强化学习机械臂控制大作业说明

自动化与感知学院

2025年10月

目录

大作业目标

实践目标

利用仿真环境（PyBullet）设置基于Gym的强化学习仿真环境，

学习强化学习仿真环境的重要组成部分（观察、动作、奖励、环境交互），

训练强化学习网络，完成机械臂抓取任务的路径规划。

能力提升

掌握强化学习的基本概念；

熟悉强化学习在机器人控制领域的应用；

提高编程实践能力和问题解决能力。

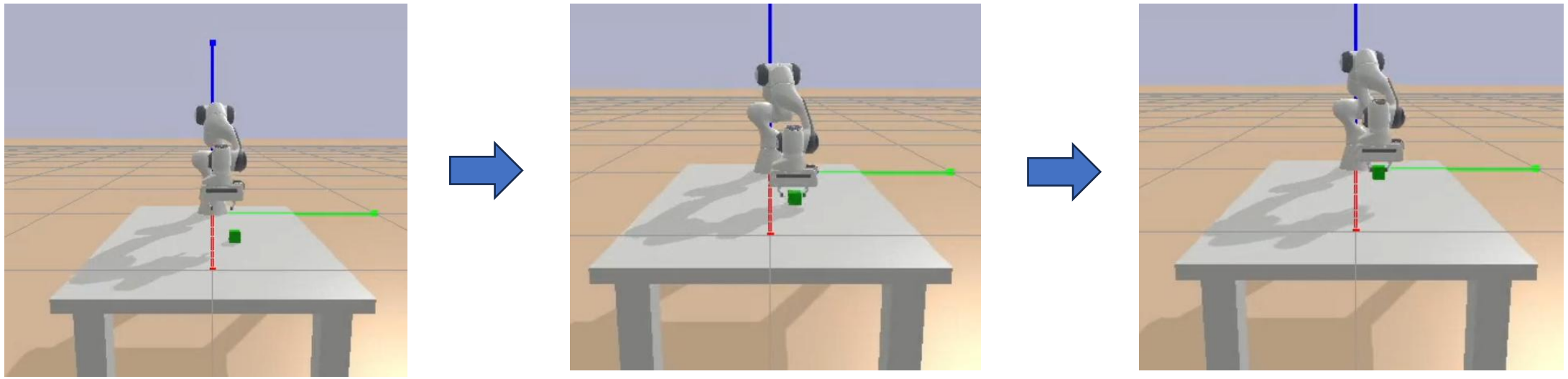
任务介绍

任务目标

任务目标

使用仿真环境pybullet完善抓取方块任务的仿真环境，并使用强化学习算法（DQN），完成抓取方块任务的操作策略的训练和优化。

为简化任务设置，减少训练时间，本次大作业只需要完成靠近目标物体的任务。即机械臂夹爪距离目标物体的距离小于阈值（0.005m）则视为任务成功。

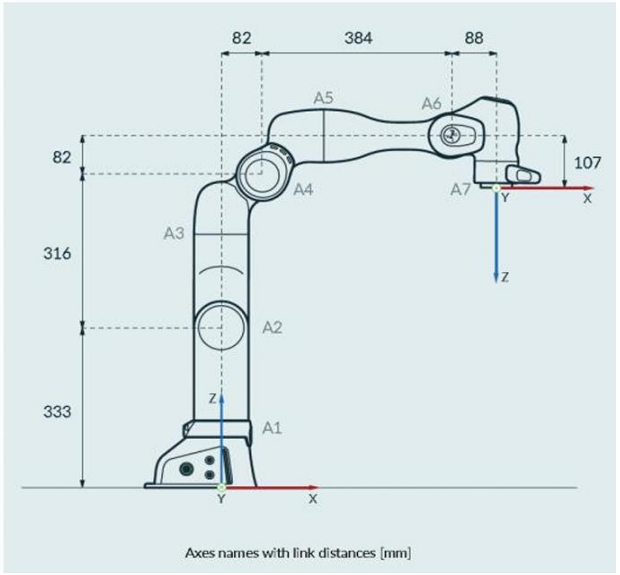


任务介绍

任务环境

智能体：

Franka Emika Panda机器人，7-轴机械臂，它的规格为3kg载重，850mm臂展。

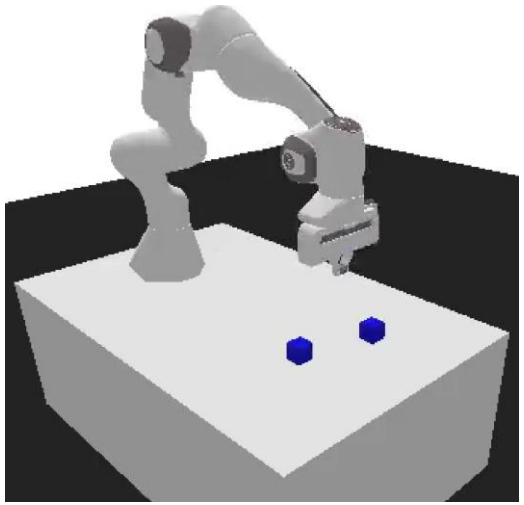


观察：

- 机械臂关节位置（9维）：7维机械臂关节+2维夹爪
- 机械臂夹爪位姿：7维（3维位置+4维四元数）
- 机械臂夹爪与目标物体的距离之差：3维
- 目标物体位姿：7维（3维位置+4维四元数）

动作（离散空间）：

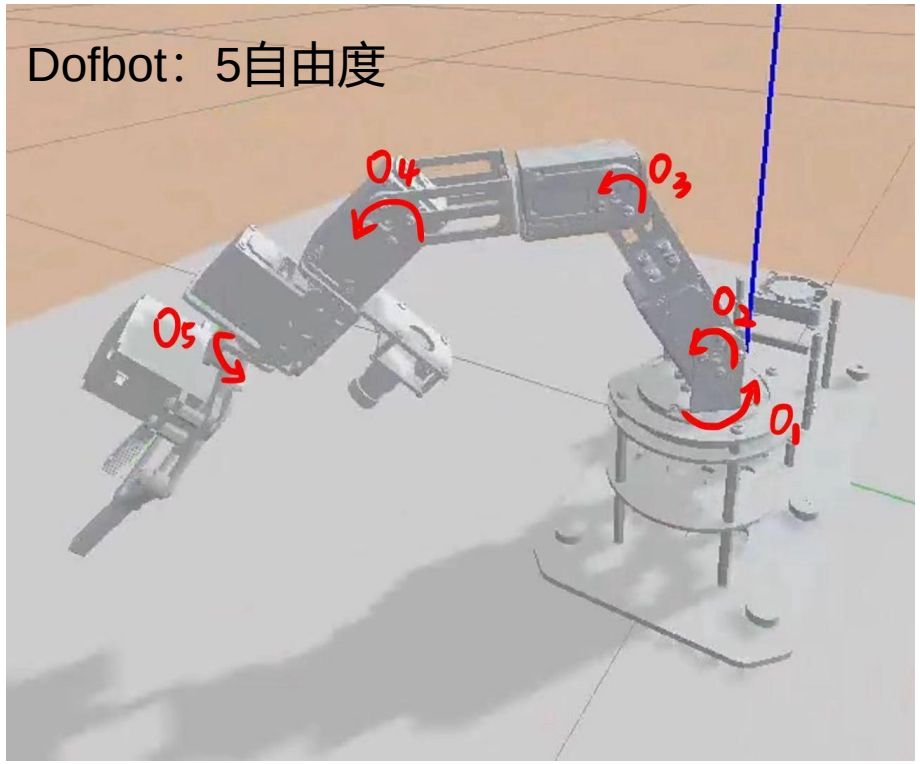
- 动作1：夹爪沿x轴负方向移动0.01m
- 动作2：夹爪沿x轴正方向移动0.01m
- 动作3：夹爪沿y轴负方向移动0.01m
- 动作4：夹爪沿y轴正方向移动0.01m
- 动作5：夹爪沿z轴负方向移动0.01m
- 动作6：夹爪沿z轴正方向移动0.01m
- 动作7：保持静止

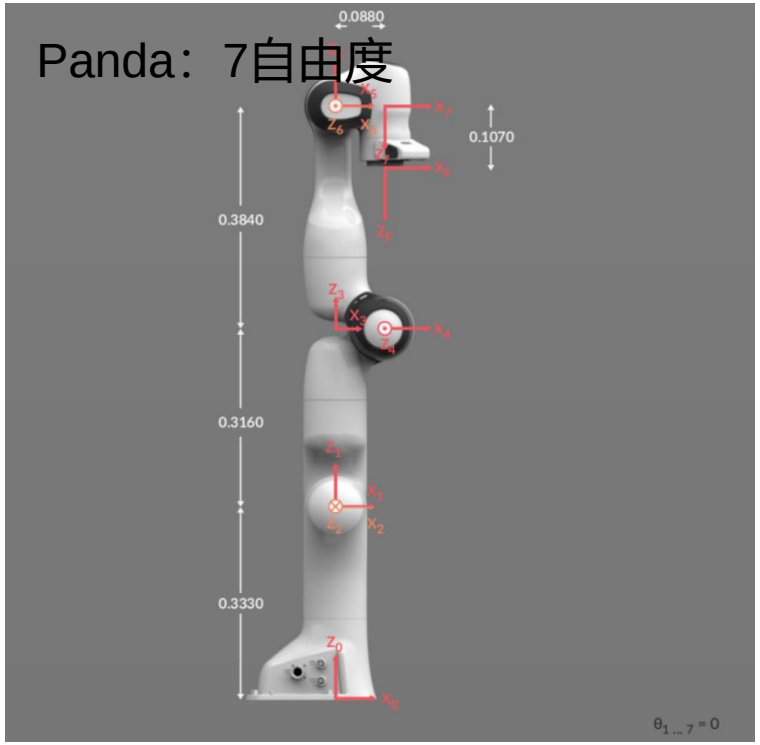


任务介绍

机械臂自由度

动作空间：笛卡尔空间（末端夹爪的位姿）
控制逻辑：输入动作→末端位姿→（逆运动学）关节位置





RL动作采样逻辑：在笛卡尔空间中随机采样

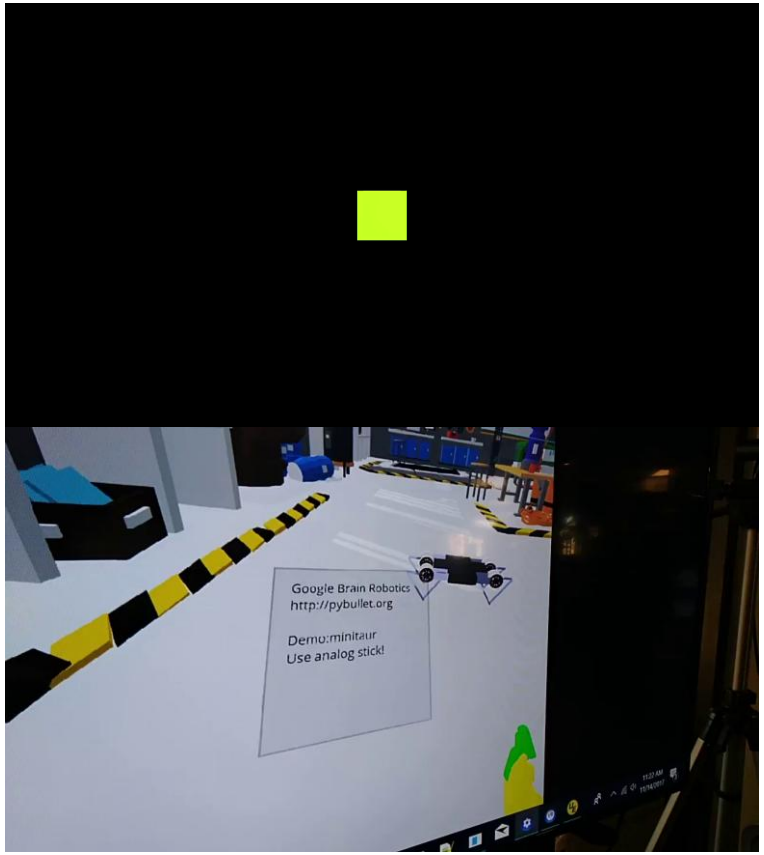
自由度不足：逆运动学可能无解

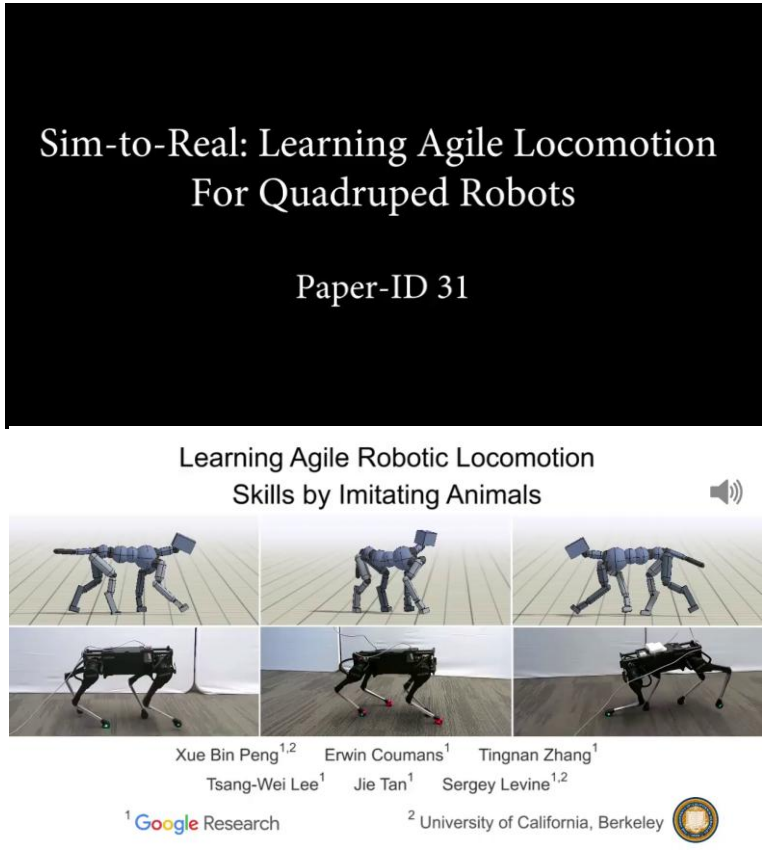
仿真环境搭建

Pybullet

PyBullet 基于著名的开源物理引擎 bullet 开发，封装成了 Python 的一个模块，用于机器人仿真和学习。PyBullet 支持加载 URDF、SDF、MJCF 等多种机器人描述文件，并提供正/逆向运动学、正/逆向动力学、碰撞检测等功能。(https://pybullet.org/wordpress/)，Bullet 物理 SDK 包括 PyBullet 机器人示例，如模拟的 Minitaur 四足机器人、使用 TensorFlow 推理的仿人机器人跑步和 KUKA 机械臂抓取物体。

PyBullet安装：pip install pybullet





仿真环境搭建

Gym环境

Gym介绍

Gym 是 OpenAI 提供的一个用于开发和测试强化学习算法的工具库，包含多种标准化的环境接口（如机器人控制、游戏等）。它通过统一的 API，方便用户创建、交互和评估各种强化学习任务。

Gym环境组成

1. 状态空间（observation space)：描述环境的状态，例如位置、速度等。状态可以是连续的或离散的。
2. 动作空间（action space)：描述智能体可以采取的动作，例如移动方向、速度等。动作空间也可以是离散或连续的。
3. 环境接口
   - env.reset()：重置环境，返回初始状态。
   - env.step(action)：执行动作，返回下一个状态、奖励、是否结束和额外信息。
   - env.render()：渲染环境，用于可视化。
   - env.close()：关闭环境，释放资源。

抓取任务训练

深度学习框架Pytorch

PyTorch 是一个开源的机器学习库，主要用于进行计算机视觉（CV）、自然语言处理（NLP）、语音识别等领域的研究和开发。

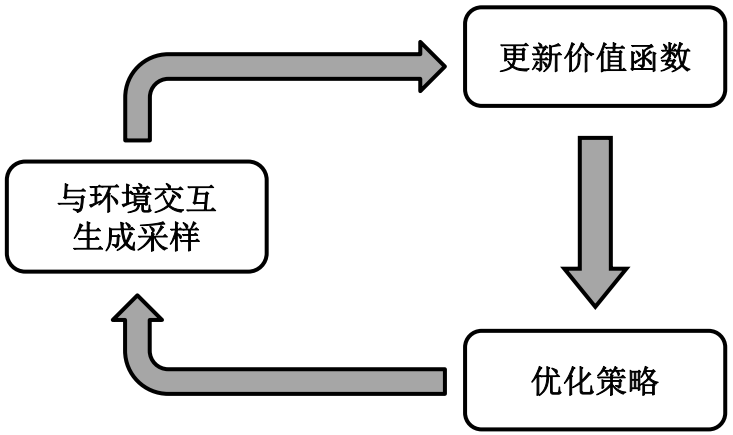
◆ 张量操作：
- 创建张量：torch.tensor(data)；torch.rand(size)
- 张量属性：张量形状.shape；张量的数据类型.dtype；张量所在设备.device
- 形状操作：矩阵乘法torch.matmul(x, y)；返回最大值的索引torch.argmax(x, dim)；计算softmax torch.softmax(x, dim)

◆ torch.nn 模块：构建和训练神经网络的核心模块
- nn.Module ： 所有自定义神经网络模型的基类
- 损失函数：均方误差损失（nn.MSELoss）、交叉熵损失（nn.CrossEntropyLoss）等
- 容器类：nn.Sequential：允许将多个层按顺序组合起来，形成简单的线性堆叠网络。
- 线性层函数：torch.nn.Linear(in_features, out_features)
- 激活函数：torch.nn.ReLU()；torch.nn.Tanh()…

抓取任务训练

网络结构

$$Q_\theta(s,a) \leftarrow r(s,a) + \gamma \max_{a'} Q_\theta(s',a')$$



$$a = \arg\max_a Q_\theta(s,a)$$

所以这里的动作空间只能是离散的！！

Q-network:

```python
# ALGO LOGIC: initialize agent here:
class QNetwork(nn.Module):
    def __init__(self, env):
        super().__init__()
        self.network = nn.Sequential(
            nn.Linear(np.array(env.single_observation_space.shape).prod(), 120),
            nn.ReLU(),
            nn.Linear(120, 84),
            nn.ReLU(),
            nn.Linear(84, env.single_action_space.n),
        )

    def forward(self, x):
        return self.network(x)
```

优化策略:

```python
q_values = q_network(torch.Tensor(obs).to(device))
actions = torch.argmax(q_values, dim=1).cpu().numpy()
```

更新价值函数:

```python
# update target network
if global_step % args.target_network_frequency == 0:
    for target_network_param, q_network_param in zip(target_network.parameters(), q_network.parameters()):
        target_network_param.data.copy_(
            args.tau * q_network_param.data + (1.0 - args.tau) * target_network_param.data
        )
```

抓取任务训练

网络结构：DQN

训练流程伪代码

初始化容量为 N 经验回放单元 D
用随机参数 θ 初始化当前值网络 Q
用一致的参数 $\theta^- = \theta$ 初始化目标网络 $\hat{Q}$
for $e = 1, 2, \ldots, E$
  $k \leftarrow 1$
  选择一个随机的初始状态 s_1
    while 目标状态未到 and $k \leq T$
      随机选择一个有效动作 a_k
      记录下一个状态 s_{k+1} 和对应的奖励 r_k | 经验回放
      将 $(s_k,\ a_k,\ r_k,\ s_{k+1})$ 存入经验回放单元
      从经验回放单元随机取出一批 $(s_j,\ a_j,\ r_j,\ s_{j+1})$
      取 $y_j = \begin{cases} r_j & s_{j+1} \text{ 为终止状态} \\ r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \theta^-) & \text{其他} \end{cases}$ | 双网络
      使用梯度下降法更新 $\theta \leftarrow \arg\min_\theta \frac{1}{2} \sum_i \left\| Q(s_j, a_j; \theta) - y_j \right\|^2$ | 双网络
      每隔 C 步，同步参数 $\hat{Q} = Q$ | 双网络
      $k \leftarrow k + 1$
  end while
end for

作业（机械臂强化学习控制部分）

任务目标

使用仿真环境pybullet搭建抓取方块任务的仿真环境，并使用强化学习算法，完成抓取方块任务的操作策略的训练和优化。

报告要求

- 任务一：完善仿真环境中的**奖励函数**。
- 任务二：定义**动作空间和观测空间**。
- 任务三：在任务二的基础上完善**step函数**，根据输入动作下发机械臂控制指令。利用test_gym.py验证动作的可行性。
- 任务四：基于DQN算法，完成**操作策略的训练**，记录训练曲线，并测试效果。

作业（机械臂强化学习控制部分）

任务一：在代码panda_env.py中，完善仿真环境中的**奖励函数**。

提示：可以考虑以下几个方向

- **靠近奖励**：目标方块和夹爪（TCP）的距离越近，奖励越大。
- **到达奖励**：当目标方块和夹爪的距离小于目标阈值时，给予奖励。

```
# TODO: 完善reward function
def _get_reward(self):
    obs = self._get_obs_dict()
    info = self._get_info()

    reward = 0
    return reward
```

[要求]
设计合理的奖励函数，使得强化学习网络能够根据该奖励函数学习到正确的动作

作业（机械臂强化学习控制部分）

- 任务二：**定义动作空间和观测空间。**

[要求1] 完成环境初始化中的observation space和action space的定义。

```
# TODO: observation space
# if obs_mode == "state": ## 训练模式下使用
#     self.observation_space =
# elif obs_mode == "state_dict":  ## 字典形式，方便读取数据
#     self.observation_space =

# TODO: action space
# self.action_space =
```

[要求2]完成_get_obs函数，从环境中得到观测信息。

```
def _get_obs_dict(self):
    Observation = self._panda.getObservation()

    # TODO:  add suitable observations here

    return Observation
```

获取物体位姿函数：

- getBasePositionAndOrientation(objectUniqueId)：返回位置列表（包含3个浮点数）以及方向列表（包含4个浮点数，按 [x, y, z, w] 顺序排列）。

作业（机械臂强化学习控制部分）

- 任务三：在任务二的基础上完善**step函数**，根据输入动作下发机械臂控制指令。利用test_gym.py验证动作的可行性。

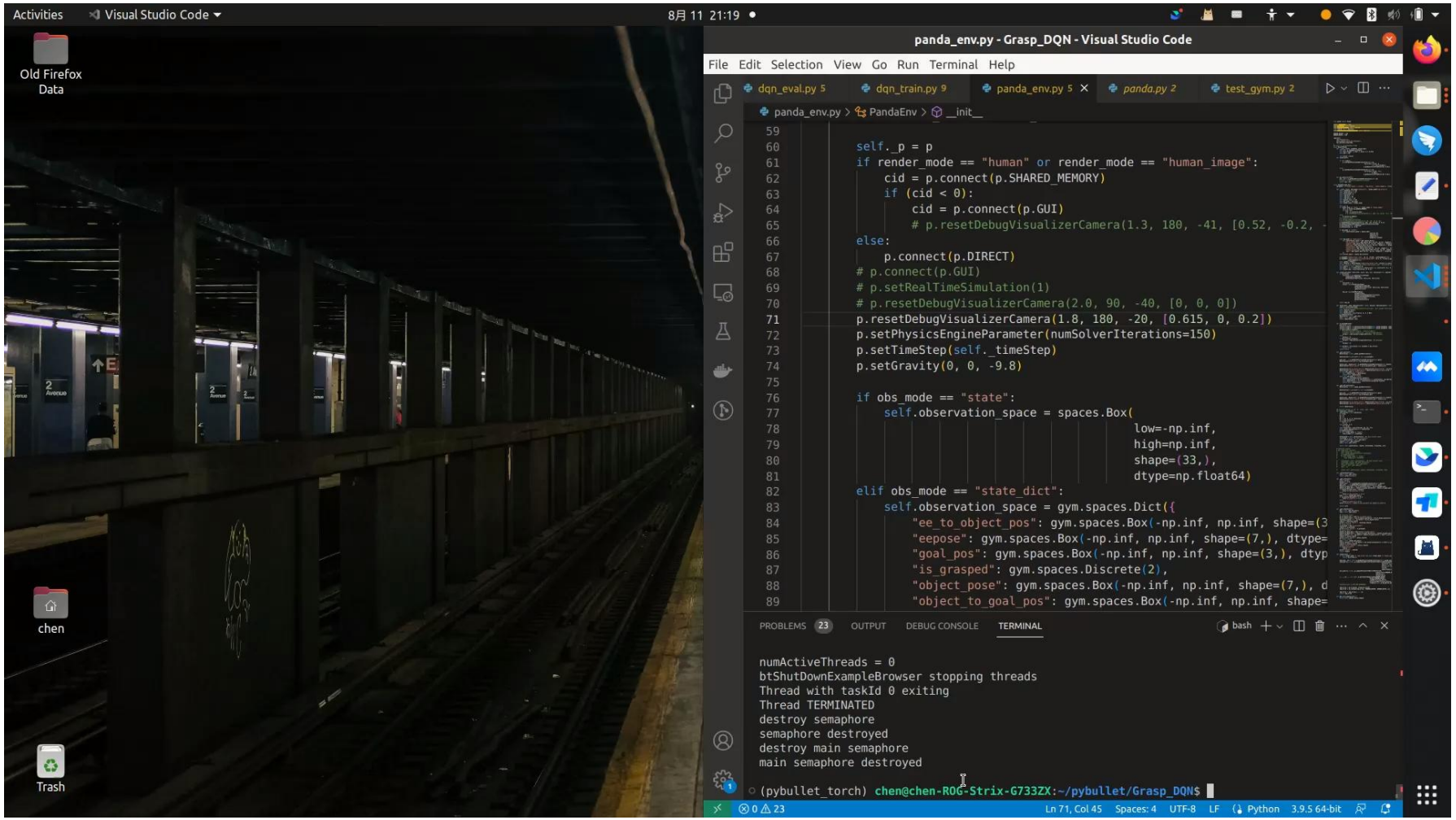
```python
## discrete action: -dx, dx, -dy, dy, -dz, dz, static
def step(self, action):

    # TODO: Define suitable realAction here
    # self.realAction = np.array([dx, dy, dz, 0.04])


    if self.terminated:
        self.realAction = np.array([0, 0, 0, 0])
    self._panda.applyAction(self.realAction)
    p.stepSimulation()
    if self.render_mode == "human":
        time.sleep(self._timeStep)

    terminated = self._termination()  ## task success check
    truncated = False  ## step limitation
    self._observation = self._get_obs()
    reward = self._get_reward()
    info = self._get_info()

    return self._observation, reward, terminated, truncated, info
```

Step函数详解：

1. 将输入的离散动作转换为机械臂任务空间的动作：机械臂夹爪目标位置、方向和夹爪开合程度
2. 计算逆运动学，将机械臂夹爪位置转换为关节空间位置
3. 利用PD控制下发控制指令
4. 返回下一时间步的观测信息、任务是否完成标志以及获得的奖励。

注意：每一步位置的变化量的大小需要仔细考虑

作业（机械臂强化学习控制部分）

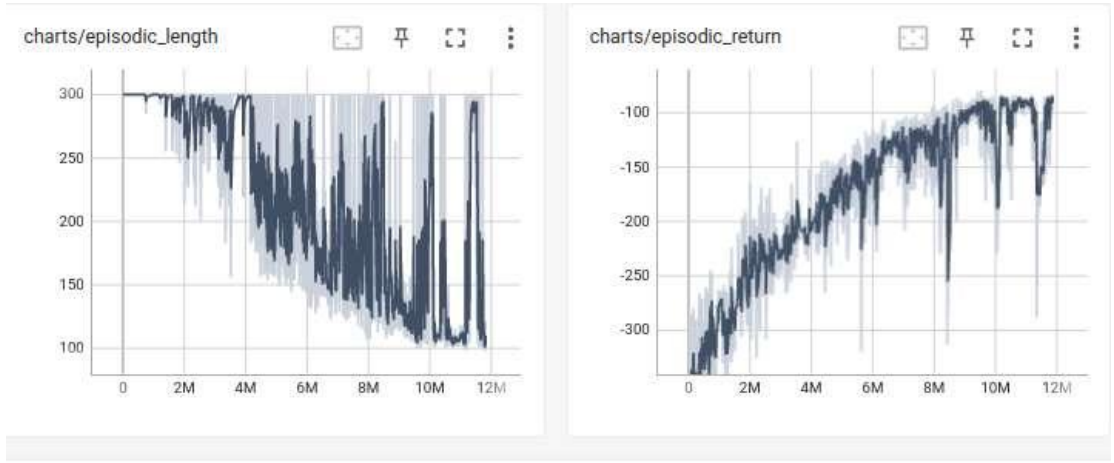


运行test_gym.py，在命令行中输入动作，在可视化窗口中验证动作设计的合理性

作业（机械臂强化学习控制部分）

任务四：基于DQN算法，完成**操作策略的训练**，记录训练曲线，并测试效果。

训练曲线示例：



[要求]

1. 基于提供的DQN代码，训练抓取任务的强化学习策略
2. 使用tensorboard添加训练信息记录，并可视化训练数据
3. 测试策略效果，在提交附件中包含视频形式的结果展示

训练曲线记录工具：tensorboard

```
pip install tensorboard
```

记录事件：

```
from torch.utils.tensorboard import SummaryWriter

# 初始化SummaryWriter
writer = SummaryWriter('runs/experiment_name')
```

记录信息：

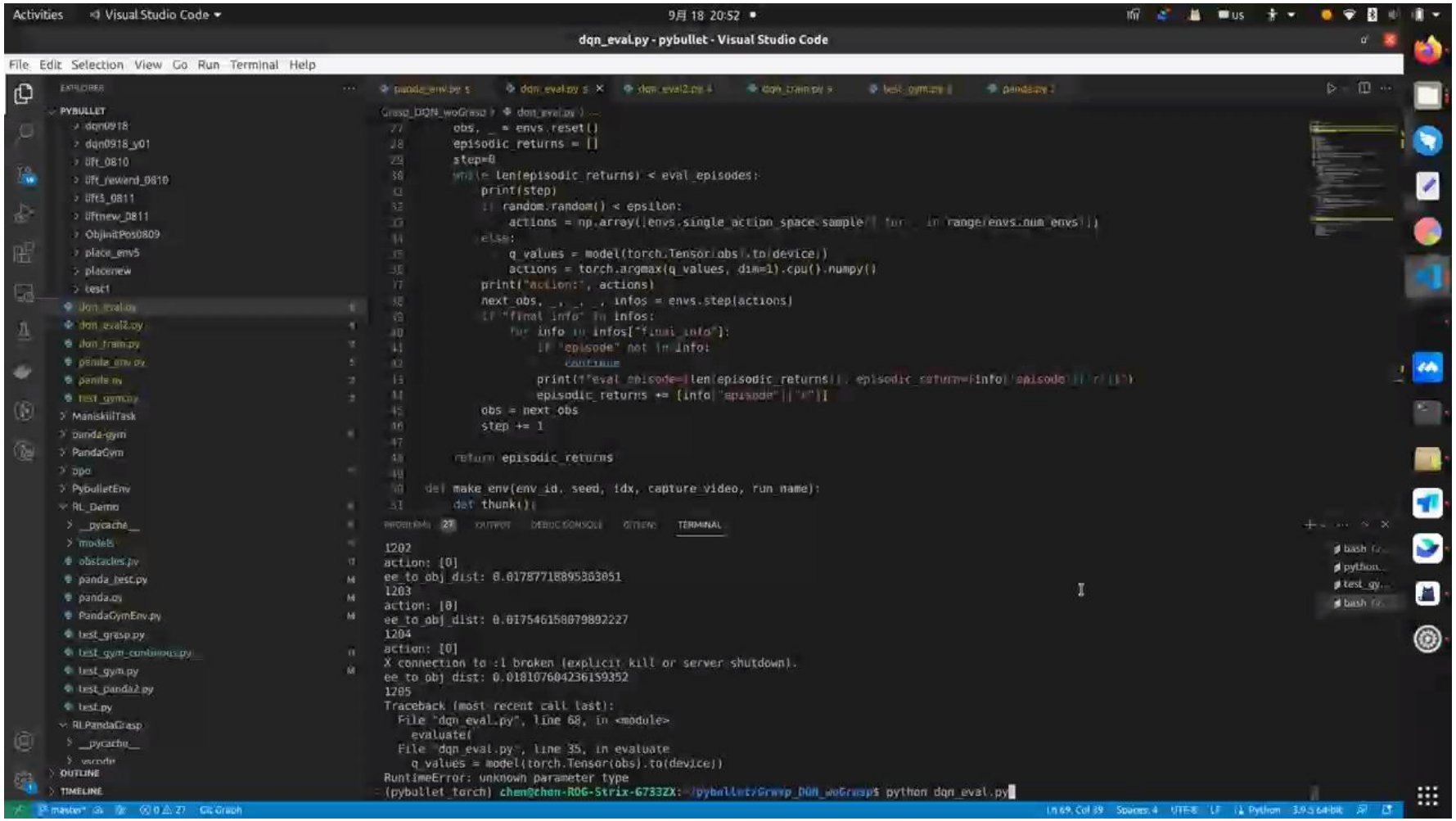
```
writer.add_scalar("losses/td_loss", loss, global_step)
writer.add_scalar("losses/q_values", old_val.mean().item(), global_step)
```

信息可视化：

tensorboard --logdir=runs/

作业（机械臂强化学习控制部分）

抓取策略展示



作业（机械臂强化学习控制部分）

报告要求

- 任务一：完善仿真环境中的**奖励函数。**

  [要求] 设计合理的奖励函数，使得强化学习网络能够根据该奖励函数学习到正确的动作

- 任务二：定义**动作空间和观测空间。**

  [要求]

  1. 完成环境初始化中的observation space和action space的定义
  2. 完成_get_obs函数，从环境中得到观测信息。

- 任务三：在任务二的基础上完善**step函数**，根据输入动作下发机械臂控制指令。利用test_gym.py验证动作的可行性。

- 任务四：基于DQN算法，完成**操作策略的训练**，记录训练曲线，并测试效果。

  [要求]

  1. 基于提供的DQN代码，训练抓取任务的强化学习策略
  2. 使用tensorboard添加训练信息记录，并可视化训练数据
  3. 测试策略效果，测试20次，记录成功率；并在提交附件中包含视频形式的结果展示
  4. 通过可视化的训练数据和策略效果，分析此次训练的结果，并提出未来可能的改进方向

拓展任务

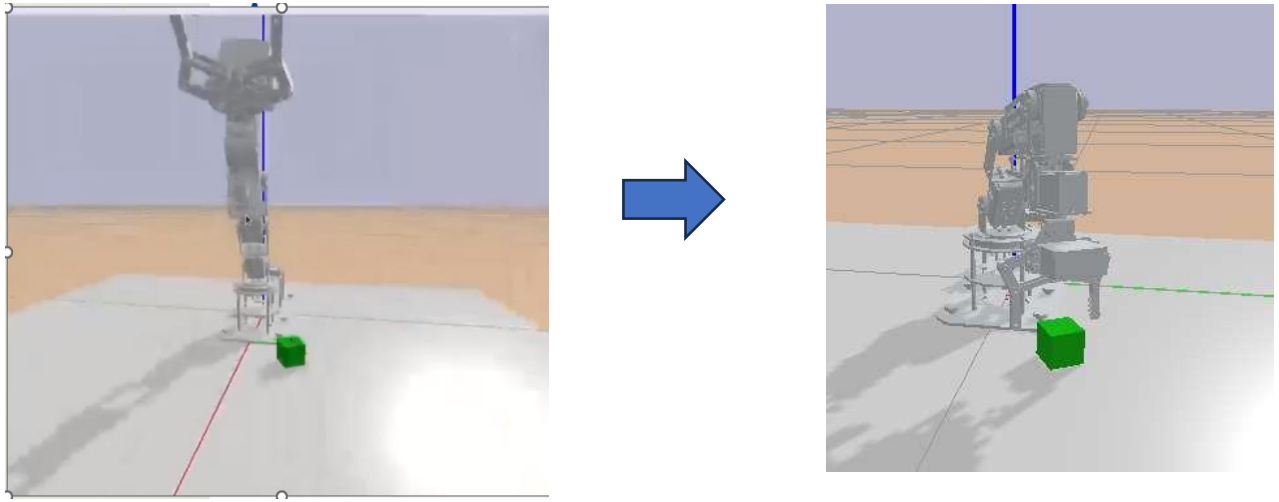
思考：有没有办法可以通过强化学习训练Dofbot这种缺少自由度的机械臂?

答案：

- 确保完成任务的途径点，机械臂末端都是可达的
- 将关节空间作为动作空间，不存在解逆运动学的问题

支持的算法：可用于**连续动作空间**的强化学习算法

- PPO、SAC…. (操作领域用的较多，算法效果好)



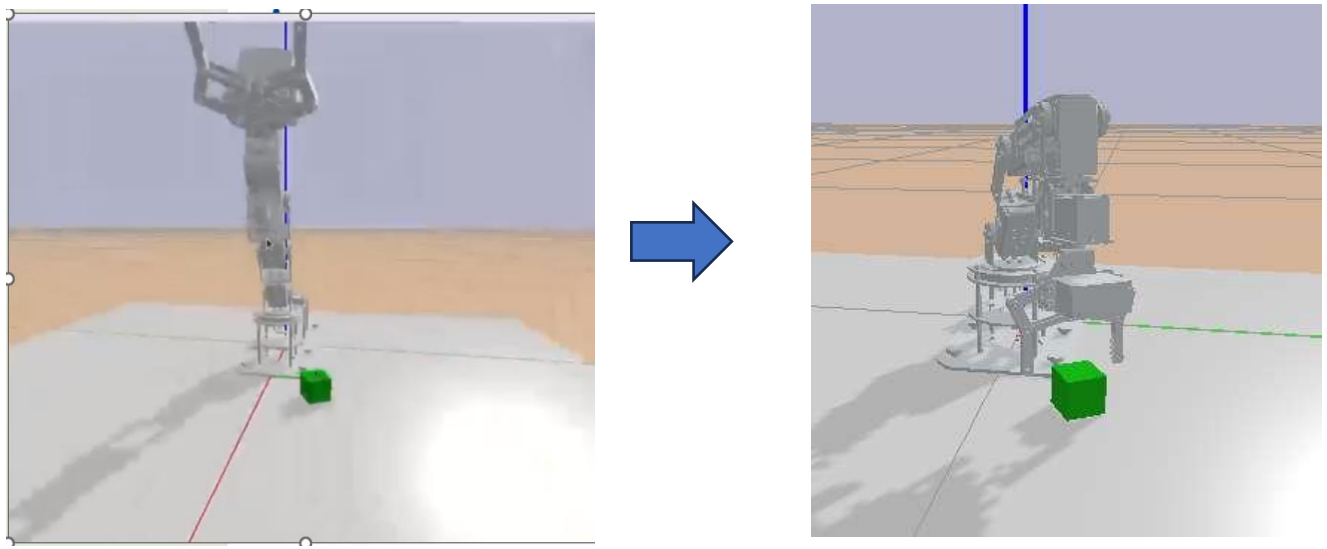
拓展任务

任务介绍

任务目标

使用仿真环境pybullet完善抓取方块任务的仿真环境，并使用强化学习算法（SAC），完成抓取方块任务的操作策略的训练和优化。

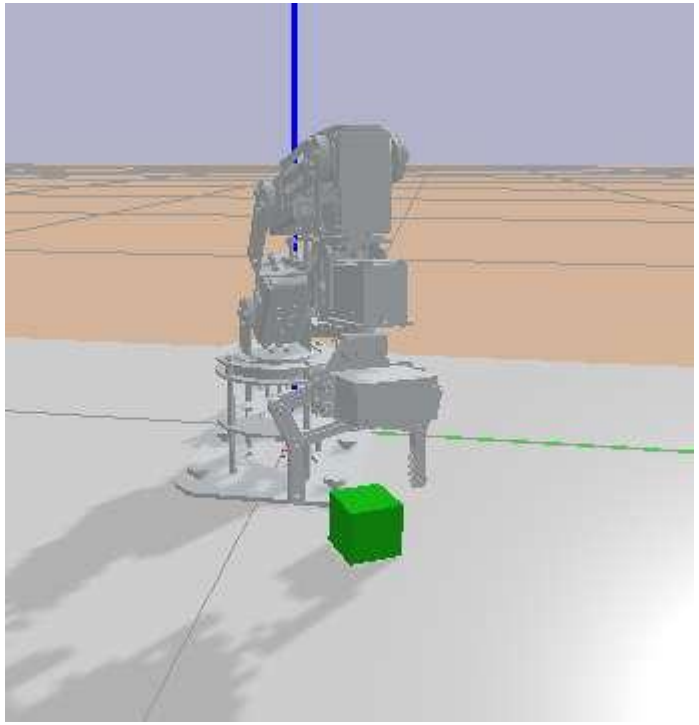
为简化任务设置，减少训练时间，本次任务只需要完成靠近目标物体的任务。即机械臂夹爪距离目标物体的距离小于阈值（0.01m）则视为任务成功。



任务环境

智能体：

Dofbot机械臂，5自由度。

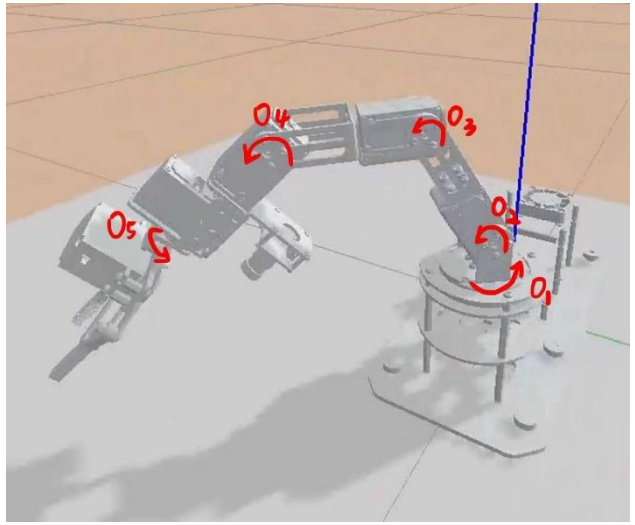


观察：

- 机械臂关节位置（6维）：5维机械臂关节+1维夹爪
- 机械臂夹爪位姿：7维（3维位置+4维四元数）
- 物体在机械臂夹爪坐标系下的位姿： 7维（3维位置+4维四元数）

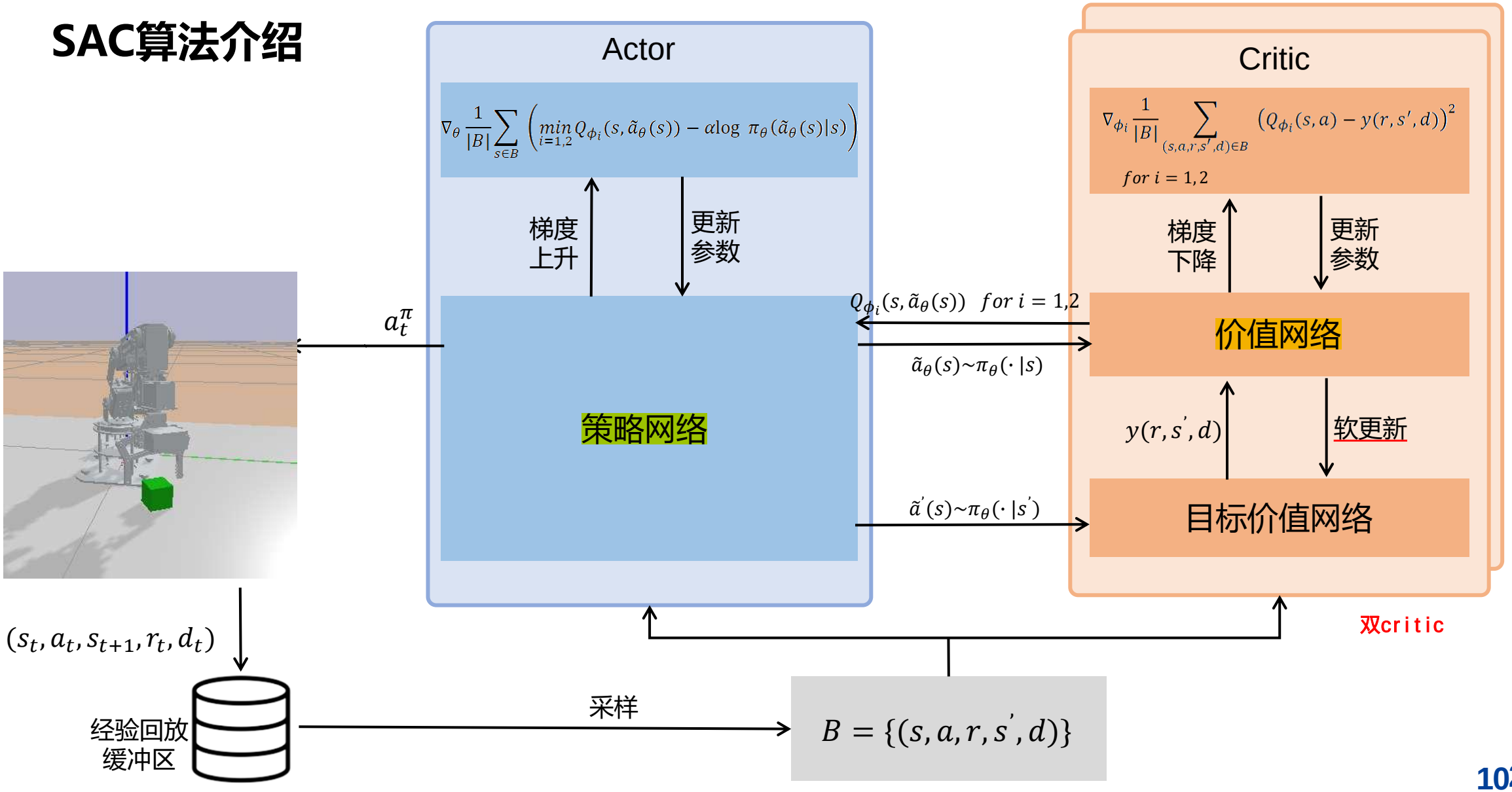
动作（连续空间）：

- 机械臂5个关节的变化位置



方法介绍

SAC算法介绍



方法介绍

训练流程伪代码

Algorithm 1 Soft Actor-Critic (SAC)

Input: 初始策略参数 θ, 价值网络参数 ϕ_1, ϕ_2, 空经验回放缓冲区 $\mathcal{D}$

初始化目标价值网络参数: $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$

repeat

  得到状态 s, 采样动作 $a \sim \pi_\theta(\cdot|s)$

  在环境中执行动作 a

  得到下一时间步的 s', 奖励 r, 回合结束标志 d

  存储转移信息 (s, a, r, s', d) 到 $\mathcal{D}$ 中

  if 回合结束 **then**

    重启环境

  end if

  if 达到更新间隔 T_{update} **then**

    for j in T_{grad} **do**

      采样一个批次的转移信息 $B = \{(s, a, r, s', d)\} \sim \mathcal{D}$

      计算目标 Q 值:

$$y(r, s', d) = r + \gamma(1-d)\left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}' \mid s')\right), \quad \tilde{a}' \sim \pi_\theta(\cdot \mid s')$$

      通过梯度下降更新价值网络: $\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_{\phi_i}(s, a) - y(r, s', d)\right)^2$ for $i = 1, 2$

      通过梯度上升更新策略网络: $\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s) \mid s)\right)$

      对目标价值网络进行软更新: $\phi_{\text{targ},i} \leftarrow \tau \varphi_{\text{targ},i} + (1-\tau)\phi_i$ for $i = 1, 2$

算法介绍网址：Soft Actor-Critic —

拓展任务（机械臂强化学习控制部分）

报告要求

- 任务一：完善仿真环境中的**奖励函数。**

  [要求] 设计合理的奖励函数，使得强化学习网络能够根据该奖励函数学习到正确的动作

- 任务二：定义**动作空间和观测空间。**

  [要求]

  1. 完成环境初始化中的observation space和action space的定义
  2. 完成_get_obs函数，从环境中得到观测信息。

- 任务三：在任务二的基础上完善**step函数**，根据输入动作下发机械臂控制指令。
- 任务四：通过求解逆运动学，找到逆运动学可解的位置，放置方块。
- 任务五：基于SAC算法，完成**操作策略的训练**，记录训练曲线，并测试效果。

  [要求]

  1. 基于提供的SAC代码，训练抓取任务的强化学习策略
  2. 使用tensorboard添加训练信息记录，并可视化训练数据
  3. 测试策略效果，测试50次，记录成功率；并在提交附件中包含视频形式的结果展示
  4. 通过可视化的训练数据和策略效果，分析此次训练的结果，并提出未来可能的改进方向

Dofbot机械臂仿真环境

关节角度控制

求解逆运动学

获取关节位置

获取关节速度

```python
def joint_control(self,dqpos):
    self.desire_qpos = self.desire_qpos + dqpos
    jointPoses = self.desire_qpos
    for i in range(self.numJoints):
        p.setJointMotorControl2(bodyUniqueId=self.dofbotUid, jointIndex=i, controlMode=p.POSITION_CONTROL,
                                targetPosition=jointPoses[i], targetVelocity=0, force=200,
                                maxVelocity=10.0, positionGain=0.3, velocityGain=1)
    self.jointPositions, self.gripperAngle = self.get_jointPoses()
    self.endEffectorPos, self.endEffectorOrn, self.endEffectorEuler = self.get_pose()
    return self.endEffectorPos, self.endEffectorOrn, self.endEffectorEuler

def setInverseKine(self, pos, orn):
    if orn == None:
        jointPoses = p.calculateInverseKinematics(self.dofbotUid, 4, pos,
                                                  self.ll, self.ul, self.jr, self.rp)
    else:
        jointPoses = p.calculateInverseKinematics(self.dofbotUid, 4, pos, orn,
                                                  self.ll, self.ul, self.jr, self.rp)
    return jointPoses[:self.numJoints], self.gripperAngle

def get_jointPoses(self):
    jointPoses= []
    for i in range(self.numJoints+1):
        state = p.getJointState(self.dofbotUid, i)
        jointPoses.append(state[0])
    return jointPoses[:self.numJoints], self.gripperAngle

def get_qvel(self):
    jointVels= []
    for i in range(self.numJoints+1):
        state = p.getJointState(self.dofbotUid, i)
        jointVels.append(state[1])
    return np.array(jointVels[:self.numJoints])
```

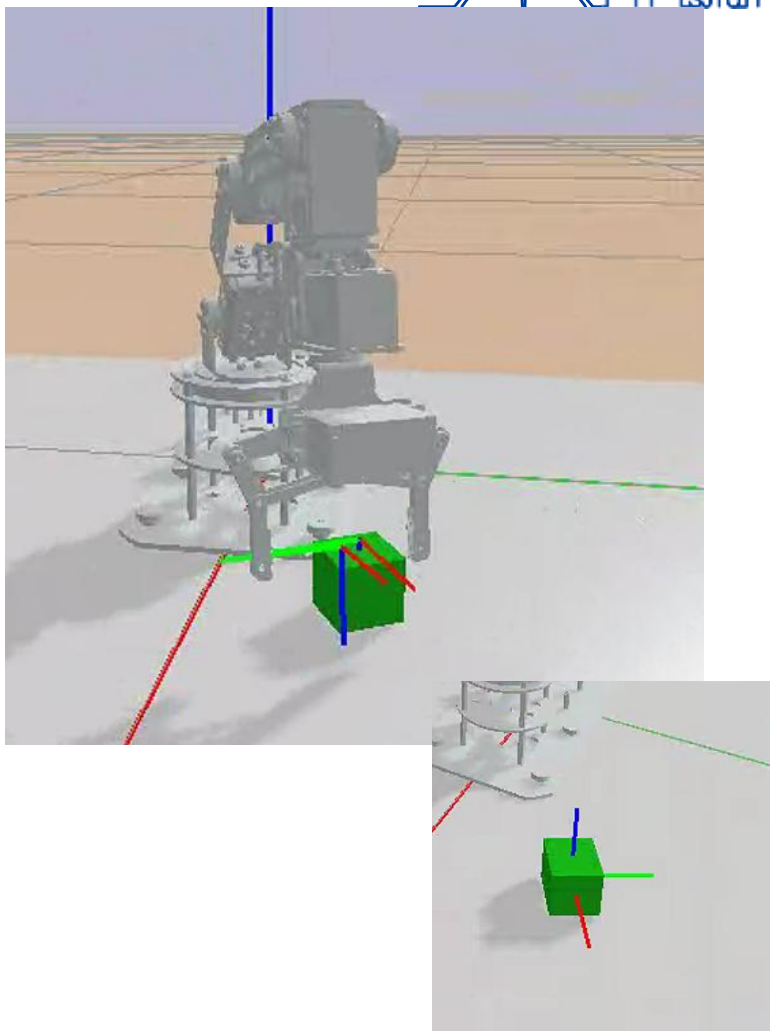
拓展任务（机械臂强化学习控制部分）

任务一：在代码dofbotGymEnv.py中，完善仿真环境中的**奖励函数**。

提示：可以考虑以下几个方向

- **靠近奖励**：目标方块和夹爪（TCP）的距离越近，奖励越大。
- **姿态奖励**：夹爪姿态与期望抓取姿态的差越小，奖励越大。
- **到达奖励**：当目标方块和夹爪的距离小于目标阈值时，给予奖励。

```
# TODO: design suitable reward funyion
def _get_reward(self):
    obs = self._get_obs_dict()
    info = self._get_info()

    reward = 0
    return reward
```



[要求]
设计合理的奖励函数，使得强化学习网络能够根据该奖励函数学习到正确的动作

拓展任务（机械臂强化学习控制部分）

- 任务二：定义**动作空间和观测空间。**

[要求1] 完成环境初始化中的observation space和action space的定义。

为了训练方便，一般设置动作空间大小为[-1, 1]

```
# TODO: define observation space and action space
# self.observation_space =
# self.action_space =
```

[要求2]完成_get_obs函数，从环境中得到观测信息。

```
# TODO: complete observation
def _get_obs(self):
    Observation = self._panda.getObservation()

    # TODO: add suitable observation items here


    if self.obs_mode == "state_dict":
        self._observation = Observation
        return self._observation
    elif self.obs_mode == "state":
        values = list(Observation.values())
        self._observation = np.concatenate([v if isinstance(v, np.ndarra
        self._observation = self._observation.astype(np.float32)
        return self._observation
```

获取物体位姿函数：

- getBasePositionAndOrientation(objectUniqueId)：返回位置列表（包含3个浮点数）以及方向列表（包含4个浮点数，按 [x, y, z, w] 顺序排列）。

拓展任务（机械臂强化学习控制部分）

- 任务三：在任务二的基础上完善**step函数**，根据输入动作下发机械臂控制指令。

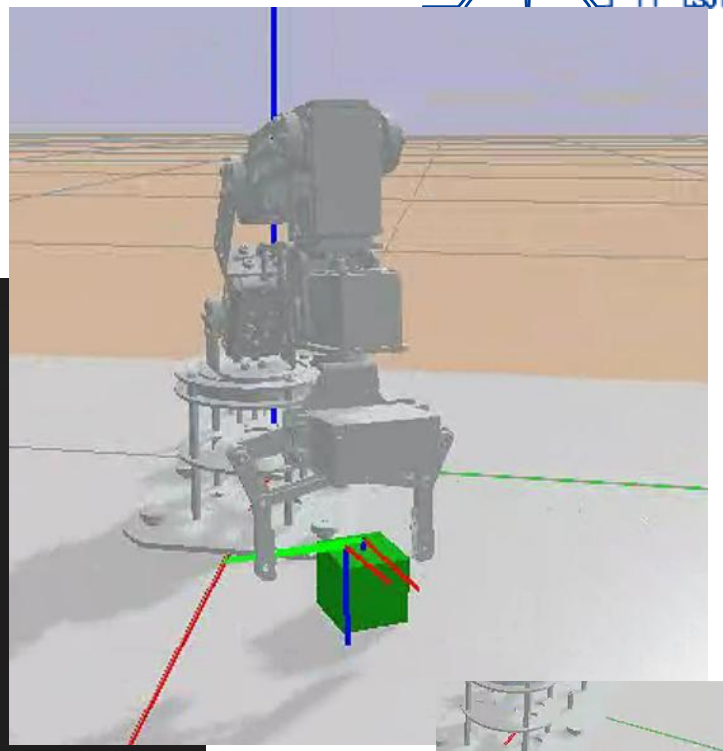
  提示：注意动作空间的范围，要转换为真实的机械臂关节位置变化量的大小

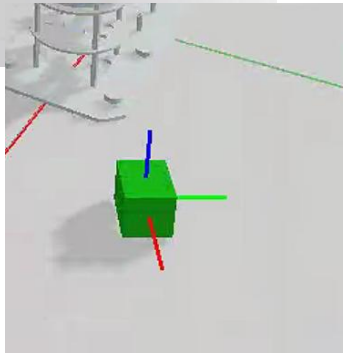
```python
def step(self, action):
    """
    action - qpos np.array(5), gripper keeps open
    """
    # TODO: complete step control of dofbot

    for i in range(self.simuRepeatNum):
        p.stepSimulation()

    if self.render_mode == "human":
        time.sleep(self._timeStep)
    terminated = self._termination()
    truncated = False
    self._observation = self._get_obs()
    reward = self._get_reward()
    info = self._get_info()
    return self._observation, reward, terminated, truncated, info
```

拓展任务（机械臂强化学习控制部分）

- 任务四：通过求解逆运动学，找到逆运动学可解的位姿，放置方块。

```
class Object:
    def __init__(self, urdfPath, block,num):
        self.id = p.loadURDF(urdfPath)
        self.half_height = 0.015 if block else 0.0745
        self.num = num

        self.block = block
    def reset(self):

        if self.num==1:
            p.resetBasePositionAndOrientation(self.id,
                                              np.array([ 0.20, 0.1,
                                                        self.half_height]),
                                              p.getQuaternionFromEuler([0, 0,np.pi/6]))
```



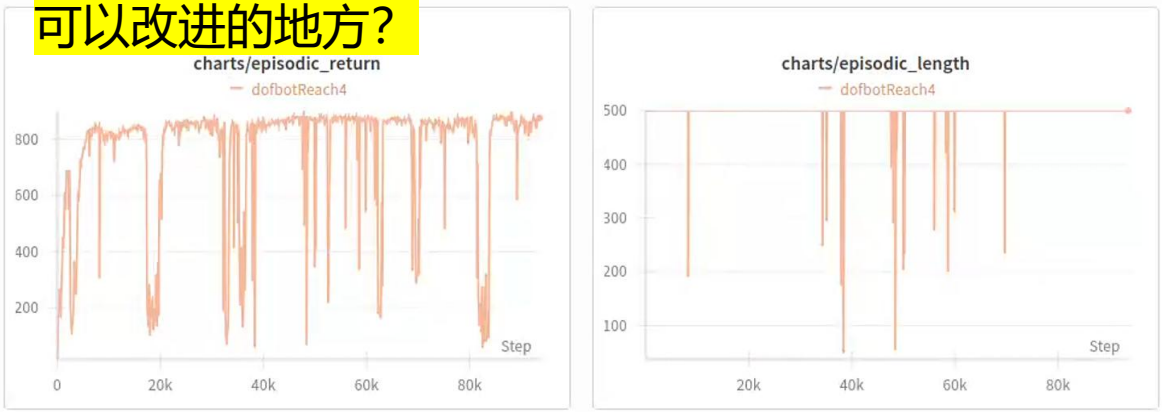


拓展任务（机械臂强化学习控制部分）

任务五：基于SAC算法，完成**操作策略的训练**，记录训练曲线，并测试效果。

训练曲线示例：

思考：为什么训练曲线会有很大的波动？有没有可以改进的地方？



[要求]

1. 基于提供的SAC代码，训练抓取任务的强化学习策略
2. 使用tensorboard添加训练信息记录，并可视化训练数据
3. 测试策略效果并分析原因，在提交附件中包含视频形式的结果展示

训练曲线记录工具：tensorboard

```
pip install tensorboard
```

记录事件：

```
from torch.utils.tensorboard import SummaryWriter

# 初始化SummaryWriter
writer = SummaryWriter('runs/experiment_name')
```

记录信息：

```
writer.add_scalar("losses/td_loss", loss, global_step)
writer.add_scalar("losses/q_values", old_val.mean().item(), global_step)
```

信息可视化：

tensorboard --logdir=runs/

结果展示