Selected Topics in Reinforcement Learning - Final Project

Introduction

Get ready for a thrilling project where you'll be training a model to take the wheel in a race car game! This is a first-person perspective racing game, where each step offers you only the current view. There are no boundaries - you can choose any algorithm or training framework. So, put on your thinking cap, strive for the top scores, and outshine the competition. Let's hit the track!

Lab Description

- Given the environment modified from ***racecar_gym***, train a neural network to control the racecar.
- Race against the clock and cover the longest distance you can!
- Observe environmental characteristics and choose your method.
- Design your mechanism to train a neural network to control the racecar.
- Try to apply different potential methods!

Important Date

- Available demo time
  - 12/1, 12/8, 12/22 (Mon.) evening
  - You need to fill out the form in advance, just like the previous labs)
- Report
  - Submission deadline: 2025/12/21 23:59

Turn in

- You have to turn in your report (.pdf)
- Notice: zip all files with the name “RL_Project_StudentId_Name.zip”
  - E.g., “RL_Project_313551072_郭建廷.zip”

    “RL_Project_313551072_郭建廷.zip”

    |_____ report.pdf
  - Wrong format deduction: -5pts

Scoring Criteria

Your total project score = report * 0.2 + demo performance * 0.8

The demo focuses on performance, and the report will pay more attention to the effort spent and things learned.

Part 1: Performance (demo)

- You **NEED to DEMO** to get performance grades.
- Different maps can be tested with different models.
- **Pre-recorded action sequences are prohibited** during the demonstration.
- You have several chances to demo, and the **highest** score will ultimately be chosen. (including the bonus)
- The distribution of demo scores
  - `Circle_cw` map:

        - Pass the Circle_cw threshold: **20%** of the demo score.
        - Ranking: **20%** of the demo score.
    - `Austria` map:
        - Pass the threshold 1/2: **15%** of the demo score.
        - Pass the threshold 2/2: **5%** of the demo score.
        - Ranking: **20%** of the demo score.
    - `Unknown` map:
        - You can use any of your models to evaluate on this map, the map is not released (it may be larger noise or new map).
        - Ranking: **20%** of the demo score.
    - Bonus:
        - If you use the same model to evaluate both maps and you pass **both the Circle_cw threshold and Austria threshold 1/2**: **+5%** of the demo score. (up to 100)

Part 2: Report

- In this section, we mainly score based on the effort and learning people put into "**Reinforcement Learning**" and "**Neural Network Training**."
    - If your content DOES NOT relate to at least one of the topics mentioned above, you will receive NO score in the report part. For instance, you might craft an "if-else" program that could achieve a decent demo performance score. However, if that is your only effort, you will receive **no points** in the report section.
- Report content and score distribution
    - Methodology Introduction (**20%** of the report score)
        - Introduction of the methodology used for each map
        - Explanation for the choice of methods for each map
    - Experiment Design and Implementation (**20%** of the report score)
        - Description of the training processes
        - Neural network architectures

        - Details of the hyper-parameters
        - List of packages, tools, or resources used
    - Method Comparison and Evaluation (**35%** of the report score)
        - Try different methods and show their results. (Including a diagram is better.)
        - Comparing the effectiveness of different approaches
        - Analyzing the successful and unsuccessful cases
        - Discussing the key observations and insights
    - Challenges and Learning Points: (**15%** of the report score)
        - Encountered challenges during the implementation process
        - The learnings from these challenges
    - Future Work: (**10%** of the report score)
        - The proposal of ideas for potential improvements
        - The suggestion of additional research directions for the future

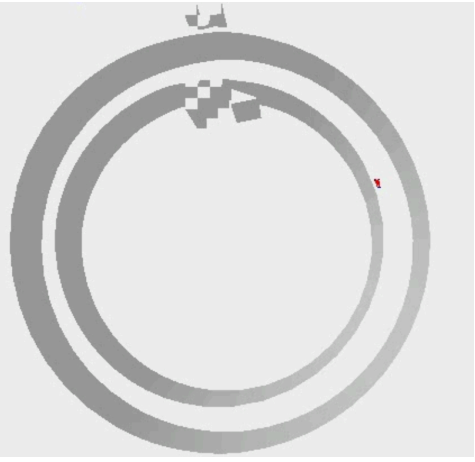
Environment

- The environment is modified from racecar_gym.
- When scoring, a pre-set parameter environment is used (see demo for details).
- Evaluation settings
    - Observation: a single 3x128x128 image
    - Action: `motor` and `steering` (both range from -1 to 1).
    - Default Reward: progress reward (difference of progress)
        - You may adjust the reward function according to your needs during your training sessions.
        - During the evaluation, we look at the progress you have completed when the time ends, not the total reward.

    - Terminated:
        - Maps:
            - `Circle_cw`: env time limit, student time limit, **collision**
            - `Austria`: env time limit, student time limit
            - `Unknown`: env time limit, student time limit
        - Please refer to the demo part.
    - Info: velocity, acceleration, pose, time, progress, lap, …
    - 0.02 sec a frame
    - During evaluation, we **may introduce some randomness**, **such as adding a bit of noise to your action or randomizing the starting point**. Therefore, please ensure that you train a stable model!
        - **The randomness used may vary on different demo days.**
- You can read the code or the official GitHub page for more information.
    - Note that the scoring is done **with the environment we provide** to everyone. The original GitHub is for reference only.
    - For training, you can change the environment to help you train. But be aware that the environment for scoring is preset.
- When you're training or testing, please set `render_mode = "rgb_array_birds_eye"`
    - Do NOT use render_mode = "human" in training or testing

Map: Circle_cw

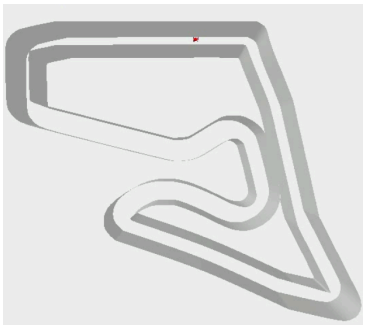
- Threshold: finish at least 1 lap

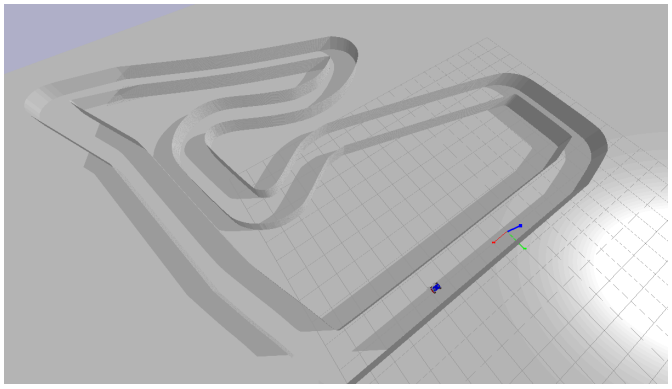


Map: Austria

- Threshold 1/2: pass the first curve (no collision)
- Threshold 2/2: pass the second curve (no collision)

The curve on the right side of the red car in the image below is the first curve; the lower right corner of the image below is the second curve.





The starting point is this car's position

Demo

The basic process during the demo

- Connect to the server opened by the TA (refer to server.py and client.py)
  - The *server* will return an `observation` to the *client*
  - The *client* sends an `action` to the *server*, then the server responds with a `terminated`

  ※ Please refer to the part about how to use server.py and client.py
- Explain the program and your design to the TA
- For the Circle_cw and Austria environments, **different models can be used** to score.
  - As mentioned in the previous part of the scoring criteria

Evaluation settings

- Env time limit: The time that the racing car can run in the game; that is, the time in the environment.
  - Note that every 0.02 seconds a frame
- Model time limit: the real-world time spent by the client
  - Timing Principle: Accumulate the time **between the server providing an observation and the reception of the subsequent action**
  - Your network delay may affect this
  - You may need to check the network settings to connect to our server.

  ※ Model Time Limit may be adjusted depending on the real cases
- Score: Lap + progress - 1 at the end of time
  - e.g., if you ran halfway (progress = 0.5) in the first lap and the time is up, then your score is 1 + 0.5 -1 (because the lap starts counting from 1)

The detailed settings:

Map	Env Time Limit	Model Time Limit (min)	Recover	CollisionStopWhenEval
Circle_cw	25 sec	10 min	No	Yes
Austria	100 sec	20 min	Yes	No
Unknown	100 sec	10 min	Yes	No

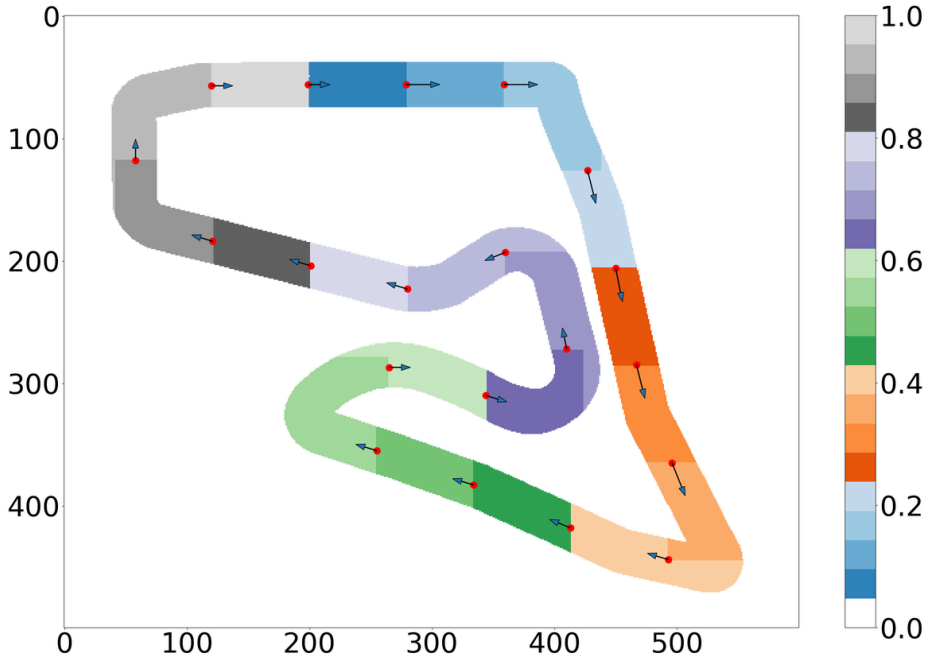
Name	Scenario	reset_when_collision
Circle_cw	circle_cw_competition_collisionStop	False
Austria	austria_competition	True
Unknown	Unknown	True

About the collision in Austria

- In Austria, it allows wall collisions, but there will be penalties. The harder you hit, the more seconds will be deducted from your remaining time.
  - Velocity: (x, y, z, roll, pitch, yaw)
  - Time Penalty $= 30 + 10 \times \sum_{v \in \text{velocity}} v^2$

    - e.g., In Austria, hit the wall at the 20th second with velocity = [2,0,0,0,0,0], the remaining seconds to run are: $100 - 20 - (30 + 10 \times 4) = 10$ sec

- This picture of Austria is divided into 20 parts according to progress. In Austria, after a collision, the car will be revived to the next area.



Austria's 20 sections of track

Leaderboard

- After the start of the demo, the weekly updated ranking information will be announced.

RL-Topics Final Project Rank

Rank	SID	Score	Env Time	Acc Time	Video
1	0716092	0.6507	39.580	239	Link

Basic Usage

Install

```
pip install -e .
```

- Suggested environment:
  - Ubuntu 22.04
  - Python 3.10
  - gymnasium==0.28.1
- You can refer to the `README.md` to build your environment.
- You can refer to the `Dockerfile` to construct your container.
  - You can refer to `container.md`

Construct the gymnasium env: `RaceEnv`

```
# Ensure you have installed racecar_gym so you can directly import the env
from racecar_gym.env import RaceEnv

env = RaceEnv(
    scenario=SCENARIO, # e.g., 'austria_competition', 'circle_cw_competition_collisionStop'
    render_mode='rgb_array_birds_eye',
    reset_when_collision=True, # Only work for 'austria_competition' and 'austria_competition_collisionStop'
)
```

Then, you can interact with it, for example:

```
obs, info = env.reset()
terminated = False
while not terminated:
    action = (1.0, 0.0) # Motor and steering
    obs, rew, terminated, truncated, info = env.step(action)
```

Evaluation by Yourself

- `server.py`
  - Open the environment and address client queries.
    - Provide observations if requested by the client.
    - Receive actions and respond with termination information.
  - Arguments:
    - `--port`: port
    - `--sid`: student id or any description
    - `--scenario`: what scenario to run
  - If you want to test for several episodes, please close the program and **restart it after each test**.
  - This is written using the Flask framework.
  - In the demo stage, the TA will open the server for you.
- `client.py`
  - Argument: `--url`: what address to connect
  - The purpose of client.py is to interact with the server using **HTTP** requests, transmitting action, and getting observations.
  - You can write your client, as long as you can correctly interact with the server.
  - An example (random agent) is given default.
    - You have to **modify it to use your neural network model**.
- Example of evaluating the random agent
  1. Run the server

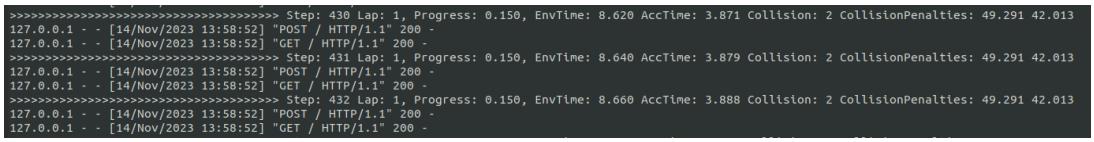
     ```
     # Example
     python server.py --port 33333 --sid tester --scenario austria_competiti
     on
     ```

  2. Run the client

```
python client.py --url http://127.0.0.1:33333
```

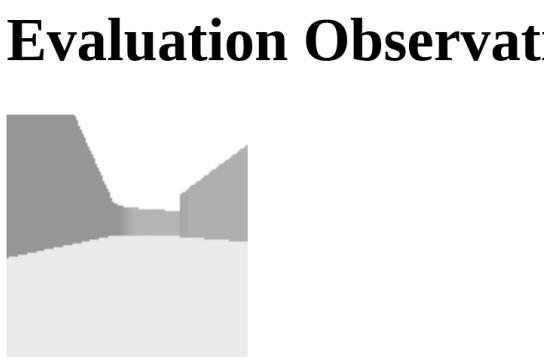
3. If they interact successfully:

   You can view information about the current status, as shown below, in the server's output:

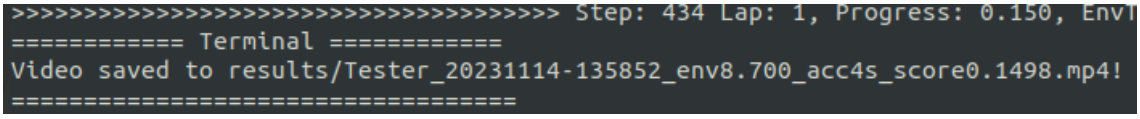


4. When they are running, you can go to `${url}/realtime` to watch the current testing observation.

   Example: open the web browser and go to `http://127.0.0.1:33333/realtime`

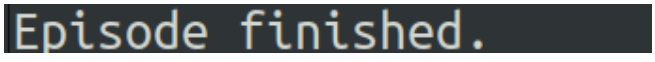


5. After the episode is terminated, the video will be saved.



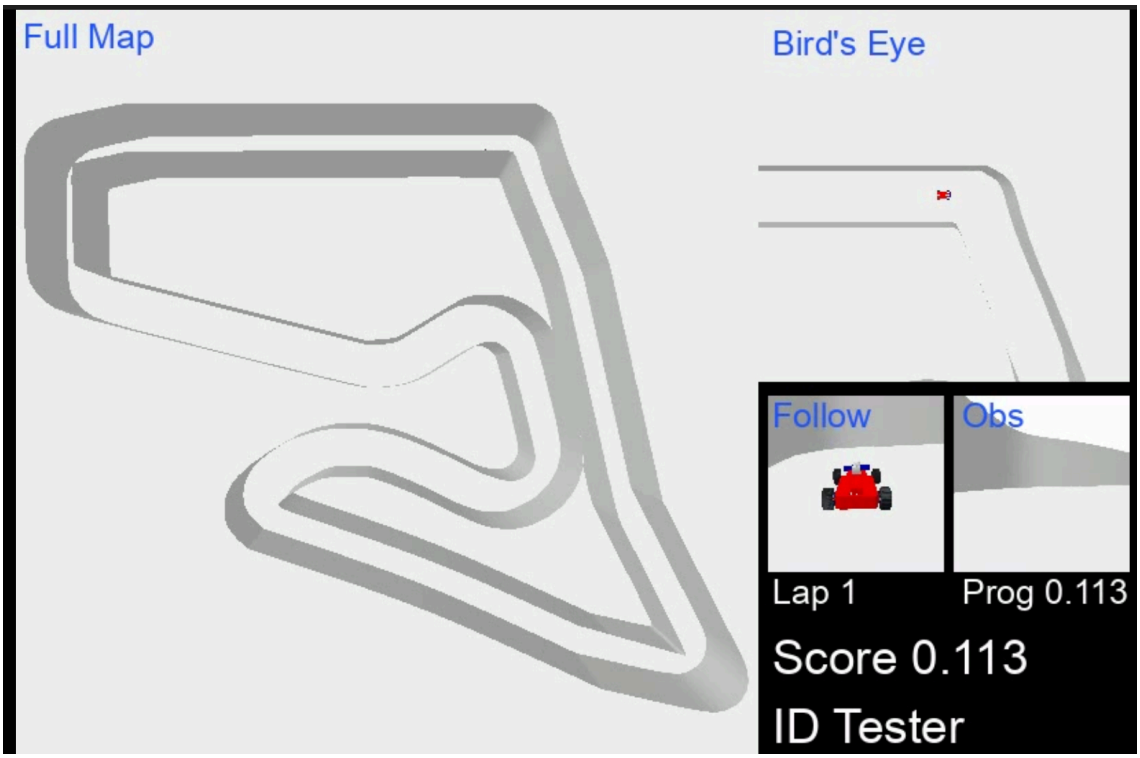
server

   The client's output will display "Episode finished.”



6. Play the video and you can see the episode

   The video contains multiple perspectives, the score, and the ID.



Note

- Not limited to algorithms.
- Not limited to programming frameworks, you can carve your own RL algorithm, or use other packages (such as Ray RLlib, Stable-Baselines3, and so on)
- Depending on the situation, rules may be adjusted or more information may be provided to everyone
- Feel free to ask questions in the discussion area!

Reference

- A. (n.d.). GitHub - axelbr/racecar_gym: A gym environment for a miniature racecar using the pybullet physics engine. GitHub. https://github.com/axelbr/racecar_gym

※ The cover photo is by toine G on Unsplash