ME491 : Final Term Project

Due Date : 8th December, 2020, 5:00 PM

1 Introduction

The goal of this project is to **climb an obstacle as high as possible using reinforcement learning with ANYmal**. It is mandatory to use RaiSim, a multi-body physics engine for robotics and AI. You can refer RaiSim document website: `http://raisim.com/index.html`. For your convenience, a basic template for the project is provided as a git repository: `https://github.com/HuboLabKaist/ME491TermProject`. For the evaluation, you need to submit files which can reproduce your result and a report that contains explanation and discussion about the result.

2 Setup

Currently, RaiSim is available both in Windows and Linux. But, including the official document, instructions will be provided only for Linux systems. **It is recommended to work on Ubuntu 18.04** for this project. Alternatively, if it is unavailable to setup due to lack of storage or computational ability of your PC, **you can instead use Google Colab** which will enable you to establish development environment with minimal effort.

Install system requirements. You can skip this if already set. You should reboot after the last command is finished

```
$ sudo apt update && sudo apt upgrade
$ sudo apt install git cmake
$ sudo ubuntu-drivers autoinstall
```

Install RaiSim dependencies. If you want to use python2, replace 'python3' and 'pip3' with 'python' and 'pip',

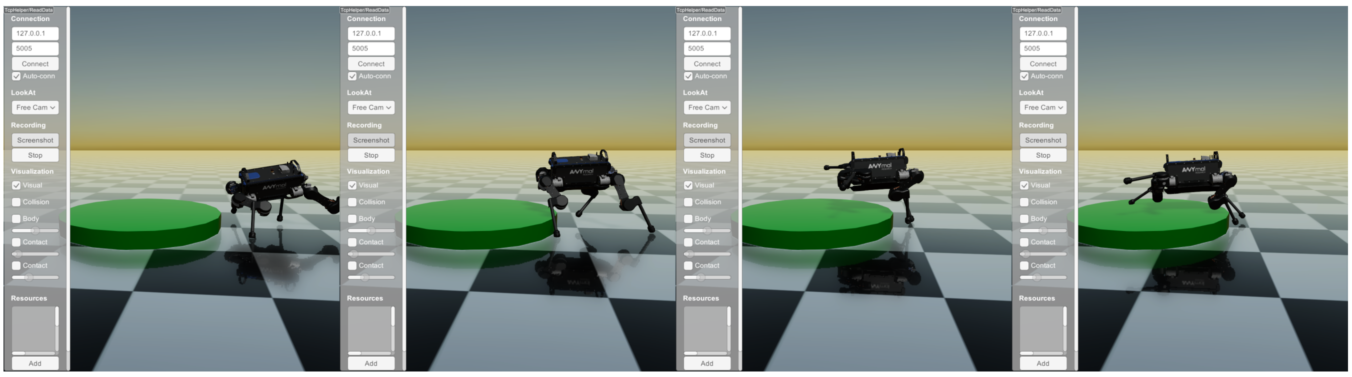


Figure 1: ANYmal climbing an obstacle

respectively.

```
$ sudo apt install libeigen3-dev minizip ffmpeg vulkan-utils
$ sudo apt install python3-dev python3-setuptools python3-pip
$ pip3 install --upgrade matplotlib ruamel.yaml
```

Create your workspace and installation directory. You should replace 'your/path/to/work(install)space' with actual path in your PC.

```
$ mkdir workspace && mkdir raisim_install
$ echo "export␣WORKSPACE=your/path/to/workspace" >> ~/.bashrc
$ echo "export␣LOCAL_INSTALL=your/path/to/installspace" >> ~/.bashrc
$ source ~/.bashrc
```

Clone the raisim library and the term project template repository.

```
$ cd $WORKSPACE
$ git clone https://github.com/raisimTech/raisimLib.git
$ git clone https://github.com/HuboLabKaist/ME491TermProject.git
```

Copy the license key of RaiSim to the designated location.

```
$ cd $WORKSPACE
$ cp your/path/to/license raisimLib/rsc
$ cp your/path/to/license ME491TermProject/raisimGymTorch/env/envs/rsg_anymal/anymal
```

Finally, build raisim.

```
$ mkdir raisimLib/build && cd raisimLib/build
$ cmake .. -DCMAKE_INSTALL_PREFIX=$LOCAL_INSTALL && make install -j4
```

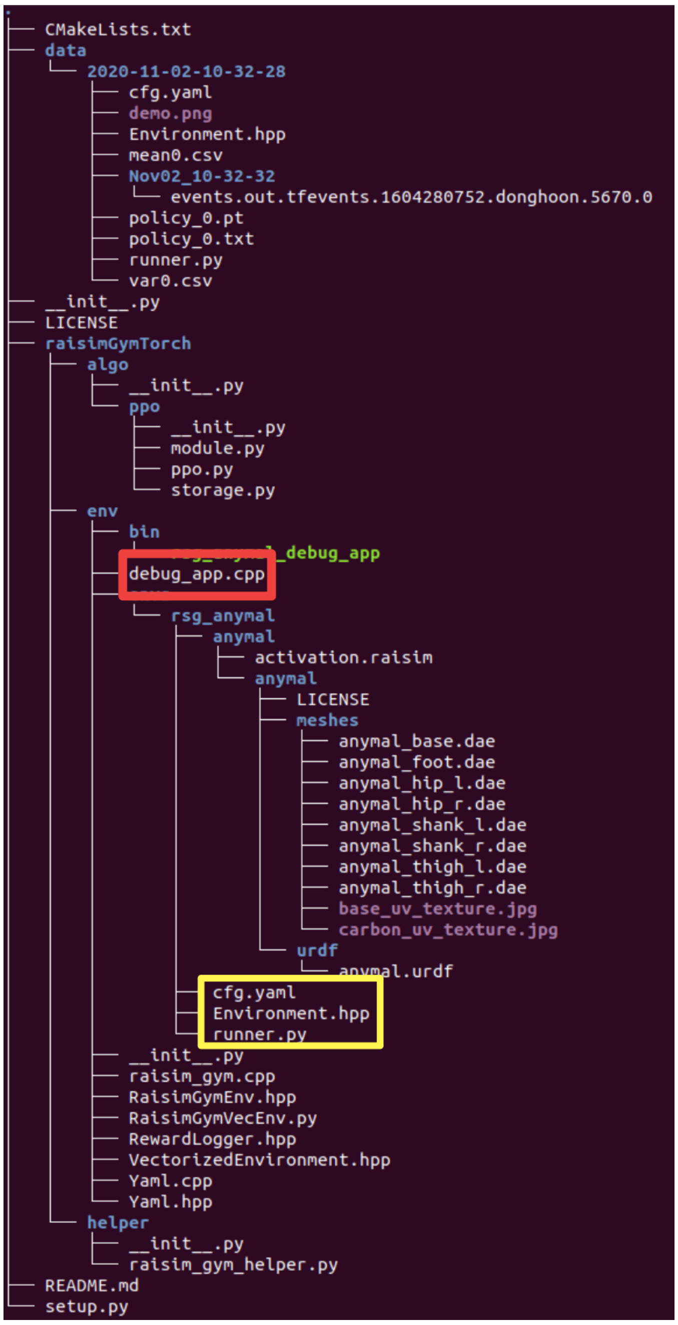
2.1 (Optional) Google Colab

Example code for Colab is provided. Only thing you need to do is upload the example code in your Google Drive then launch it. Detailed instructions are given in the file. You can also refer an official document of Google Colab: `https://colab.research.google.com/notebooks/intro.ipynb#recent=true`. Note that you cannot visualize the learning process inside Colab. You have to download the learned policy from Google drive and follow instructions below to visualize the result.

3 Assignment

The only task you need to achieve is simple: **Climb an obstacle as high as possible**. If the setup process is finished properly, there would be 2 directories in your work space: *'raisimLib'* and *'ME491TermProject'*. For the assignment, what you only need to exploit is the template repository, *'ME491TermProject'*. You don't have to look over *'raisimLib'* at all. Structure of the directory is depicted in Figure 2.

To accomplish the goal of this project, all the files need to be modified is remarked with yellow box in Figure 2. The learning environment and hyper-parameters can be adjusted by *'cfg.ymal'*



```
.
├── CMakeLists.txt
├── data
│   └── 2020-11-02-10-32-28
│       ├── cfg.yaml
│       ├── demo.png
│       ├── Environment.hpp
│       ├── mean0.csv
│       ├── Nov02_10-32-32
│       │   └── events.out.tfevents.1604280752.donghoon.5670.0
│       ├── policy_0.pt
│       ├── policy_0.txt
│       ├── runner.py
│       └── var0.csv
├── __init__.py
├── LICENSE
├── raisimGymTorch
│   ├── algo
│   │   ├── __init__.py
│   │   └── ppo
│   │       ├── __init__.py
│   │       ├── module.py
│   │       ├── ppo.py
│   │       └── storage.py
│   ├── env
│   │   ├── bin
│   │   │   └── [illegible]_debug_app
│   │   ├── debug_app.cpp
│   │   ├── [illegible]
│   │   │   └── rsg_anymal
│   │   │       ├── anymal
│   │   │       │   ├── activation.raisim
│   │   │       │   └── anymal
│   │   │       │       ├── LICENSE
│   │   │       │       ├── meshes
│   │   │       │       │   ├── anymal_base.dae
│   │   │       │       │   ├── anymal_foot.dae
│   │   │       │       │   ├── anymal_hip_l.dae
│   │   │       │       │   ├── anymal_hip_r.dae
│   │   │       │       │   ├── anymal_shank_l.dae
│   │   │       │       │   ├── anymal_shank_r.dae
│   │   │       │       │   ├── anymal_thigh_l.dae
│   │   │       │       │   ├── anymal_thigh_r.dae
│   │   │       │       │   ├── base_uv_texture.jpg
│   │   │       │       │   └── carbon_uv_texture.jpg
│   │   │       │       └── urdf
│   │   │       │           └── anymal.urdf
│   │   │       ├── cfg.yaml
│   │   │       ├── Environment.hpp
│   │   │       └── runner.py
│   │   ├── __init__.py
│   │   ├── raisim_gym.cpp
│   │   ├── RaisimGymEnv.hpp
│   │   ├── RaisimGymVecEnv.py
│   │   ├── RewardLogger.hpp
│   │   ├── VectorizedEnvironment.hpp
│   │   ├── Yaml.cpp
│   │   └── Yaml.hpp
│   └── helper
│       ├── __init__.py
│       └── raisim_gym_helper.py
├── README.md
└── setup.py
```

Figure 2: File tree of the template directory.

and *'Environments.hpp'*. There are comment-remarked regions in *'Environments.hpp'* so that you can easily modify it. Also you can directly affect learning process by revising *'runner.py'*. It will be somewhat sufficient to accomplish the goal by just revising these files. However, in case you want to improve further, feel free to modify any part in the template or just build your own from the scratch.

Test the example code :

Build environment using python setup tools. Note that whenever you revise cpp or hpp files, you have to compile again by launching this

```
$ cd $WORKSPACE/ME491TermProject
$ python3 setup.py develop --user --Debug --CMAKE_PREFIX_PATH "$LOCAL_INSTALL"
```

Run main learning script

```
$ python3 raisimGymTorch/env/envs/rsg_anymal/runner.py False
```

If you want to test learned policy,

```
$ python3 raisimGymTorch/env/envs/rsg_anymal/runner.py True
```

Note that content of the example code is irrelevant with this project.

Visualization :
Launch raisimUnity which is included in raisimLib. Its location is *'raisimLib/rasimUnity/linux/raisimUnity.x86_64'*. Pictures of Figure 1 are example scenes which can be shown when you launch raisimUnity.

Save and load :
While the learning process is going on, it automatically saves related information in sub-directory of *'data'* directory which is named by its generated date and time. You can test each policy by revising *'runner.py'*. Note that *'Environment.hpp'* and *'cfg.yaml'* are also generated together in the directory to represent the learned environment.

(Optional) Trouble Shooting :
Try using *'debug_app.cpp'* covered in red box in Figure 2 with other tools such as GDB. It would enable you to debug C++ codes much easier than in python scripts. Note that you should switch 'render' option from True to False in '*cfg.yaml*' when you launch the debug app.
Run debug app

```
$ cd $WORKSPACE/ME491TermProject/raisimGymTorch/env
$ bin/rsg_anymal_debug_app envs/rsg_anymal/anymal envs/rsg_anymal/cfg.yaml
```

4 Report

Final report should contain these contents:

- Brief explanation about applied reinforcement learning algorithm
- How hyper-parameters or size of batch affect the performance.
- How observation or action is adjusted and the reason why.
- Discussion about result.(*e.g.* number of iterations(or samples) taken, height of obstacle)
- Future work or possible improvements.

It is recommended for the report to be about 10 pages long.

5 Submission

Please **submit a zip file named '(student number)_(name).zip' consisted of the report and files required to reproduce the result.** All the files required are automatically generated in the sub-directories of *'data'* directory which is mentioned above. *'Environment.hpp'* and *'cfg.yaml'* should be included to implement learned environment. Then, among several policies distinguished by the number of iterations, choose one that gives the best result. Corresponding csv files for mean and variance should be also included.

For example :

```
20200000_name.zip
 ├─ final_report.docx
 ├─ Environments.hpp
 ├─ cfg.yaml
 ├─ policy_N.pt
 ├─ meanN.csv
 └─ varN.csv
```

where N represents the number of iterations

6 Contact

If you have any question, please contact TA Donghoon Youm(ydh0725@kaist.ac.kr) or Joowoong Byun (woong164@kaist.ac.kr)