Programming a Testbed for Contract-Based Design

Jason Song, jasonaeroastro@outlook.com
University High School, Class of 2020
USC Viterbi Department of Electrical and Computer Engineering, SHINE 2019

Introduction

The Cyber-Physical System Design (DesCyPhy) Lab at USC investigates contract-based design methodologies for the development of safer and more reliable embedded systems such as autonomous vehicles and aircraft. This project aims to program a testbed for validating the methodology in the form of a small-scale self-driving car.

Skills Learned

- Basic Python programming
- Connecting via SSH (Secure Shell)
- How to install software on Linux OS by using terminal window
- Engineering mindset:
  - Being patient when faced with difficulties, fixing errors one at a time
  - Being prepared for all possible outcomes
  - Consulting others (online and in-person) for solutions
- Being methodical and brainstorming different scenarios that the self-driving car could find itself in
- I have grown to appreciate the process of asking myself about what it is exactly that I am trying to do instead of scrambling through a maze of code.

A Raspberry Pi computer was used on the vehicle. https://www.raspberrypi.org

How the Self-Driving Car Works

1. The Raspberry Pi camera takes a picture of the track in front of it.
2. The color image is converted to a grayscale image.
3. The grayscale image is inverted. The black lines become white.
4. A graph of the intensities is generated, and a Butterworth filter is used to deal with noise.
5. The car makes a decision:
   - If there is one peak on the left side of the graph, the car turns right. If there is a peak on the right side, the car turns left.
   - If there are two peaks (for two lines), the car goes straight.
   - Speeds are preset. Motor and servo are controlled via pulse width modulation.
6. Back to Step 1 (continuous stream of images and commands)

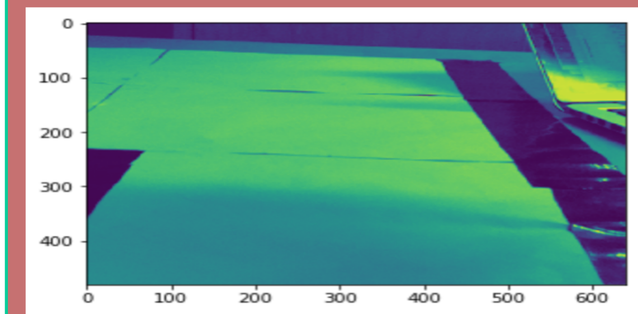


Figure 1

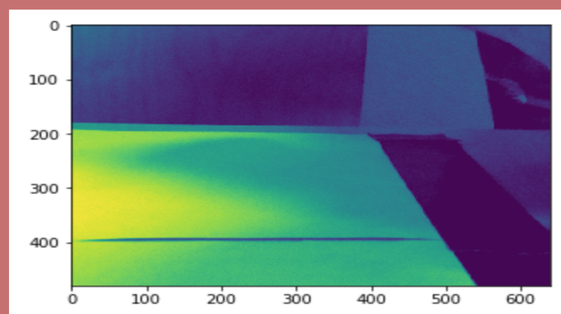
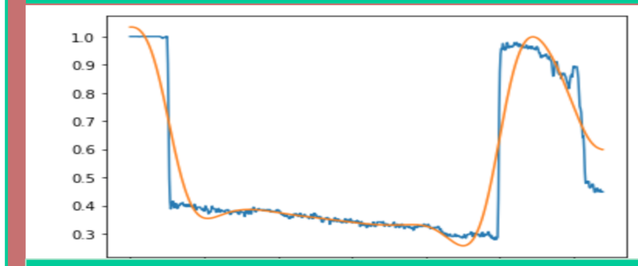
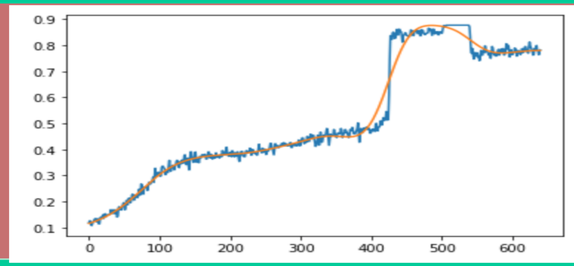


Figure 2



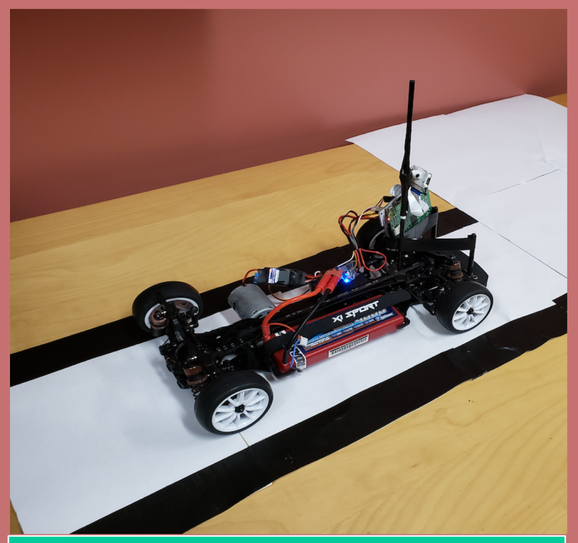


Graphs of intensities generated by the Raspberry Pi computer. The orange curve is generated after application of the Butterworth filter, which deals with the noise in the image. In Figure 1, the two black lines result in two peaks in the graph, guiding the car to go straight. In Figure 2, one black line and peak on the right make the car turn left.

Reflection

The project being my first experience with computer science and hardware, I struggled at first without fully understanding why software was not being downloaded properly, and I was often clumsy with hardware. I was initially frustrated by all the errors appearing on my computer's terminal window.

However, as I started to gain an understanding of computer science and the codes on my screen, I began to appreciate the process of debugging. I grew more patient and tried to be more methodical. My mindset has changed: instead of looking for a straight path to what I want, I am now open to the possible twists and turns. I must say that this process has become rather enjoyable.



Remotely testing the car's servo on a track. PC: Pratham Gandhi

Advice for Future SHINE Students

- Do not hesitate to ask questions and seek clarification.
- Be prepared for progress to come very slowly with many failures.
- When faced with many inexplicable errors, do not panic. Be patient, and deal with them one at a time. I have spent entire six-hour workdays resolving errors.
- Be open to learning new skills and knowledge. You are not here at SHINE to only display how much you know.

Acknowledgements

Thank you to *Professor Pierluigi Nuzzo*, my PhD mentors *Chanwook Oh* and *Yinghua Hu*, my lab mate *Pratham* and *the SHINE staff* for their guidance and collaboration on my project!