RUTGERS
WINLAB | Wireless Information Network Laboratory

Autonomous Self Driving Vehicular Project 2023

RASCAL: Robotic Autonomous Scale Car for Adaptive Learning

Adarsh Narayanan (UG), Joshua Menezes (UG), Christopher Sawicki (UG), Tommy Chu (UG), Brandon Cheng (UG), Ruben Alias (UG), Aleicia Zhu (HS), Ranvith Adulla (HS), Arya Chhabra (HS), Suhani Sengupta (HS)

RASCAL Overview

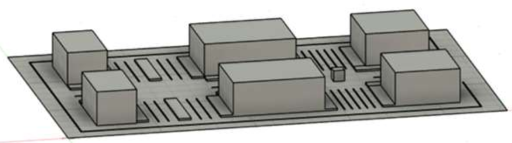
In our project, we hope to provide greater simulation accuracy with our design for a ~1/14 scale platform for miniature smart cars for repeatable tests in the smart city environment. Our design includes mechanisms such as Ackerman steering and differential gearing system to emulate the behavior of real cars, as well as sensors to calculate feedback. Using this car, we trained an imitation learning model to drive using camera input.

Goal

Create a self-driving car that emulates the mechanics of a real car, controlled autonomously by an imitation learning algorithm

Methodology

- Prototype 3D printed car
- Control hardware components
- Record training data
- Train machine learning model to associate control to image

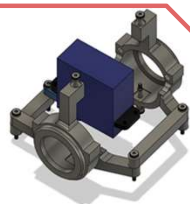


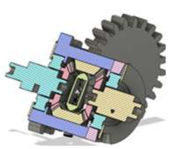
Smart City Model

Physical Design

Ackerman Steering

- Inside wheel travels in an arc with a smaller radius during turns
- Allows car to turn front wheels at correct angles
- Eliminates skidding during turns





Differential Gear System

- Back wheels are powered by stepper motor
- Allows left and right wheels to turn at different speeds, to accommodate different turn radii

Additional components added since *2022 Smart Car Hardware Design* project:

Adafruit BNO005 Absolute Orientation Sensor

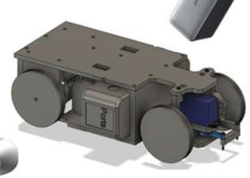
- Orientation data
- Used for odometry

Anker 737 Power Bank

- Improved battery for addition of camera

Intel RealSense D435

- RGB and depth camera



3D Model of RASCAL 2.0

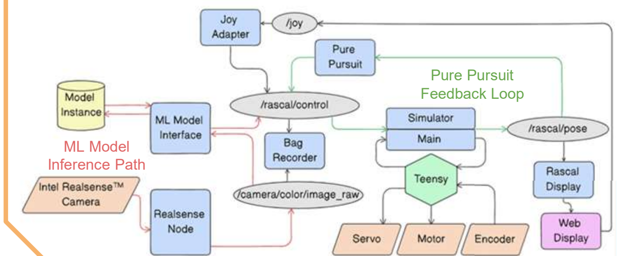
Software Architecture

ROS - Robot Operating System

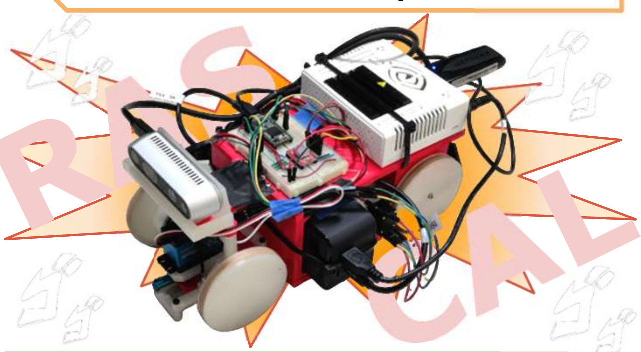
- Project runs code in ROS **nodes**, which communicate using **topics**

Modularity

- Nodes can be interchanged while others are kept the same
- **Main** can be replaced by the **Simulator** node, for parallel development without hardware
- Specifics for Arduino hardware can be found in the 2022 "Mini Smart Car Hardware Design" poster



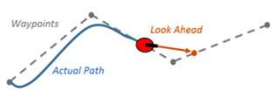
Software Architecture Diagram



Pure Pursuit & Odometry

Motivation

- Pure pursuit drives much more consistently than manual control
- Useful for gathering training data

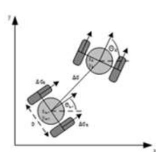


Pure Pursuit

- Finds a lookahead point on a path given the path and current position
- Internal calculations drive the car in arcs to the lookahead point

Odometry

- Uses car angle (from IMU) and distance traveled (from encoder) to calculate the car's position with arcs
- Use this feedback to constantly adjust back on to the path



This work was supported in part by the NSF REU program and the donation from nVERSES CAPITAL

Imitation Learning Model

Goal

Train a PyTorch machine learning model to drive using video input

Data Collection

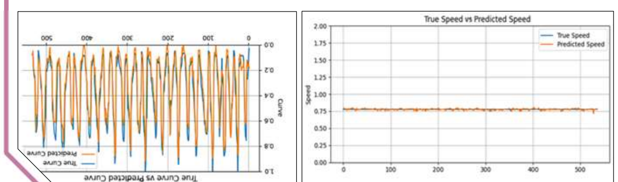
- Drive with pure pursuit algorithm
- Collect training data using the camera and control input
- Synchronize data to associate image frames with control input
- Clean the data and split it into training and test sets

Training the Model

- Train model to learn speed and curve associated with an image
- Output speed and curve when given a **new** image
- Compare with expected value (output from pure pursuit)

Running the Model

- Give the model live image feed and receive signals to control car



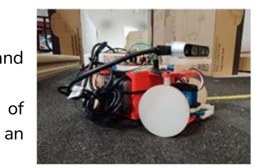
Conclusion & Future Work

Conclusion

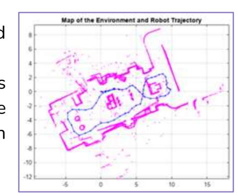
Our hardware and software redesigns resulted in vastly improved performance in operation. Using pure pursuit, we were able to accurately follow a defined path with manageable drift, and gather data to train our imitation model to drive in a loop in the smart city.

Future Work

- Improve design for easier assembly and mass production
- Incorporate an additional input of desired action, such as left or right at an intersection
- Process data with time or previous state, like stopping at stop signs or waiting for traffic lights
- Train to output a desired point instead of direct controls and use pure pursuit
- Implement SLAM (Simultaneous localization and mapping) to improve accuracy of odometry using depth camera



Driving in the Smart City



SLAM

WINLAB