# Autonomous Vehicle System

University of Essex

Object Oriented Programming

System Implementation

Description:

The program is to help with the data functions of a self-driving car it includes the following classes: User interface which takes the driver destination choice, Localization class, monitors where the current location is, the Destination class which finds the destination, the navigation class which finds and creates the route and passes the info to data store class which passes onward to the central control unit for decision making to be sent to the vehicle interface or Physical car class which has functionally in the form of steering, acceleration and deceleration. The program also need to uses the sensor class which passes raw data to the perception class for analysis and detection. This is sent to the central control unit for decision making. Which generates corrective responses used by the car.

Running the program:

Requirements:

* Python

1. Open the OOPSelfdrivingcar file
2. Run the SelfDriveCar.py file and follow the prompts.

Development process:

Firstly, understanding what the scope of what the program was and determining what it is I wanted the program to do was a tough first step, this is due to the task being so broad and the functioning system of a self-driving car being so large. I decided to keep the system relatively simple and use the classes that I chose in my UML design as they were broad representations of the key model related to the system.

I started the development process by coding all the classes and define them this took the longest time as I had to change many aspects from each one of the classes to be better able to function with the transferring of data between them.

I organized them based on the user input and the transfer of information towards the vehicle interface or car.

I utilized queues and stacks for the navigation and perception data respectively as this would give the central control unit the best ability to make decisions based on the most relevant information that corresponds to each other.

The data store and control unit classes were important for the requirements of this assignment as it utilizes both the searching and storing algorithms by using the data generated from other classes.

Using the help of the built-in python debugger, I was successfully able to remove all the syntax errors from the program and create two testing files that firstly tested the destination module with assert statements and the secondary testing file tested all the components used in the program together.

When running the program, it should generate continuous corrective responses that will be printed for the user. These responses should be utilized via the car system on the way to the set destination.

I initially wanted the system to also show the angles of steering that should be changed and the changes that should be made to the speed according to the object detected however I was unable to implement this with the given time constrains.

Ultimately, the program represent the outline of a self-driving car system that shows in a basic sense how a full autonomous car system would function.

References:

Reitz, K. and Schlusser, T. (2016) *The hitchhiker''S guide to python: Best practices for development*. Sebastopol, CA: O''Reilly Media, Inc.

Reddy, P.P. (2019) *Driverless car : Software modelling and design using Python and tensorflow*, *EasyChair Home Page*. EasyChair.

Real Python (2023) *Python code quality: Tools & Best Practices*, *Real Python*. Real Python. Available at: <https://realpython.com/python-code-quality/>

*Sorting, searching and algorithm analysis* (2022) *Sorting, searching and algorithm analysis - Object-Oriented Programming in Python 1 documentation*. Available at: https://python-textbok.readthedocs.io/en/1.0/Sorting\_and\_Searching\_Algorithms.html