Final Project Report

The text in yellow is newly added.

The text in red is revised.

Group member:

Wanglibo Chen (wach5503@colorado.edu)

Yuwangyang Fan (yufa4450@colorado.edu)

Tongxin Zhu (tozh3775@colorado.edu)

Project name:

Traffic Simulator

Content:

a. Abstract

For this final project, we will create an environment simulating a map with objects. The objects act as traffic lights. They function as the indicator of decision making. The robot should find its own path and make its decision along the path. The robot will start at a spot and move on a map with grids which contains traffic lights on designated interactions. It will proceed when its sensor detects a green light and change direction or wait until the light changes to green when its sensor detects a red light. We will set the color of the lights randomly changing to allow the robot to make decisions based on the sensor detection. The robot will finally stop at the target which we set to be the destination.

The robot will go to the goal point from the start point with a goal seeking algorithm which is affected by bearing error, heading error and distance error. And there are several traffic lights on the map, when the robot gets close to the traffic light, it will make an action depending on the state of the traffic light -- red signal or green signal, and then it will keep going to the target.

b. Equipment

E-puck

Webot environment

Python-based control

Light sensor (Does not use)

Generic Traffic Light object (Newly added)

c. Deliverables and Implementation Plan

i) Webot world set up: (Lead: Wanglibo Chen)

1. Need a E-puck with light sensors.
2. Need a map with grids.
3. Need blocks with colors as the signal light. The color will change every few minutes. => change to object Generic traffic light.
4. Need a start position and a target position.

ii) Implement robot-side controller: (Deadline: 12/11/2020)

1. Line following (Tongxin)

    1)Collect the reading from the sensor.

    2)Follow the line depending on the reading from the sensor.

    We find that in the grid map the line following algorithm meets a problem that there are too many cross lines to disturb the sensor. So we decided to use Goal seeking which we learned in lab3.

We meet a problem that the robot cannot reach the goal point, and the main reason could be the function of the target position collection/ calculation is not perfect.

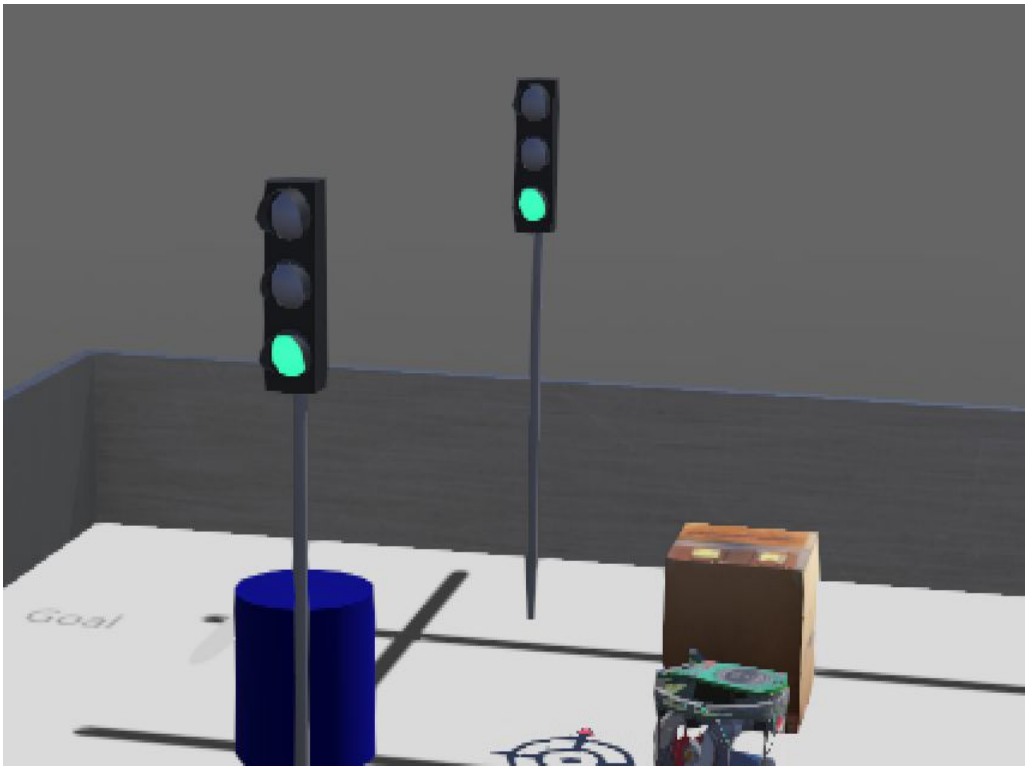
2. Light sensor detection (Wanglibo)

    1)Collect the reading from the light sensor.

    2)Make movements(stop/turning) depending on the reading with the color it represents.

    We find that there is an object called Generic Traffic Light which already has the signal swift function and we decide to use the traffic light instead of the color blocks.

    1) Add the object Generic Traffic Light into world



2) Revise the supervisor, get the light state of the traffic light such as “Green” , “Red”.

```
def init_supervisor():
    global supervisor, robot_node, target_node, A_node, B_node, C_node
    supervisor = Supervisor()

    # do this once only
    robot_node = supervisor.getFromDef("EPUCK")
    target_node = supervisor.getFromDef("Goal")
    A_node = supervisor.getFromDef("lightA")
    B_node = supervisor.getFromDef("lightB")
    C_node = supervisor.getFromDef("lightC")
    if robot_node is None:
        sys.stderr.write("No DEF e-puck node found in the current world file\n")
        sys.exit(1)
```

3) Add the stop and go algorithm to control the robot. When the robot gets close to the traffic light and acts depends on the state of the traffic light.

```
#start light section
a_pose, b_pose, c_pose = csci3302_final_supervisor.supervisor_get_light_pose()
a_state, b_state, c_state = csci3302_final_supervisor.supervisor_get_light_state()

a_dis = sqrt(pow(a_pose[0] - pose_x, 2) + pow(a_pose[1] - pose_y, 2))
b_dis = sqrt(pow(b_pose[0] - pose_x, 2) + pow(b_pose[1] - pose_y, 2))
c_dis = sqrt(pow(c_pose[0] - pose_x, 2) + pow(c_pose[1] - pose_y, 2))

if min(a_dis, b_dis, c_dis) < 0.05:
    if a_state == "red" or b_state == "red" or c_state == "red":
        rightMotor.setVelocity(0.0)
        leftMotor.setVelocity(0.0)
        right_wheel_direction = WHEEL_STOPPED
        left_wheel_direction = WHEEL_STOPPED

#end light section

print(a_dis, b_dis, c_dis)
```

3. Code framework and combination (Yuwangyang)

    1)Import packages for robot, python libraries.

    2)Set up and define parameters for the robot and the environment.

    3)Combine functions implemented by other members.

d. Demo

We will give an instruction to the robot asking the robot from start point to target point. The robot should follow the calculated path and the indication of the traffic lights. The demo includes the representation of the map or the world, the algorithm of the controller, the structure of the controller, and the final resulting action of the robot. We will make a video of the expected result of the robot and a video of the actual result of the robot. According to the class format, we can either share our screen in zoom to present our project or upload the video of the result.

Link of Github: https://github.com/TongxinZHU/CSCI3302-Final-Project