机器学习纳米学位开题报告

项目名称：训练四轴飞行器学会飞行

项目背景

四轴飞行器或四轴飞行器无人机在个人和专业应用领域都变得越来越热门。它易于操控，并广泛应用于各个领域。大多数四轴飞行器有四个提供推力的发动机，通过接受一个推力大小和俯仰/偏航/翻滚控件简化飞行控件。传统的方法是通过诸如 PID 控制器的外在媒介手动控制，但如果使用机器学习算法，让四轴飞行器自己学会飞行，则是一件非常有趣也有价值的研究。就像自动驾驶一样，具备自主巡航能力的无人机将会极大的提升工作效率，也会扩展出更多的应用领域。这个项目将采用强化学习算法，在一个模拟环境中训练四轴飞行器学会自主飞行。采用强化学习，是因为它非常适合这个任务，如果把该任务进行抽象，会发现这是一个典型的连续空间中的强化学习问题。

本项目采用深度强化学习的算法设计模型，深度强化学习是传统强化学习算法在连续空间的扩展，该领域的里程碑是 DeepMind 发表在 nature 的文章 Human-level control through deep reinforcement learning（https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf）。这片文章提出的 DQN 算法结合了深度学习，在视频游戏类任务中表现出色。但由于飞行器的动作空间也是连续的，这个任务并不适合采用 DQN 算法（DQN 只适用于离散动作空间）。DQN 本质上是 value-based 算法，主要考虑值函数的更新，智能体在交互过程中需要通过诸如 Epsilon 贪婪算法的手段间接的选择动作输出。后来又发展出了更加直接的 policy-based 方法，典型的就是策略梯度算法，这类算法不考虑值函数，直接输出策略函数，代表性的有随机策略梯度 Policy Gradient Methods for Reinforcement Learning with Function Approximation(http://webdocs.cs.ualberta.ca/~sutton/papers/SMSM-NIPS99.pdf)，确定性策略梯度 Deterministic Policy Gradient Algorithms（DPG）(http://link.zhihu.com/?target=http%3A//jmlr.org/proceedings/papers/v32/silver14.pdf)。在此基础上，又发展出了结合 value-based 和 policy-based 的方法，行动者-评论者（actor-critic），这种算法有两个模型组成，policy 网络是 actor（行动者），输出动作。value 网络是 critic（评价者），用来评价 actor 网络所选动作的好坏。

目前最具代表性的 actor-critic 方法是 google 提出的 DDPG（深度确定性策略梯度）算法 CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING(http://link.zhihu.com/?target=https%3A//arxiv.org/pdf/1509.02971v2.pdf)。DDPG 中，actor 网络的输入是 state，输出 action，以 DNN 进行函数拟合，对于连续动作 NN 输出层可以用 tanh 或 sigmod，离散动作以 softmax 作为输出层则达到概率输出的效果。critic 网络输入为 state 和 action，输出为 Q 值。

问题描述

在该项目中，需要通过 ROS 平台控制一架模拟四轴飞行器，并使用强化学习算法训练四轴飞行器完成四个任务：起飞(Takeoff)，悬停(Hover)，降落(Landing)，和整套动作(Combine)。我们可以把飞行器看作智能体，飞行器在模拟器中飞行就是智能体跟环境的交互，而飞行器的动作包括 3 个方向的线性推力和 3 个方向的扭矩，而这里关键的奖励则需要根据任务的要求去设计。因此，这是一个典型的连续空间中的强化学习问题，而且动作空间也是连续的，具体来说，状态向量有 7 维，而动作向量有 6 维。

第一个起飞任务已经设置好。后面的任务都需要先进行定义，设计一个合适的方案，包括奖励机制和目标，状态表示和奖励函数等。对每个任务，都需要选择合适的强化学习算法，然后定义合适的智能体类完成它们。每个任务都要记录下阶段累积奖励，以评估智能体的学习性能。

数据和输入

该项目的输入是四轴飞行器的 7 维状态向量，包括三个位置向量(pose.position.x, pose.position.y, pose.position.z)，表示飞行器在三维空间中的坐标，和四个方向向量（pose.orientation.x, pose.orientation.y, pose.orientation.z,pose.orientation.w），分别表示沿 x,y,z 坐标的飞行方向和旋转方向。在具体的任务中，这些状态向量并不是都需要的，比如起飞任务就只需要考虑位置，忽略方向，在设计奖励时，主要是根据飞行高度来判断是否起飞，因此奖励函数只需要 pose.position.z 就可以；对于悬停任务，也是主要考虑高度，但由于要保证飞行器位置的稳定（不能飞出边界），也需要用到 x,y 坐标，为了更有效的控制飞行姿态，方向向量可能也是需要考虑的；而降落任务，飞行方向很重要，要保证飞行器向下方飞行，因此着重考虑 pose.orientation.z 和 pose.position.z，其他的几个向量对于考察飞行姿态可能也是需要的。因此在不同的任务中，根据具体情况限制状态空间，选择不同的维度子集，对比那种效果最好。

解决方案

这个项目的完成可以分为两个部分：任务的定义和智能体的编写。

第一个部分，创建任务。takeoff 已经事先定义好，其他三个任务都可参照 takeoff 的模板进行修改，Hover 任务：在 takeoff 的基础上更改奖励函数，把智能体在指定高度停留的时间作为主要奖励标准，延长阶段持续时间（起飞时间加上预期悬停时间）。Landing：初始条件设置为从指定高度开始，为了实现缓慢降落，我们需要定义智能体的瞬时下降速度，并通过奖励函数每隔一小段时间检查智能体的状态，确保它在规定的时间内下降速度维持在合理的范围，并且水平方向的平移不会过大。Combine：建立一个端对端任务，结合 takeoff，Hover，Landing，设置分段奖励函数，并且考察飞行器动作的连贯性，确保它在切换动作时不会出现位置和速度的突变。

第二部分是本项目的核心内容，需要选择合适的算法，实现智能体类完成四个任务，智能体和任务的关联是通过任务类在每个时间步调用智能体的 step()方法完成。我在这里选择的方案是为四个任务分别创建单独的智能体类，但使用一

个核心算法作为模板，针对不同的任务进行适当的更改，如使用不同的状态表示，调整超参数等。选择的算法是 DDPG（深度确定性策略梯度），采用行动者-评论者模型，分别定义两个 MLP 模型作为 Actor 和 Critic，并创建一个缓冲区存储学习经验。针对不同的任务，要对智能体进行相应调整。比如在 Takeoff 任务中，可以对状态空间和动作空间进行限制，为了成功起飞，只需要用到位置而可以忽略方向，且可以只考虑线性推力而忽略扭矩。但对于 hover 任务，为了保证飞行器在指定高度维持位置不变，需要考虑它的方向。而在降落任务中，我们需要加入扭矩来产生合适的偏角。另外，每种任务需要的参数也有所不同。

基准模型

这里选择项目自带的随机策略搜索（RandomSearch）模型作为基准模型，与我自己定义的 DDPG 模型做对比。RandomSearch 模型是一个非常简单的随机模型，在每个时间步，智能体都会随机选择动作，因此选择每种动作组合的概率都是均等的，这就导致智能体无法进行学习，不能从环境反馈的奖励中学到经验，也就无法提升性能。在前期我已经试验过使用 RandomSearch 训练智能体进行 takeoff 任务，效果十分糟糕，飞行器会毫无章法的翻滚旋转，无法起飞，记录下累积奖励后发现，阶段奖励值一直维持在很低的水平，无法提升。用这个模型跟我的 DDPG 模型对比，可以清楚的反应出 DDPG 模型的优势。如果 DDPG 模型训练的智能体可以很好的完成任务，并获得上升的奖励曲线，就足以说明改模型的成功。

评估标准

该项目的评估标准主要是考察飞行器对每个任务的完成度：第一个任务—起飞，要求使飞行器平稳起飞，达到目标高度（地面以上 10 个单元）；第二个任务—悬停，要求飞行器在起飞后在指定高度悬停一段时间；第三个任务—降落，要求飞行器从指定高度平稳且缓慢的着落；最后，要设计一个端对端任务，完成连贯的从起飞，到悬停，再到降落三个动作。

项目设计

1. 项目工作流程

1) 开发环境配置

首先安装 ROS 虚拟机，下载项目代码，安装需要的环境包，然后下载四轴飞行器的模拟器软件，将模拟器连接至虚拟机。

2) 熟悉控制工作流

这个项目的配置和运行比较复杂，环境配置好后，需要在虚拟机中编写代码，通过 ROS 命令控制主机中的模拟器，观察模拟器中的智能体是否能完成各项任务的预期目标，然后记录下阶段累计奖励。项目中提供了一些示例代码，定义了一个随机策略搜索的智能体，它的性能十分糟糕，我们需要以它为模板，定义新的智能体以完成任务。

3) 任务一：起飞

使用 DDPG 算法，编写新智能体，运行预先定义好的 takeoff，实现平稳起飞，高度达到地面 10 个单位以上。达成目标后，用 txt 文件记录下阶段奖励数据。

4) 任务二：悬停

创建新任务 hover，以 takeoff 为模板，重新定义奖励函数，主要考察智能体在指定高度停留（水平位置的变化限制在一定范围内）的时间。在原有智能体的基础上做些调整，比如改变状态表示，修改超参数等。

5) 任务三：降落

创建新任务 landing，重设初始条件，让智能体从指定高度开始阶段，奖励函数要考察智能体下降的速度，这里不仅要衡量智能体降落的时间，还要每隔一段时间检查飞机的位置，以确保“平稳”着落。

6) 任务四：结合动作

最后一个任务，需要找到合适的方案完成从起飞到悬停再到降落的一系列组合动作。这里打算尝试两种不同的方式，分开训练三个智能体分别完成三项任务，或训练一个智能体完整一整套动作，最后比较两种方案的性能。完成整套连贯动作后，同样要记录下阶段奖励数据。

7) 数据可视化，分析讨论

把每项任务的奖励数据绘制成可视化图表，观察累积奖励曲线的变化趋势，比如是否循序渐进，或者有无剧烈突变的部分，如果曲线出现异常（比如短暂下降或起伏过大），分析可能的原因。根据任务的完成情况和奖励曲线评价每个任务中智能体的性能，这里可以把最后 10 个阶段的奖励平均值作为量化指标。

8) 评价性能，提出更合理的方案

在完成任务目标的基础上，思考如何提升智能体的性能。

这里打算从两个方面进行尝试：参数调整和改变算法。

可以先维持原有算法，对模型结构做出一些大的调整，可以更新 Actor 和 Critic 模型，改变网络结构，采用不同的优化器等，或者改变一些参数。也可以尝试改变状态表示，比如在起飞任务中考虑方向和扭矩等。

如果在原有算法的基础上改进提升不明显，也可以尝试不同的算法。比如尝试更简单的模型：首先策略梯度 PRPO，PPO，由于本项目的动作空间是连续的，理论上并不适合采用 value-based 的算法，但也可以尝试，比如采用 DQN 模型，用于对比策略梯度算法和行动者-评论者方法。

2. 算法分析

本项目计划采用的主要算法是 DDPG（深度确定性策略梯度），DDPG 基于行动者-评论者方法，结合了 value-based 方法和 policy-based 方法。非常适合连续状态空间和连续动作空间的强化学习，所以符合本项目的需求。

DDPG 的主要结构包含两个网络，一个策略网络（Actor），一个价值网络（Critic）。策略网络输出动作，价值网络评判动作。两者都有自己的更新信息。策略网络通过梯度计算公式进行更新，而价值网络根据目标值进行更新。同时，DDPG 采用了 DQN 的成功经验，即采用了样本池和固定目标值网络这两项技术。算法把智能体和环境的交互分成两部分，训练和学习，在训练阶段，智能体不断跟环境交互并获得反馈，使用一个缓存区存储学习经验；在学习阶段，批量从缓存区中取样，更新模型参数。

具体来说，算法大致流程如下：每个时间步，Actor 选择当前状态下采取的动作，并把相应的经验 (s,a,s,r) 放入缓存；每隔一段时间，当缓存中有足够的经验时，从中批量取出样本用于训练 Critic 模型；Critic 模型负责生成策略梯度，用以训练 Actor 模型，更新参数。这里还有一个关键的地方是设置两套 Actor 和 Critic 模型的副本，一个本地副本，一个目标副本，拆分被更新的参数和生成目标值的参数。目标副本作为“固定 Q 目标”，本地副本用于实时更新和输出。在用多个经验进行训练后，我们可以将新学习的权重（来自本地模型）复制到目标模型中，进行软更新。最后还需要加入一个噪点模型，这里采用 Ornstein–Uhlenbeck 噪点，赋予智能体一定的随机探索能力。