**Line follower in Vrep**

Carolyn Cressman and Ruben Vazquez

**INTRODUCTION**

Ruben and I decided to create a virtual line follower robot for our final project. At first, we were going to use ROS to create our robot, but we didn’t know how to use ROS well enough to utilize it as we would like in the one month timeframe allotted.

Ruben found a software called Vrep which is an IDE for creating CAD demonstration of robotic systems. This software is for education use and is free to use. Vrep comes preloaded with several robotic systems pre-made and scripted. These systems include the robotic snake, the line follower, the kilobot, and many other existing projects. The scripts in Vrep can be written in several languages including C/C++, Java, Lua, and Python.

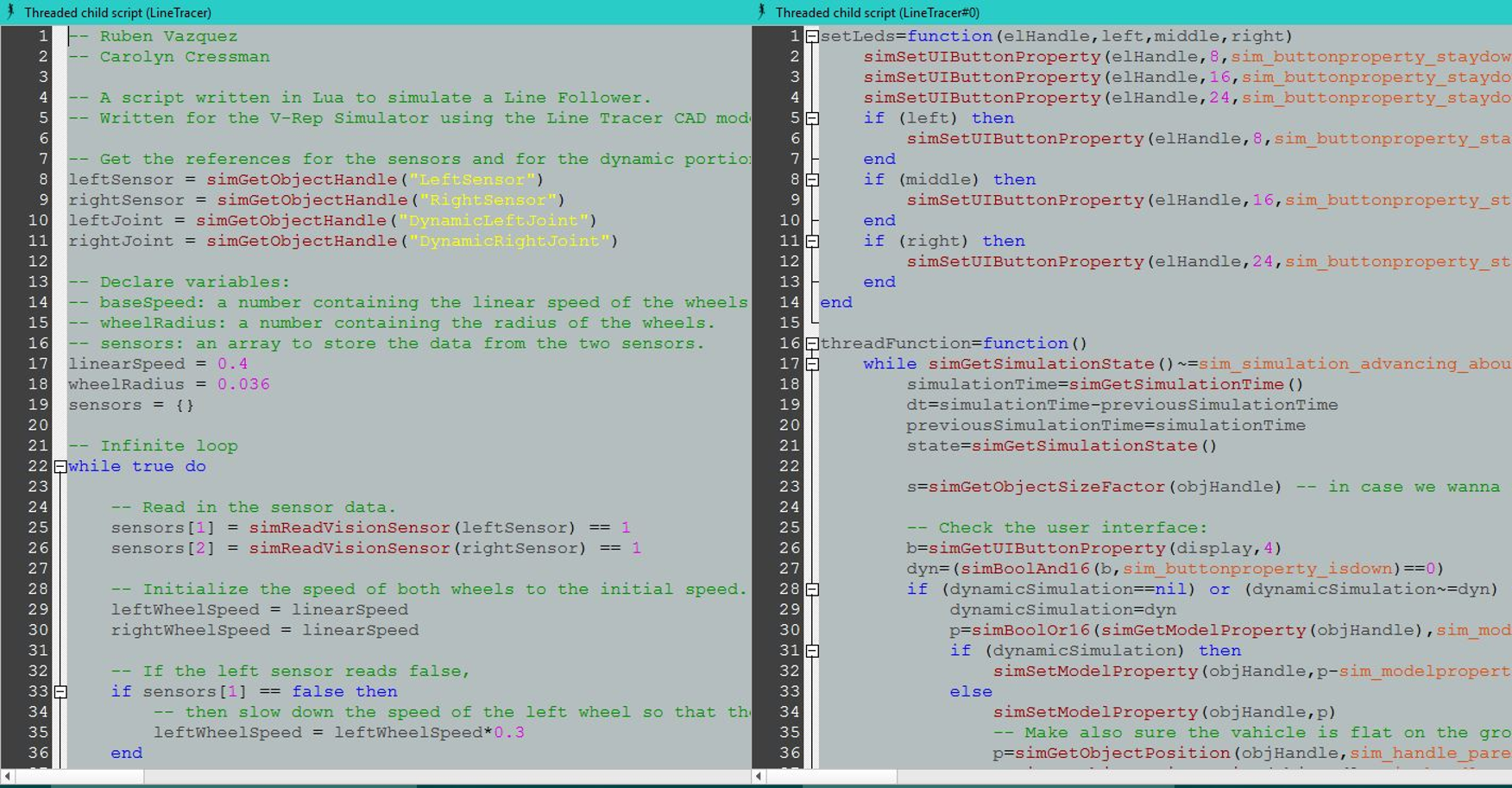
Ruben and I decided to use the existing line follower code as a base. We edited the code to simplify the code and methods. The original line follower code is written in Lua and our edited code is also written in Lua.

**METHOD**

We decided to try to optimize the line follower code by removing one sensor from the original design which has 3 vision sensors. We cleaned the code up to only include what is needed and keep the code simple. Below is an image displaying the two versions of the line follower; on the right, is our optimized version of the original design and on the left, is the original code.

An abbreviated explanation of how our code works is as follows:

* Get references for the sensors and dynamic portions of the wheels
* Declare Variables: base speed, wheel radius, sensor array
* Main code inside infinite loop
  + Read sensors
  + True/False comparison of sensor reading to 1
  + Initialize wheel speeds
  + Adjust wheel speeds based on sensor comparison results
    - If False, the New Speed = Current Speed \* 0.3
    - Else, do nothing



**DISCUSSION**

To test our system, we placed both the original design and our design on the same track together and ran the simulation. To make the results easy to see we colored the original system pink and our system orange. Our design performed as well as the original despite only having 2 vision sensors. In both systems, we did experience an issue where if the simulation ran for a few laps that the line followers would wonder off the track.

If we had more time Ruben and I would’ve liked to attempt making the system work with only one sensor. We would’ve liked to make more tracks and test both systems on them in order to achieve a more accurate and well tested conclusion. Also, if we had the time we would’ve attempted making a better performing and more complicated equation to adjust the wheel speeds.

**CONCLUSION**

In conclusion, our design performed well compared to the original line follower design and the original code can be optimized further by removing one of the three vision sensors on the design. We believe there is room for improvement in both systems though, the current equation to adjust the wheel speeds is too simplified and could be more efficient. Inclusion of computer vision might improve these design; if the line follower used computer vision it could adjust the wheel speed more efficiently by taking the percentage of the sensor image that is the line color into account in the adjustment calculations.