Simulated Parking Report

B09901133 林天行 B09902005 盧冠綸

December 26 2022

1 Abstract

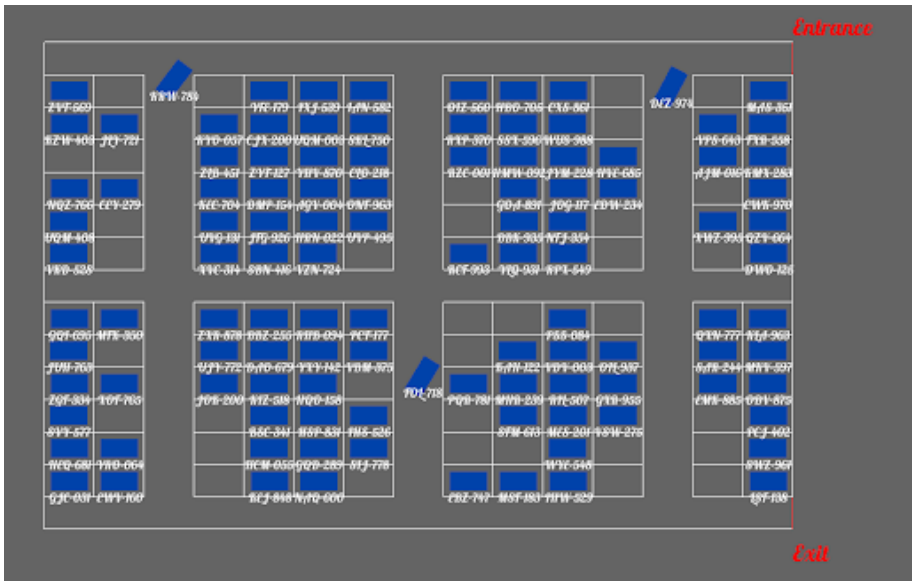
Variety applications regarding intelligent vehicles were brought up in the lectures. However, there isn't much mention about parking system for intelligent vehicles. Thus, in this project, we implement a parking system and show by simulation that it is a good solution to manage the parking of intelligent vehicles.

2 Introduction

As the use of autonomous car becomes widespread, there are clearly some other existing system that may have some room for improvement. One of them is the parking problem of self-driving car.

Currently, it seems that intelligent vehicles can work well in traditional parking lots. However, we start to wonder is it able to design a new parking lot for self-driving cars, which may be able to increase convenience, reduce redundant space in a parking lot, and maximize the number of cars a parking lot can contain.

Motivated by this idea, we do a simulation of a parking lot for intelligent vehicles. You can see `https://www.youtube.com/watch?v=Z2qO128PWHM` for the demo.



1: An image of our parking lot

3 Source code

The source code of the implementation can be found in

`https://github.com/B09902005/parking`.

How to install the environment:

- **Linux**:

```
sudo apt-get install make g++
sudo add-apt-repository ppa:allegro/5.2
sudo apt-get install liballegro*5.2 liballegro*5-dev
```

- **Mac OS**:

```
brew install allegro
brew install pkg-config
```

- **Windows**: Currently, we still do not have Windows version of the code. (This is also our future goal)

How to compile and run the code:

```
make
./bin/main
```

4 Implementation

1. **environment:** We use C++ to implement the parking lot, and we use the library "Allegro 5" to visualize it.

2. **structure:** In our files, most of our code are in `main.cpp`, `menu.cpp`, `parking.cpp`, and `car.cpp`.

   `main.cpp` controls the switching between menu and the scene of the parking lot. `menu.cpp` deals with the menu (The scene that has "press any key to play" on the screen). `parking.cpp` deals with what the parking lot does. `car.cpp` deals with the behaviors of the cars.

3. **parking lot:** The parking lot has the information of all the cars and all the parking spaces. If there is a car coming in, the parking lot will allocate an empty parking space for the car. If the parking is full, then the car will be stuck at the entrance until there is a car leaving the parking lot.

4. **cars:** In our code, every car is an object, which has the following attributes:

   - x (and y): the location of the car.
   - angle : the direction the car is facing at.
   - brake_timer : if nonzero, then the car is slowing down. For every tick, The value of brake_timer approaches zero by 1 if nonzero.
   - wheel_timer : if nonzero, then the car is turning left or right. For every tick, The value of wheel_timer approaches zero by 1 if nonzero.

- cell : the parking space that the car is going to.
- state : what the car is doing (the car may be entering the parking lot, leaving the parking lot, or letting the inner car out)
- priority : whether the car is waiting for other cars.

After receiving the destination (the parking space allocated, or the exit of the parking lot), the car will decide its route, so it'll know where to turn, to slow down, and to stop. For every tick, it moves ahead (x and y will be modified) for a little bit according to its angle and brake_timer.

5. **communication:** In our implementation, we use V2I for the communication between cars and the parking lot. That is, the parking lot has the information of all the cars. If a car wants to enter or leave the parking lot, the parking lot will know and allocate a destination for the car.

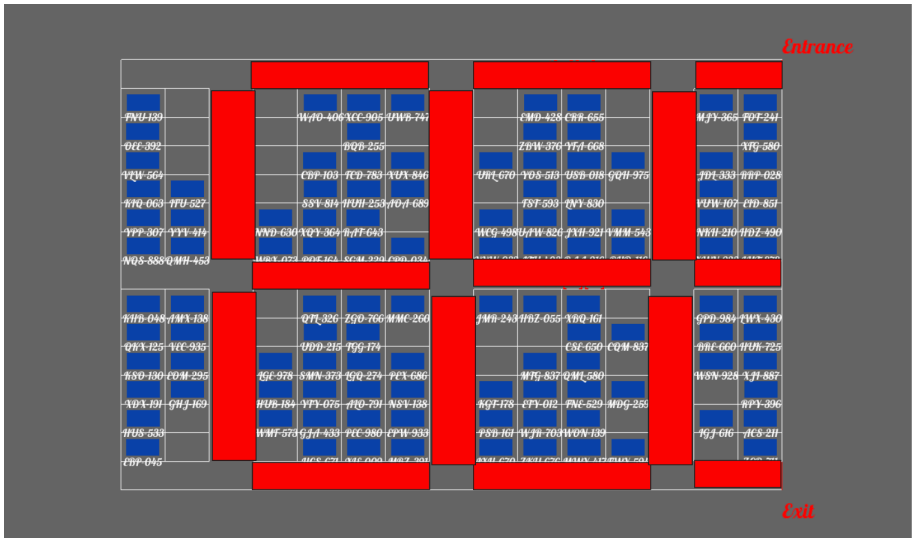
   And, if an inner car wants to leave the parking lot, but it is blocked by the car in front of it (as the picture below shows, when the car in the red rectangle wants to get out, it will be blocked by the car on its left), then it will inform the parking lot, and the parking lot will send a signal to the outer car to let the inner one out.

2: The car in the rectangle wants to go out

You can check `https://youtu.be/orxvTwTLWdI` to see how it works.

In our implementation, we do not use V2V because V2I is already enough to let the parking lot to work smoothly. However, in real world, we think it is still necessary to have V2V to make a better efficiency and prevent from some potential dangers in case of some errors of the parking lot.

6. **conflict zones:** We use conflict zones to avoid collisions. In our implementation, each red rectangle in the picture below can be seen as a conflict zone.

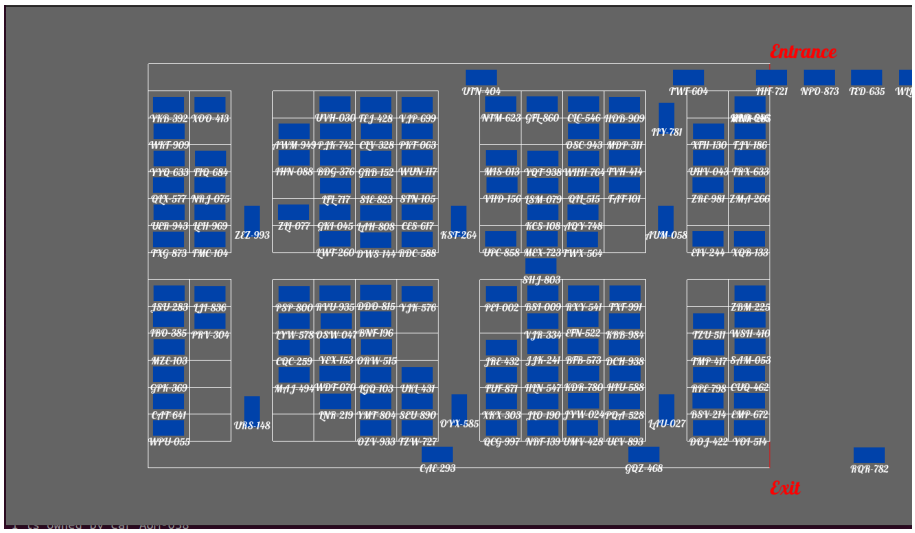


3: Conflict zones

Only one car can be in a conflict zone at any time period. When a car wants to enter a conflict zone, it has to ask for the right to use the conflict zone first. If there is another car in the conflict zone, then the car will not get the right. This way, it has to wait until that car leaves the conflict zone.

We set the conflict zones on each road sections instead of intersections (which is different from the course slides) because cars may leave its parking space at any time, and we do not know when will a car leave. Thus, if we don't set the conflict zones on those road segments, the leaving cars will collide with the car passing through the road segment.

5 Bugs



4: When cars get stuck inside the parking lot

As mentioned in our presentation, there is one time that all the cars will be blocked in the entrance (as the image above) when we run the code. However, we can't reproduce the bug again, we have tried several times and all of the results went well. It makes us hard to tell whether it is because there are some errors in our code or because this is a special scenario not in our consideration.

Currently, we are still figuring a way to find out where the bug is. We think that using several fixed seeds might help us to find it.

6 Limitation

Clearly, there are some limitations about the parking lot of this simulation. The first one is that the parking lot only works for autonomous car, as traditional cars can't receive or follow the instructions from the parking lot. The second one is what if human drivers decide to take over the control? It is very likely that human will care most about their own interest. In the parking case, their interest might

be the time to find a parking spot or the waiting time to leave the parking lot. Thus, human drivers might just ignore the space allocated by parking lot and park their car in the nearest space even if the parking lot allocated it for an other car. The third problem needs to put into consideration is if there is an autonomous car that is out of control (this case doesn't involve human, such as one of its component malfunctions) and couldn't follow or miss the instruction of the parking lot, it will probably end up all the cars are blocked and cars can't go out, as there is only one exit now.

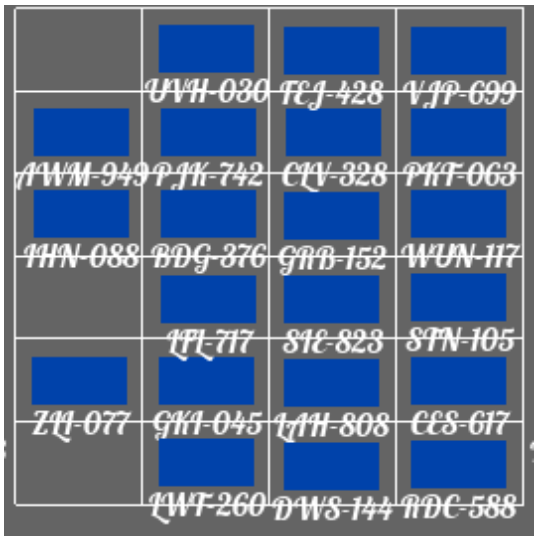
7 Conclusion

In this project, we have simulated a parking lot for intelligent vehicles. We use Allegro 5 to visualize the environment. In our implementation, the parking lot has the information of all the cars and all the spaces. Every car is an object, that has several attributes helpful for parking the car. For communication, we use V2I between cars and the parking lot. An important feature of this parking lot is that if an inner car wants to leave, but blocked by the outer car, the parking lot will know it and tells the outer car to let the inner car go out. We use conflict zones to avoid collision, and the conflict zones are set on each road section.

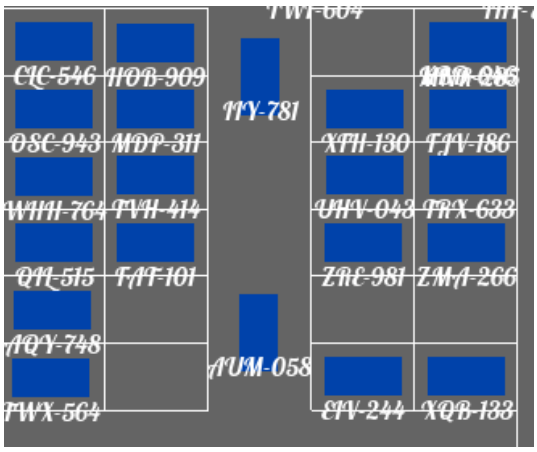
However, there are several limitations. The parking lot now can only park autonomous car. If human drivers take control over the autonomous car or there is malfunction of the components in a car, there might occur collision or all the other cars were blocked by one single car.

In our design, we can park more cars in the parking lot since we make more use of the space. In the traditional parking lot, a design of parking lot such as figure 5, which has more than 2 consecutive parking spaces in a row, can't exist, since the inner car may be stuck inside forever if no outer car wants to get out. Thus, most

of the parking lots use the design in figure 6, which only has 2 consecutive parking spaces in a row, to ensure that any car can go out whenever it wants.



5: More efficient space usage of parking lot



6: Space usage of traditional parking lot

8 Future goals

1. Make the program able to run on Windows operating system.
2. Find the reason why the bug mentioned above happens and fix it.
3. Consider the scenario when there are both human-driving cars and self-driving cars.

4. Develop an error-detect system to detect if there is a selfish human driver takes over control or there are cars malfunctioning.

5. Add V2V communication for better efficiency and prevent potential dangers in case of some errors of the parking lot.

9 Contributions

member	contributions
B09901133 林天行	Minor of the code, fix some bugs, presentation, Major of the report
B09902005 盧冠綸	Majority of the code, presentation, report

10 References

The codes under the `src/utils` directory (`imageProcess.cpp`, `imageProcess.hpp`, `log.cpp`, `log.hpp`), which are aim for drawing the image using Allegro 5 and generating logs on terminal, are based on `https://github.com/B09902005/sprout_hw2`, and this part is the completed work before this semester.