1 Introduction

This report will provide an overview of the advancements and activities achieved during the fourth month of the competition. The team has improved the project architecture and lane-following algorithm, and now we're using a perspective-based approach using the sliding window technique. This led to increased accuracy and precision; thus, our algorithm is more reliable. We are also working on object detection and transferring our architecture from C++-based to ROS-based.

2 Planned activities

1) Project architecture and transfer to ROS
2) Improving lane following and implementing speed control
3) Adaptive ROI
4) Intersection detection
5) Traffic sign detection
6) Creating a physical and virtual testing environment

3 Status of planned activities

3.1 Project architecture and transfer to ROS

Status: Ongoing 50%

We understood the brain project and did a whole new project that runs only on Raspberry Pi. We took many ideas from the original project architecture that was provided, but we adapted them to fit our algorithms. We have a Decision-maker process that aims to orient the car on the map and choose the process that will send commands to the serial handler. For example, Decision-maker will start the Lane-Follower process, and Lane-Follower will send commands to Serial-Handler. Lane-Follower will throw a trigger when the horizontal line is detected, and Decision-maker has the task of choosing a new process. Although it did work for the qualifications, this architecture is unscalable and hard to debug, so we plan on transferring our code to ROS instead of typing thread-handling from the ground up by ourselves.

Difficulties: We have never worked with ROS, so learning it has proved difficult and slow.

3.2 Improving lane following and implementing speed control

Status: Ongoing 80%

We changed the lane-following algorithm completely and now use a method that involves changing perspective. We perform the sliding window technique on the bird's eye view of the terrain. The program operates in 4 mods when both lines are detected, when only left or right lines are detected, and when none are detected. The algorithm is robust and works well.
We made an algorithm to detect stop lines (horizontal lines) using a sliding window, but now we calculate the middle line and follow it with a sliding window. The vehicle slows down gradually and stops before the stop line.

3.3 Adaptive ROI

Status: Finished

We change ROI when the vehicle gets closer to the stop line, so it won't have problems with steering when there is a crosswalk after the stop line. It is done gradually on every frame, giving precision but computationally expensive.

3.4 Intersection detection

Status: Finished

We detect intersections with the help of detecting stop lines and have made predefined routes for intersection navigation.

3.5 Sign detection

Status: Ongoing 80%

Using the TensorFlow framework, we have built a deep-learning model to recognize traffic signs in the Colab environment. Our first challenge was finding a good training dataset. We have researched several options and picked German Traffic Sign Dataset from Kaggle; since it doesn’t contain all signs, we plan to add our photos to complete it. We are using 2 CNN layers and two dense layers. We have also used the Adam optimizer and dropout technique. The algorithm works well on the classification of 3 sign classes.
If we can acquire a Jetson Nano, we will instead implement a YOLO model that we found online and has excellent accuracy in detecting all the objects the car might encounter.

3.6 Creating a physical testing environment

Status: Ongoing

We will get a 15x15m map, but we are still determining whether we will have enough time before the competition to test and work on our algorithms.

4 General status of the project

We are not very satisfied with the progress since the qualifications, which haven’t been great, primarily due to our inability to find and create a suitable testing environment on which we can continue to work on the more advanced features of our car.

5 Upcoming activities

1) Traffic light detection
2) Finding the path and orientation in the map
3) Implementing remaining traffic signs actions
4) Implementing other advanced features