Lab 4

目录

介绍

在这个项目中，你将使用numpy实现智能机器人和仿真专题中学到的各种算法，帮助pacman走出迷宫。

代码库发生了大幅改变，请从教学网上下载lab4。

本项目包括一个自动评分器，供你在机器上对答案进行评分。

用以下指令可以对所有题目进行评测。

```
python autograder.py
```

环境配置

对于Mac/Linux和安装了Visual Studio的Windows环境

```
pip install numpy scipy panda3d
pip install swig box2d-py
```

如果你的Windows环境没有Visual Studio，我们提供了python3.8/3.9/3.10/3.11两个版本编译好的Box2D库，你只需安装numpy scipy panda3d就可以运行作业文件。其他的python版本需要你先安装Visual Studio后按以上的命令安装。

注意：除去Windows上以上四个版本的python可以使用我们提供的软件包以外，其他环境需要把文件夹中的 `Box2D` 文件夹删除，然后自己pip安装box2d-py（Windows需要先安装Visual Studio，并注意在安装时要勾选"使用C++的桌面开发"

如果下载速度过慢可以使用清华源安装 (https://mirrors.tuna.tsinghua.edu.cn/help/pypi/)

请不要使用其他的库，以免影响评分。

你需要修改的文件:

info.yaml 姓名学号

`answerLocalization.py` 定位任务中需要你实现的部分

`answerPDControl.py` 控制任务中需要你实现的部分

`answerPlanning.py` 规划任务中需要你实现的部分

你可能需要阅读的文件:

`taskLocalization.py` 蒙特卡洛定位

`taskPDControl.py` PD控制器

`taskPlanning.py` 运动规划

`physicsWrapper.py` 获取仿真环境中的Pacman位置和速度信息

`util.py` 实现算法时的可能会用到的数据结构。

你可以忽略的文件:

`visualizer.py` 3维显示窗口

`loadMap.py` 导入地图的程序

`simuScene.py` 物理仿真后端

`autograder.py` 自动评分器

需要修改并提交的文件: 提交所有需要修改的文件。你可以使用以下指令来打包答案文件。

```
python compressAnswers.py
```

评估: 我们使用`autograder.py`对你的提交进行评分，测试用例与本地给出的样例文件相同。你可以在本地运行评分器对你的代码进行评测，分数即为最终分数，注意完整测试的时间可能比较久。

学术诚信: 我们会将你的代码与课堂上其他提交的代码进行逻辑查重。如果你拷贝了别人的代码，并做一些微小的修改，我们会很容易发现，请不要尝试。我们相信你们会独立完成作业。

新的pacman规则

与之前的lab相比，lab4的pacman改成了连续地图。你需要规划路径，通过PD控制器来控制pacman在有限步内吃掉地图上的全部食物。

由于改为连续地图，地图文件的格式也发生了变化，一个典型的地图文件如下

```
%%%%%%%%%%
%        P %
%.%%%%%%%%
%.       ..%
%%%%%%%%%%
```

其中P代表pacman的起始位置，`%`代表以该位置为中心有边长为1的正方形墙壁。其中pacman为半径为0.25的圆盘，`.`为代表食物，没有质量和体积，只要pacman与食物有接触即视为吃掉食物。

所有用于评分的地图已经公布。测试地图中都没有鬼，且最外圈全部是障碍物。

Question 1 (8 points): 机器人定位

已知迷宫地图，给定一些机器人的运动轨迹，使用第21节课上的内容实现蒙特卡洛定位。

阅读`taskLocalization.py`，并按照要求在`answerLocalization.py`中实现对应要求。

需要你实现的部分有：

1. generate_uniform_particles 在地图上的空地均匀采样，生成初始粒子

2. calculate_particle_weight 根据Pacman实际接收到的激光雷达信号和你估计的粒子位置接收到的激光雷达信号，计算该粒子的重采样权重，具体的计算公式为
$w = exp(-k \cdot norm(\hat{x}_i - x_i))$
其中x_i和$\hat{x}_i$分别是16线激光雷达传感器在估计位置和真实位置的接收数据，`norm`可以直接使用L2距离，系数k可以由你来调整。每个粒子按照上述公式计算权重后还需要进行归一化，这一步不需要你来实现。

3. resample_particles 根据归一化后的重采样权重对粒子进行重采样（在原来粒子的位置和朝向上加高斯噪声得到新粒子），采样出来的粒子要符合上一步计算出来的权重。注意在原有粒子上加的高斯噪声是必需的，通过不断地加噪和重采样筛选使估计的机器人姿态更加精确。可以选择在采样时直接加噪，然后下一步仅更新粒子姿态，也可以这里采样得到多个重复的粒子，然后下一步更新姿态添加噪声。高斯噪声的方差可以由你调整。

   可能的重采样方式是：将粒子总数乘以每个粒子的权重，并在每个粒子周围采样该数量的点，剩余的位置用均匀采样补全粒子总数。也可以用轮盘赌的方式，先按权重从大到小排序，求权重的前缀和，每次在0-1中均匀采样一个值，在前缀和数组中找到对应位置，该位置就对应着这次采样出来的粒子，这样的方式采样出来的分布就是原来的粒子权重分布。

4. apply_state_transition 实现粒子姿态的移动，框架会提供每一步机器人真实的移动距离和朝向的改变值，你需要仿照给出的粒子方式更新粒子的位置和朝向，先更新朝向，并在新朝向上按照给定距离移动。

5. get_estimate_result 移动结束，根据当前估计的众多粒子得到最终结果，当然你也可以直接选权重最大的

实现完成后运行

```
python taskLocalization.py
```

来可视化你的实现结果，可以修改`load_scene_and_run_gen`函数中的`data_num`来可视化你的算法在不同测试样例上的表现。

运行

```
python autograder.py --q q1
```

来评分。评分规则为你的定位算法计算出来的机器人姿态与答案姿态进行比较，按照差异大小进行评分。

注意autograder可能会报错`python: command not found`，此时请修改autograder.py开头的PYTHON_PATH为你的python程序的绝对位置（Linux和苹果用户可以在shell中用`which python`，windows用户可以在powershell中用`gcm python`

Question 2 (4 points): 运动控制

实现一个PD控制器，在无障碍地图上控制pacman在到达食物的位置。
阅读`taskPDControl.py`，并按照要求在`answerPDControl.py`中实现`calc_pd_force`函数，注意要使用函数提供的`kp`和`kp`，以免影响评分，这两个参数不需要你修改。

实现完成后运行

```
python taskPDControl.py
```

来可视化你的实现结果，可以修改`task_pd_control`函数的`food_idx`参数来可视化你的算法在不同测试样例上的表现。

运行

```
python autograder.py --q q2
```

来评分。评分器会根据在指定时间你控制的Pacman是否到达正确位置来判断你的实现是否正确。

Question 3 (8 points): 运动规划

给定一个有多个食物的pacman地图，使用part2中实现的PD控制器，控制pacman将地图上的食物尽可能全部吃掉，吃掉食物的顺序由框架给出。

阅读`taskPlanning.py`，你需要实现的部分在`answerPlanning.py`

`answerPlanning.py`提供了快速探索随机树的大致框架，提供了一个`PlanningMap`对象用于检查与障碍物的交互，你可以在`simuScene.py`中查看用法。
你主要需要实现以下部分：

1. find_path函数：每次吃掉一个食物后会调用这个函数，提供下一个食物的位置，你需要规划出一条由当前位置前往下一个食物路径，一个由路径上的点组成的列表。
2. get_target函数：每一个仿真时间步都会调用一次，计算该步的PD target，框架之后会用你计算出的PD target调用task2代码计算这一步施加到Pacman质心上的力，来驱动Pacman运动。

除以上两个函数的接口之外，`answerPlanning.py`中的其他部分你都可以自由修改，这里需要注意的是，由于我们在仿真环境中驱动角色，Pacman不一定能准确到达你指定的位置，在一定情况下可能需要重新规划路径。

实现完成后运行

```
python taskPlanning.py
```

来可视化你的实现结果，可以修改`main`中的`seq_file`来可视化你的算法在不同测试样例上的表现。

运行

```
python autograder.py --q q3
```

来评分。评分器将根据你驱动Pacman将地图上食物全部吃完所需要的时间步进行打分。

Acknowledgement

Designed and implemented by Yulong Zhang and Libin Liu from the MoCCA Lab. Thanks all professors and TAs of the 2023 Introduction to AI team for the valuable discussions.