OuterSense

Automated Driving Using External Perception

Individual Lab Report - ILR07
September 27, 2023

Team E - Outersense

Author:
Jash Shah

Team Members:

Atharv Pulapaka
Dhanesh Pamnani
Jash Shah
Ronit Hire
Shreyas Jha

THE ROBOTICS INSTITUTE

Carnegie Mellon University

Contents

1 Individual Progress

1.1 Simulation Setup

Between the previous progress review (PR) and the current one, we have made significant developments in establishing a Gazebo simulation environment. The primary objective of this simulation environment is to facilitate testing and validation of our recently introduced planning subsystem for the current semester.

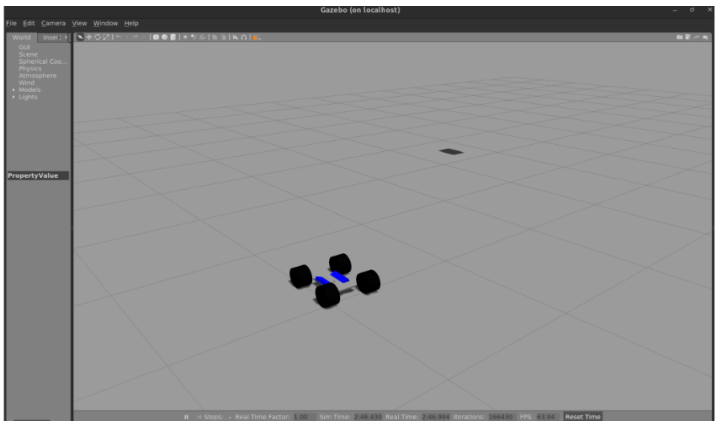

Figure 1: Gazebo setup

The rationale behind creating this Gazebo simulation environment is twofold. Firstly, it provides a safe and controlled platform for evaluating and refining our planning algorithms. Directly applying these algorithms to the physical vehicle can pose risks, particularly when they haven't undergone thorough testing. Secondly, practical constraints, such as the finite battery life of the real car, make it impractical to conduct extensive testing over extended periods.

The current state of our Gazebo environment features a single Ackermann vehicle. This vehicle is capable of receiving and executing commands in a custom message format, which includes the following key elements:

- `Header header`: This component furnishes essential metadata and context for each command, ensuring accurate tracking and timestamping.
- `float32[] acc_input`: These values signify acceleration inputs, governing changes in the vehicle's speed, including acceleration and deceleration.
- `float32[] steering_input`: These values dictate steering inputs, specifying the vehicle's maneuvers, including turning angles and directions.
- `float32[] current_state`: This section encapsulates the vehicle's present state, encompassing information about its position, orientation, and other pertinent attributes.
- `float32 flag`: This flag serves as a control signal, enabling specific actions or modes of operation within the simulation environment.

Together, these components establish a robust communication and control system that facilitates interaction between our planning subsystem and the simulated Ackermann vehicle. This interaction is fundamental for conducting comprehensive testing and validation of our planning algorithms within the secure digital realm. This approach addresses safety concerns and accommodates practical constraints related to real-world testing, such as limited battery capacity.

1.2 Planning Literature review

1.3 Planning Subsystem and Lane Centering:

In addition to our ongoing simulation efforts, we've given significant attention to developing the planning subsystem. Our primary focus, particularly in the initial phase, is to create a streamlined planner designed to guide our vehicles to the lane center. This task, though seemingly simple, presents various complexities that require careful consideration and evaluation of algorithms.

Lane Centering Algorithms:

Our literature review covered various planning algorithms for lane centering. Choosing the right algorithm is crucial for the effectiveness of our planning subsystem. Here's an overview of some prominent algorithms we've examined:

Stanley Controller:

The Stanley Controller is an algorithm specifically designed for autonomous vehicles, focusing on tracking the lane center by considering lateral and heading errors.

A* Algorithm (A-Star):

A* is known for guaranteeing optimal paths by minimizing a cost function. It's adaptable to different road configurations but can be computationally intensive and requires a detailed map of the road.

RRT (Rapidly-Exploring Random Trees):

RRT is a sampling-based algorithm suitable for handling dynamic obstacles or uncertain environments. It can adapt to environmental changes but doesn't guarantee path optimality and may require many iterations.

Static Obstacle Avoidance:

Our project involves using bird's-eye-view cameras to gain insights into the vehicle's surroundings. This aerial perspective enables us to develop path planning strategies for lane centering and static obstacle avoidance. We'll continue exploring and evaluating additional planning algorithms that integrate data from these external infrastructure sensors to navigate our vehicles safely in dynamic and complex environments.

2 Challenges Faced

2.1 Simulation setup

One of the primary hurdles I encountered during the setup of the Gazebo simulator revolved around dependency issues. Given the necessity to integrate an off-the-shelf Ackermann vehicle model, ensuring compatibility with the various components of the robotic operating system (ROS) and Gazebo became a non-trivial task. The precise coordination of software versions, ROS packages, and Gazebo plugins proved to be a complex undertaking. Achieving the correct alignment of these dependencies was crucial to successfully execute and simulate the Ackermann vehicle model.

Moreover, the second significant challenge involved configuring the vehicle to interpret and act upon our custom message format. Initially, the message format employed was intricate and, at times, convoluted, making it challenging to establish effective communication between our control system and the vehicle. Addressing this challenge required a meticulous process of message format refinement, ensuring that the vehicle could seamlessly interpret and respond to commands in our desired format. This endeavor demanded considerable effort to streamline the messaging protocol, ultimately enabling the vehicle to accurately respond to our control instructions.

These challenges, while demanding, have significantly contributed to the robustness and effectiveness of our simulation environment. Overcoming these obstacles not only improved compatibility and communication but also laid a strong foundation for conducting comprehensive testing and experimentation within the Gazebo simulator.

2.2 Planning literature review

In our planner selection process, we are faced with a crucial dilemma: opting for a simple, quick-to-implement planner that meets our current needs or going for a complex one with extensive capabilities for future expansion.

2.2.1 Simplicity's Appeal:

A simple planner has its merits. It swiftly addresses our immediate requirements, ensuring we are getting our project off the ground quickly. However, we must be cautious not to oversimplify to the point where it limits future enhancements or adaptability.

2.2.2 Potential for Growth:

On the other hand, a complex planner offers extensive capabilities and future-proofing. Yet, it demands more time, effort, and resources to implement. It can also pose a steeper learning curve and delay project objectives.

2.2.3 Finding the Middle Ground:

Our approach is to choose a planner that's "simple but not simplistic." We are aiming for a solution that serves our immediate needs while staying open to future improvements. This means selecting a planner with a clear, extensible design. It strikes the right balance between addressing our current goals and accommodating future innovations.

Ultimately, our decision centers on finding the planner that hits this balance, moving us forward with our current tasks while allowing room for future growth.

3 Teamwork

In terms of teamwork, we are flexible and collaborate seamlessly to swiftly address high-priority tasks.

- **Ronit Hire**:

  Ronit played a pivotal role in managing the perception subsystem. His key contribution involved the modification of the existing codebase to implement Aruco markers for detecting and tracking the vehicle's markers. This adaptation was crucial not only as a backup system in case non-fiducial markers failed but also as a robust solution to enhance control system reliability.

- **Dhanesh Pamnani**:

  Dhanesh primarily focused on the literature review and developing the planning architecture framework. His thorough research in the planning domain, coupled with our collaborative work, greatly contributed to our planning subsystem's progress.

- **Shreyas Jha**:

  Shreyas integrated the VESC IMU package into the RC car's software, ensuring its accuracy through calibration. He also worked on optimizing the VESC driver's odometry data for each RC car.

- **Atharv Pulapaka**: Atharv took charge of managing multiple cars independently by implementing MPC (Model Predictive Control) packages. He aimed to incorporate distance constraints into the control system, but due to its non-linearity, achieving this constraint proved to be challenging.

4 Future work

4.1 Personal

On a personal front, my upcoming tasks will focus on the following areas:

1. Gazebo Environment improvement: I will add modular functionality to the current gazebo environment by adding a ground plane that mimics our track and also tries to add obstacles to the path.

2. Planning subsystem: I will also work to create a simple framework that will be used in the planning subsystem to try to make the car follow a center lane.

4.2 Team

As for the future work of the team, we wish to achieve the following goals.

1. Autonomously drive 2 vehicles on a track spanned by 3 infrastructure sensors.

2. Implementation of prototype global planner that steers to the center of the lane

3. Experiment with 2 different approaches for data association instead of relying on unique fiducial markers.