

基于深度强化学习的 平面虚拟软体手臂模型控制

Control Strategy of
Planar Virtual Soft Manipulator Model
Based on Deep Reinforcement Learning



王高天 PB18020686



BACKGROUND KNOWLEDGE

软机器人的建模 & 控制

软机器人

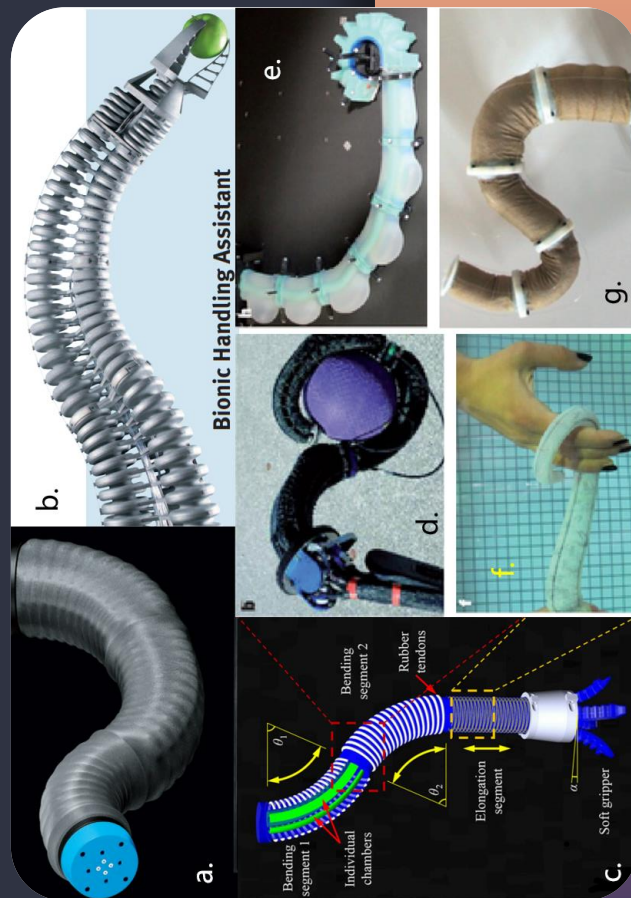
Soft Robotics

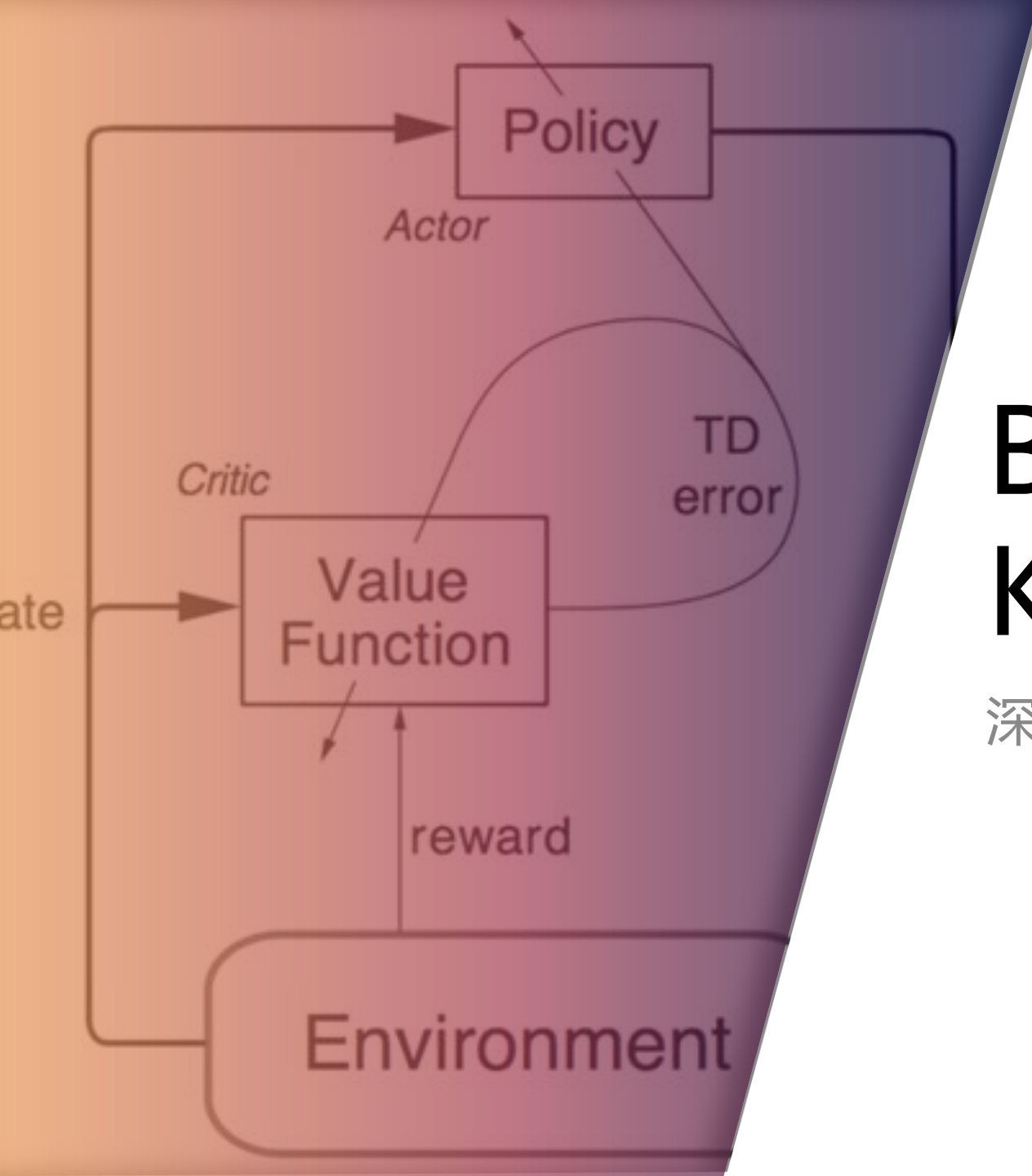
简介

- 完全柔性的形态和运动模式（章鱼手臂、象鼻、软体动物等）
- 适用于在复杂、非结构化的工作环境（不规则物体的抓取、日常生活服务、灾难营救、野外场景）
- 安全性高，人机交互任务的理想执行器
- 传统机器人：利于建模，高度结构化

建模 & 控制

- 难以建模
 - 非线性，外力干扰，结构多样
- 建模
 - 分段定曲率
 - 有限元分析
- 控制
 - 依赖工作空间的闭环反馈系统
 - 简单的模型
 - 学习类方法





BACKGROUND KNOWLEDGE

深度强化学习

强化学习

Reinforcement Learning

机器人上的应用

- 强化学习方法分类

- Policy-Based & Value-Based
- Model-Based & Model-free

- 面临的挑战

- 控制维度高，数据量大
- 现实问题（修理，磨损，人类监督，无法暂停）
- 模型不精确
- 奖励函数不好定义

深度强化学习相关算法

- Q-learning: 离散状态 & 动作空间
- DQN(Deep Q Network):
深度神经网络端到端拟合Q值；不能解决高维动作空间
- Policy Gradient:
动作策略梯度；无法及时得知所有动作值
- Actor-Critic:
结合Policy-Based & Value-Based；复杂问题不稳定
- DDPG(Deep Deterministic Policy Gradient)
经验回放 (Experience Replay) 和 Fix Q target

```

# 总训练次数
# 单次训练的最大步长

# 为True时训练，为False时展示训练结果

# 引入环境
# 状态维数
# 动作维数
# 动作范围

m, s_dim, a_bound)

# 引入强化学习算法

# 开始训练

for i in range(MAX_EPISODES):
    env.reset()
    r = 0.
    for j in range(MAX_EP_STEPS):
        #env.render()
        #上一行开启时是在可视化环境下进行训练，即可看到手臂的训练过程
        a = rl.choose_action(s)      # 选择动作
        s_, r, done = env.step(a)    # 得到下一步的状态，奖励，环境
        rl.store_transition(s, a, r, s_) # 保存策略
        ep_r += r
        if rl.memory_full():         # 当记忆库被填满时开始学习
            rl.learn()
        s = s_
        if done or j == MAX_EP_STEPS-1: # 在控制栏中显示训练次数，是否达标，总奖励
            print('Ep: %i | %s | ep_r: %.1f | step: %i' % (i, '---' if not done else ' ', ep_r, j+1))
            break
    rl.save()

if eval():
    # 开始展示评估表现
    rl.restore()
    # 载入动作策略
    env.render()
    # 打开可视化环境
    env.viewer.set_vsnc(True)
    s = env.reset()
    while True:
        env.render()
        a = rl.choose_action(s)
        s, r, done = env.step(a)

3
3
34 if ON_TRAIN:
35     train()

```

ALGORITHM

算法介绍

ENVIRONMENT

环境设置

项目	内容
操作系统	Windows 10 x86
CPU	Intel(R) Core(TM) i7-8550U@ 1.99GHz
运行内存	8.00GB
语言	Python 3.7.0
深度学习框架	Tensorflow 1.14.0
库	Pyglet; Numpy; Tensorflow

- Open AI gym框架
 - 主循环
 - 虚拟环境
 - 强化学习算法

```

1  #-*- coding: utf-8 -*-
2  """
3  Created on Sat Apr 25 21:56:51 2020
4  @author: Vector Wang
5  """
6  from env import ArmEnv
7  from rl import DDPG
8
9  MAX_EPISODES = 4000          # 总训练次数
10 MAX_EP_STEPS = 200           # 单次训练的最大步长
11 #ON_TRAIN = False
12 ON_TRAIN = True              # 为True时训练，为False时展示训练结果
13
14 env = ArmEnv()                # 引入环境
15 s_dim = env.state_dim         # 状态维数
16 a_dim = env.action_dim        # 动作维数
17 a_bound = env.action_bound    # 动作范围
18
19 rl = DDPG(a_dim, s_dim, a_bound) # 引入强化学习算法
20
21 steps = []
22 def train():                  # 开始训练
23
24     for i in range(MAX_EPISODES):
25         s = env.reset()
26         ep_r = 0.
27         for j in range(MAX_EP_STEPS):
28             #env.render()
29             #上一行开启时是在可视化环境下进行训练，即可看到手臂的训练过程
30             a = rl.choose_action(s) # 选择动作
31             s_, r, done = env.step(a) # 得到下一步的状态，奖励，环境
32             rl.store_transition(s, a, r, s_) # 保存策略
33             ep_r += r
34             if rl.memory_full():
35                 rl.learn() # 当记忆库被填满时开始学习
36             s = s_
37             if done or j == MAX_EP_STEPS-1: # 在控制栏中显示训练次数，是否达标，总奖励和完成时的步数
38                 print('Ep: %i | %s | ep_r: %.1f | step: %i' % (i, '---' if not done else 'done', ep_r, j))
39                 break
40         rl.save()
41
42 def eval():                   # 开始展示评估表现
43     rl.restore()              # 载入动作策略
44     env.render()              # 打开可视化环境
45     env.viewer.set_vsync(True)
46     s = env.reset()
47     while True:
48         env.render()
49         a = rl.choose_action(s)
50         s, r, done = env.step(a)
51
52
53
54 if ON_TRAIN:                  # 判断是训练还是展示评估
55     train()
56 else:
57     eval()

```

MAIN

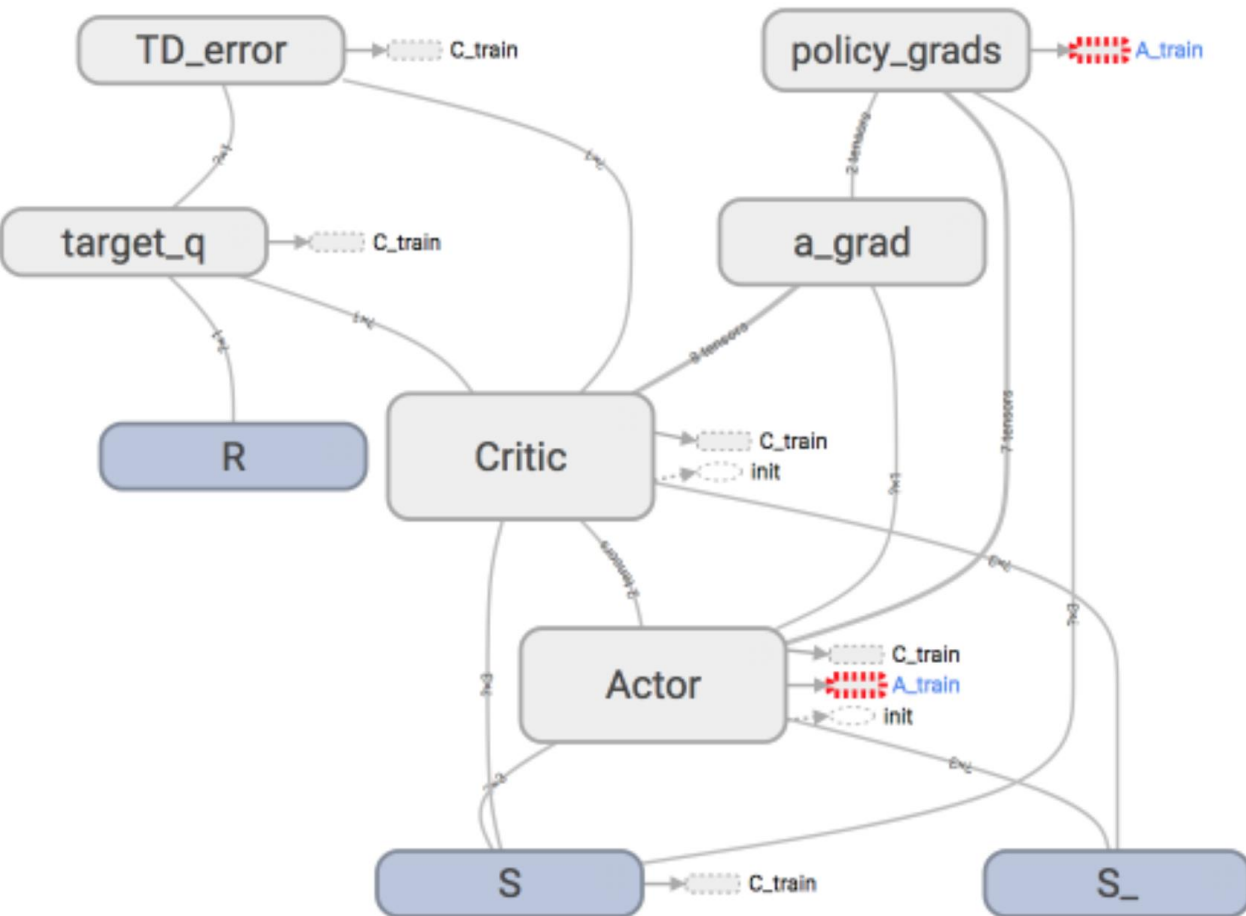
主循环

- 设置总训练次数与单次训练的步长
- 引入虚拟环境和强化学习算法
- ON_TRAIN=True时开始训练
- ON_TRAIN=False时展示结果


```

1 #-*- coding: utf-8 -*-
2 """
3 Created on Sat Apr 25 21:56:54 2020
4
5 @author: Vector Wang
6 """
7 import tensorflow as tf
8 import numpy as np
9
10 # 超参数
11 LR_C = 0.0005

```



```

50 # 计算q_target
51 q_target = self.R + GAMMA * q_
52 # 计算 td_error 误差并反向传递
53 td_error = tf.losses.mean_squared_error(labels=q_target, predictions=q)
54 self.ctrain = tf.train.AdamOptimizer(LR_C).minimize(td_error, var_list=self.ce_params)
55 # 使 q 最大化
56 a_loss = - tf.reduce_mean(q)
57 self.atrain = tf.train.AdamOptimizer(LR_A).minimize(a_loss, var_list=self.ae_params)
58
59 self.sess.run(tf.global_variables_initializer())

```

DDPG

强化学习算法

- Actor: 两个神经网络，梯度下降法学习
 - Eval net: 输出实时动作
 - Target net: 预测Critic的Q target中的动作，更新 Value
- Critic: 两个，算Q target和TD-error并反向传递
 - Eval net: 及时更新参数
 - Target net: 给出更新时的梯度下降强度
- 随机从记忆库抽取n个记忆学习;
- 开始时增强探索，记忆库满后降低探索性

```

1 import numpy as np
2 import pygame
3
4
5 class ArmEnv(object):
6     viewer = None
7     dt = .1 # 手臂更新速率
8     action_bound = [-1, 1] # 手臂动作范围
9     goal = {'x': 100., 'y': 100., 'l': 40} # 目标方块的参数
10    state_dim = 9 # 状态维数
11    action_dim = 2 # 动作维数
12
13    def __init__(self): # 手臂初始化
14        self.arm_info = np.zeros( # 手臂的参数（段数，长度，方位角）
15            2, dtype=[('l', np.float32), ('r', np.float32)])
16        self.arm_info['l'] = 100 # 手臂长度设置
17        self.arm_info['r'] = np.pi/6 # 手臂方位角设置
18        self.on_goal = 0 # 初始化判断为未达目标
19
20    def step(self, action):
21        done = False
22        action = np.clip(action, *self.action_bound) # 定义动作
23        self.arm_info['r'] += action * self.dt # 更新动作
24        self.arm_info['r'] %= np.pi * 2 # 限制角度大小
25        # 限制根部段在上半平面
26        if (0 > self.arm_info['r'][0]) or (self.arm_info['r'][0] > np.pi):
27            self.arm_info['r'][0] = np.pi - (self.arm_info['r'][0] + np.pi) % (np.pi)
28
29        (a1l, a2l) = self.arm_info['l'] # 两根手臂的长度
30        (a1r, a2r) = self.arm_info['r'] # 两根手臂的方位角
31        a1xy = np.array([200., 0.]) # 手臂根部位置
32        a1xy_ = np.array([np.cos(a1r), np.sin(a1r)]) * a1l + a1xy # 第一关节，尖端的位置
33        finger = np.array([np.cos(a1r + a2r), np.sin(a1r + a2r)]) * a2l + a1xy_
34        # 对目标点到第一关节的位移，目标点到尖端的位移进行归一化
35        dist1 = [(self.goal['x'] - a1xy_[0]) / 400, (self.goal['y'] - a1xy_[1]) / 400]
36        dist2 = [(self.goal['x'] - finger[0]) / 400, (self.goal['y'] - finger[1]) / 400]
37        r = -np.sqrt(dist2[0]**2 + dist2[1]**2) # 设置奖励为目标点到尖端距离的负
38
39        # 完成了任务后获得的奖赏
40        if self.goal['x'] - self.goal['l']/2 < finger[0] < self.goal['x'] + self.goal['l']/2:
41            if self.goal['y'] - self.goal['l']/2 < finger[1] < self.goal['y'] + self.goal['l']/2:
42                r += 1.5
43                self.on_goal += 1
44            if self.on_goal > 50: # 任务完成：尖端在目标区域内停留50步
45                done = True
46        else:
47            self.on_goal = 0
48
49        # 得到目前的手臂状态
50        s = np.concatenate((a1xy_/200, finger/200, dist1 + dist2, [1. if self.on_goal else 0.]))
51        return s, r, done

```

ENVIRONMENT

虚拟环境 / 模型

- 分开运行，提高效率

ArmEnv用于计算，Viewer用于可视化（pygame）

- 奖励设置

未达到目标范围：尖端到目标点距离的负值


达到目标范围：在目标内停留n步，奖励

+1.5(n<50)

- 手臂特征：

段长，夹角，关节位置，尖端位置，到目标距离

- 随机运动程序：不用训练即可测试环境



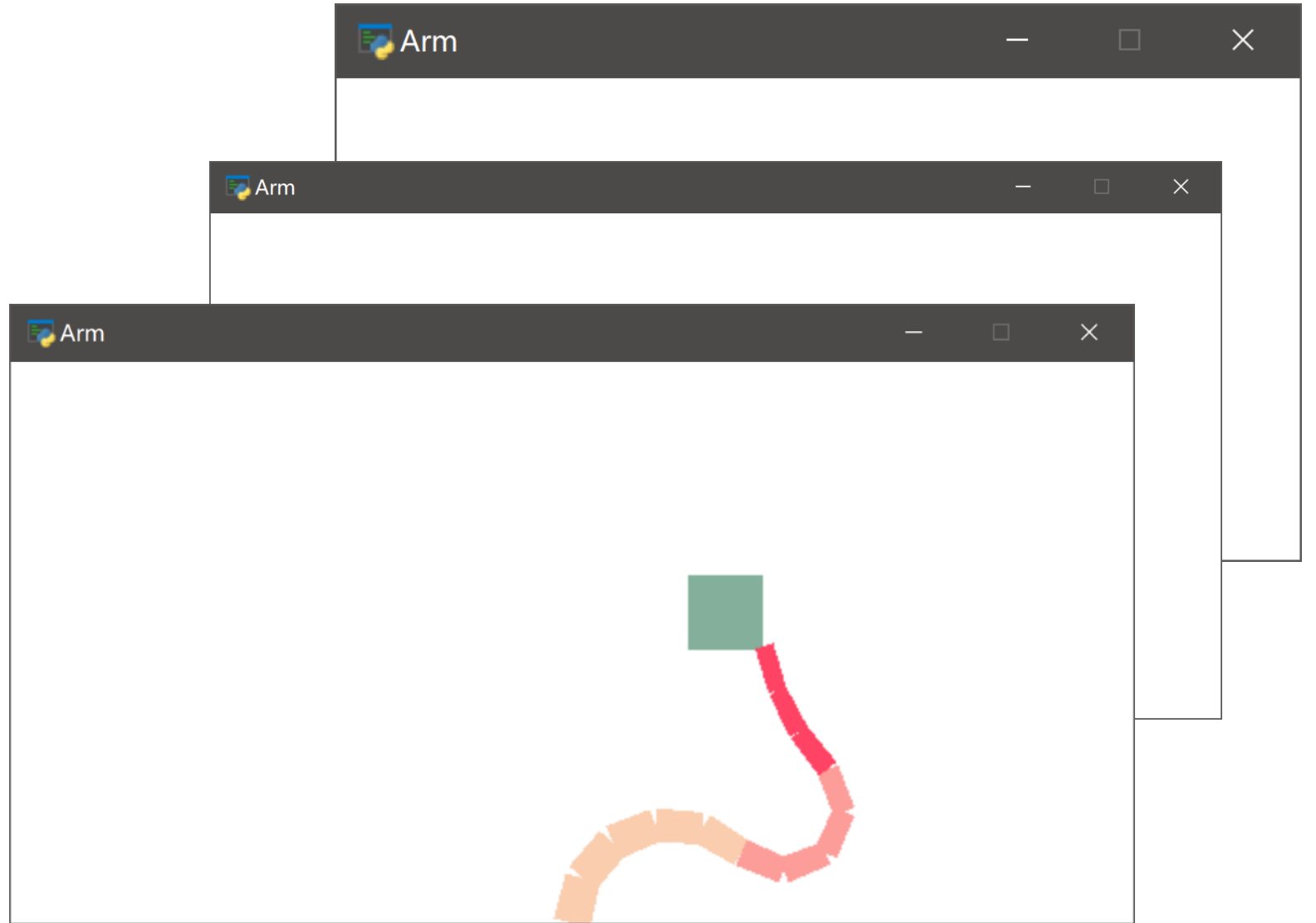
TESTS & RESULTS

测试及结果

MODEL

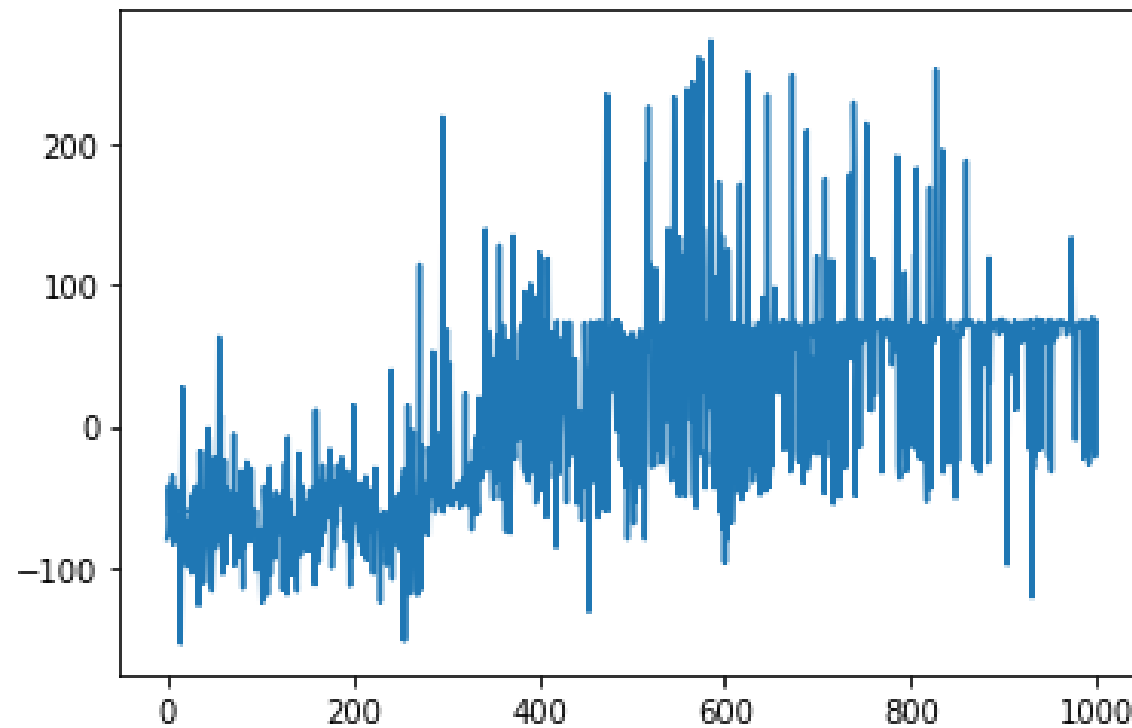
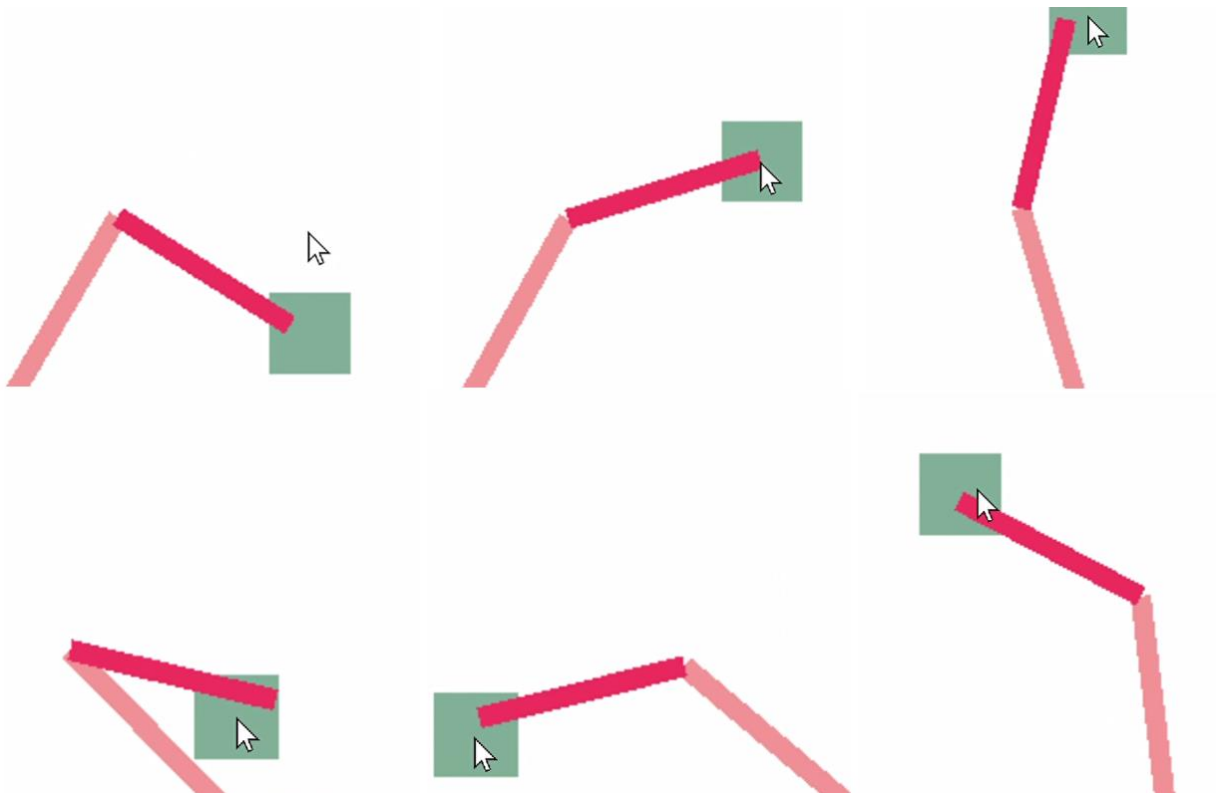
模型展示

- 二自由度手臂
 - 最简单，训练效果最好
- 三自由度手臂
 - 具有冗余自由度
 - 长，中，短
- 软体手臂
 - 分段定曲率模型



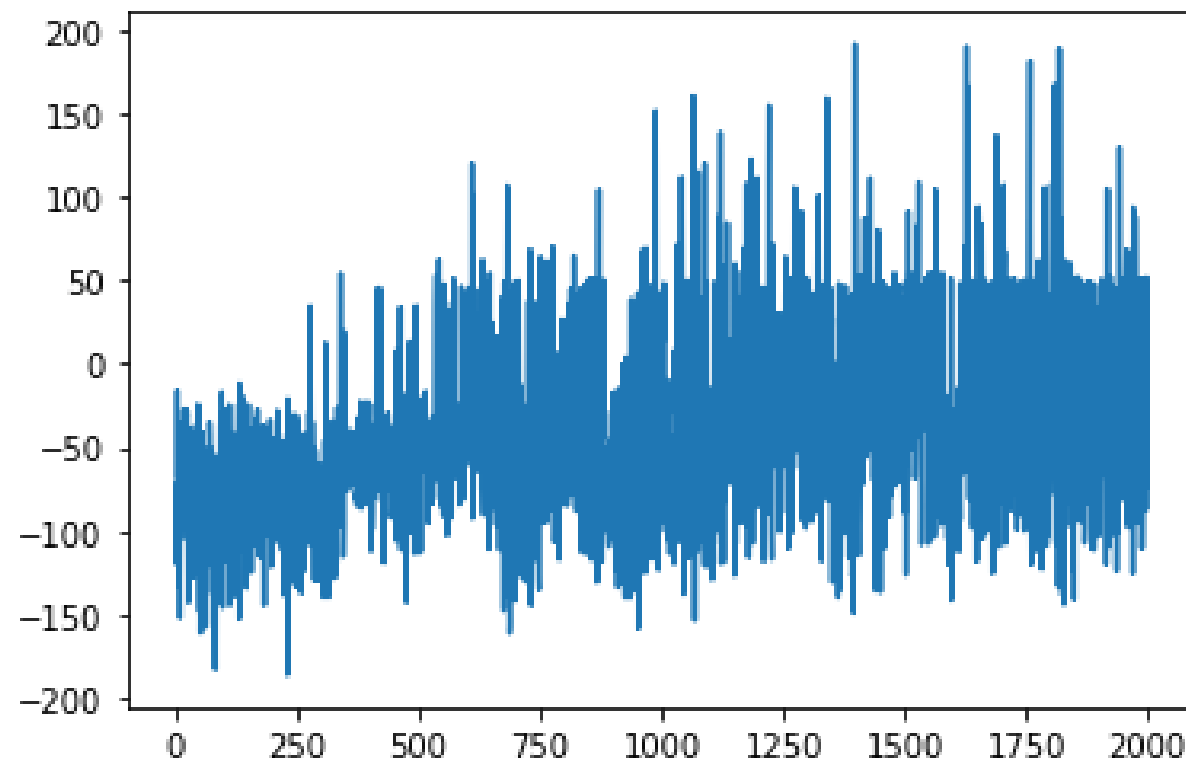
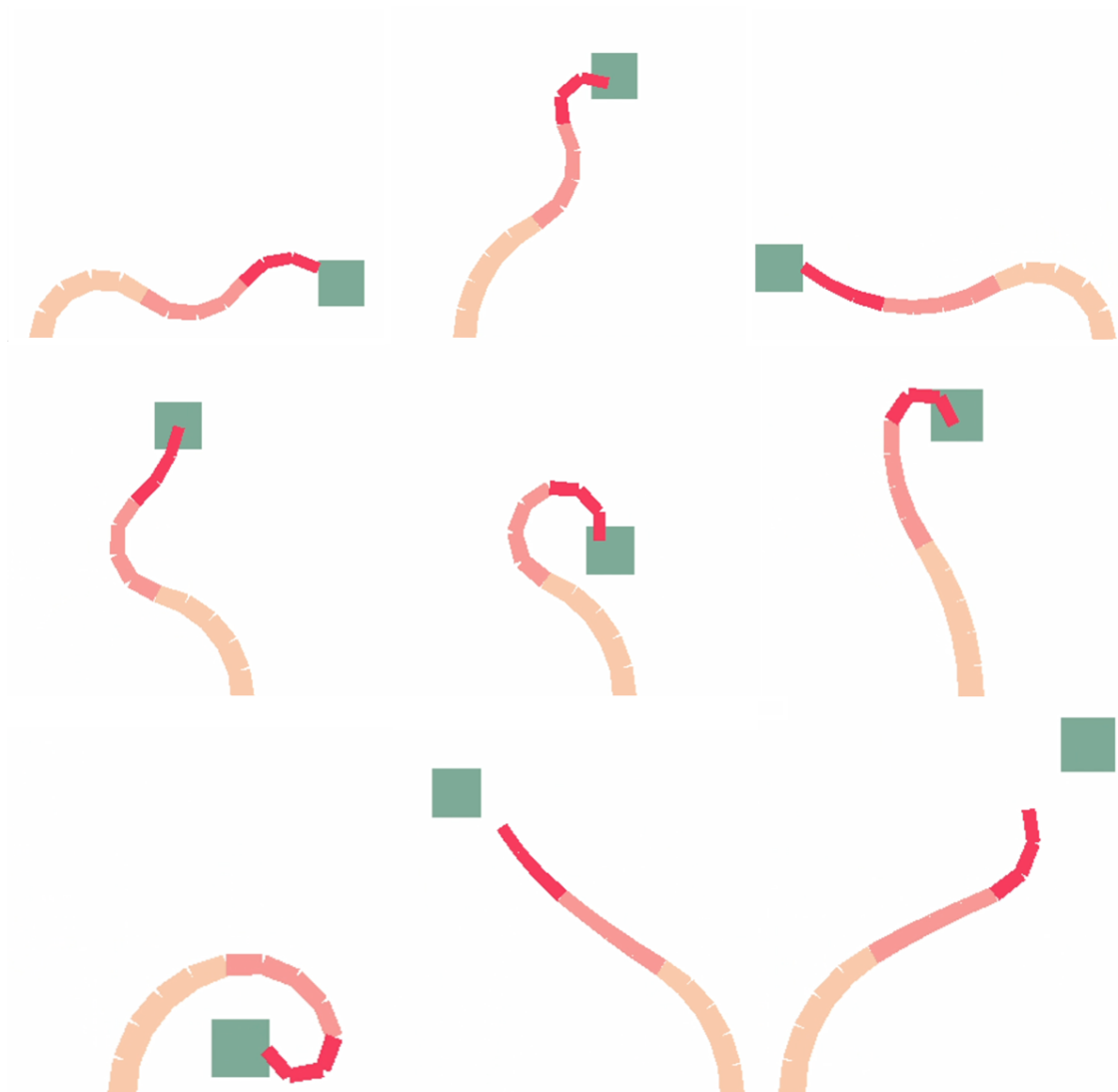
RESULTS

训练结果



RESULTS

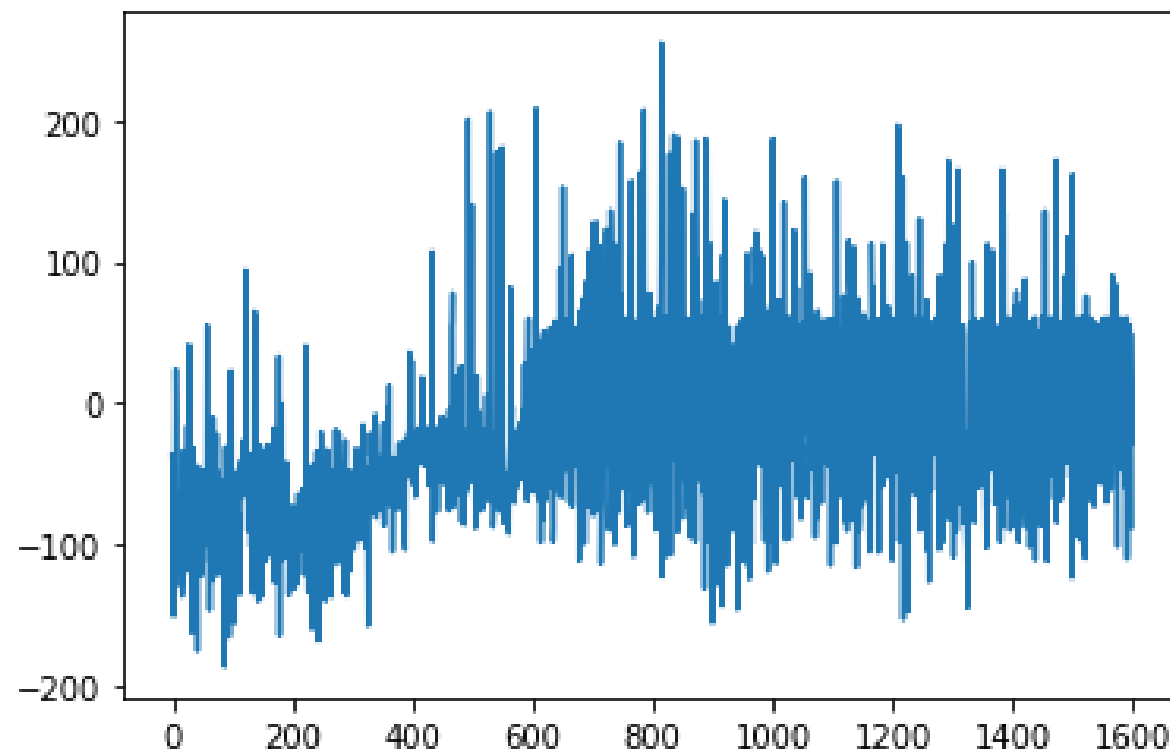
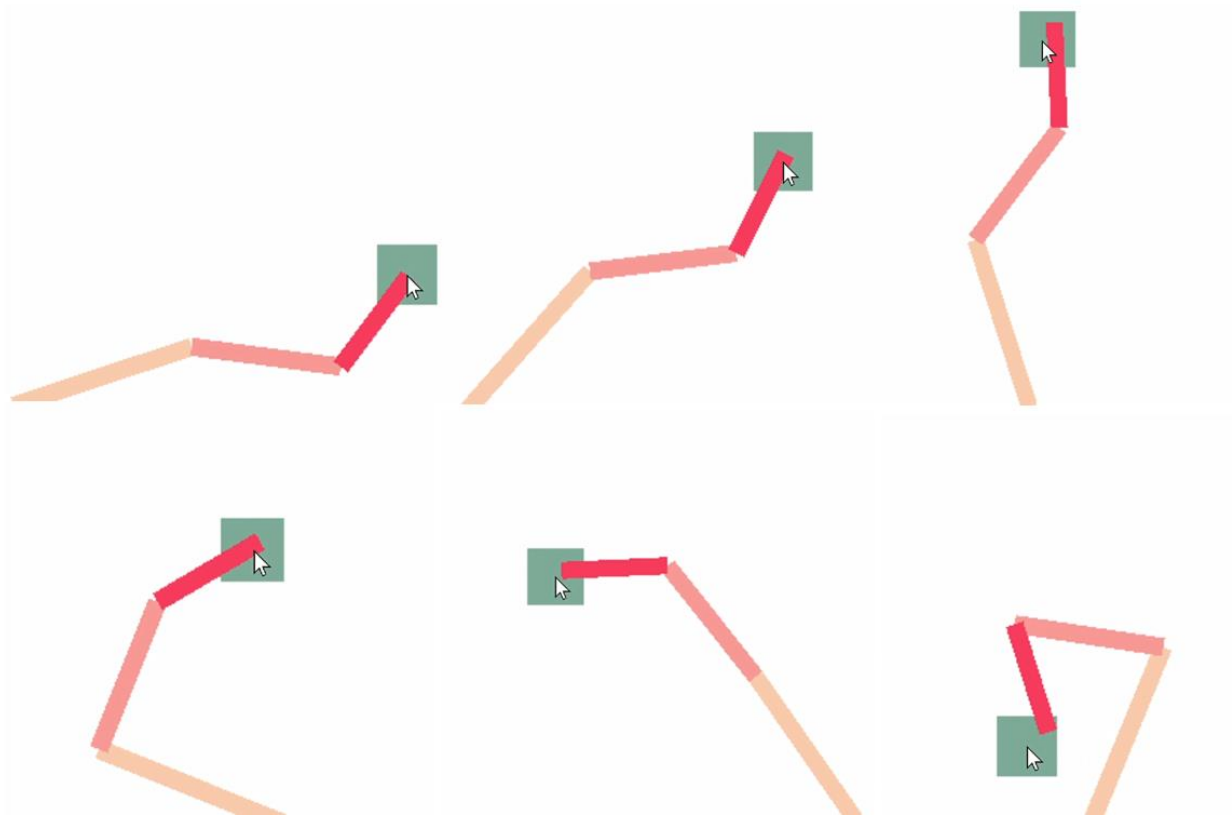
训练结果



RESULTS

训练结果

训练次数对训练效果的非理想化影响



CONCLUSION

总结

成果

基于课上学到的Python知识和自学的深度强化学习相关知识实现了对一种平面软体手臂模型的点到点控制

研究过程

知识学习调研，算法选择，设计模型，建立虚拟环境，
编写DDPG算法，分析评估训练结果并调试

CONTRIBUTION

贡献

- 验证了将目前最前沿的深度强化学习算法之一DDPG作为控制软体手臂的一种途径的可行性
- 提出了一种循序渐进的建模方法，降低了编程难度，减少了出现bug的概率，且方便调试
- 发现并通过仿生学的知识解释了利用强化学习对机械臂进行训练时的模型结构的影响
- 分析了强化学习训练过程中训练次数对训练效果的非理想化影响

LIMITS

限制

- 最简单的定曲率模型，相较于其他描述软体手臂的模型来说过于理想化
- 二维情况下利用DDPG控制软手臂可行并不代表其在三维情况下就能有很好的表现
- 均在虚拟环境中进行，还没在机器人上跑过

THANKS FOR LISTENING!



王高天 PB18020686