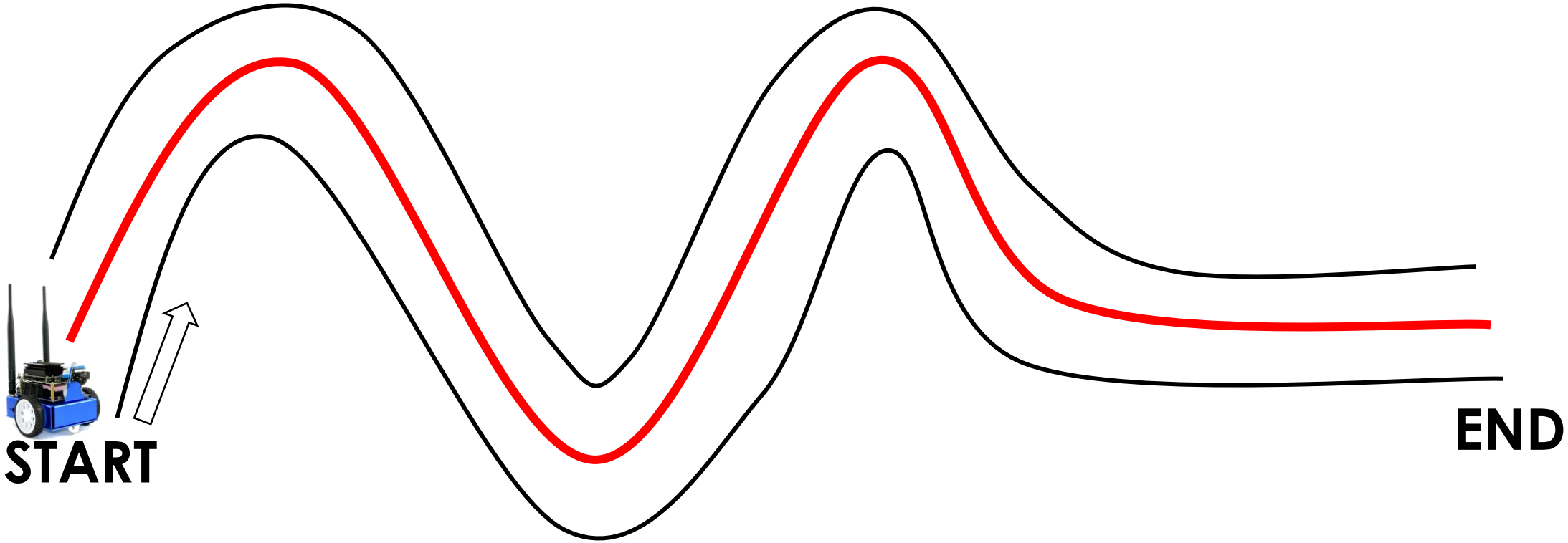
# Robotic Navigation and Exploration

Final Project

Min-Chun Hu anitahu@cs.nthu.edu.tw
CS, NTHU

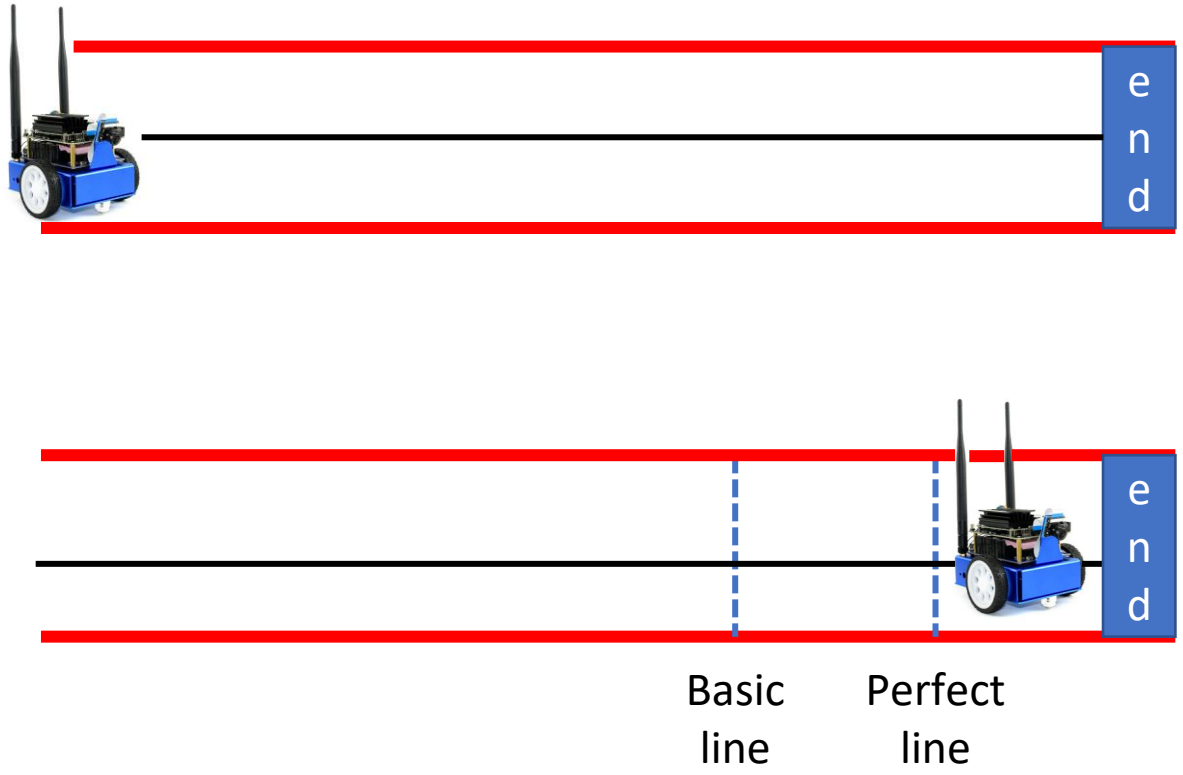
# Basic-Automatic tracking schematic map

- You should let your jetbot follow the line in the demo.



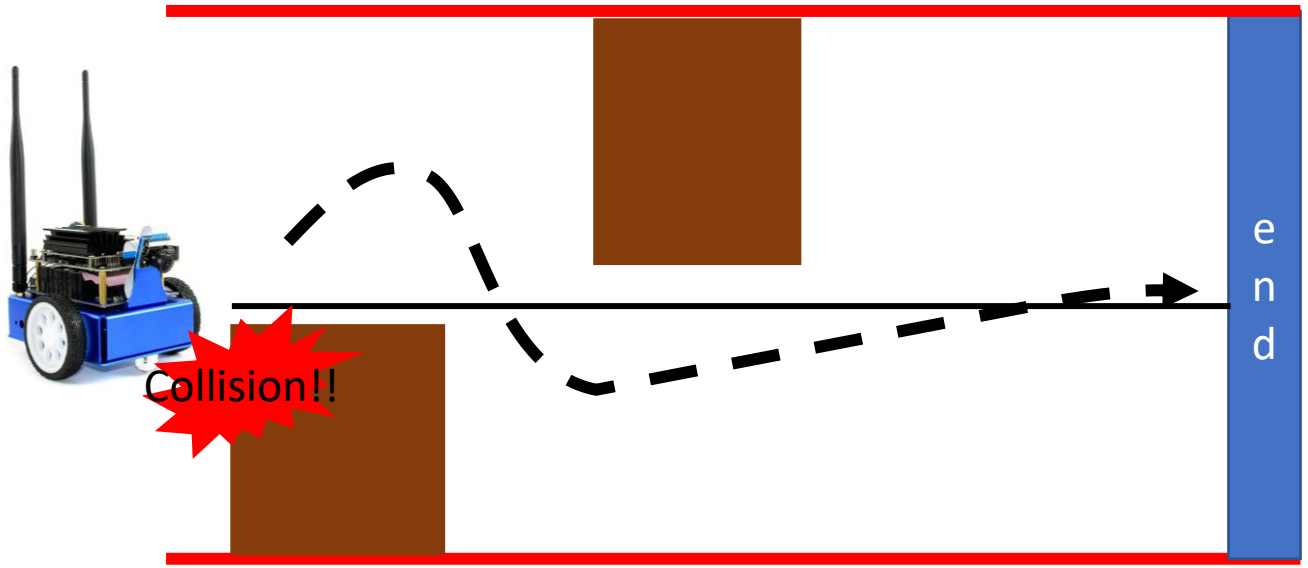
# Advance – Parking schematic map

- Try to park on the end sign as near as you can.

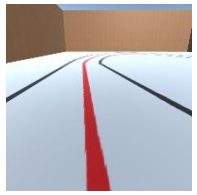


# Advance – Avoidance schematic map

- You should bypass the obstacles to reach the goal.



# JetBot Simulator



Camera View

Message Server

Control Signal

unity

Physical Simulator

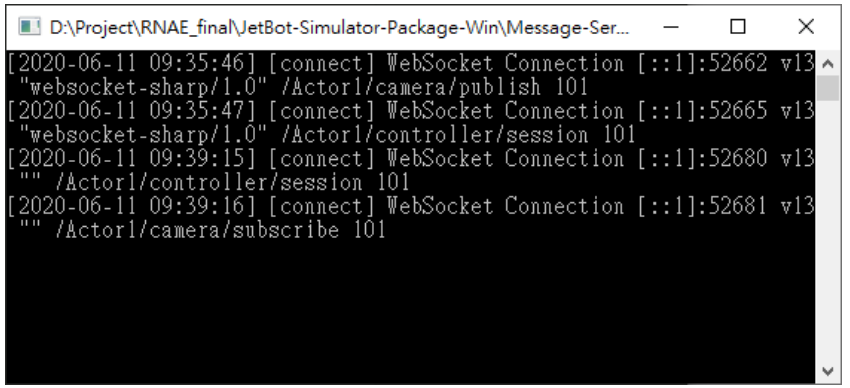
python™

Wrapper

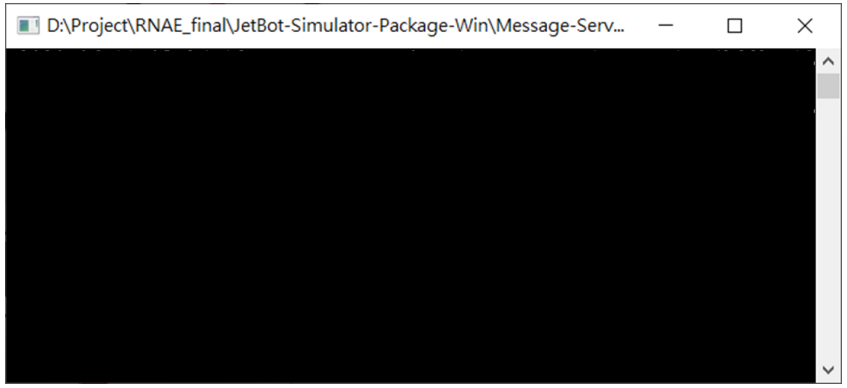
Link: https://reurl.cc/z8EK3e (for windows)

# Message Server

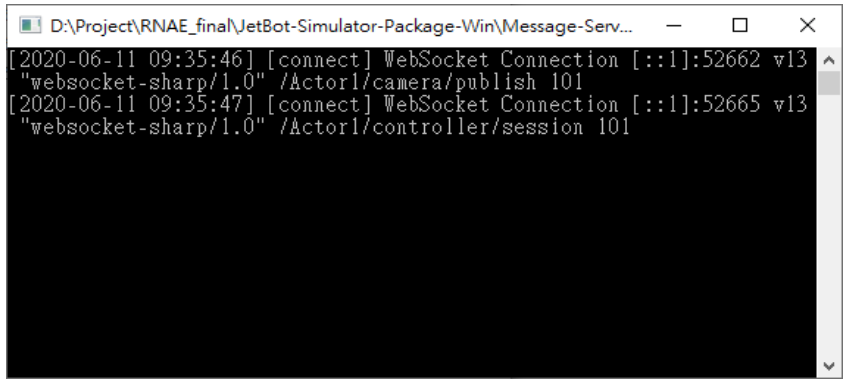
- Dataserver.exe
  - Run Server
  - Run Unity3D Simulator
  - Run Python Client




Run Python Client

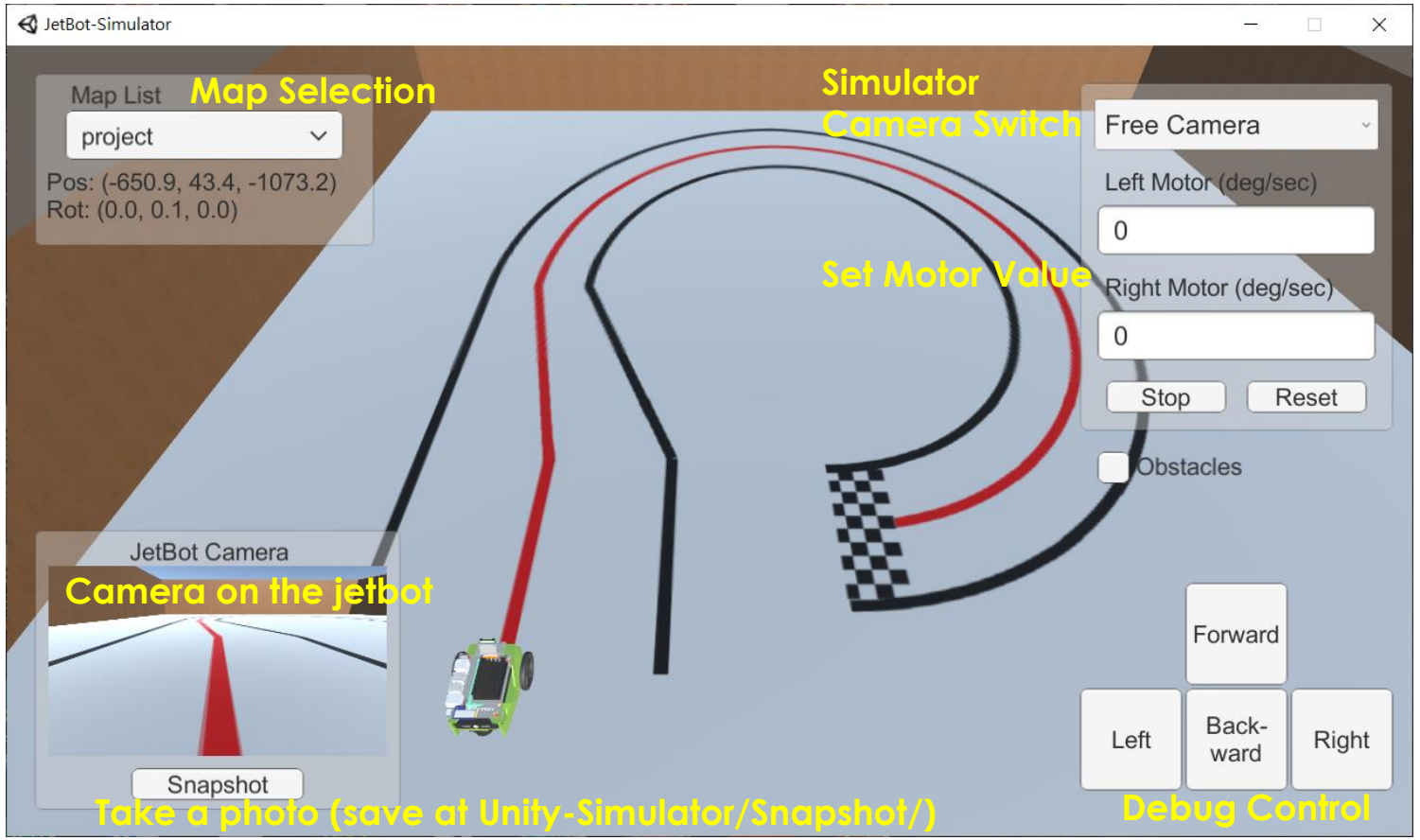



Run Server

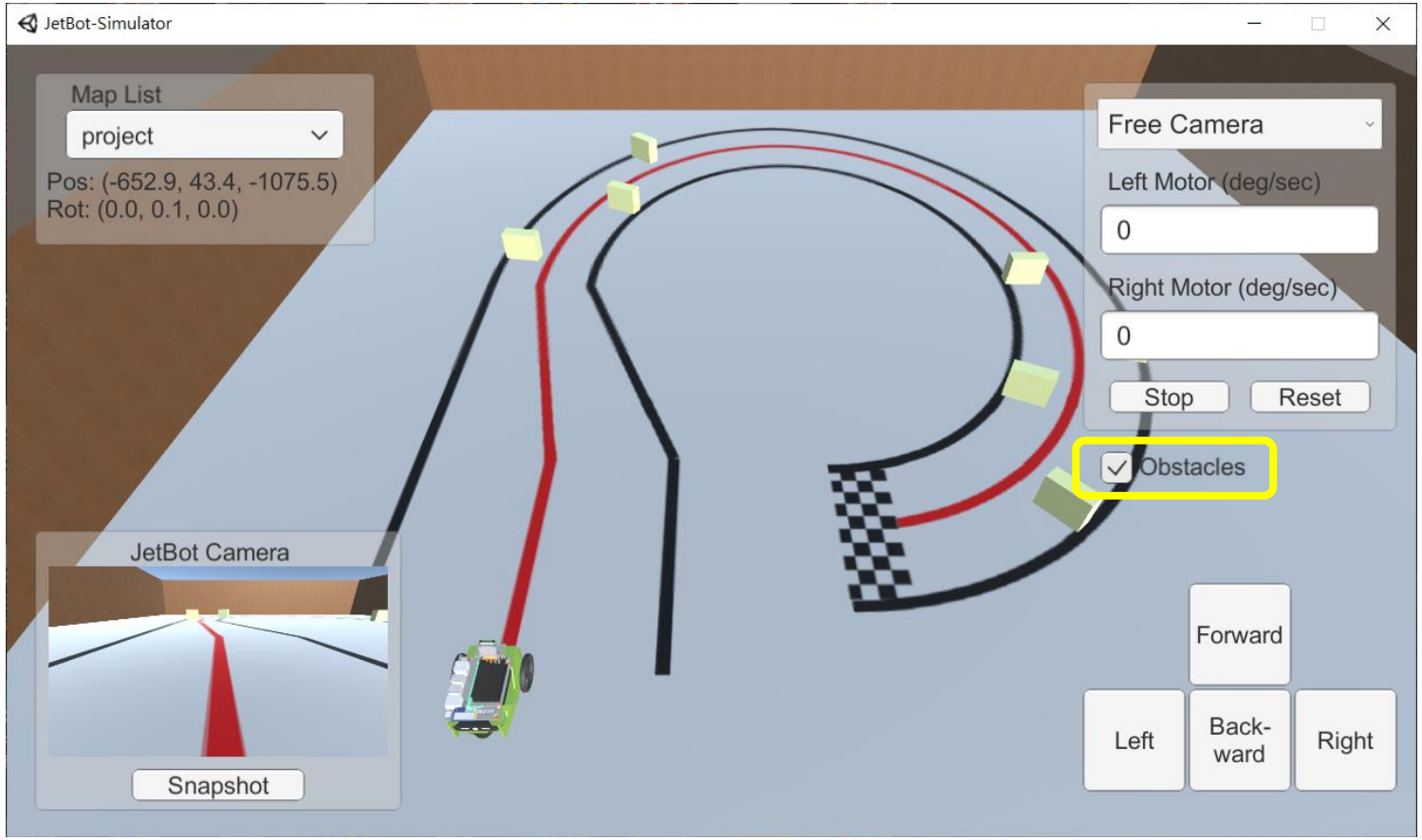



Run Unity3D Simulator

# Unity3D Simulator

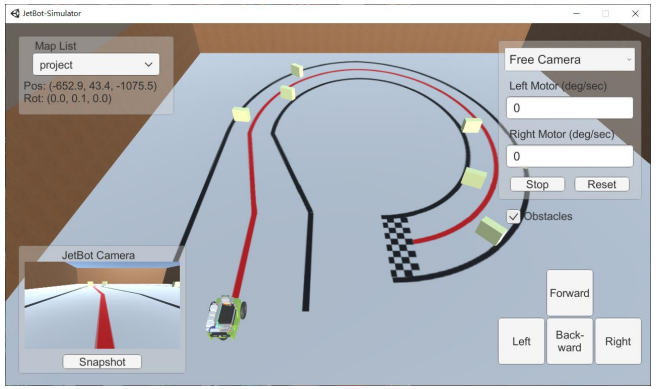


# Obstacles

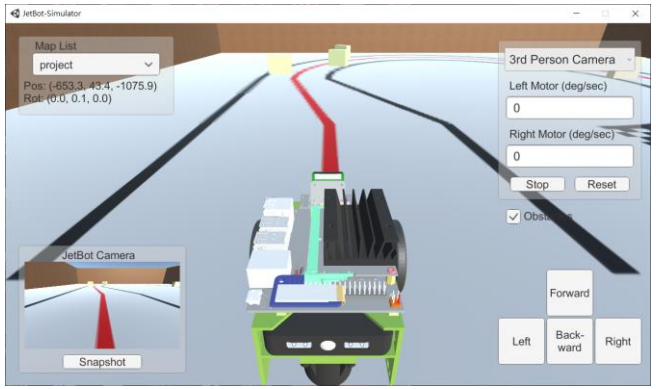

JetBot-Simulator
Map List
project
Pos: (-652.9, 43.4, -1075.5)
Rot: (0.0, 0.1, 0.0)
Free Camera
Left Motor (deg/sec)
0
Right Motor (deg/sec)
0
Stop
Reset
Obstacles
JetBot Camera
Snapshot
Forward
Left
Back-ward
Right

# Camera Switch

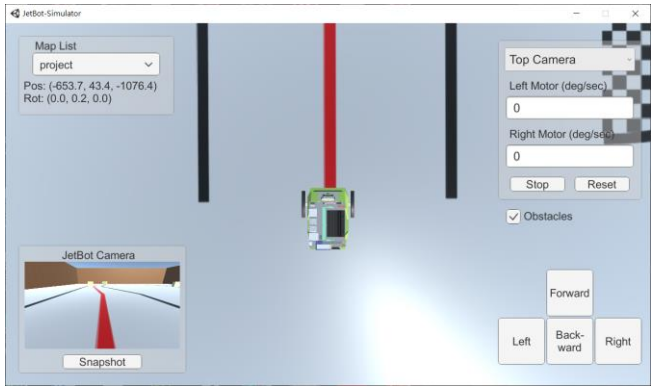
- **Free Camera**: The camera can be controlled by mouse.
- **3rd Person Camera**: The camera is above behind the vehicle.
- **Top Camera**: The camera is above the vehicle and captures the top view with orthogonal projection.




Free Camera




3rd Person Camera




Top Camera

# Python Wrapper

- jetbotSim
  - Camera
    - Wait for the camera data published by Unity3D simulator and invoke the callback function.
  - Robot
    - Send JSON-format control message to Unity3D simulator.

**Note:** The websocket library is required

```
pip install websocket
pip install websocket-client
```

# Example

```
from jetbotSim import Robot, Camera
import cv2

frames = 0
def execute(change):
    global robot, frames
    print("\rFrames", frames, end="")
    frames += 1

    # Control Example
    if frames == 1:
        robot.forward(0.2)
    if frames == 80:
        robot.left(0.05)

    # Visualize
    img = cv2.resize(change["new"],(640,360))
    cv2.imshow("camera", img)

robot = Robot()
camera = Camera()
camera.observe(execute)
```

# Control API

- **robot.set_left_motor(value)**
  - Left_motor = value
- **robot.set_right_motor(value)**
  - Right_motor = value
- **robot.set_motor(value_l, value_r)**
  - Left_motor = value_l
  - Right_motor = value_r
- **robot.add_motor(value_l, value_r)**
  - Left_motor += value_l
  - Right_motor += value_r

**Note:**
The unit of motor value shown on Unity3D simulator (**deg/sec**) is different from the unit of control value in python wrapper (**m/sec**).

# Control API

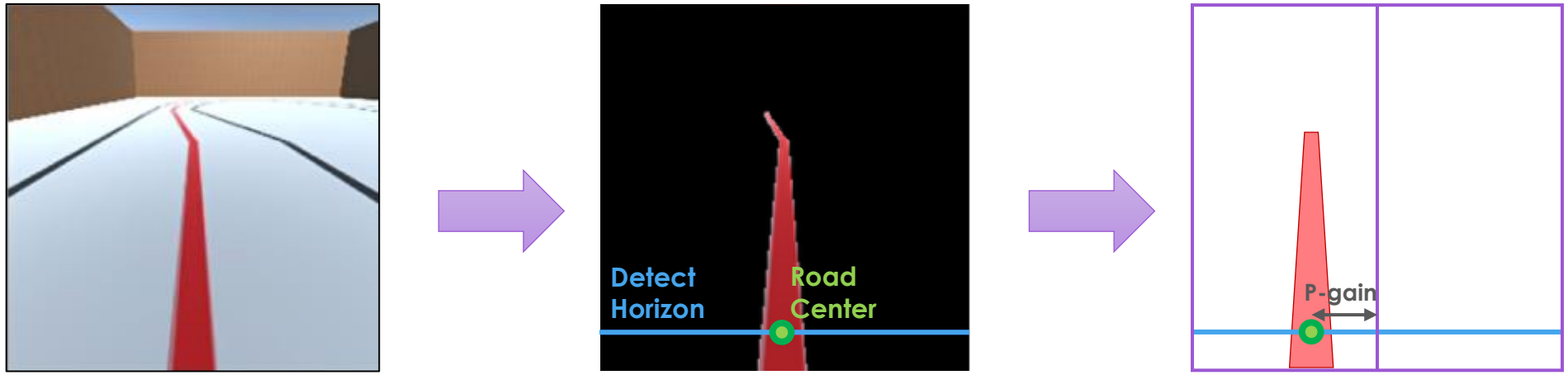
- **robot.forward(value)**
  - Left_motor = value
  - Right_motor = value
- **robot.backward(value)**
  - Left_motor = -value
  - Right_motor = -value
- **robot.left(value)**
  - Left_motor = -value
  - Right_motor = value
- **robot.right(value)**
  - Left_motor = value
  - Right_motor = -value

# Control API

- **robot.stop()**
  - Left_motor = 0
  - Right_motor = 0
- **robot.reset()**
  - Left_motor = 0
  - Right_motor = 0
  - Set to origin

# Hint

- You can apply image processing and simple rule-base algorithm to detect the center of road and utilize P-control to track the road.



# Real World Demo

# Demo Process

- The demo process contains two parts (for both test and final map).
  - First part
    - **Automatic tracking & Parking**
  - Second part
    - Automatic tracking & Parking & **Avoidance**

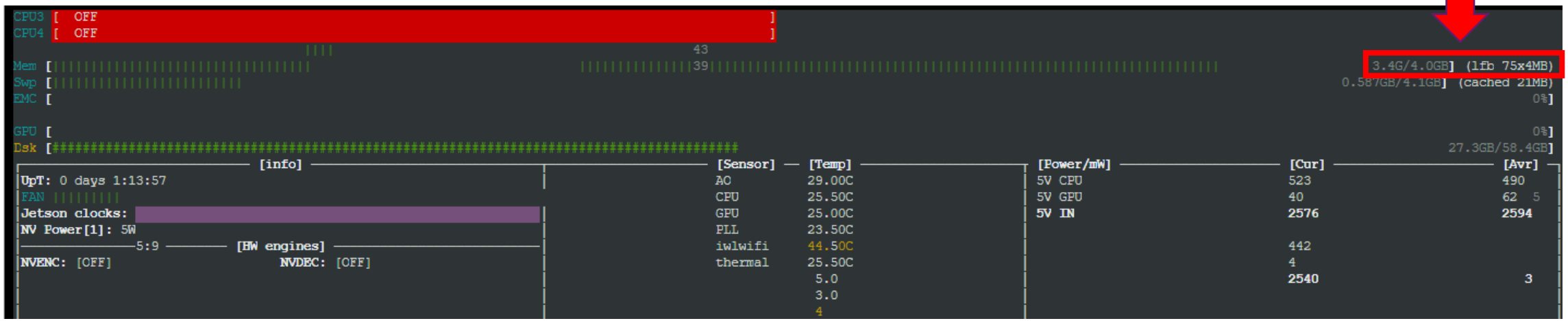
# Real world score – Details

| | Test map | Real map |
|---|---|---|
| Automatic tracking | Tracking red line:<br>4 points | Tracking red line:<br>8 points |
| Parking | Basic line: 1 point<br>Perfect line: 3 points | Basic line: 2 points<br>Perfect line: 6 points |
| Avoidance | 1 obstacle:<br>3 points<br>(A little collision is ok) | 6 obstacles:<br>1 point/obstacle<br>(A little collision is ok) |
| | You can only restart at the origin | You can restart at the previous position 3 times |

# Hint 1 - memory usage

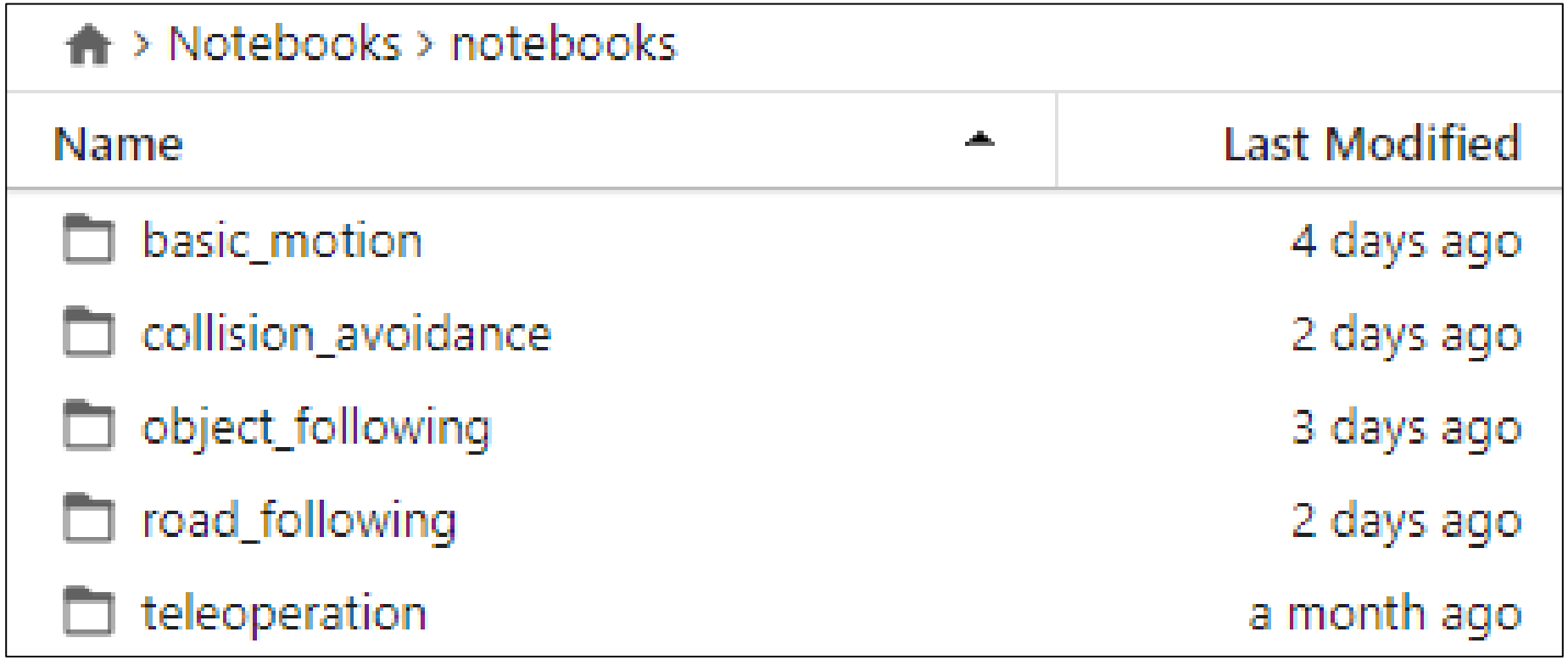
- `sudo -H pip3 install jetson-stats`
- `sudo jtop`

memory



# Hint 2 - reference code

**JupyterLab**
**http://<jetbot_ip_address>:8888**



# Hint 3 - others

- Recommend model: lightweight model, ex: **shufflenet**, **mobilenet**
- **Do not update pytorch**
- If your jetbot have any problem, you can restart the jetbot first!!!
  - (Ex: camera dead)