仿真环境介绍

——基于数据驱动机械手设计

撰写人：张中 联系方式：15868103331

目录

仿真环境的系统搭建 2

1. 搭建 python 环境 2
2. 安装强化学习模块 2
3. 安装强化学习模块配套的基础环境模块 3
4. 安装物理引擎 3
5. 安装外部的数学库 3
6. 安装 IDE 3

仿真环境的主要函数 4

1. 自动创建机械手模型的函数 4
2. 选择物体以及确定物体大小的函数 7
3. 基于物体位置和形状初始化机械手位置的函数 8
4. 基于物体形状初始化机械手各关节角度的函数 8
5. 抓取规划以及拿到抓取评价的函数 9
6. 评价函数 9

强化学习的框架以及环境中需要定义的函数 10

强化学习模块 spinningup 的使用 11

仿真环境的系统搭建

本环境需要 python3.6+环境下运行，所以需要先搭建一个 python 的环境，并在下面安装物理引擎 pybullet，强化学习模块 spinningup，以及和强化学习模块配套的基础环境模块 gym。为了方便编辑程序，还可以下个编辑程序的 IDE——pycharm。因为本环境里的一些评价函数用了一些外部的数学库，所以需要安装下 cvxpy 和 sympy。安装流程如下（ubuntu 系统下）：

1. 搭建 python 环境

下载 anaconda，创建一个 python3.6 的环境（因为之后东西需要在 python3.6 以上的环境运行，ubuntu 默认的 python 是 2.7 和 3.5）。

conda create -n py3.6 python=3.6 (创建一个 python3.6 的环境，命名为 py3.6)

conda activate py3.6（激活环境 py3.6)

2. 安装强化学习模块

下载 spinningup。

参考网站 https://spinningup.openai.com/en/latest/user/installation.html

sudo apt-get update

sudo apt-get libopenmpi-dev

git clone https://github.com/openai/spinningup.git

cd spinningup

pip install -e .

3. 安装强化学习模块配套的基础环境模块

在 py3.6 的环境下，安装 openai gym(需要外网，不然非常非常慢)。

参考网站：https://gym.openai.com/docs/

pip install gym.或者

git clone https://github.com/openai/gym

cd gym

pip install -e .

4. 安装物理引擎

pybullet 的 tuturial 请参考

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

pip install pybullet

5. 安装外部的数学库

pip install cvxpy

pip install sympy

6. 安装 IDE

为了方便代码编辑和运行，可以额外下个 pycharm。(安装办法请直接百度)

然后将 pycharm 里环境核设置成 anaconda 创建的 py3.6 中的 python 核.

设置流程为 File->settings->Project->Python Interpreter.将其修改为第一步

anaconda 创建的 py3.6 环境名下的 python3.6。

仿真环境的主要函数

我们将整个强化学习的环境定义为一个类

class Self_reconstructed_hand(gym.Env)

在类里我们需要定义环境。

1．自动创建机械手模型的函数

create_hand

（urdfRootPath,hand_state,base_xyz,link_length_list,link_width):

该函数需要输入：

1.urdfRootPath：文件存放地址。

2.handstate：手的具体构成信息 ，其维度为

finger_number*(link_number+joint_number)，其实 link_number 和

joint_number 是一样的。handstate 每一行都代表一根手指的具体构成信息，

每行中，每两个数构成一个 link 和一个 joint 的信息，其中，第一个数代表

link 的长度选择，第二个数表示 joint 的转角选择。

3.base_xyz：代表手掌的尺寸。

4.link_length_list：代表 link 可选长度的 list，假如 len(link_length_list)=10,

表示有 10 种可选长度。

5.link_width：定义每个 link 的粗细。

在该函数中，我们初步定义了手指的生长位置，一个有 10 个生长点，分别为对侧各五个，然后每个 joint 可以有两个方向旋转的选择（弯曲和外展，抛弃了内旋，因为这不是必要的)。每根手指最多长 4 个 link 和 joint。所以 handstate 的维度此时为 10×(4+4) (其实每根手指最多长多少指节由 hand_state 这个输入决定，handstate 的维度应该为 10x(n+n))。其中 handstate[n][index]中，n 为 0-9 的任一值，当 index 为偶数时，可选的数字为 1-10 的整数，代表了 link 的长度。index 为奇数时，可选的数字为 1 或者 2,1 代表该自由度是弯曲，2 代表该自由度是外展。

特别注意：因为 pybullet 导入 urdf 时，如果 urdf 是相同的文件名，实际导入 urdf 只会导入最初始的一次，所以当我们在运行 pybullt.remove(self.urdf_name) 再 pybullet.loadURDF()时候，实际不再次导入 urdf 文件（即使删除 urdf 文件再重新创建)，所以在 create_hand 这个函数中，我们每生成一个新的构型，urdf 文件名的名字都需要修改，一种取巧的办法是在文件名后面加一个随机浮点数。

最后该函数返回四个值：

1.urdf_file_name：生成 urdf 的文件名。

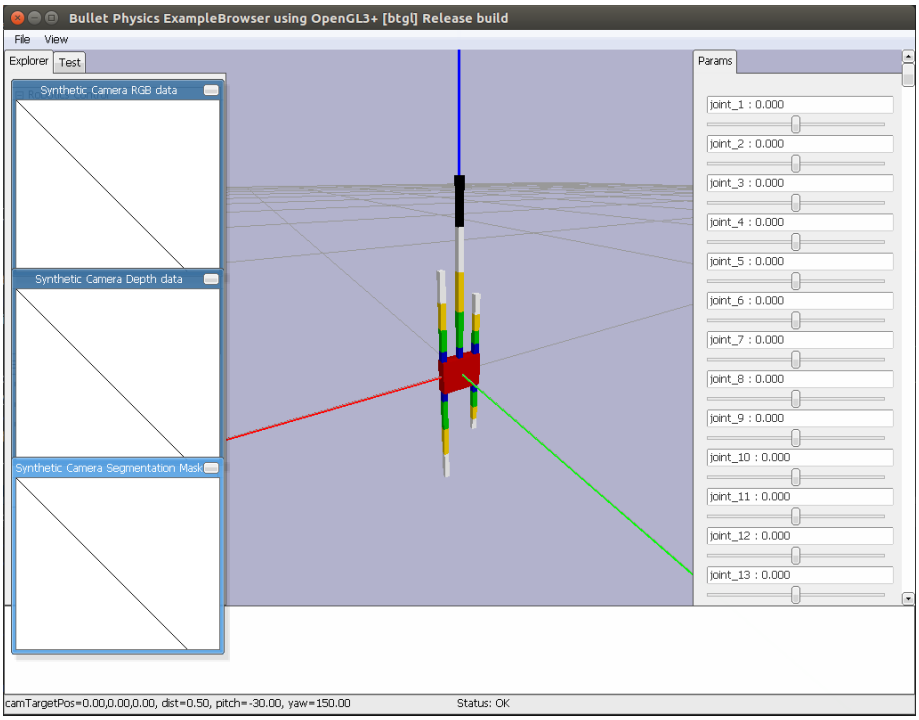
2.finger_abduction_adduction_index:表示生成 joints 中外展自由度的索引

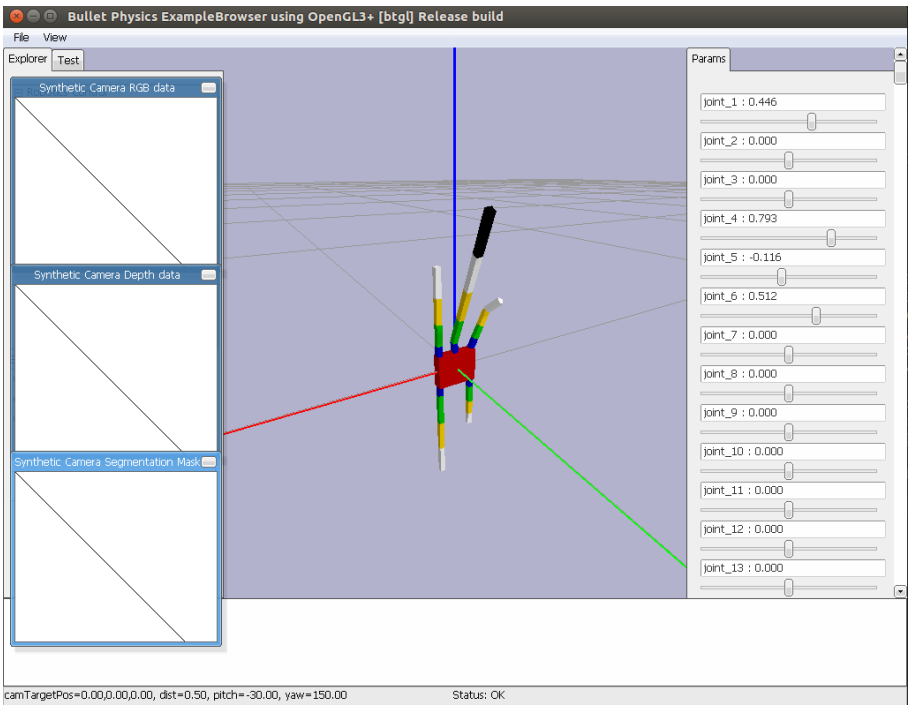
3.finger_abduction_adduction_in_which_side:表示生成 joints 中外展自由度在手掌的哪一侧，如果在上侧面，则为 1,若在下侧面，则为-1。

4.finger_flex_index：表示生成 joints 中弯曲自由度的索引。

for example，图中红线代表 x 轴正方向，绿线代表 y 轴正方向，蓝线代表 z 轴正方向，我定义的手掌正面是面向 y 轴正方向的，在 z 轴正方向上会有 5 个生长点，side 标记为 1;在 z 轴负方向会有 5 个生长点，side 标记为-1:

```
hand_state = [[0 for col in range(self.single_finger_joint_number * 2)] for row in range(self.finger_number)]
hand_state[0] = [1, 2, 2, 1, 3, 1, 4, 1, 0, 0, 0, 0]
hand_state[2] = [1, 2, 10, 1, 10, 1, 10, 1, 10, 1, 0, 0]
hand_state[4] = [1, 2, 5, 1, 6, 1, 7, 1, 0, 0, 0, 0]
hand_state[5] = [1, 2, 4, 1, 3, 1, 2, 1, 0, 0, 0, 0]
hand_state[9] = [1, 2, 10, 1, 10, 1, 10, 1, 0, 0, 0, 0]
print('hand_state is {0}'.format(hand_state))
urdf_file_name = create_hand(self.urdfRootPath,hand_state,[0.06,0.016,0.05],
                             [0.016, 0.020, 0.024, 0.028, 0.032, 0.036, 0.040, 0.044, 0.048, 0.052],0.008)
```





2. 选择物体以及确定物体大小的函数

choose_object(self,object_index,object_width_or_radius,object_height):

该函数输入为：

1.object_index：代表物品标号，0对应长方体，1对应圆柱体，2对应球，3代表圆锥。

2.object_width_or_radius：定义了物体的宽度/半径。

3.object_height：定义了物体的高度。

该函数会根据参数往仿真环境里加载对应信息的物体。

最后返回三个值：

1.self.object_Uid：表示物体在仿真环境里对应的索引。

2.central_width：物体中心到最外边缘的距离。

3.central_height：物体中心的高度。

3. 基于物体位置和形状初始化机械手位置的函数

reset_hand_position_and_orientation(self,object_index,central_width,central_height,theta,offset):

该函数是为了设置手的起始位置，其起始位置是根据不同物体的种类以及不同物体的大小决定的。其中物体中心默认被放在 x=0,y=0的位置。若没有offset，手的起始位置可大概看成是围绕物体中心的位置，贴着物体表面放置，以便于包裹。

theta：表示手相对于物体绕 z 轴的旋转角度。

offest：表示手远离物体的程度（距离）。

该函数不返回数值，直接 reset 了手在仿真环境里的初始位置。

4. 基于物体形状初始化机械手各关节角度的函数

choose_initial_hand_abduction_adduction_joint_degree(self,object_index,finger_abduction_adduction_index,finger_abduction_adduction_in_which_side)

该函数是为了根据不同物体设置手初始位置时候外展自由度的角度，

当 index=0和1时，因为抓的是圆柱和长方体，不需要外展，所以外展对应的弧度全为0。

当 index=2时，大概流程就是从-2到2中随机选个整数，乘上30角度赋值给对应的外展自由度。

该函数返回外展自由度的弧度值，若没有外展自由度，则返回空白 list。

5．抓取规划以及拿到抓取评价的函数

grasp_plannar():

该函数是为了让已经根据物体类别和大小摆好的机械手进行一个包裹抓取动作。抓取主要分为两步，第一步是为了让各机械手指收拢让其碰到物体并停止，第二步是让所有手指进一步包裹去抓牢物体。在进行完抓取动作以后，可以通过接触点是否形成力凸包来判断物体是否抓稳，抓稳的话再计算抓取的评价函数。

6．评价函数

get_force_convex_hull_minimum_distance(self, frictional_coefficient, central_width):

输入摩擦系数和物体的宽度。

该函数用于计算当前物体与机械手接触点形成的凸包大小。

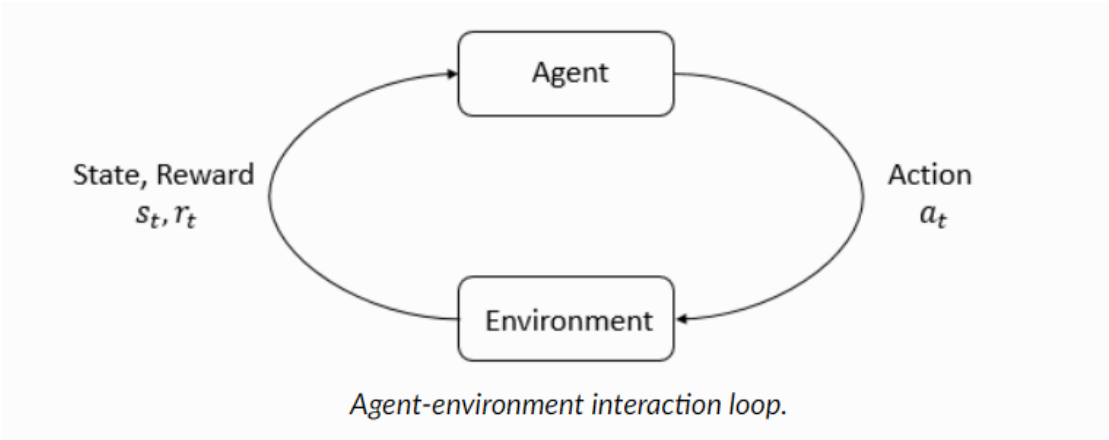
get_robotic_hand_minimum_energy(self, hand_state, frictional_coefficient, W_g, joint_torque_limit):

输入机械手的构型，摩擦系数，物体重力，关节力矩限制

该函数用于计算维持物体抓取状态下机械手所需要的最小能量。

上述两者函数的具体计算都放在 grasp_quality_measurement_functions.py 里，里面的注释已经详细解释了函数的理论基础和计算。

强化学习的框架以及环境中需要定义的函数



Agent-environment interaction loop.

上图的程序语言为：

o, r, d, _ = env.reset()

o, r, d, _ = env.step(a)

其中 o 为上图中的 State（一般也叫做 observation），r 为上图中的 Reward，a 为上图中的 Action。我们上述第二部分讲的仿真环境即为 env，也就是上图中的 Environment。

step 函数定义了环境是如何接收 action 并作出相应反馈的。

reset 函数进行环境的初始化。

强化学习模块 spinningup 的使用

请参考 spinningup 的 tutorial：https://spinningup.openai.com/en/latest/

要想使上述的环境能在 spinningup 中的强化模块中被正常使用，环境需要继承下 openai gym 定义的环境类。另外需要将环境重命名为规定的格式，重命名环境参考教程如下：

https://github.com/openai/gym/blob/master/docs/creating-environments.md