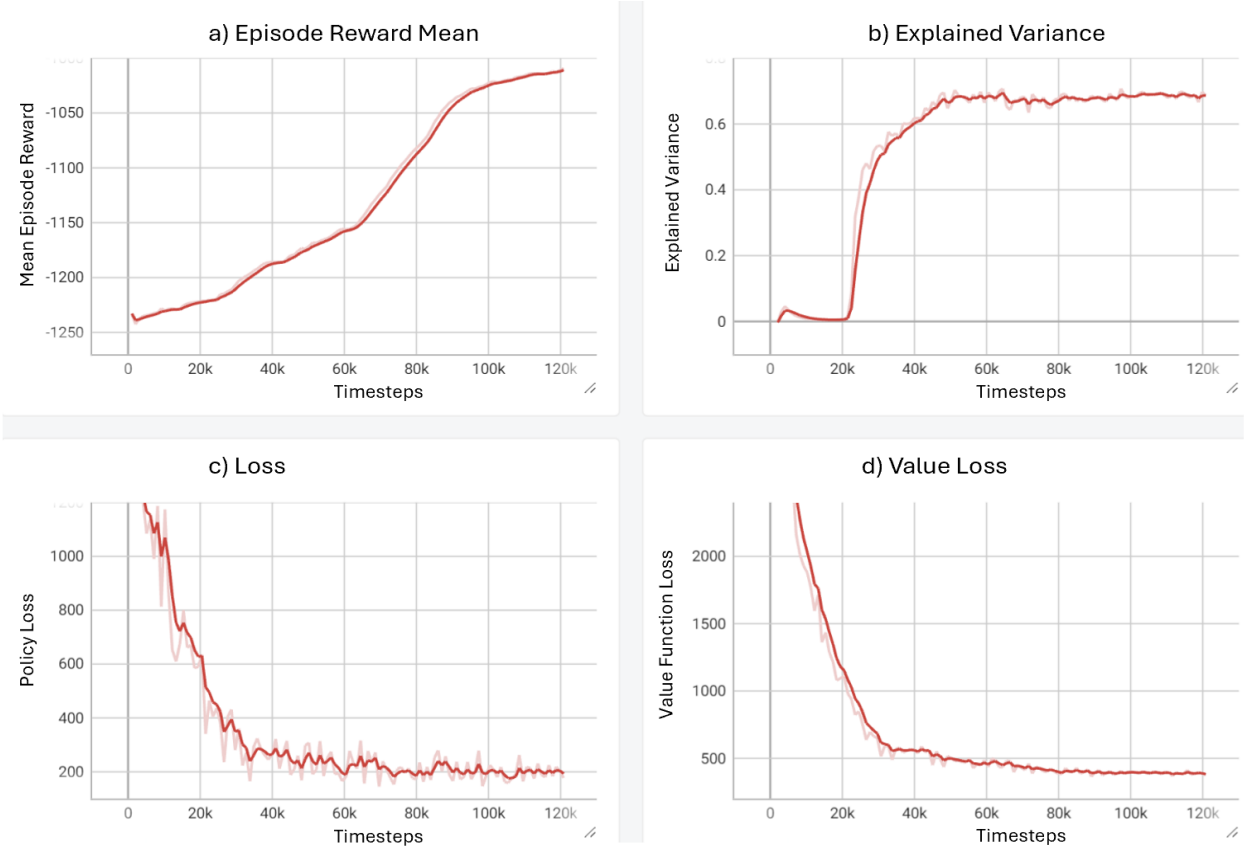

Figure 15: Training Results from Tracking Error

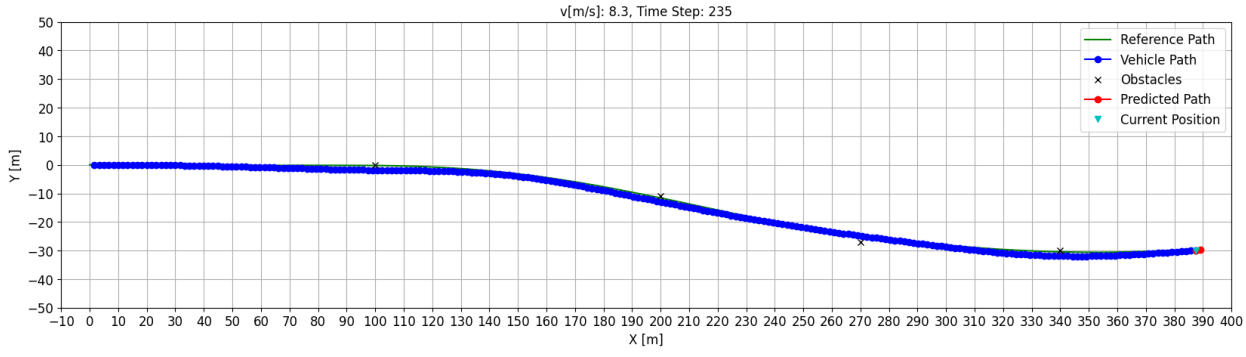

Figure 16: Results for Path Prioritizing Tracking Error

For the steering angle, the tensor board metrics are shown in Figure 17. The results are very similar to Figures 13 and 15. In Figure 17(a) there is an increase and decrease between 0-50,000 timesteps, then an increase after 50,000 timesteps until it converges. The explained variance in Figure 17(b) shows that it is not able to predict as well as the computation time and tracking error; however, Figures 17(c) and 17(d) show that the plot fluctuates but still decreases. This is due to the explained variance. It was not able to predict the results as well, which resulted in a fluctuating loss. In this training, it shows that the training was still able to learn as the rewards increased. However, it gave a prediction horizon value of 24. A value of 25 would have been more ideal, as a higher prediction value results in a lower steering angle. The value 24 could have been shown as the explained variance was not as good as the other observations. After training and testing the path with a prediction horizon value was 24 was plotted in Figure 18. The plot in Figure 18 shows the path taken by the vehicle. In the plot, there is a difference in the smoothness of the vehicle. The path was also taken differently from Figures 14 and 16. When prioritizing the steering angle, the vehicle goes above the obstacle at [270, -27]. By taking this path, the vehicle was able to have a lower steering angle than when the prediction horizon value was 10 or 15.

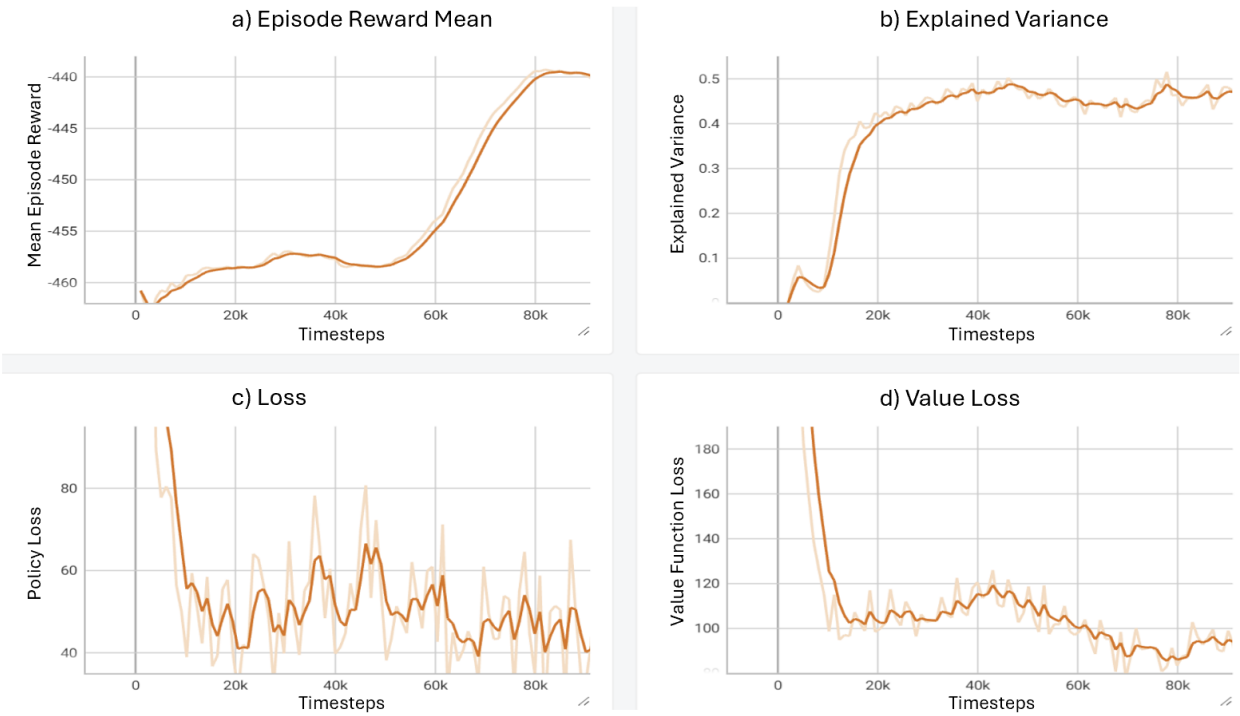

Figure 17: Mean Rewards for Steering Angle

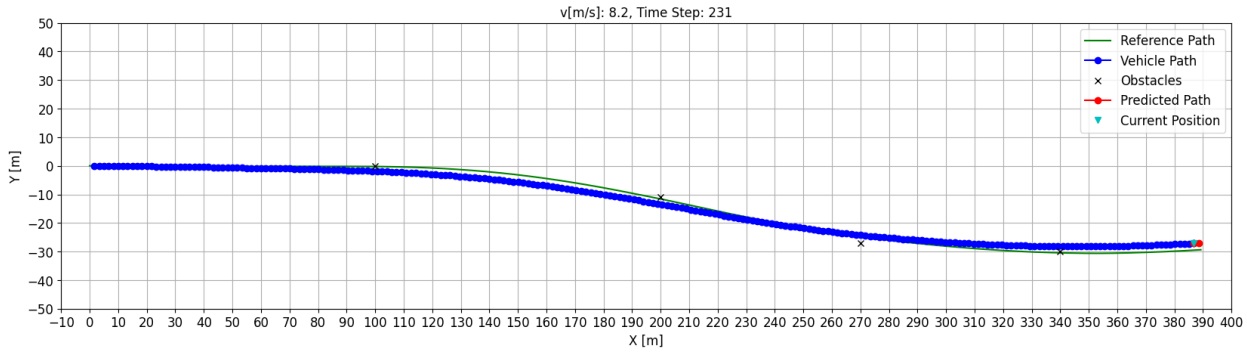

Figure 18: PPO Prediction Horizon

With RL using PPO, the agent was able to successfully minimize the rewards and train the agent to observe the computation time, tracking error, and steering angle. Table 2 shows the prediction horizon when prioritizing each criterion. The results showed that based on each criterion, PPO was able to predict the correct value given the path and the obstacles in the road.

Table 2: Prediction Horizon Values

Trial	Prediction Horizon
Computation Time	10
Tracking Error	10
Steering Angle	24

4 Conclusion

This study explored the effectiveness of fixed, adaptive, and RL-based PPO prediction horizon selection in Frenet Frame-based path planning for AV. Unlike other literature where the metric was not mentioned or an FFM with RL was not implemented, this research compared the fixed, adaptive, and RL approaches to picking the optimal prediction horizon value. Moreover, many papers use datasets available online, we collected our data to create the waypoints of the path. This allowed the simulation to be done on the road in Norman, Oklahoma. With the path data created, we conducted a simulation that will help the Nissan Leaf pick the best prediction horizon based on the obstacles on the road and the path. Based on the trajectory of the prediction horizon, the computation time, tracking error, and steering angle were calculated. By evaluating each approach, we gained insight into their strengths and limitations. At a fixed prediction horizon of 15, it provided low computation time but could not adjust to different horizons based on the distance to the obstacle. Hence, the adaptive approach was introduced, where the vehicle will have an adjusted prediction horizon based on the distance to the obstacle. This method led to smoother obstacle avoidance and a lower steering angle compared to the fixed horizon, but at the cost of a higher tracking error. The PPO–based reinforcement method outperformed both methods by learning the

optimal prediction horizon dynamically. Unlike the other two approaches, PPO uses learned experience to select the best prediction horizon based on real-time observation. The results showed that PPO achieved the lowest value in all three criteria tracking error (0.8226), computation time (0.0686), and steering angle (0.1710) when its metrics were prioritized. Therefore, if a vehicle wants to make fast decisions on the road. A low prediction horizon will be needed, and the PPO can be adjusted to weigh the computation time more heavily. The same can be done for tracking errors and steering angles. If the driver wants to stay on the road as much as possible, then the tracking error can be prioritized. For tracking errors, the prediction horizon should be low (10), and this can be prioritized in PPO by adjusting the weight in the tracking error reward function. If the driver wants to have a smooth ride and no jerks, the steering angle prediction horizon needs to be high (25). The findings demonstrate that RL-based prediction horizon selection offers a more efficient and adaptable solution for autonomous vehicle path planning. For future work, we can implement the three reward functions to consider all three criteria when choosing the best prediction horizon, train in more diverse driving scenarios, increase stability in PPO training, as well as add more reward functions and penalties to simulate real-world scenarios.

5 References

[1] J. Guo, U. Kurup, and M. Shah, "Is it Safe to Drive? An Overview of Factors, Metrics, and Datasets for Driveability Assessment in Autonomous Driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 8, pp. 3135–3151, Aug. 2020, doi: 10.1109/TITS.2019.2926042.

[2] B. Li, Y. Ouyang, L. Li, and Y. Zhang, "Autonomous Driving on Curvy Roads Without Reliance on Frenet Frame: A Cartesian-Based Trajectory Planning Method," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 15729–15741, Sep. 2022, doi: 10.1109/TITS.2022.3145389.

[3] S. Luo, X. Li, and Z. Sun, "An Optimization-based Motion Planning Method for Autonomous Driving Vehicle," in *2020 3rd International Conference on Unmanned Systems (ICUS)*, Harbin, China: IEEE, Nov. 2020, pp. 739–744. doi: 10.1109/ICUS50048.2020.9275009.

[4] J. S. Ho, B. C. Tan, T. C. Lau, and N. Khan, "Public Acceptance towards Emerging Autonomous Vehicle Technology: A Bibliometric Research," *Sustainability*, vol. 15, no. 2, p. 1566, Jan. 2023, doi: 10.3390/su15021566.

[5] J. Huang, Z. He, Y. Arakawa, and B. Dawton, "Trajectory Planning in Frenet Frame via Multi-Objective Optimization," *IEEE Access*, vol. 11, pp. 70764–70777, 2023, doi: 10.1109/ACCESS.2023.3294713.

[6] A. Benloucif, A.-T. Nguyen, C. Sentouh, and J.-C. Popieul, "Cooperative Trajectory Planning for Haptic Shared Control Between Driver and Automation in Highway Driving," *IEEE Trans. Ind. Electron.*, vol. 66, no. 12, pp. 9846–9857, Dec. 2019, doi: 10.1109/TIE.2019.2893864.

[7] L. H. Meftah, A. Cherif, and R. Braham, "Improving Autonomous Vehicles Maneuverability and Collision Avoidance in Adverse Weather Conditions Using Generative Adversarial Networks," *IEEE Access*, vol. 12, pp. 89679–89690, 2024, doi: 10.1109/ACCESS.2024.3419029.

[8] P. Wang, S. Gao, L. Li, B. Sun, and S. Cheng, "Obstacle Avoidance Path Planning Design for Autonomous Driving Vehicles Based on an Improved Artificial Potential Field Algorithm," *Energies*, vol. 12, no. 12, p. 2342, Jun. 2019, doi: 10.3390/en12122342.

[9] K. Shibata, N. Shibata, K. Nonaka, and K. Sekiguchi, "Model Predictive Obstacle Avoidance Control for Vehicles with Automatic Velocity Suppression using Artificial Potential Field," *IFAC-Pap.*, vol. 51, no. 20, pp. 313–318, 2018, doi: 10.1016/j.ifacol.2018.11.050.

[10] S. Dixit *et al.*, "Trajectory Planning for Autonomous High-Speed Overtaking in Structured Environments Using Robust MPC," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2310–2323, Jun. 2020, doi: 10.1109/TITS.2019.2916354.

[11] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *J. ACM*, vol. 40, no. 5, pp. 1048–1066, Nov. 1993, doi: 10.1145/174147.174150.

[12] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive Active Steering Control for Autonomous Vehicle Systems," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 3, pp. 566–580, May 2007, doi: 10.1109/TCST.2007.894653.

[13] Z. Chen, J. Lai, P. Li, O. I. Awad, and Y. Zhu, "Prediction Horizon-varying Model Predictive Control (MPC)for Autonomous Vehicle Control," Jan. 18, 2024, *In Review*. doi: 10.21203/rs.3.rs-3850749/v1.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 28, 2017, *arXiv*: arXiv:1707.06347. doi: 10.48550/arXiv.1707.06347.

[15] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, “Deep Reinforcement Learning framework for Autonomous Driving,” *Electron. Imaging*, vol. 29, no. 19, pp. 70–76, Jan. 2017, doi: 10.2352/ISSN.2470-1173.2017.19.AVM-023.

[16] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, “Optimal trajectory generation for dynamic street scenarios in a Frenét Frame,” in *2010 IEEE International Conference on Robotics and Automation*, Anchorage, AK: IEEE, May 2010, pp. 987–993. doi: 10.1109/ROBOT.2010.5509799.

[17] W. Xu, J. Pan, J. Wei, and J. M. Dolan, “Motion planning under uncertainty for on-road autonomous driving,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China: IEEE, May 2014, pp. 2507–2512. doi: 10.1109/ICRA.2014.6907209.

[18] J. Ziegler, P. Bender, T. Dang, and C. Stiller, “Trajectory planning for Bertha — A local, continuous method,” in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, MI, USA: IEEE, Jun. 2014, pp. 450–457. doi: 10.1109/IVS.2014.6856581.

[19] M. I. Makarov, “A LOCAL PATH PLANNING ALGORITHM FOR AVOIDING OBSTACLES IN THE FRENET FRAME,” *Control Sci.*, no. 3, pp. 56–61, Jul. 2024, doi: 10.25728/cs.2024.3.5.

[20] Q. Song *et al.*, “Autonomous Driving Decision Control Based on Improved Proximal Policy Optimization Algorithm,” *Appl. Sci.*, vol. 13, no. 11, p. 6400, May 2023, doi: 10.3390/app13116400.

[21] A. Hasankhani, Y. Tang, and J. VanZwieten, “Integrated path planning and control through proximal policy optimization for a marine current turbine”.

[22] Z. Arjmandzadeh, M. H. Abbasi, H. Wang, J. Zhang, and B. Xu, “A Comparative Study on Autonomous Vehicle Local Path Planning Through Model Predictive Control and Frenet Frame Method,” *SAE Int. J. Connect. Autom. Veh.*, vol. 7, no. 4, pp. 12-07-04–0029, Oct. 2024, doi: 10.4271/12-07-04-0029.

[23] C. S. Arvind and J. Senthilnath, “Autonomous Vehicle for Obstacle Detection and Avoidance Using Reinforcement Learning,” in *Soft Computing for Problem Solving*, vol. 1048, K. N. Das, J. C. Bansal, K. Deep, A. K. Nagar, P. Pathipooranam, and R. C. Naidu, Eds., in Advances in Intelligent Systems and Computing, vol. 1048. , Singapore: Springer Singapore, 2020, pp. 55–66. doi: 10.1007/978-981-15-0035-0_5.

[24] A. Boyali, S. Mita, and V. John, “A Tutorial On Autonomous Vehicle Steering Controller Design, Simulation and Implementation,” 2018, *arXiv*. doi: 10.48550/ARXIV.1803.03758.