Roboracer-UI-Control: An Interactive ROS-based System Parameter Tuning

raorane

March 2025

1 Introduction

This report presents the development of Roboracer-UI-Control, an interactive system designed to enhance the understanding and visualization of robotics concepts, particularly waypoint following using the Pure Pursuit algorithm. The system integrates ROS (Robot Operating System) nodes, custom message types, and a user-friendly Gradio interface to allow real-time parameter tuning and visualization.

2 Motivation

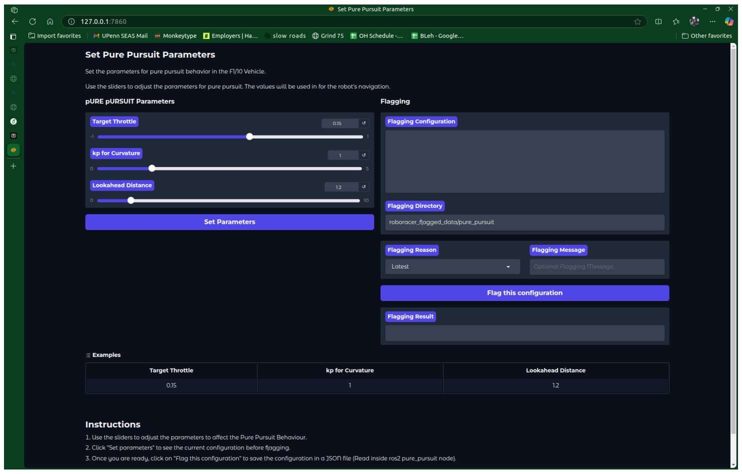
The primary motivation behind Roboracer-UI-Control is to simplify complex robotics concepts for students. By providing an interactive platform for visualizing and manipulating parameters in real-time, students can gain a more intuitive understanding of concepts such as wall following with PID control, gap following, and Pure Pursuit algorithm for waypoint navigation.

3 System Architecture

The Roboracer-UI-Control system consists of several interconnected components:

- Waypoint Logging Node
- Waypoint Following Node with Adjustable Parameters
- Custom ROS Message for Parameter Sharing
- Parameter Publishing Node
- Gradio Interface for Parameter Tuning
- Foxglove Visualization

4 Using the Interface



Set Pure Pursuit Parameters

Set the parameters for pure pursuit behavior in the F1/10 Vehicle.

Use the sliders to adjust the parameters for pure pursuit. The values will be used in for the robot's navigation.

pURE pURSUIT Parameters

Target Throttle: 0.15 (-1 to 1)

kp for Curvature: 1 (0 to 5)

Lookahead Distance: 1.2 (0 to 10)

Set Parameters

Flagging

Flagging Configuration

Flagging Directory: roboracer_flagged_data/pure_pursuit

Flagging Reason: Latest

Flagging Message: Optional Flagging Message.

Flag this configuration

Flagging Result

Examples

Target Throttle	kp for Curvature	Lookahead Distance
0.15	1	1.2

Instructions

1. Use the sliders to adjust the parameters to affect the Pure Pursuit Behaviour.
2. Click "Set parameters" to see the current configuration before flagging.
3. Once you are ready, click on "Flag this configuration" to save the configuration in a JSON file (Read inside ros2 pure_pursuit node).

Figure 1: Roboracer Control Interface

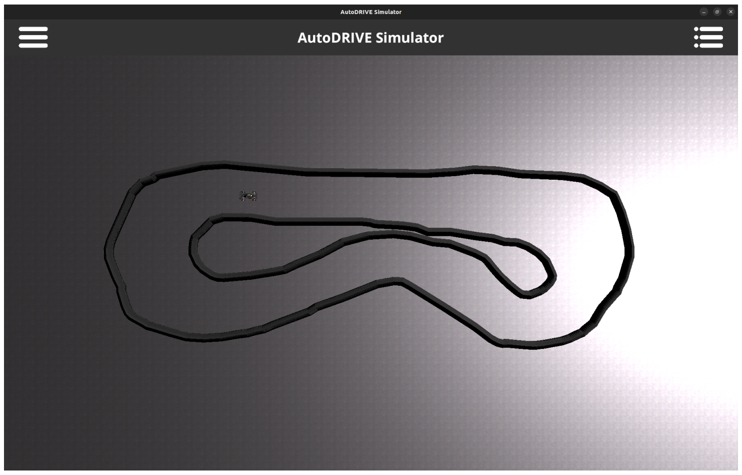


Figure 2: Autodrive UI showing real-time vehicle performance

The Roboracer-UI-Control interface provides intuitive controls for adjusting the key parameters that affect the vehicle's waypoint following behavior. These parameters can be modified in real-time, allowing users to observe immediate changes in the vehicle's performance.

4.1 Parameter: Target Throttle

The target throttle parameter controls the maximum forward velocity of the vehicle. In the implementation, this value is automatically adjusted based on the steering angle to ensure stable cornering:

```
throttle_multiplier = 1.0 - 0.5 * np.abs(steering_angle)
throttle = self.target_throttle * throttle_multiplier
```

This adaptive throttle mechanism reduces speed during sharp turns (when steering angle is high) and allows higher speeds on straighter sections. The default value is set to 0.15, which represents a moderate speed suitable for learning.

4.2 Parameter: Kp for Curvature

The Kp parameter is a proportional gain that directly affects how aggressively the vehicle responds to the calculated path curvature:

```
curvature = 2 * y_ego * self.wheelbase / (x_ego**2 + y_ego**2)
steering_angle = self.kp * curvature
```

A higher Kp value results in more aggressive steering responses, which can help the vehicle stay on track during sharp turns but may cause oscillations at high speeds. The default value is 2.0, providing a balanced response for most scenarios.

4.3 Parameter: Lookahead Distance

The lookahead distance determines how far ahead the algorithm looks for the next waypoint to follow:

```
distances[distances < self.lookahead_distance] = np.inf
idx = np.argmin(np.abs(distances))
```

This parameter has a significant impact on the vehicle's path-following behavior:

- A smaller lookahead distance makes the vehicle follow the path more precisely but can result in jerky movements.
- A larger lookahead distance creates smoother trajectories but may cut corners on sharp turns.

The default value is set to 1.2 meters, which provides a good balance between precision and smoothness for the Roboracer platform.

The interface allows students to experiment with these parameters and observe in real-time how changes affect the vehicle's behavior, providing an intuitive understanding of the Pure Pursuit algorithm and its tuning considerations.

5 Implementation Details

5.1 Waypoint Logging Node

This ROS node is responsible for recording waypoints as the robot moves through its environment. The implementation details of this node are crucial for creating accurate path data.

5.2 Waypoint Following Node

The core of the system is a ROS node that implements the Pure Pursuit algorithm for waypoint following. Key parameters include:

- Throttle: Maximum throttle for the car, adjusted based on steering angle
- Kp: Proportional gain multiplied by the calculated curvature to the next waypoint
- Lookahead Distance: Distance threshold for selecting the next target waypoint

5.3 Custom ROS Message

A custom message type was created to efficiently share parameters between nodes. This approach optimizes performance by avoiding continuous file reads.

5.4 Parameter Publishing Node

This node reads parameter values from a JSON file and publishes them to a dedicated ROS topic, enabling dynamic parameter updates without system restarts.

5.5 Gradio Interface

A Gradio-based user interface was developed to provide an intuitive means of adjusting parameters. This interface updates the JSON file read by the parameter publishing node.

5.6 Foxglove Visualization

Foxglove Studio is utilized to visualize key metrics such as distance from walls and waypoint positions, enhancing the user's understanding of the system's behavior.

6 Challenges

6.1 System Integration

One of the main challenges faced during development was the integration of various components into a cohesive system. This included ensuring seamless communication between ROS nodes, the Gradio interface, and the visualization tools.

7 Results and Discussion

The Roboracer-UI-Control system was tested with real-time parameter adjustments, which provided valuable insights into the impact of different parameters on vehicle performance. A video demonstrating these adjustments can be viewed at the following link:

`https://www.youtube.com/watch?v=mbhb9hKrOSA&feature=youtu.be`

The video shows how changing the parameters affects the car in real time. Along with the Gradio interface, we see real-time video of the car in the Autodrive simulator. When the lookahead distance is set too high, the vehicle struggles at corners, as seen in the video. This highlights the tradeoff between smoothness and responsiveness in the Pure Pursuit algorithm, where a larger lookahead distance can cause the car to cut corners on sharp turns.

Additionally, waypoints are visualized in RViz, along with car frames, to help users better understand how the vehicle follows the planned path. The combination of these visualizations, along with real-time parameter tuning, provides a comprehensive learning experience, helping students grasp the nuances of the system's behavior and the impact of each parameter.

8 Future Work

Several avenues for future development have been identified:

- User-defined Waypoint Mapping: Allow students to create and edit their own waypoints on the map, providing hands-on experience with path planning and smoothing techniques.
- Real-time Occupancy Mapping: Implement simultaneous localization and mapping (SLAM) to generate occupancy maps in real-time as the robot explores its environment.
- Performance Optimization: Refine the codebase to enable higher speeds while maintaining accurate waypoint following.
- Advanced Control Algorithms: Incorporate additional control algorithms beyond Pure Pursuit for comparative studies.