3월 1주차 레포트

#1차원 상에서 Goal 지점 찾기

1. 보상 설정

[수정 전 코드]

```
## Move에 대한 보상
1. Action이 주어진 Config와 유사한가?
  >> REWARD_CONFIG 부여

2. Step이 많은가?
  >> REWARD_STEP 부여

## Task에 대한 보상
1. Goal 지점에 도달하였는가?
  >> YES
    >> REWARD_GOAL 부여

  >> NO
    >> 위치 범위를 벗어나는가?
      >> YES
        >> REWARD_AWAY 부여

      >> NO
        >> Nothing
```

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/8bf29988-2cb9-4428-9db6-64fbd88e2105/My_Viewer-2022-02-25_22.42.13.mp4

[1차 수정 후 코드]

```
## Move에 대한 보상
1. 위치 범위를 벗어나는가?
  >> Action 재설정 : 범위 내에 분포하도록 Action 재설정
  >> REWARD_AWAY 부여

2. Action이 주어진 Config와 유사한가?
  >> REWARD_CONFIG 부여

3. Step이 많은가?
  >> REWARD_STEP 부여

## Task에 대한 보상
1. Goal 지점에 도달하였는가?
  >> REWARD_GOAL 부여

2. Goal 지점에 가까워지고 있는가?
  >> REWAWRD_DIFF 부여
```

2. 환경 설정

```
변화 없음
```

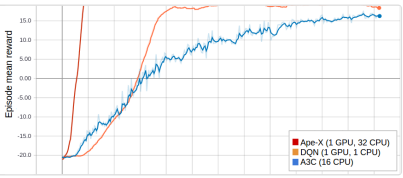
(1). ppo.DEFAULT_CONFIG 정보

[출처] :

Ray v1.10.0

We're hiring! The RLlib team at Anyscale Inc., the company behind Ray, is hiring interns and full-time reinforcement learning engineers to help advance and maintain RLlib. If you have a background in ML/RL and are interested in making RLlib the industry-leading open-source RL library, apply here today.

https://docs.ray.io/en/latest/rllib-algorithms.html




```
# Adds the following updates to the (base) `Trainer` config in
# rllib/agents/trainer.py (`COMMON_CONFIG` dict).
DEFAULT_CONFIG = with_common_config({
    # Should use a critic as a baseline (otherwise don't use value baseline;
    # required for using GAE).
    "use_critic": True,
    # If true, use the Generalized Advantage Estimator (GAE)
    # with a value function, see https://arxiv.org/pdf/1506.02438.pdf.
    "use_gae": True,
    # The GAE (lambda) parameter.
    "lambda": 1.0,
    # Initial coefficient for KL divergence.
    "kl_coeff": 0.2,
    # Size of batches collected from each worker.
    "rollout_fragment_length": 200,
    # Number of timesteps collected for each SGD round. This defines the size
    # of each SGD epoch.
    "train_batch_size": 4000,
    # Total SGD batch size across all devices for SGD. This defines the
    # minibatch size within each epoch.
    "sgd_minibatch_size": 128,
    # Whether to shuffle sequences in the batch when training (recommended).
    "shuffle_sequences": True,
    # Number of SGD iterations in each outer loop (i.e., number of epochs to
    # execute per train batch).
    "num_sgd_iter": 30,
    # Stepsize of SGD.
    "lr": 5e-5,
    # Learning rate schedule.
    "lr_schedule": None,
    # Coefficient of the value function loss. IMPORTANT: you must tune this if
    # you set vf_share_layers=True inside your model's config.
    "vf_loss_coeff": 1.0,
    "model": {
        # Share layers for value function. If you set this to True, it's
        # important to tune vf_loss_coeff.
        "vf_share_layers": False,
    },
    # Coefficient of the entropy regularizer.
    "entropy_coeff": 0.0,
    # Decay schedule for the entropy regularizer.
    "entropy_coeff_schedule": None,
    # PPO clip parameter.
    "clip_param": 0.3,
    # Clip param for the value function. Note that this is sensitive to the
    # scale of the rewards. If your expected V is large, increase this.
    "vf_clip_param": 10.0,
    # If specified, clip the global norm of gradients by this amount.
    "grad_clip": None,
    # Target value for KL divergence.
    "kl_target": 0.01,
    # Whether to rollout "complete_episodes" or "truncate_episodes".
    "batch_mode": "truncate_episodes",
    # Which observation filter to apply to the observation.
    "observation_filter": "NoFilter",

    # Deprecated keys:
    # Share layers for value function. If you set this to True, it's important
    # to tune vf_loss_coeff.
    # Use config.model.vf_share_layers instead.
    "vf_share_layers": DEPRECATED_VALUE,
})
```

3. Rendering

[수정 전 코드]

```
## Task 성공 여부에 따라 진행
self.done = False
>> Task 성공 X

self.done = True
>> Task 성공 O

  >> self.done = True 될때까지만 진행
```

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/41382756-8cda-4713-b9b8-b3aa613253ec/2월_4주차_비디오_1차_바_-_1차_수.mp4

[수정 후 코드]

```
## Task 성공 여부에 따라 변화
self.done = False
>> Task 성공 X

self.done = True
>> Task 성공 O

  >> self.done = True 될 시,
  >> self.env.reset() 진행
    >> self.goal 변경 & self.episode_reward 초기화
      >> Task 목표 지점을 변경한다.
      >> Episode에 대한 Reward를 초기화
  >> 계속 반복 진행
```

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0522cdc7-bbc4-4f73-a4fb-23201f1b897b/My_Viewer-2022-02-25_22.42.13.mp4

#2차원 상에서 Goal 지점 찾기

1. 보상 설정

[수정 전]

```
 ## Move에 대한 보상
1. 위치 범위를 벗어나는가?
  >> Action 재설정 : 범위 내에 분포하도록 Action 재설정
  >> REWARD_AWAY 부여

2. Action이 주어진 Config와 유사한가?
  >> REWARD_CONFIG 부여

3. Step이 많은가?
  >> REWARD_STEP 부여

## Task에 대한 보상
1. Goal 지점에 도달하였는가?
  >> REWARD_GOAL 부여

2. Goal 지점에 가까워지고 있는가?
  >> REWAWR_DIFF 부여
```

[1차 수정 후]

```
## Move에 대한 보상
1. Step이 많은가?
  >> REWARD_STEP = - STEP_COUNT * 0.1
```

```
## Task에 대한 보상
1. Goal 지점에 도달하였는가?
  >> REWARD_GOAL = 5000

2. Goal 지점에 가까워지고 있는가?
  >> REWARD_DIFF = 1/(Distance+0.5)
```

2. 환경 설정

```
## Action Space & Observation Space
>> 2차원으로 설정
```

3. Rendering

```
## Task 성공 여부에 따라 변화
self.done = False
>> Task 성공 X

self.done = True
>> Task 성공 O

  >> self.done = True 될 시,
  >> self.env.reset() 진행
    >> self.goal 변경 & self.episode_reward 초기화
      >> Task 목표 지점을 변경한다.
      >> Episode에 대한 Reward를 초기화
  >> 계속 반복 진행
```

4. 결과 동영상

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/73ee2f61-aa73-49de-9cd0-90ab06328ed9/My_Viewer-2022-02-25_23.52.04.mp4

5. 보완해야할 점들

```
1. Goal 지점을 찾기까지 너무 오래 걸린다.
  (1). 보상함수들의 가중치 변화를 준다
  (2). 보상함수들을 변경한다

2. 이전의 3~4프레임 정도의 이동이 색이
옅어지며 없어지면 어떻게 이동하는지 보일 것 같다.

3. Window상에서 Episode Reward가
업데이트하는 것이 보여졌으면 좋겠다.
```

Dartpy를 활용한 Deepmimic github 코드

[출처] :

senier-project/DeepMimic_Project_Final at bce3b6fce427419fbec9348101d99d582dc23620 · letcodesin/senier-project

Contribute to letcodesin/senier-project development by creating an account on GitHub.

https://github.com/letcodesin/senier-project/tree/bce3b6fce427419fbec9348101d99d582dc23620/DeepMimic_Project_Final

letcodesin/sen
project

1 Contributor 0 Issues 0 Stars

#DeepMimic 구현

1. PyBullet

(1). 설치

```
pip3 install pybullet
```

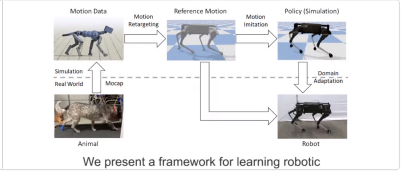
(2). 가이드 라인

[홈페이지] :

Bullet Real-Time Physics Simulation

We are developing a new differentiable simulator for robotics learning, called Tiny Differentiable Simulator, or TDS. The simulator allows for hybrid simulation with neural networks. It allows different automatic differentiation backends, for forward and reverse mode gradients. TDS can be trained using Deep

https://pybullet.org/wordpress/

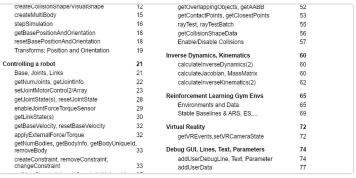


[책] :

PyBullet Quickstart Guide

PyBullet Quickstart Guide Erwin Coumans, Yunfei Bai, 2016-2022 Visit desktop doc, forums, github discussions and star Bullet! Introduction 2 Hello PyBullet World 3 connect, disconnect, bullet_client 3 setGravity 7 loadURDF, loadSDF, loadMJCF 7 saveState, saveBullet, restoreState 11 createColl...

https://docs.google.com/document/d/10sXEhzFRSnvFcl3XxNGhnD4N2SedqwdAvK3dsihxVUA/edit#heading=h.2ye70wns7io3



2. 캐릭터 파일 분석

(1). Deepmimic 저자의 Github 상에서의 캐릭터의 구성

```
## Github상에서 존재하는 txt파일 형식의 Skeleton 파일

Root - 1
  >> type : None
  >> weight : 6
  >> shape : sphere

Chest - 2
  >> type : Spherical
  >> weight : 14
  >> shape : sphere

Neck - 3
  >> type : Spherical
```

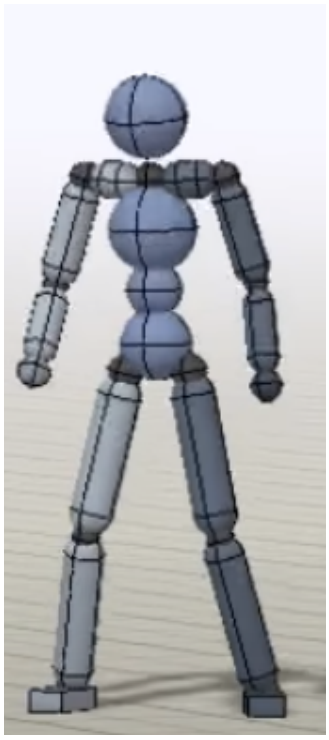
```
  >> weight : 2
  >> shape : sphere

Right_hip - 4
  >> type : Spherical
  >> weight : 4.5
  >> shape : capsule

Right_knee - 5
  >> type : revolute
  >> weight : 3
  >> shape : capsule

Right_ankle - 6
  >> type : Spherical
  >> weight : 1
  >> shape : box

Right_shoulder - 7
  >> type : Spherical
  >> weight : 1.5
  >> shape : capsule

Right_elbow - 8
  >> type : revolute
  >> weight : 1
  >> shape : capsule

Right_wrist - 9
  >> type : fixed
  >> weight : 0.5
  >> shape : sphere

Left_hip - 10
  >> type : Spherical
  >> weight : 4.5
  >> shape : capsule

Left_knee - 11
  >> type : revolute
  >> weight : 3
  >> shape : capsule

Left_ankle - 12
  >> type : Spherical
  >> weight : 1
  >> shape : box

Left_shoulder - 13
  >> type : Spherical
  >> weight : 1.5
  >> shape : capsule

Left_elbow - 14
  >> type : revolute
  >> weight : 1
  >> shape : capsule

Left_wrist - 15
  >> type : fixed
  >> weight : 0.5
  >> shape : sphere
```

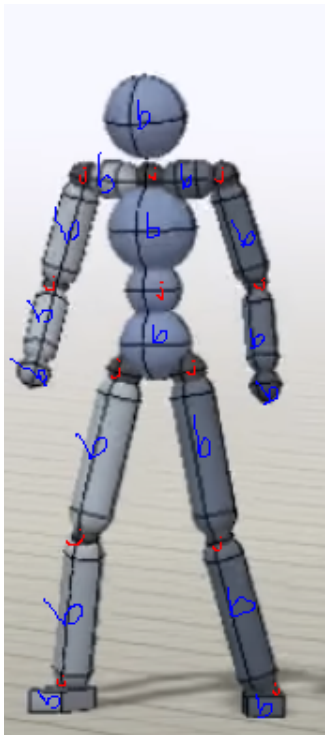
(2). Deepmimic 저자의 Github 상에서의 human 캐릭터의 Frame 파일 구성

```
[        0.0333320000,        0.0000000000,        0.8475320000,        0.0000000000,        0.9986780000,        0.0141040000,
>> 총 44개의 Dof들에 대한 정보를 지니고 있다.

>> bvh에서는
  >> spherical : 3 Dof
  >> revolute : 1 Dof
  >> free : 6 Dof
  >> fixed : 0 Dof

>> txt 파일 형식의 Skeleton에서의 type구성
  >> spherical : 8개 >> 24 Dof
  >> revolute : 4개 >> 4 Dof
  >> fixed : 2개 >> 0 Dof
  >> None : 1개 >> 6?? Dof
  >> 하지만 저자의 Deepmimic 캐릭터에는 총 29개의 Body가 존재한다
```

```
>> 15개의 Body Shape + 14개의 Joint Shape로 구성
>> 17개의 Body Shape + 12개의 Joint Shape로 구성
```

(3). Deepmimic 저자의 Youtube에 올라와 있는 캐릭터 분석





>> 17개의 Body Shape와 12개의 Joint Shape로 구성되어있다.

>> **b_파랑색** : Body Shape

>> **j _ 빨강색** : Joint Shape

>> 그렇다면 Dof는 어떻게 설정되어있기에 44개의 Dof를 구성하는가?

[가정]

1. parent에 종속되어있는 joint는 parent의 joint형태를 따라간다.

```
## 위의 가정을 참으로 생각하고 Dof를 계산해보자
[Body Shape]
>> Spherical : 8 >> 24 Dof
>> Revolute : 4 >> 4 Dof
>> Free or None : 1 >> 6 Dof ??
>> Fixed : 2 >> 0 Dof
  >> 34 Dof or 28 Dof

Root - Free or None
Chest - Spherical
Neck - Spherical

Right
  Right_Hip - Spherical
  Right_Knee - Revolute
  Right_Ankle - Spherical

  Right_Shoulder - Spherical
  Right_Elbow - Revolute
  Right_Wrist - Fixed

Left
  Left_Hip - Spherical
  Left_Knee - Revolute
  Left_Ankle - Spherical
```

```
  Left_Shoulder - Spherical
  Left_Elbow - Revolute
  Left_Wrist - Fixed

[Joint Shape]
Example - Parent Joint (Child Joint)

Waist - Root (Chest) : Free or None
Neck - Neck (None) : Spherical

Right
  Clavicle - Chest (Shoulder) : Spherical
  Hip - Hip (Knee) : Spherical
  Knee - Knee (Ankle) : Revolute
  Ankle - Ankle (None) : Spherical

Left
  Clavicle - Chest (Shoulder) : Spherical
  Hip - Hip (Knee) : Spherical
  Knee - Knee (Ankle) : Revolute
  Ankle - Ankle (None) : Spherical

>> Spherical : 7 >> 21 Dof
>> Revolute : 2 >> 2 Dof
>> Free or None : 1 >> 6 Dof ??
  >> 29 Dof or 23 Dof

>> 전체 Dof
  >> 44 Dof가 아니다.
```

2. Joint shape들은 무조건 Revolute 형태로 존재한다.

```
## 위의 가정을 참으로 생각하고 Dof를 계산해보자
[Body Shape]
>> Spherical : 8 >> 24 Dof
>> Revolute : 4 >> 4 Dof
>> Free or None : 1 >> 6 Dof ??
>> Fixed : 2 >> 0 Dof
  >> 34 Dof or 28 Dof

Root - Free or None
Chest - Spherical
Neck - Spherical

Right
  Right_Hip - Spherical
  Right_Knee - Revolute
  Right_Ankle - Spherical

  Right_Shoulder - Spherical
  Right_Elbow - Revolute
  Right_Wrist - Fixed

Left
  Left_Hip - Spherical
  Left_Knee - Revolute
  Left_Ankle - Spherical

  Left_Shoulder - Spherical
  Left_Elbow - Revolute
  Left_Wrist - Fixed

[Joint Shape]
Example - Parent Joint (Child Joint)

Waist - Root (Chest) : Revolute
Neck - Neck (None) : Revolute

Right
  Clavicle - Chest (Shoulder) : Revolute
  Hip - Hip (Knee) : Revolute
  Knee - Knee (Ankle) : Revolute
  Ankle - Ankle (None) : Revolute

Left
  Clavicle - Chest (Shoulder) : Revolute
  Hip - Hip (Knee) : Revolute
  Knee - Knee (Ankle) : Revolute
  Ankle - Ankle (None) : Revolute
```

```
>> Revolute : 10 >> 10 Dof

>> 전체 Dof
  >> 44 Dof or 38 Dof
  >> 위의 가정일 확률이 높다
  >> PD Control로 위의 가정이 맞는지 확인해보자
  >> Fixed Joinit >> Dartpy Skeleton에서는 weld로 사용
```

[결과] - 홈페이지를 꼼꼼히 살펴보자

```
[
  duration of frame in seconds (1D),
  root position (3D),
  root rotation (4D),
  chest rotation (4D),
  neck rotation (4D),
  right hip rotation (4D),
  right knee rotation (1D),
  right ankle rotation (4D),
  right shoulder rotation (4D),
  right elbow rotation (1D),
  left hip rotation (4D),
  left knee rotation (1D),
  left ankle rotation (4D),
  left shoulder rotation (4D),
  left elbow rotation (1D)
]
총 44개의 정보
>> 1개는 time step
>> 43개 Dof (Quaternions(w,x,y,z) + Revolute) 형식
```

(4). Quaternion과 Euler

3. 캐릭터 구현

Dartpy 관련 함수들 사이트

[출처] :

> DART: dart::dynamics::Skeleton Class Reference
>
> Compute forward kinematics. In general, this function doesn't need to be called for forward kinematics to update. Forward kinematics will always be computed when it's needed and will only perform the computations that are necessary for what the user requests.
>
> https://dartsim.github.io/dart/v6.6.2/d3/d19/classdart_1_1dynamics_1_1Skeleton.html

(1). txt파일 형식을 skel 파일 형식으로

>> capsule 형식으로 되어있는 부분은 어떻게 해결할 것인가?

>> 직접 capsule을 그릴 것인가?

```
1.gluSSolidSphere 함수와 gluCyliner을 활용하는 방법

2.직접 Wireframe형식으로 그리는 방법
```

>> 사각기둥으로 대체할 것인가?

```
1. glutSolidSphere(크기)
```

>> sphere 형식으로 되어있는 부분은 어떻게 해결할 것인가?

>> 직접 sphere을 그릴 것인가?

```
1.glutSolidSphere(크기,slice,stacks)
```

>> 사각기둥으로 대체할 것인가?

```
1. glutSolidSphere(크기)
```

[Joint Shape 들을 제외한 Sphere 형식의 캐릭터 구현]

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/dcd6dfd8-cf1e-45e7-9a6e-2d25ab293a96/Deepmimic_캐릭터_구현_기.mp4

>> Joint Shape들의 경우에는 mass가 0이다. 어떻게 Joint Shape들을 움직일 것인가?

>> setForce를 활용하지 못한다. 어떻게 풀 것인가?

>> parent와 연결된 지점의 위치를 추출하여

>> setTransformation을 통해서 위치를 지정해줄 것인가?

>> 질량을 아주 작게라도 설정하여 줄 것인가?

>> Joint Shape들은 진짜 관절부위처럼 Body Shape들을 이어주는 역할을 하는 것이 아니다.

>> 단순히 시각적인 효과를 위해서 연결되어 있는 것이다.

>> 생략을 해도 될까?

>> Yes

>> 그렇다면 frame 파일에서 어떤 것이 Joint Shape들의 정보인지를 알아서

>> 제거하여 사용할 것인가?

>> 만약 Shape들의 순서에 따라서 Dof Frame 파일들이 정리되어있다면

>> 제일 끝에서부터 Joint들의 정보를 지워나가면 된다. |

>> No

>> 생략이 불가능하다면 왜 존재하는 것일까?

4. Walk 파일 읽기

(1). 코드

```
## Read_txt.py

import numpy as np
import sys
import os
import re
```

```
INF=[[1,3,4,4,4,4,1,4,4,1,4,1,4,4,1]]

def Read_txt():
    f = open("humanoid3d_walk.txt","r")
    lines = f.readlines()

    for i in range(4,43):
        numbers = re.findall("\d+\.\d+",lines[i])
        for j in range(len(numbers)):
            numbers[j] = float(numbers[j])

        List_1=[]
        Count = 0
        for k in INF[0]:
            List_2=[]

            for l in range(k):
                List_2.append(numbers[Count+l])

            List_1.append(List_2)
            Count += k

        INF.append(List_1)

if __name__=="__main__":
    sum = 0
    for i in INF[0]:
        sum += i
    Read_txt()
    print(INF[1][0][0])
```

(2). 코드 결과값

##Read_txt.py로 첫번째 프레임들 정보 추출한 결과값

```
tot4766@tot4766:~/intern/python/rllib/deepmimic$ python3 Read_txt.py
[[0.033332], [0.0, 0.847532, 0.0], [0.998678, 0.014104, 0.000698, 0.049423], [0.998813, 0.009485, 0.0
4756, 0.004475], [1.0, 0.0, 0.0, 0.0], [0.96494, 0.024369, 0.057555, 0.254922], [0.249116], [0.999366
, 0.009952, 0.032654, 0.010098], [0.985498, 0.064407, 0.093243, 0.126297], [0.170571], [0.992755, 0.0
20901, 0.088824, 0.078178], [0.391532], [0.982879, 0.101391, 0.05516, 0.143619], [0.965942, 0.188459,
 0.142246, 0.105854], [0.581348]]
```

##humanoid3d_walk.txt. 첫번째 프레임들 정보

```
[      0.0333320000,        0.0000000000,        0.8475320000,        0.0000000000,        0.9986780000,
0.0141040000,       -0.0006980000,       -0.0494230000,        0.9988130000,        0.0094850000,       -0.0475600000,
-0.0044750000,        1.0000000000,        0.0000000000,        0.0000000000,        0.0000000000,        0.9649400000,
0.0243690000,       -0.0575550000,        0.2549220000,       -0.2491160000,        0.9993660000,        0.0099520000,
0.0326540000,        0.0100980000,        0.9854980000,       -0.0644070000,        0.0932430000,       -0.1262970000,
0.1705710000,        0.9927550000,       -0.0209010000,        0.0888240000,       -0.0781780000,       -0.3915320000,
0.9828790000,        0.1013910000,       -0.0551600000,        0.1436190000,        0.9659420000,        0.1884590000,
-0.1422460000,        0.1058540000,        0.5813480000],
```

>> 무분별하게 연결되어있는 프레임 데이터들을

```
[
  duration of frame in seconds (1D),
  root position (3D),
  root rotation (4D),
  chest rotation (4D),
  neck rotation (4D),
  right hip rotation (4D),
  right knee rotation (1D),
  right ankle rotation (4D),
  right shoulder rotation (4D),
  right elbow rotation (1D),
  left hip rotation (4D),
  left knee rotation (1D),
  left ankle rotation (4D),
  left shoulder rotation (4D),
  left elbow rotation (1D)
]
총 44개의 정보
```

```
>> 1개는 time step
>> 43개 Dof (Quaternions(w,x,y,z) + Revolute) 형식
```

>> 이것과 같이 각각에 알맞는 숫자들로 나누어서 데이터를 구성하였다.