CE 6308 / CS 6396 / EEDG 6308

Real-Time Systems

Assignment 3

Spring 2023

Instructor: Farokh Bastani

Deep Padmani (DMP210005)

Rubina Parveen (RXL220014)

Roshni Johnson Nambiaparambil (RXN200022)

Venkat Kishan Bommali (VXB200009)

Sai Vishwanth Yalamanchili (SXY210012)

Objective

The objective of this task is to program the Ego vehicle to navigate through a parking lot, while searching for an available parking spot and then park the vehicle in the empty space, all using only the sensor data and actuators provided by the RTOS. The constraints of the task include staying within the boundaries of the parking lot, avoiding collisions with parked cars, and avoiding collisions with pedestrians. The program should enable the vehicle to scan the parking lot for open spots, evaluate whether the space is large enough for the vehicle, and maneuver the vehicle into the empty spot safely and efficiently

Requirements

- Problem 1 – We were asked to search for the empty parking spot in the parking lot.
- Problem 2 – We had to park the Ego vehicle safely in the parking lot.

Designated Sensors and Actuators

- **Sensors:** DeviceRegsitry.pixels, DeviceRegistry.compass, DeviceRegistry.speedometer, DeviceRegistry.lidar, DeviceRegistry.gps
- **Actuators:** DeviceRegistry.speedControl, DeviceRegistry.brakeControl, DeviceRegistry.steeringControl.

Implementation

For the implementation of this project, we created the 5 tasks given below using the sensors, actuators and state of the Ego Vehicle.

Task List

```
protected TaskInterface[] tasks = new TaskInterface[]
    {
        new InitialPosTask(),
        new ObstacleDetect(),
        new RightWallFollow(),
        new WallFinder(),
        new SearchParkingSlot(),
        new ParkingTask(),
};
```

Sensors Used

```
devices.pixels
devices.compass
devices.lidar
devices.gps
```

The **`device`** is an instance of **`DeviceRegistry`**.

Actuators Used

```
devices.speedControl
devices.steeringControl
devices. brakeControl
```

To store the current state of the vehicle, we are using Ego Vehicle's memory and we created the states as listed below.

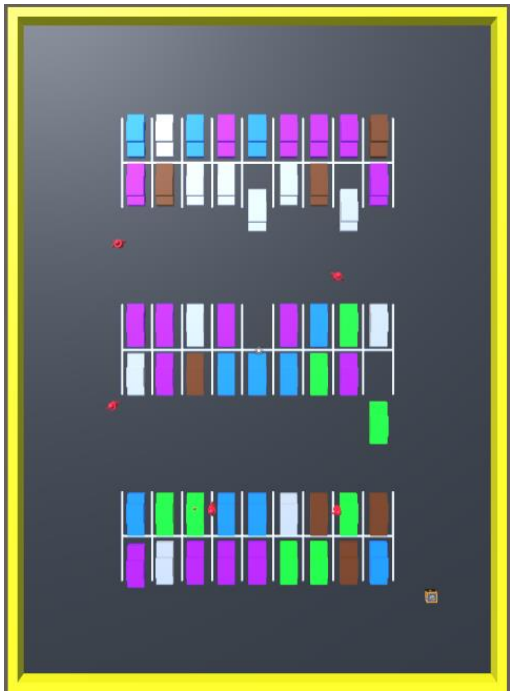
```
devices.memory
```

```
STATE_INITIAL_ORIENTATION = 0;
STATE_TAKING_INITIAL_POSITION = 1;
STATE_INITIAL_STATE_REACHED = 2;
STATE_INITIAL_DIRECTION_CALIBRATION = 3;
STATE_MOVING_FORWARD_WITH_3F_SPEED = 4;
STATE_OBJECT_DETECTION = 5;
STATE_OBJECT_DETECTED = 6;
STATE_WALL_DETECTED = 7;
STATE_CAR_DETECTED = 8;
STATE_MOVING_FORWARD_WITH_1F_SPEED = 9;
STATE_LANE_DETECTION = 10;
STATE_TURNING_RIGHT = 11;
STATE_TURNING_LEFT = 12;
STATE_U_TURN_1 = 13;
STATE_ U_TURN_2 = 14;
STATE_PARKING_DETECTED = 15;
STATE_READY_FOR_PARKING = 16;
STATE_PARKING_EGO_VEHICLE = 17;
STATE_PARKING_DONE = 18;
STATE_PEDS_DETECTED = 19;
```

Task 1: `InitialPosTask()`

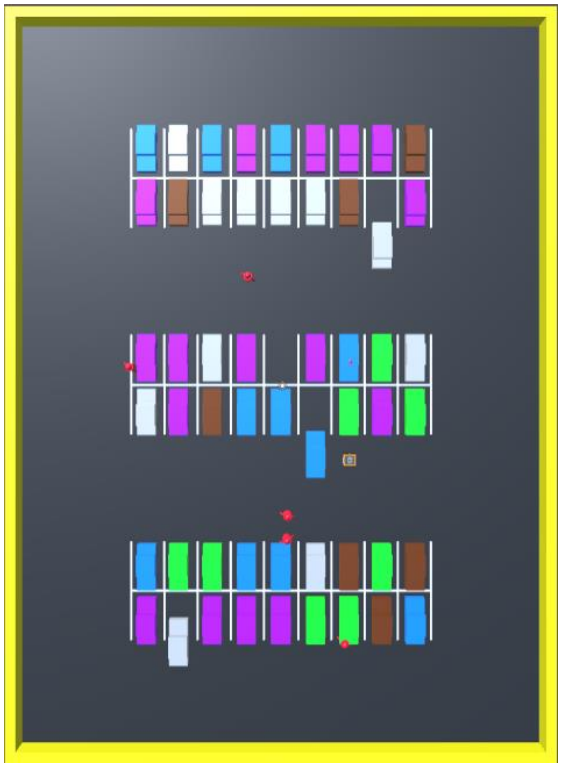
Using this task, the vehicle will take its initial position by aligning itself away from the wall and near to the lane one. The vehicle will move left or right depending on its position and according to value of **`compass`**.
Using the states `STATE_INITIAL_ORIENTATION = 0` and `STATE_TAKING_INITIAL_POSITION = 1` it will orient itself towards south and move towards lane one. Using the `STATE_INITIAL_STATE_REACHED = 2` and `STATE_INITIAL_DIRECTION_CALIBRATION = 3` it will start facing the east direction.

Dependencies: `DeviceRegistry.compass`, `DeviceRegistry.gps`, `Device.memory`, `DeviceRegistry.steeringControl`



The vehicle has taken the initial position to move forward in the lane 1.

Task 2: `ObstacleDetect()` Using this task, the vehicle can detect obstacles like pedestrians and cars, and at that time, it will wait until that object cannot clear the vehicle route. The vehicle will perform this task using lidar and pixel sensors, as shown below.



Object Detection by ego vehicle.

```
// return true if car in front of vehicle
//(0, 124, 255) (0, 255, 27) (172, 0, 255) (192, 192, 192) (101, 33, 0)
if((devices.pixels[5,4,0] == 0 && devices.pixels[5,4,1] == 124 && devices.pixels[5,4,2]
== 255) ||
(devices.pixels[5,4,0] == 0 && devices.pixels[5,4,1] == 255 && devices.pixels[5,4,2] ==
27) ||
(devices.pixels[5,4,0] == 172 && devices.pixels[5,4,1] == 0 && devices.pixels[5,4,2] ==
255) ||
(devices.pixels[5,4,0] == 192 && devices.pixels[5,4,1] == 192 && devices.pixels[5,4,2] ==
192) ||
(devices.pixels[5,4,0] == 101 && devices.pixels[5,4,1] == 33 && devices.pixels[5,4,2] ==
0)||
(devices.pixels[4,4,0] == 0 && devices.pixels[4,4,1] == 124 && devices.pixels[4,4,2] ==
255) ||
(devices.pixels[4,4,0] == 0 && devices.pixels[4,4,1] == 255 && devices.pixels[4,4,2] ==
27) ||
(devices.pixels[4,4,0] == 172 && devices.pixels[4,4,1] == 0 && devices.pixels[4,4,2] ==
255) ||
(devices.pixels[4,4,0] == 192 && devices.pixels[4,4,1] == 192 && devices.pixels[4,4,2] ==
192) ||
 (devices.pixels[4,4,0] == 101 && devices.pixels[4,4,1] == 33 && devices.pixels[4,4,2] ==
0)){
            return true;
        }
        return false;
    }

public bool checkPedestrains(DeviceRegistry devices)
{
     // return true if pedestrain in front of vehicle
  if((devices.pixels[5,4,1] == 0 && devices.pixels[5,4,0] == 255 &&devices.pixels[5,4,2]
== 27) ||
        (devices.pixels[4,4,1] == 0 && devices.pixels[4,4,0] == 255 &&
devices.pixels[4,4,2] == 27)){
            return true;
        }
        return false;
    }
```

Task 3: `RightWallFollow()`

We have used the right-hand rule or the wall follower strategy therefore this task moves the ego vehicle in a straight path in the lanes. The states of the Ego vehicle are as follows -

`STATE_MOVING_FORWARD_WITH_3F_SPEED = 4;` For moving in the lanes with speed 3f

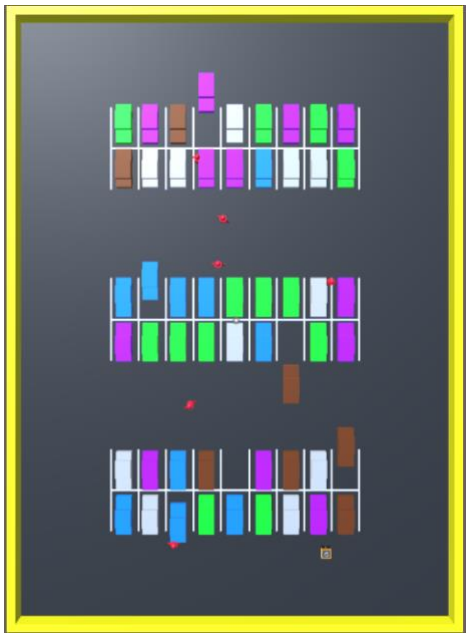
`STATE_OBJECT_DETECTION = 5;` For detects if there is an object in the path

`STATE_MOVING_FORWARD_WITH_1F_SPEED = 9;` For moving towards another lane with 1f speed

`STATE_LANE_DETECTION = 10;` Detects the other lanes

`STATE_U_TURN_1 = 13;` Used while moving from lane 2 to 3 and 4 to 5

Dependencies:
`DeviceRegistry.lidar`, `DeviceRegistry.gps`, `Device.memory`, `DeviceRegistry.pixels`



Ego Vehicle is moving in the forward direction using the right-hand rule.

Task 4: `WallFinder()`

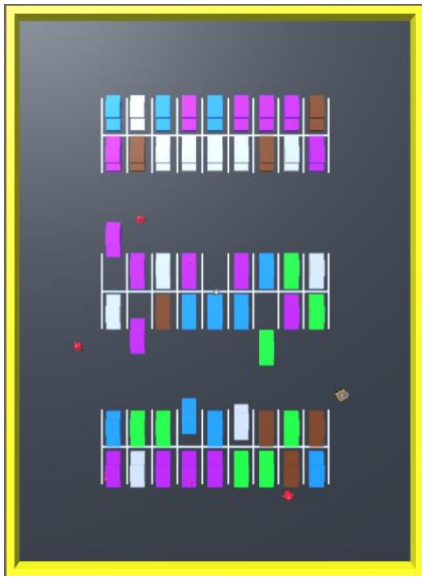
As the name suggests, the Ego vehicle is trying to find the outer walls. According to the GPS values, it will align itself away from the wall. Following states are used for this task:

`STATE_OBJECT_DETECTED = 6;` Indicates that something is in the path of the ego vehicle.
`STATE_WALL_DETECTED = 7;` Tells the Ego vehicle that the object is a wall

`STATE_TURNING_RIGHT = 11`; Taking a right turn away from the wall
`STATE_TURNING_LEFT = 12;` Taking a left turn away from the wall

`STATE_ U_TURN_2 = 14`; Using this it will take a left turn after detecting the lane

Dependencies:
`DeviceRegistry.compass`, `DeviceRegistry.gps`, `Device.memory`, `DeviceRegistry.steeringControl`.



The Ego vehicle is taking a U turn to travel from lane 2 to lane 3

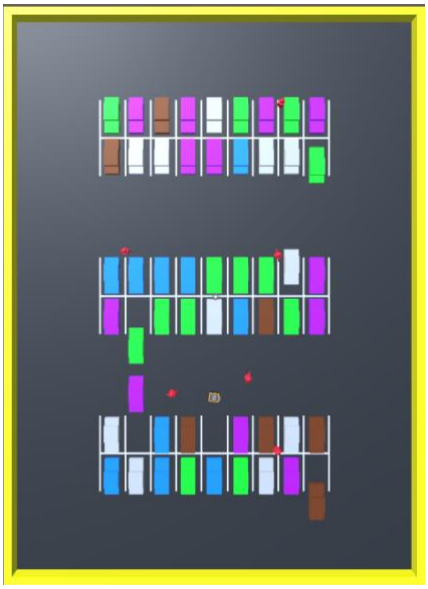
Task 5: `SearchParkingSlot()`

Using this task, the Ego vehicle is searching for the empty parking slot using lidar sensors 3 and 4. If it detects the empty space, it stops there. The states used here are:

`STATE_OBJECT_DETECTION = 5;` Used for detecting the object

`STATE_PARKING_DETECTED = 15;` Indicates that an empty spot is detected

Dependencies:
`DeviceRegistry.steeringControl, DeviceRegistry.lidar, DeviceRegistry.memory, DeviceRegistry.gps`



The Ego vehicle has detected empty parking slot and is now starting to orient itself towards the parking slot.

Task 6: `ParkingTask()`

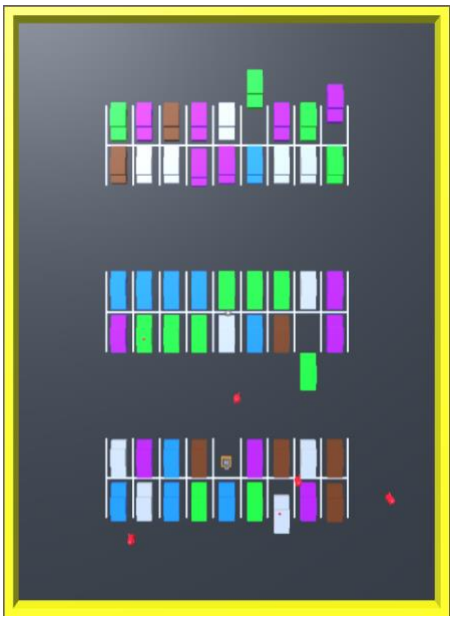
We are using this task to park the Ego Vehicle into the empty parking slot. The states we used here are:

`STATE_READY_FOR_PARKING = 16;` Using this state the ego vehicle will orient itself towards the parking slot.

`STATE_PARKING_EGO_VEHICLE = 17;` The state helps the Ego vehicle to move forward while detecting the car in front of it using the lidar.

`STATE_PARKING_DONE = 18;` This is the state of the Ego vehicle when it is successfully parked.

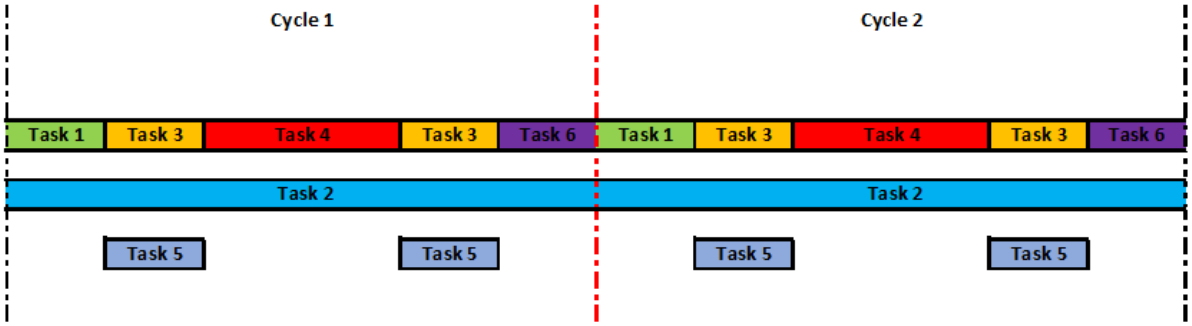
Dependencies:
`DeviceRegistry.compass, DeviceRegistry.steeringControl, DeviceRegistry.memory, DeviceRegistry.lidar, DeviceRegistry.gps`



The Ego vehicle is parked.

Task Graph for 2 cycles of simulation

Cycle 1



Task 1	InitialPosTask()
Task 2	ObstacleDetect()
Task 3	RightWallFollow()
Task 4	WallFinder()
Task 5	SearchParkingSlot()
Task 6	ParkingTask()

Task 1: InitialPosTask ( )

Task 2: `ObstacleDetect` ( )

Task 3: RightWallFollow( )

Task 4: WallFinder( )

Task 5: SearchParking( )

Task 6: ParkingTask( )

Cycle 2 is same as Cycle 1