Design Manual

ALVN

Autonomous LiDAR and Vision based Navigator

By
Ciaran Maye
C00253212

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

Table of Contents

Table of Figures

Introduction

This project aims to develop a system for a vehicle to autonomously detect a 'Guideline' and its surroundings and navigate them accordingly. The system uses a Sunfounder PiCar-V, Slamtec RPLidar A2M8 sensor, and a 1080p webcam to develop an 'Obstacle Avoidance System' using image processing, edge/line detection, and LiDAR spatial mapping.

Architecture

The architecture of this project can be described as a modular and object-oriented design, consisting of several interconnected classes representing different functionalities within the autonomous vehicle system. The central class, ALVN, composes the other modules, including LaneFollower, ObstacleAvoider, and PathFollower, and interacts with the hardware components through SunFounder PiCar-V SDK classes.

The project leverages the capabilities of the Sunfounder PiCar-V, Slamtec RPLidar A2M8 sensor, and a 1080p webcam to gather data from the environment. The software is written in Python and utilizes the Sunfounder PiCar-V SDK and RPLidar Python library to interact with the hardware components.

The project's architecture can be broken down as follows:

ALVN class.
This is the main class that initializes and manages all other components, both hardware and software. It establishes the composition relationship with the LaneFollower, ObstacleAvoider, and PathFollower modules and uses the SunFounder_PCA9685, SunFounder_PiCar_V_Front_Wheels, and SunFounder_PiCar_V_Back_Wheels classes to interact with the hardware.

PathFollower class.
This module focuses on following a single line using image processing techniques. It also calculates the steering angle and sends commands to the ALVN class to control the vehicle's movement.

ObstacleAvoider class.
This module is responsible for obstacle detection and avoidance using data from the RPLidar A2M8 sensor. It processes the LiDAR data to identify obstacles, calculates the appropriate steering angle, and sends commands to the ALVN class to control the vehicle's movement.

Components

Hardware

Sunfounder PiCar-V kit for Raspberry Pi 3
Slamtec RPLidar A2M8 sensor
120° wide angle camera

Software

Raspbian operating system
Python 3.10
Sunfounder PiCar-V SDK
RPLidar Python library
OpenCV Python library

User Interface

The user interface is a command-line menu-based system that allows the user to choose between different modes and control the vehicle.

Application Navigation Sequence

Initialization

The components, including the camera, LiDAR, and motors, are initialized automatically through the ALVN class.

Menu

The user is presented with a menu with the following options:

1. Follow a path.
2. Avoid obstacles.
3. Follow path and avoid obstacles
4. Reset Hardware.
5. Quit.

The user chooses a mode, and the vehicle behaves as follows:

Follow a path.
The vehicle follows a single line and maintains its position as close as possible to it.

Avoid obstacles.
The menu presents a follow up question of which obstacle avoidance algorithm to use.
The vehicle travels forward and avoids any obstacles in its path.

Follow path and avoid obstacles.
The vehicle follows a single line while avoiding obstacles.

Reset Hardware
Reset all hardware to default positions.

Error Handling

The program relies on the command-line interface for displaying errors and incorporates multiple exception handling mechanisms. However, it is essential to remember that this is a proof of concept, and error handling is still under development.

RPLidarException: The program handles RPLidarException by restarting the lidar device. If the device encounters an error such as "Incorrect descriptor starting bytes," it stops the lidar, disconnects it, waits for a short period, reconnects it, and starts the motor again before continuing.

KeyboardInterrupt: If the user interrupts the program using a keyboard interrupt (e.g., pressing Ctrl+C), the program stops the car and exits gracefully.

PortNotOpenError: In case the lidar device's serial port is not open, the program attempts to reconnect by stopping the lidar, disconnecting it, waiting for a brief period, reconnecting it, and starting the motor again.

Generic Exception Handling: For any other unexpected exceptions, the program stops the car, displays the error message, and exits. This ensures that the program doesn't leave the car running in an unsafe state.

Diagrams

Class Diagram

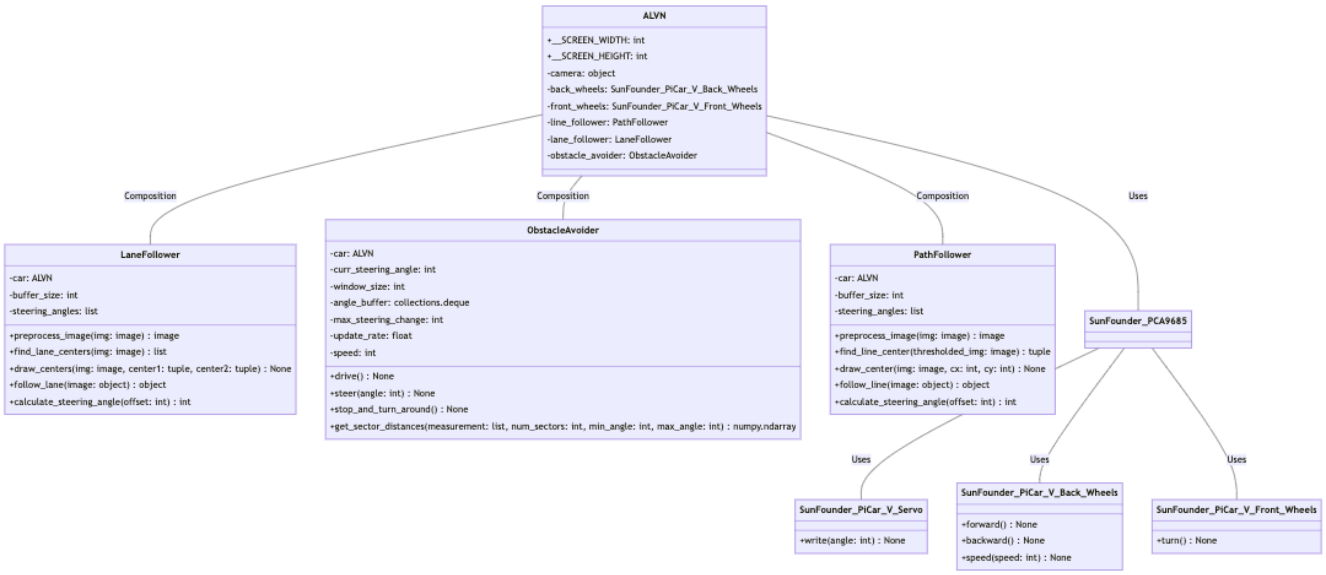

Figure 1 : Class Diagram

State Diagram

This following diagram depicts the different states that the vehicle can transition between during the menu options **“**Follow path and avoid obstacles” and “Follow lanes and avoid obstacles”.
The vehicle starts in the Idle state, waiting for user input. From there, the user can choose to enter one of the states. If the Follow Path or Follow Lane states encounter an obstacle, they will transition to the Obstacle Detection state, where the vehicle will attempt to navigate around the obstacle. Once the obstacle is cleared, the vehicle will return to the original state it was in before encountering the obstacle. Finally, the user can choose to quit the program, which will transition the vehicle to the Shutdown state, where it will stop all movement and power off.

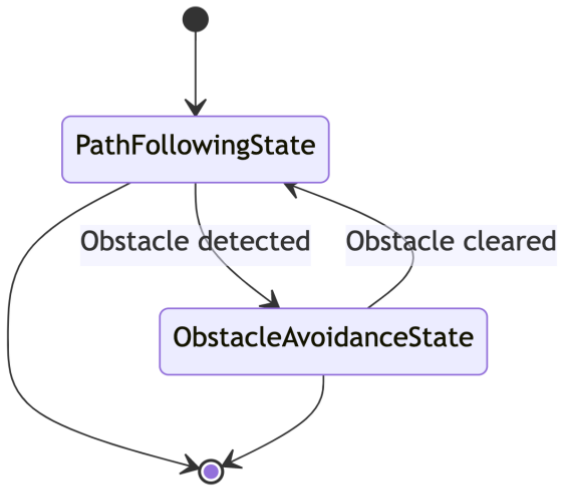

Figure 2 : State Diagram for Path Following and Obstacle Avoidance

Sequence Diagrams

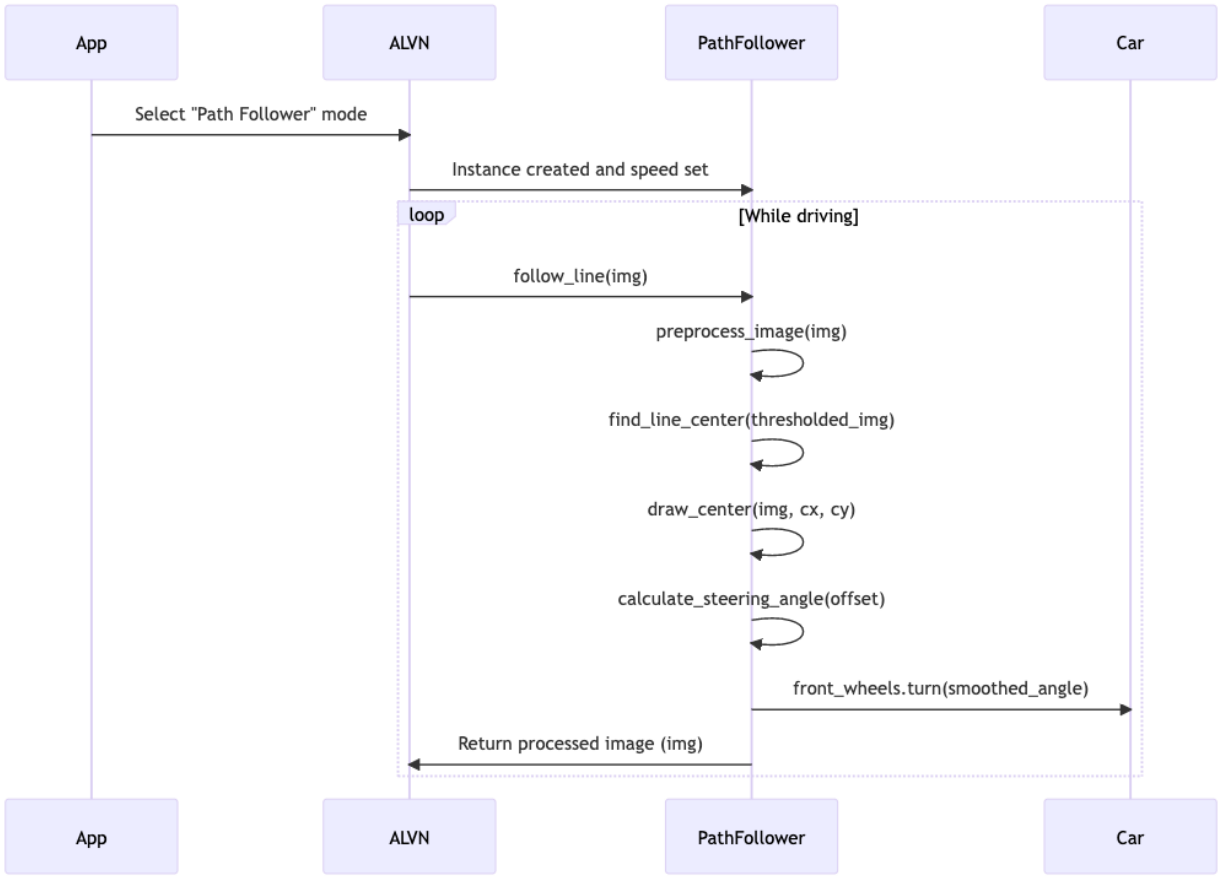

Figure 3 : Sequence Diagram for Path Follower

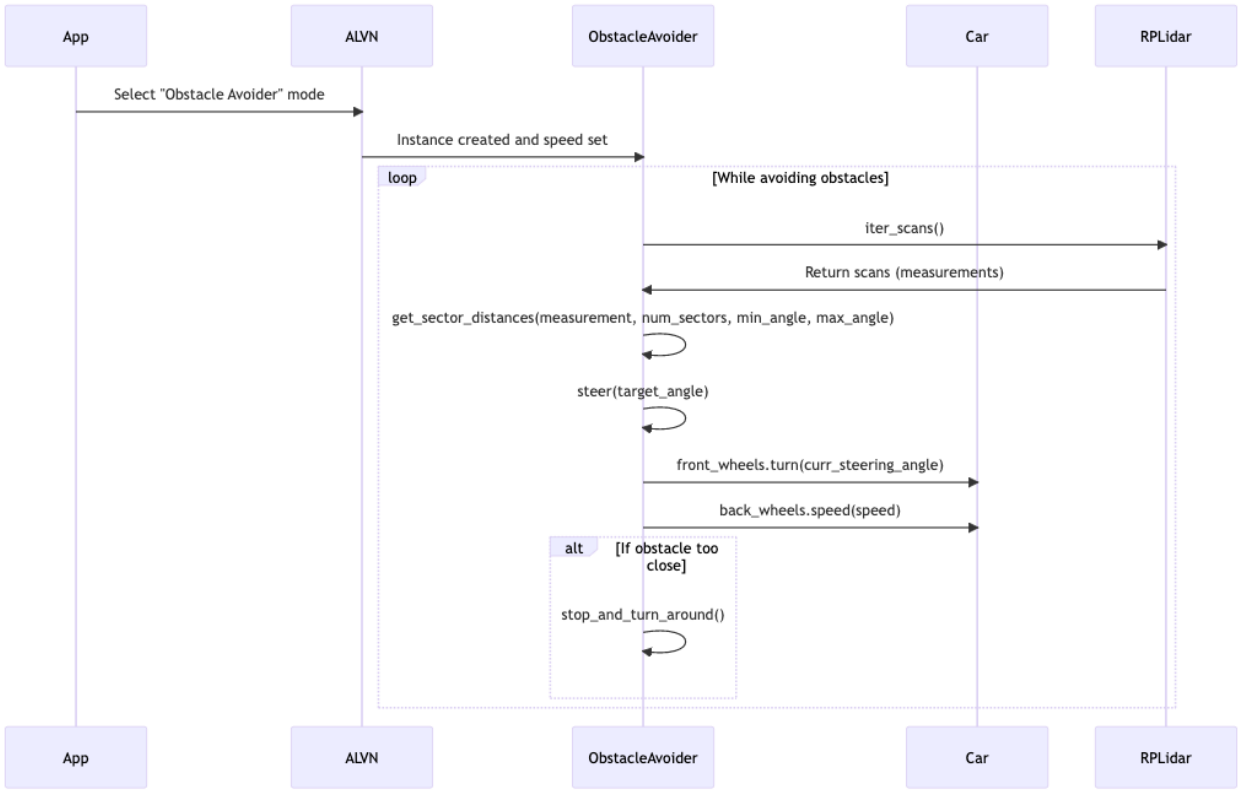

Figure 4 : Sequence Diagram for Obstacle Avoider

3D printed RPLidar mount

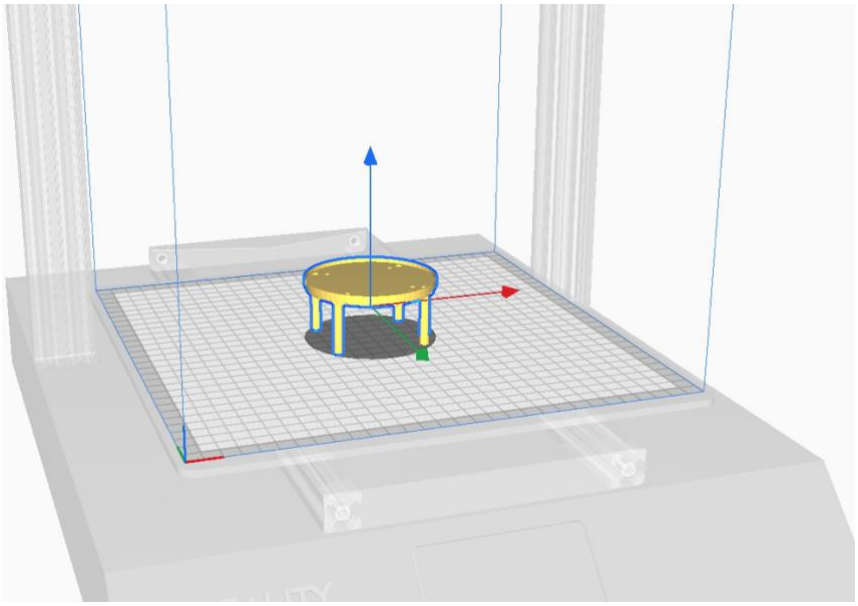

Figure 5: 3D Model 1

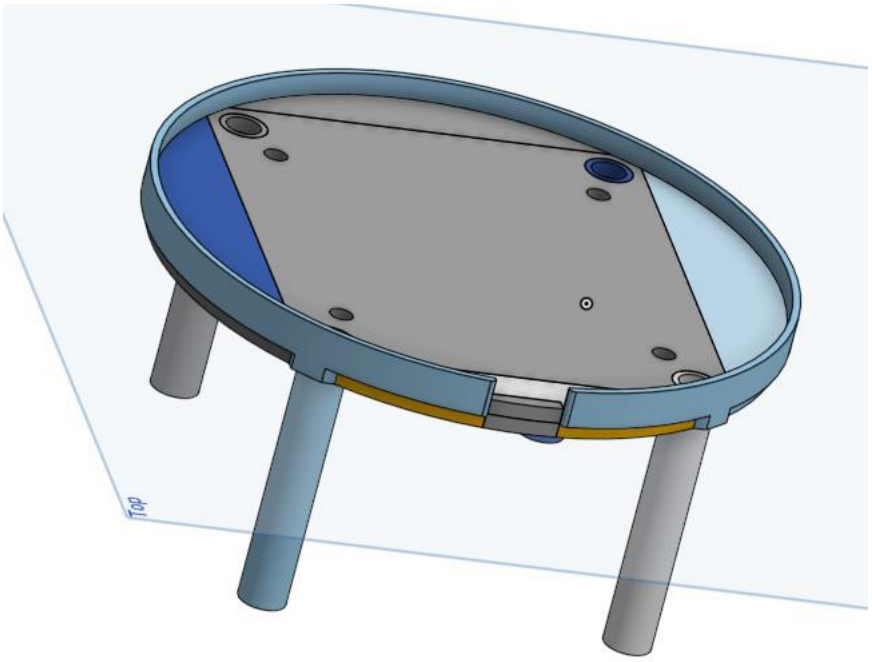

Figure 6: 3D Model 2