

摘要

无人机系统目前在日常生活的众多场景中都得到了十分广泛的应用，而自主导航和目标追踪是无人机领域中十分重要的两项技术。传统算法实现无人机自主导航任务主要依赖雷达传感器和 SLAM 模块输出的高精度地图信息，但是雷达设备通常价格较为昂贵，不适合小型无人机系统携带，并且高精度地图的建立过程往往过于耗时；而在目标追踪任务中，普通的视觉检测算法只能得到目标物体的二维坐标信息，丢失的深度信息需要用额外方法进行估计，但这种传统算法的实现流程准确性较低且会消耗较多计算资源。因此为了解决以上问题，本文基于深度学习和强化学习的方法提出了一种端到端实现、无需地图先验的无人机自主导航算法以及基于 3D 目标检测算法的无人机目标追踪系统，本文的主要工作如下：

- (1) 本文以 SAC 算法为基础，额外设计了课程学习过程以及自注意力网络层，提出了改进的 CLSA-SAC 算法，该算法可以更加有效地学习到导航与避障行为，解决样本效率问题，进而完成较为复杂的自主导航任务。
- (2) 为实现目标追踪任务，无人机需要利用视觉传感器实时解算物体的位置信息，因此本文提出了一种基于 CenterNet 的 3D 目标检测网络 3D-CenterNet，为其设计了新的 3D 参数回归层和特征融合层，并在训练过程中引入了多参数解耦损失，有效增加了训练稳定性，提高了 3D 目标检测模型的性能表现。
- (3) 本文基于 Airsim 与 UE4 平台，设计并搭建了高像素与高性能的仿真环境，用于训练和测试无人机自主导航与目标追踪任务。该仿真环境支持域随机化等操作，通过域随机化可以有效减少模型在测试时出现的域漂移现象，增强模型训练的稳定性。同时仿真环境还封装了 gym 接口，可以方便地实现与环境的交互操作。最终，结合上述已经实现的自主导航与 3D 目标检测算法，本文在仿真环境中实现了无人机目标追踪系统，相比于传统实现方式本文所设计的方法有着更快的执行速度，并大大减少了计算复杂度，降低了算力需求。

关键词：深度强化学习；无人机自主导航；3D 目标检测；无人机目标追踪

Abstract

Recently, the UAV systems are widely used in many scenarios in daily life. In the field of UAV, the autonomous navigation and target tracking are two significant technologies. The traditional algorithm to realize the autonomous navigation mostly relies on the radar sensor and the high-precision map information outputted by the SLAM module, but the radar is expensive and the establishment of high-precision map is too time-consuming. As to the target tracking task, the ordinary visual detection algorithm can only obtain the two-dimensional coordinate information of the target, and the lost depth information needs to be estimated by additional methods. This traditional pipeline leads to low accuracy and consumes more computing resources. In order to solve the above problems, this paper proposes an end-to-end and mapless UAV autonomous navigation algorithm based on deep reinforcement learning, and implements a UAV target tracking system based on 3D target detection and autonomous navigation algorithm. The main contributions are as follows.

(1) This paper proposes an improved CLSA-SAC algorithm based on the SAC algorithm with the additional curriculum learning process and the self-attention mechanism, which can help the agent learn navigation and obstacle avoidance behaviors more efficiently.

(2) In order to achieve the target tracking, it is necessary to compute the three-dimensional coordinate information of the object in real time. Therefore, this paper proposes a new 3D object detection network, 3D-CenterNet, implemented with the 3D parameter regression layer and a new feature fusion layer. Then the multi-parameter decoupling loss is utilized in the training time, which effectively increase the model stability and performance.

(3) Based on the Airsim and UE4 platform, this paper builds a photo-realistic and high-performance simulation environment for training and testing the UAV autonomous navigation and target tracking tasks. This simulation environment supports the environment domain randomization, which can effectively reduce the domain drift phenomenon during testing and enhance the stability of training. Besides, the environment is encapsulated the gym interface, which can be easily interacted with. Finally, the UAV target tracking system is realized in the simulation environment based on the designed autonomous navigation and 3D target detection algorithm in this paper. Compared with the traditional implementation, this system has a

fast executing speed and greatly reduces the computational complexity with little computing power requirements.

Key Words: deep reinforcement learning; UAV autonomous navigation; 3D object detection; UAV target tracking

目 录

| | |
|-------------------------------------|----|
| 摘要 | I |
| Abstract | II |
| 第1章 绪论 | 1 |
| 1.1 课题研究背景与意义 | 1 |
| 1.2 国内外研究现状及发展趋势 | 3 |
| 1.2.1 基于强化学习的无人机自主导航相关理论及发展概况 | 3 |
| 1.2.2 3D 目标检测算法的相关理论及发展概况 | 4 |
| 1.3 本文具体内容与章节安排 | 7 |
| 第2章 强化学习算法与仿真系统介绍 | 9 |
| 2.1 强化学习算法介绍 | 9 |
| 2.1.1 强化学习的定义与建模 | 9 |
| 2.1.2 价值函数 | 11 |
| 2.1.3 贝尔曼方程 | 12 |
| 2.1.4 蒙特卡洛方法与时序差分算法 (TD) | 14 |
| 2.1.5 基于时序差分法的 Q-learning | 16 |
| 2.1.6 策略梯度算法与 Actor-Critic 算法 | 17 |
| 2.2 本文仿真环境介绍 | 19 |
| 2.3 本章小结 | 22 |
| 第3章 基于深度强化学习的无人机自主导航 | 23 |
| 3.1 问题分析 | 23 |
| 3.1.1 状态空间设计 | 24 |
| 3.1.2 动作空间设计 | 25 |
| 3.1.3 奖励函数设计 | 27 |

| | |
|---------------------------------------|-----------|
| 3.2 基于主流 RL 算法训练的无人机自主导航模型比较 | 29 |
| 3.2.1 PPO 算法 | 29 |
| 3.2.2 SAC(Soft Actor Critic) 算法 | 31 |
| 3.2.3 模型设计与算法比较 | 34 |
| 3.3 改进的 CLSA-SAC 算法 | 38 |
| 3.3.1 基于课程学习方法训练 SAC 模型 | 38 |
| 3.3.2 基于自注意力机制提高避障性能 | 42 |
| 3.3.3 CLSA-SAC 算法 | 45 |
| 3.4 本章小结 | 49 |
| 第 4 章 基于深度学习的 3D 目标检测 | 50 |
| 4.1 问题分析 | 50 |
| 4.2 基于深度学习的 2D 及 3D 目标检测算法 | 50 |
| 4.2.1 单阶段 2D 目标检测算法 | 51 |
| 4.2.2 基于深度学习的 3D 目标检测算法 | 54 |
| 4.3 改进的 3D 目标检测算法 3D-CenterNet | 58 |
| 4.3.1 问题描述 | 58 |
| 4.3.2 模型结构 | 59 |
| 4.3.3 损失设计 | 64 |
| 4.3.4 KITTI 数据集 | 64 |
| 4.3.5 训练过程与实验结果 | 65 |
| 4.4 本章小结 | 67 |
| 第 5 章 无人机目标追踪算法的软件系统实现 | 70 |
| 5.1 无人机目标追踪算法框架 | 70 |
| 5.1.1 3D 目标检测与目标匹配过程 | 71 |
| 5.1.2 卡尔曼滤波 | 73 |
| 5.1.3 配合自主导航模块完成目标追踪 | 75 |
| 5.2 具体实验效果 | 76 |
| 5.3 本章小结 | 77 |

| | |
|-------------------------|----|
| 结论 | 79 |
| 参考文献 | 81 |
| 附录 A 相关证明 | 89 |
| 攻读学位期间发表论文与研究成果清单 | 94 |
| 致谢 | 95 |

第1章 绪论

1.1 课题研究背景与意义

近年来，以无人机、无人车为代表的无人系统在日常生活中的各个领域都得到了十分迅猛的发展。其中，无人机在航拍、侦查、救援等诸多场景中都得到了广泛的应用。传统的由人手控无人机虽然可以很好地执行既定任务，但是在大规模任务场景下，这种方式会消耗大量不必要的人力资源，使得无人机的使用门槛与使用成本随之变高。因此，为了更加广泛地应用无人机，对于无人机自主飞行的相关研究成为了该领域内学者们关注的重点。在众多热点技术中，无人机自主导航和目标追踪是其中较为关键的两项基础技术^[1-3]。无人机自主导航通常是指，无人机利用机载传感器（相机，激光雷达等）以及高精度地图信息进行自主决策飞行并安全抵达目的地的过程；本文所研究的无人机目标追踪技术则是在自主导航的基础上，结合视觉或者雷达传感器的信息，实现对移动目标的实时跟踪飞行。这两项技术在无人机系统中有大量的应用，也是实现其他任务的基础技术手段。

通常，根据实现方式的不同，无人机自主导航算法可以分为以下两种：一种是传统的非机器学习类的自主导航算法^[4,5]，另一种则是基于机器学习的智能导航算法^[6,7]。传统的非机器学习自主导航算法通常由三个模块构成：定位与地图构建模块（Localization and Mapping）^[8-10]，运动规划与决策模块（Motion Planning）^[11,12]，以及底层控制模块（Controlling）^[13,14]。其中定位与地图构建模块和运动规划与决策模块也被视为无人机系统的上层算法模块，其主要作用是进行脱离底层信息的任务规划与决策，为无人机计算出一条安全合适的路径。而底层控制模块主要是接受上层算法模块的指令，控制电机输出从而完成无人机飞行任务。这三个模块通常是分开进行研究的，而传统的自主导航算法的实现主要依赖于这三个模块的协同配合：在得到地图和定位信息后，规划出无人机的最优运动轨迹，控制无人机安全抵达目标点。而本文所重点研究的基于深度强化学习^[15-17]的自主导航算法则与传统方法有很大不同，整个过程并没有明确的阶段性求解，例如建立地图，优化轨迹等过程，模型的输出是端到端计算得到的，输入环境状态信息后直接输出上层决策指令，然后结合深度学习^[18,19]技术对整个模型进行训练。这样的过程与自然界中生物的学习和进化十分相似，生物在与环境的长期接触过程中通过环境给予的反馈信息来不断增强自身的适应性以及生

存能力。生物正是在这样的过程中不断地提升与改变自身的生存策略，而这也是强化学习算法的核心思想^[20]。因此，如果将无人机类比为生物的话，无人机会利用机载传感器所收集的信息以及飞行环境反馈的奖励值进行学习，使得自主导航算法在这个过程中可以变得越来越智能，进而实现避障功能并安全到达目标地点。在诸多基于强化学习的自主导航算法中，本文重点关注免模型（model-free）的强化学习算法^[21]，由于免模型算法不需要学习环境模型，训练过程也无需大量的系统参数，仅通过个体和环境交互的经验就可以学习到相应的知识，这简化了系统构建和模型训练流程，进而可以实现基于强化学习无需地图先验（mapless）^[22,23]的无人机自主导航系统。

本文所研究的另一重点内容为无人机目标追踪算法，其是在实现自主导航算法的基础上，结合计算机视觉算法完成的。该算法主要是通过视觉传感器的图像信息解算出目标的位置，然后将该位置信息输入给自主导航算法模块，进而实现对移动目标的实时追踪。为完成上述追踪任务，无人机系统首先需要对目标物体进行视觉上的检测与跟踪^[24,25]，而通常的视觉检测算法其输出为目标在图像坐标系下的二维位置信息，但是在目标追踪的过程中需要实时解算出目标的三维位置，缺失的一维信息通常需要额外的技术进行估计^[26]。传统的深度估计方法一般有着较为严重的漂移以及较大的误差。而目前新兴的基于深度学习方法的3D目标检测技术^[27]则可以直接从二维图像中估计出物体的三维坐标信息，在减少误差的同时也可以实现端到端的训练与部署。

综上所述，虽然传统算法实现自主导航与目标追踪的主要问题在于系统的多模块耦合所带来的复杂性，以及由于各个模块之间依赖紧密，实现起来较为繁琐，使得计算资源消耗较多。但是传统算法由于各个模块的稳定性和可解释性，仍旧是目前解决无人机领域相关问题的重要方法。基于深度学习和强化学习的无人机自主导航与目标追踪算法，近几年才开始被一些学者所关注和研究，虽然目前可以在仿真系统中高效地完成训练过程，但是深度学习和强化学习算法还存在真实训练样本获取代价过高、端到端训练方式所造成的不可解释性等问题，这些困难都是目前亟待解决的。但不可否认的是基于人工智能方法的各项无人机技术已经逐渐成为该领域未来的重要发展方向，本问也是围绕上述两项技术展开算法层面的深入研究。因此，本文所深入探讨的如何改善现有强化学习算法在无人机自主导航方向的应用，以及如何实现高效的3D目标检测技术，对于简化无人机系统和用更智能的方式来实现自主导航与目标追踪任务具有较大意义。

1.2 国内外研究现状及发展趋势

1.2.1 基于强化学习的无人机自主导航相关理论及发展概况

无人机自主导航技术一直是无人机领域的研究热点与难点。近年来，随着深度神经网络的逐渐成熟，这个领域也开始大量应用深度学习与强化学习的方法来解决相关问题^[28-30]，这使得该问题的解决方案从之前的传统方法不断向现代的深度学习方向发展和迈进，也使该领域逐渐融合了人工智能、计算机视觉、自动控制等不同领域的先进成果，成为了一个多学科交叉的复杂的研究方向。

(1) 传统的无人机自主导航方法

自上世纪 90 年代以来，无人机自主导航任务的主要挑战之一就是如何安全和可靠地避免碰撞，以便无人机可以从其起始位置顺利飞行至目标位置。传统解决自主导航问题的主要方法是将自主导航系统分成多个模块，各模块协同配合共同完成自主导航任务。而其中的运动规划与决策模块主要负责计算出无人机从当前位置到达目标位置的最优轨迹，且保证飞行过程中不会与周围环境中的障碍物发生碰撞。对于传统的无人机系统来说，运动规划模块的主要输入，包括了定位与地图构建模块（SLAM）所输出的障碍物地图以及自身的定位信息，之后通过轨迹优化算法实时解算出最优运动轨迹并产生移动动作指令，最后给到底层控制模块来控制无人机飞行^[31]。而 SLAM 模块主要依赖于视觉或激光雷达的数据以及手工设计的特征描述子^[32]，来完成对无人机自身的定位以及周围环境地图的建立。为了较好地完成导航任务，对于传统方法中属于上层算法的两个模块的协同配合以及稳定性有着很高的要求。因此，传统的自主导航解决方案存在着一些较为明显的问题：SLAM 模块中环境障碍物地图的生成和更新十分耗时并会占据较多的计算资源；SLAM 模块对于高精度传感器（如激光雷达）和手工设计的特征描述子的依赖；动规划模块对于高精度先验地图的依赖等等。

(2) 基于深度强化学习的无人机自主导航

目前深度强化学习算法已经发展到了可以有效应对多种时序决策型任务的阶段，如 Atari 游戏，围棋比赛，星际争霸，自动驾驶等等^[33-35]。针对上述传统自主导航方法的缺点，在一些问题上，基于强化学习的无地图 (mapless) 自主导航与避障技术成为了一种更受欢迎也更具挑战的技术手段，其将传统的 SLAM 和运动规划模块相结合形成了一个整体的上层规划结构，这样的设计极大地减少了系统的计算量。强化学习 (Reinforcement Learning) 是一种通过奖励函数进行学习，从而解决状态信息映射到

动作空间的即时序列决策方法，对于基于深度强化学习的运动规划模块来说，可以直
接利用机载传感器的观测信息以及输入的目标位置信息，在没有全局先验地图输入的情
况下，执行安全的导航行为。虽然这样的方法具有很大的难度和挑战性，但随着深
度强化学习的快速发展^[36-38]，上述可以应用到无人机平台的自主导航决策方法如今
已可以初步实现。例如，Chen 等人^[39]认为强化学习算法可以直接通过输入 RGB 图
像来推断出局部地图，其中深度学习模型负责隐式地学习出信息，这样的设计使得他
们可以方便地训练出基于 CNN^[40] 网络的，直接输入 RGB 图像来产生动作的运动规
划模块，并且通过 Q-learning 算法^[41] 完成了较为简单的避障行为；Chaplot 等人^[42]通
过设计分层强化学习^[43]，将 SLAM 模块与探索规划模块同样设计为一个可学习的算
法植入到整个自主导航算法的框架当中，并分层设计了全局策略和局部策略，对自主
导航与探索任务进行基于强化学习的自主地图建立和路径规划；Tai 等人^[22]利用激光
雷达的点云信息以及目标点位置作为状态信息，同样实现了基于目标驱动的强化学习
算法，他们基于 MLP 网络在仿真环境中成功训练出驱动无人车进行自动避障与导航
的功能模块，并且由于激光雷达数据在仿真环境和现实环境中具有较小的差异，训练
好的模型可以直接迁移到现实环境中进行使用，同样取得了较好的效果。尽管强
化学习在很多任务上都取得了突破性的成功，但是将其直接应用到无人机自主导航模块仍
旧存在诸多问题。例如，训练过程需要大量样本，因此模型训练一般只能在虚拟的仿
真环境中进行；在测试场景和训练场景有较大变化时，算法可能因为这种域漂移现象
而失效；如何设计合适的奖励函数以提高训练效率；如何提高训练过程中的样本效率
等问题都是目前该领域亟待解决的。

1.2.2 3D 目标检测算法的相关理论及发展概况

本文所研究的无人机目标追踪算法，是建立在上文的无人机自主导航技术的基础
上，结合计算机视觉检测算法来实现的对运动目标的实时追踪，因此视觉检测技术是
本节所述重点。

(1) 传统的视觉目标检测与跟踪技术

在计算机视觉领域，较为早期的目标检测算法是按照人为划分的不同阶段进行
的，主要流程包括了感兴趣目标区域选取^[44,45]、根据目标区域和手工设计的特征描
述子提取区域特征、通过对目标外观模型进行建模与匹配、根据 SVM^[46] 等算法在目
标区域内进行分类。具体实现包含了滑动窗口或启发式窗口搜索、区域匹配、特征点

匹配等算法。常用的特征有 SIFT 特征、HOG 特征和 SURF 特征等^[47,48]。如 2001 年 Viola^[49] 等人提出的人脸检测器，通过积分图像^[50] 等技术实现了目标检测功能。2005 年基于 HOG 的检测算法被提出，这使得 HOG 成为了后续众多传统目标检测算法特征提取阶段的基础。2008 年，Felzenszwalb 等人提出的 DPM 目标检测算法^[51]，进一步提升了传统目标检测算法的精度。

2012 年，Alexnet^[40] 的出现改变了计算机视觉领域的发展，以神经网络为主体的深度学习技术开始在各种视觉领域崭露头角，在目标分类、目标检测、语义分割等方向都实现了远超传统算法的精度。其中 RCNN^[54,103,104] 系列算法是将 CNN 网络用于检测领域的开山之作，该方法相比于传统的检测算法在性能上有了突破性提升。RCNN 是经典的两阶段算法：首先，利用 selective search 算法^[45] 进行候选区域筛选，产生用于后续阶段的 2000 个候选框，包括了其在图片中的大小与位置信息；然后将这 2000 个候选框在原始图像上进行截取和缩放操作，并将两千张图片分别通过同一个 CNN 网络进行特征提取并利用 SVM 进行分类。Fast R-CNN 仍然利用 selective search 得到 2000 个候选区域，但是并没有重复执行输入每个候选区域的 CNN 前向操作，而是只针对原始图片进行了一次前向特征提取，之后在原始特征图上进行候选框特征选取，利用 RoI pooling 算法解决了候选框在特征图上尺度不同的问题，使得各个候选框可以得到相同大小的特征。Fast R-CNN 算法也取消了 SVM 分类器，直接利用神经网络进行分类和回归训练。Faster R-CNN 在 Fast R-CNN 的基础上取消了利用 selective search 进行候选区域筛选的操作，而是设计了另外一个 RPN (region proposal network) 网络进行候选区域生成，同时 RPN 可以与后续的分类回归网络进行统一训练，使得整个算法可以完全端到端实现，不再依赖其他算法产生候选区域。RPN 网络提出了预设框 (anchor) 的概念，在每一个特征点位置处，均设置了 k 个大小不一的预设先验框，这也解决了目标检测算法中的变长 (variable-length) 问题。相比于 selective search 等算法必须主动检测出物体在哪里，anchor 的操作将检测位置的问题变成了：对于每一个 anchor 先验框都需要网络预测该 anchor 是否包含了一个物体，以及怎样修改该 anchor 的大小使得其对于该物体尺寸更为合适。最后，Faster R-CNN 将每个位置的 anchor 框的预测分值降序排列并提取前 300 个作为候选区域，候选区域的减少以及共享特征卷积的方法使得 Faster R-CNN 的速度有了极大提升。RPN 的提出也对于后续目标检测算法的发展起到了至关重要的影响。

在这之后目标检测领域也提出了许多单阶段检测算法，例如 YOLO 系列、SSD 系

列等等，在兼顾检测速度的同时保障了目标检测的精度。

(2) 基于深度学习的 3D 目标检测技术

在无人机目标追踪任务中，普通的视觉目标检测算法只能获取目标在图像平面上的二维坐标信息，由于需要实时跟踪物体，因此必须要获取目标在无人机坐标系下的三维位置信息，并将此信息输入到自主决策与导航模块，才能进行相应的决策与规划，从而实现目标追踪过程。因此，无人机目标追踪任务中最为重要的一个环节就是如何利用视觉传感器输出的信息解算出目标物体的三维坐标信息。而从目标在图像中的二维信息恢复到三维空间中的位置信息本身就是一个病态问题，通常丢失的一维深度信息需要根据图像特征来进行额外估计和恢复。传统方法进行深度估计的建模复杂而且精确度很差，而基于深度学习^[57,58] 的方法则通过拟合大量数据的方式进行学习，并且可以在相应的场景下有着远超传统方法的估计准确度。但是将目标检测系统和深度估计系统相结合来计算目标物体的三维位置无疑额外增加了整体系统的计算量和资源消耗，而目前新兴的 3D 目标检测技术^[59,60] 则恰好可以解决上述问题。3D 目标检测技术是从 2D 目标检测技术发展而来的，3D 目标检测算法可以按照其输入数据类别分为三类：基于激光雷达信息的、基于单目相机信息的和基于多传感器融合信息的 3D 目标检测算法。例如，Voxelnet^[61] 利用激光雷达的信息，首先把点云分割成为不同的 3D 体素，再把不同体素中的点云通过 VFE(Voxel Feature Encoding) 转化成标准的高维特征表达，最后使用 Faster-RCNN^[54] 中的 RPN(Region Proposal Network) 网络对物体进行分类和 3D 参数回归；而基于单目相机的 3D 目标检测算法，通常是利用 2D 目标检测的模型，在引入不同的先验条件下通过对原网络模型改进并增加额外的三维参数估计层，直接预测每个检测对象的三维包围框，这大大提高了训练和检测速度，并且可以直接从单帧图像中估计出目标的三维坐标信息，使其在无人驾驶，无人机等领域有着十分重要的应用；利用融合信息的方法如 AVOD^[62]，使用 LIDAR 点云和 RGB 图像由两个子网络生成共享的特征，再提取两个特征图对应的区域进行融合，最后利用 RPN 网络输出 3D 候选框来进行 3D 物体检测。虽然基于雷达信息的 3D 目标检测算法通常有较高的检测精度，但是由于雷达装置的价格昂贵且并不适用于简便部署的无人机基础平台，因此本文重点研究单目相机输入信息的 3D 目标检测技术。

综上所述，基于深度强化学习的无人机自主导航和目标追踪方法无论在国内还是国外都属于前沿性的研究，而且大部分研究结果出现在近几年内，受到了很多研究者们的广泛关注与重视。

1.3 本文具体内容与章节安排

为实现无人机系统的自主导航与目标追踪算法，本文综合参考了目前相关领域的诸多工作，并且设置了五个章节对各部分内容进行详细地介绍，各章节具体内容编排如下：

第一章：绪论部分。本章节重点介绍了论文课题所涉及的研究背景以及国内外对该领域的具体研究情况，重点分析了本文中基于强化学习的自主导航算法相比于传统算法的优势和所面临的挑战，以及 3D 目标检测算法的发展概况，同时探讨了本课题的应用价值和应用前景，并介绍了本文的具体研究内容和章节安排。

第二章：强化学习算法与仿真系统介绍。本章节详细介绍了强化学习的算法基础，从 MDP 开始简述强化学习的构成以及经典的动态规划、蒙特卡洛、时序差分算法的各自优缺点，然后介绍了深度学习和强化学习相结合的 DQN、策略梯度、Actor-Critic 等主流算法。最后介绍了本文基于 Airsim 与 UE4 平台，设计并实现的适用于本文所需的无人机自主导航与目标追踪系统的仿真训练环境，该仿真环境可以高效地实现无人机强化学习相关任务的训练与测试，同时支持 gym 接口以及环境域随机化等操作。

第三章：基于深度强化学习的无人机自主导航算法。本章节首先介绍了如何利用强化学习算法对无人机自主导航问题进行建模分析，并通过实验证实了由于样本效率问题以及任务的复杂性(既需要学会安全避障又需要到达指定的目标地点)，主流的深度强化学习算法在自主导航任务上的表现较差。然后针对这些方法所存在的问题，本文基于 SAC 算法设计了相应的课程学习过程以及自注意力网络层，并提出了改进的 CLSA-SAC 算法。其中课程学习过程可以使强化学习模型阶段性地探索到有效样本并减轻模型的学习压力，逐渐习得最终的复杂任务。自注意力机制层则能够使模型关注到更有价值的信息，同时学到更有效的避障行为。同时在本章的最后，对自注意力机制层做了可解释性分析。

第四章：基于深度学习的 3D 目标检测算法。本章节重点介绍为完成目标追踪任务所需的基于单目图像信息的 3D 目标检测技术，提出了一种基于 CenterNet 改进的 3D 目标检测网络 3D-CenterNet，整体网络架构基于 CenterNet 实现，设计了新的 3D 参数回归层，同时为更好地提取多尺度特征，设计了一种新的特征融合层，并且在计算损失时将网络的输出进行三维空间内的解耦，并由此分别得到多个 3D 估计参数的损失，这些设计有效增加了模型训练的稳定性与最终的检测性能。

第五章：无人机目标追踪算法的软件系统实现。本章节基于第二章、第三章和第

四章的内容具体设计了无人机目标追踪算法框架，同时在仿真环境中对该算法进行了系统验证与测试，相比于传统实现方法该系统设计大大减少了计算复杂度，降低了算力需求。

第2章 强化学习算法与仿真系统介绍

深度强化学习无疑是目前人工智能领域中最为前沿的技术之一，利用深度学习的特征抽象能力以及强化学习的思想，这项技术能够解决多种复杂的现实决策问题。本章重点介绍强化学习算法原理，作为第三章内容的理论铺垫。本章主要内容可分为以下两个部分：第一部分介绍强化学习的理论以及相关算法；第二部分详细介绍了本文所设计的无人机仿真环境的系统构成与特性。

2.1 强化学习算法介绍

2.1.1 强化学习的定义与建模

强化学习是基于马尔可夫决策过程 (MDP)^[63] 的一种机器学习理论，强化学习的核心思想是在试错中进行学习，通过环境给予的奖励来不断优化和提升策略，与监督学习不同的是，奖励本身并不是明确的标签值，只在某种程度上反映了当前动作的好坏，而且强化学习中每一步的样本在时间顺序上是紧密相关的。因此，强化学习的过程可以被描述为下图2.1的形式，即个体不断地通过执行策略产生的动作来获得环境给予的奖励，利用积累的经验来不断提升自己所执行的策略的过程。

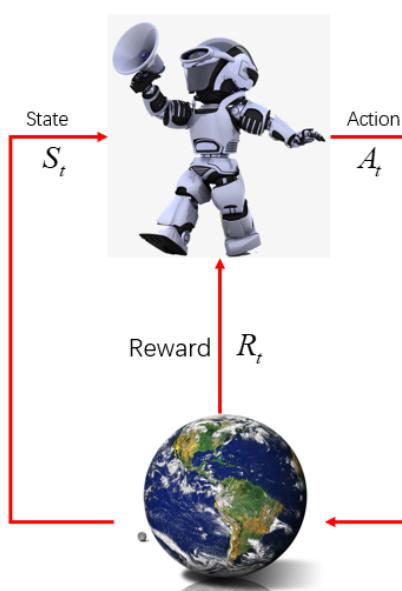


图 2.1 强化学习个体交互示意图, S_t 代表了环境状态, A_t 代表了执行动作, R_t 代表了环境奖励

强化学习通常被应用到对时序决策问题进行建模，强化学习可建模的问题主要包含以下几个要素：

(1) 环境的状态空间 S : 状态信息是环境当前情况的一种数学表示，个体在任一时刻所处的状态是唯一确定的，同时在个体施加动作后，环境状态会发生相应的变化，所有可能的状态构成了状态空间 S 。

(2) 动作空间 A : 动作是在环境中的个体所执行行为的一种数学描述，个体所有可能动作的集合构成了动作空间 A 。

(3) 奖励函数 R : 奖励函数代表着环境对个体所执行动作的反馈信息，且具体形式不被个体所知。通常环境在某一时刻反馈的奖励值，受当前时刻的环境状态、此时个体所执行动作以及下一时刻状态共同影响，因此奖励函数也可以被记为下式：

$$R_t = f(s_t, a_t, s_{t+1}) \quad (2.1)$$

(4) 个体策略 π 。强化学习所要优化和学习的核心模型，策略代表了个体在面临当前所处状态时所选取动作的依据和标准，即个体会根据策略来进行动作选取并执行。在 MDP 的建模下，策略通常被表述为一个关于当前状态的条件概率分布，即：

$$\pi(a | s) = P(A_t = a | S_t = s) \quad (2.2)$$

(5) 累积回报 (return) G : 累积回报代表了个体执行当前动作后直到这段序列结束所获得的奖励之和，同时为了综合考虑当前奖励值与未来奖励值的权重，引入了折扣因子 $\gamma^{[64]}$ ，因此累积回报可表示为下式：

$$G_t = \sum_{i=t+1}^T \gamma^{i-t-1} r_i \quad (2.3)$$

(6) 环境状态转移模型 $P_{s,s'}^a$ 。这个状态转移模型是一个概率分布，代表了在状态 s 下执行动作 a ，转移到状态 s' 的概率。实际情况中，从状态 s 转移到状态 s' ，并不只取决于当前的状态和动作，还与之前的状态相关，但是按这种情况建模会导致环境状态转移模型非常复杂，因此需要用 MDP 过程中的马尔可夫性对环境模型进行简化。即假设转化到下一个状态 s' 的概率仅与上一个状态 s 有关，与之前的所有状态都无关，这样我们既能对环境模型进行简化，同时在具体应用中如果在某种程度上将之前的状

态信息融合到当前状态的特征表示中，同样可以建模现实情况下的复杂转移模型，在实际应用中这样的处理也十分常见。同理，无人机的自主导航任务也可以被看做一个MDP过程，在接受到视觉传感器和目标点位置信息后，输出飞机的控制指令进行决策，控制输出仅取决于当前时刻无人机的状态。

强化学习依赖于以上六要素，并决定其是否能够正常运行。同时强化学习过程并没有明确个体需要了解环境的机制，也就是说强化学习建模适用于环境模型未知，且能够频繁和环境交互的情形，个体不断从环境中获得奖励反馈，积累经验，进而改进自身的策略。同时从策略的定义中可知，强化学习是一个端到端的系统，只关心输入状态和输出动作，通过和环境交互直接形成一个反馈学习机制，因此这样的特点也方便我们引入深度学习的方法对策略模型进行优化。

2.1.2 价值函数

强化学习的目标是使得个体获得最大的期望累积回报，即强化学习并不要求每一时刻下个体都做出获得最大奖励的动作，而是“趋利避害”、“目光长远”地去学习一种能够长期获得较高收益的行为。因此，强化学习的优化目标可以表达为下式：

$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t \gamma^t r(s_t, a_t) \right] \quad (2.4)$$

上式中 θ 是个体策略所对应的参数（可以利用高斯过程、神经网络等方式建模）， $p_{\theta}(\tau)$ 是此段序列在当前策略对应的MDP框架下出现的概率，可通过马尔可夫性质进行独立性分解，即：

$$p_{\theta}(\tau) = p_{\theta}(s_1, a_1, \dots, s_T, a_T) = p(s_1) \prod_{t=1}^T \pi_{\theta}(a_t | s_t) p(s_{t+1} | s_t, a_t) \quad (2.5)$$

强化学习中另外两个重要的概念就是状态价值函数与动作价值函数，状态价值函数 $V_{\pi}(s)$ 描述了在状态 s 下的期望累积回报，可以表达为下式：

$$V_{\pi}(s) = E_{\pi}(G_t | S_t = s) = E_{\pi}(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s) \quad (2.6)$$

动作价值函数 $Q_\pi(s, a)$ 描述了在当前状态 s 下执行动作 a 所能带来的期望累积回报，即：

$$Q_\pi(s, a) = E_\pi(G_t | S_t = s, A_t = a) = E_\pi(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s, A_t = a) \quad (2.7)$$

这两个价值函数都描述了个体在未来一段时间内的期望奖励总和，通常研究者们更喜欢利用动作价值函数进行估计以优化当前策略，因为动作价值函数直接反映了某一状态下选取该动作的优劣程度；而状态价值函数描述了当前状态下的潜在价值，状态价值函数通常会被利用做策略评估、作为基线（baseline）减小强化学习的方差^[65]以及计算优势值（advantage）^[66]等操作。如果 $Q_\pi(s, a)$ 能够被准确的估计出来，那么就可以利用贪婪策略获得最优策略行为，也就是说利用动作价值函数可以直接产生策略，这种思想也被视为值估计方法。例如，在无人机自主导航过程中，某一状态下无人机的视觉传感器中出现相应障碍物，左转后飞行和右转后飞行的动作价值都是 10，而沿直线飞行的动作价值只有 1，那么此时根据贪婪策略选取最大动作价值对应的动作就可以顺利地执行避障行为。

以上诸多变量均是期望的形式，也就是说强化学习的优化过程，重点关注的是各种随机变量的期望，只要这些期望是在一个合理的分布下，使得优化的参数关于期望值是光滑可导的即可。

2.1.3 贝尔曼方程

2.1.2 中介绍了状态价值函数与动作价值函数，二者之间的关系如下：

$$V_\pi(s) = E_\pi [Q_\pi(s, a) | S_t = s] = \sum_{a \in A} \pi(a | s) Q_\pi(s, a) \quad (2.8)$$

$$Q_\pi(s, a) = E_\pi [R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] = \sum_{s' \in S} P_{s,s'}^a [R_{t+1} + \gamma V_\pi(s')] \quad (2.9)$$

由此可以推出状态价值函数和动作价值函数关于自身的迭代方程：

$$V_\pi(s) = \sum_{a \in A} \pi(a | s) \sum_{s' \in S} P_{s,s'}^a [R_{t+1} + \gamma V_\pi(s')] \quad (2.10)$$

$$Q_\pi(s, a) = \sum_{s' \in S} P_{s,s'}^a \left[R_{t+1} + \gamma \sum_{a' \in A} \pi(a' | s') Q(s', a') \right] \quad (2.11)$$

式(2.10)和(2.11)是著名的贝尔曼方程^[67]，并且在特定条件下可以证明价值函数会收敛到唯一不动点，证明过程详见附录A。也就是说在固定策略下，可以通过动态规划(dynamic programming)的思想，分治迭代求解准确的价值函数。这个求解过程也被叫做策略评估(policy evaluation)，通过策略评估后收敛的价值函数能够准确反映当前策略所执行的每一步的具体价值。

如2.1.2中所述，当获得某一策略准确的动作价值函数后，可以通过使用贪婪策略的方法来执行个体的下一步动作，这样做可以获得当前条件下的最大价值函数的动作，同时改变了原有策略。这种根据价值函数贪婪地来调整策略的方法，被叫做策略改进(policy improvement)，并且可以证明，根据 $Q_\pi(s, a)$ 贪婪地选取一个最优动作，所得到新的策略的价值函数 $V'_\pi(s)$ 一定会优于原有策略的价值函数 $V_\pi(s)$ ，证明过程详见附录A。策略评估可以对确定的策略进行准确的价值函数估计，而策略改进可以提升原有策略，那么二者之间交替迭代就能够获得最优策略，其过程如图2.2所示，这个过程被称为策略迭代(policy iteration)，并且策略迭代在一定条件下会收敛到最优解，即最优策略的价值函数，这样我们就能方便地将求解最优策略的过程转换为价值函数迭代不断产生新策略的过程。

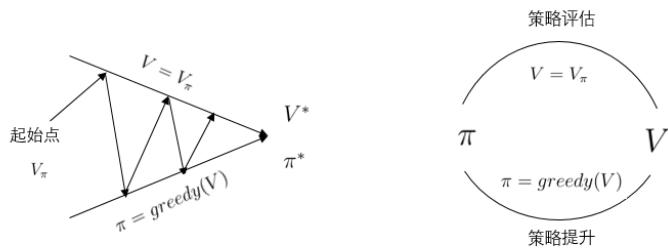


图 2.2 强化学习中 policy iteration 示意图， π^* 代表了最优策略

强化学习的最终目标就是求解最优策略，针对一个具体任务通常会存在一个或多个最优策略，最优策略所对应的价值函数定义为：

$$V_*(s) = \max_{\pi} V_{\pi}(s) \quad (2.12)$$

$$Q_*(s, a) = \max_{\pi} Q_{\pi}(s, a) \quad (2.13)$$

对于最优策略的求解，通常可以按照上文的策略迭代进行，但是策略迭代的一个最大的缺点就是每一次迭代都涉及了策略评估，而策略评估本身就是一个需要多次遍历的迭代过程，倘若遇到的问题有着复杂的状态空间，则其求解过程就变得十分漫长与繁琐。那么一种简单的思路就是，虽然策略评估是迭代进行的，理论上需要等待其收敛，但实际上是可以提前截断策略评估的过程，且提前截断策略评估过程并不影响策略迭代的收敛是可以证明的，证明过程详见附录A。一种特殊的情况就是策略评估只迭代一次，然后就立刻进行策略改进，这种算法被称之为价值迭代，其更新公式就是把策略评估和策略迭代两个公式结合在一起，这就是最优贝尔曼方程，也是值方法的核心思想：

$$V^*(s) = \max_a \left(\sum_{s' \in S} P_{s,s'}^a [R_{t+1} + \gamma V^*(s')] \right) \quad (2.14)$$

$$Q^*(s, a) = \sum_{s' \in S} P_{s,s'}^a \left[R_{t+1} + \gamma \max_{a'} Q^*(s', a') \right] \quad (2.15)$$

价值迭代也是策略迭代的一种特殊形式，策略迭代的思路是策略评估与策略改进二者同时进行并且相互作用，这种形式可以用广义策略迭代 (GPI)^[68] 来表示，而几乎所有强化学习方法都可以被表述为广义策略迭代，这些算法都包含明确定义的策略和价值函数，策略总是基于特定的价值函数进行改进，价值函数也会向对应策略的真实期望回报收敛。

2.1.4 蒙特卡洛方法与时序差分算法 (TD)

基于动态规划方法求解贝尔曼方程的弊端在于必须知道环境的状态转移分布，这就导致了很多时候基于动态规划的方法几乎没法使用。为了解决该问题，诞生了一系列免模型 (model-free) 的求解方法，其中蒙特卡洛方法是利用了统计学上的蒙特卡洛原理对期望价值函数进行无偏估计，它要求个体采样得到一幕完整的序列 (episode)，然后用这些样本来对价值函数进行估计：

$$V(S_t) = V(S_t) + \frac{1}{N(S_t)} (G_t - V(S_t)) \quad (2.16)$$

上式中 $N(S_t)$ 代表了该状态的计数值。相比于动态规划算法，蒙特卡洛算法不依赖环境转移概率，并且需要完整的一幕序列样本进行学习，同时蒙特卡洛的学习效果随着样本数量的增加而越来越好。蒙特卡洛算法的策略提升也与动态规划方法不同，通常

采用 ϵ -贪婪法进行策略改进， ϵ -贪婪法是设置一个较小的 ϵ 值并使用 $1-\epsilon$ 的概率选取最大价值的动作，而以 ϵ 的概率从剩余的动作中随机选取。这样做的目的是由于在环境转移概率未知的情况下，尽可能多的探索未曾到达的状态，使得蒙特卡洛估计的更加准确。由此可见，蒙特卡洛算法也符合 GPI 的范式。

蒙特卡洛算法虽然可以对价值函数进行无偏估计，但是它所需要的样本必须是一幕完整的序列，因为它需要累积求和来计算期望回报值进而估计价值函数。但是实际情况往往无法获得完整的序列样本，那么蒙特卡洛法就失效了。时序差分（Temporal-Difference, TD）算法综合了蒙特卡洛算法无需环境模型以及动态规划算法基于其他状态价值进行更新的优点，可以解决上述问题。对于时序差分而言，即使只有部分状态序列，仍然可以近似求解价值函数，它利用了价值函数的自举（bootstrapping）进行近似，只需要两个连续的状态和奖励值就可以进行价值函数的估计：

$$V_\pi(s) = \mathbb{E}_\pi(R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s) \quad (2.17)$$

上式中的 $R_{t+1} + \gamma V_\pi(S_{t+1})$ 就可以近似代替蒙特卡洛中计算的累积回报 G_t ，一般将 $R_{t+1} + \gamma V_\pi(S_{t+1})$ 叫做 TD 目标值（TD-target），而 $R_{t+1} + \gamma V_\pi(S_{t+1}) - V_\pi(S_t)$ 也被叫做 TD 误差（TD-error），价值函数的更新公式为：

$$V(S_t) = V(S_t) + \alpha (R_{t+1} + \gamma V_\pi(S_{t+1}) - V(S_t)) \quad (2.18)$$

由于在时序差分中没有完整的序列样本，也就没有对应的计数值 $N(S_t)$ ，所以用一个 $[0,1]$ 的系数 α 代替更新步长，同时该系数和多臂赌博机中的增量式更新系数的作用相同，可以应对非平稳（nonstationary）环境下的强化学习^[20]。时序差分法的优势在于不需要完整的一个 episode 的样本就可以进行学习，并且可以在持续进行的环境中学习，可以更快速灵活地进行价值函数的估计，这是非常具有实际意义的。另外，时序差分法在更新价值函数时使用的是基于自举的 TD 目标，也就是即时奖励和下一状态的预估价值的和来替代序列结束后所计算的累积回报，因此相比于蒙特卡罗法，时序差分方法的估计是有偏的。虽然时序差分法得到的价值函数是有偏估计，但其方差却比蒙特卡罗法得到的方差要低很多，通常比蒙特卡罗法更加高效，所以现在主流的强化学习求解方法都是基于时序差分的。当然也有很多综合蒙特卡洛和 TD 法的算法，诸如 n-TD、TD(λ)、资格迹等算法。

2.1.5 基于时序差分法的 Q-learning

Q-learning 算法是经典的基于值方法的强学习算法，它是基于时序差分法和最优贝尔曼方程的一种离轨策略算法 (off-policy)。离轨策略算法指的是采样策略和所要优化策略并不一致的算法，相比较于在轨策略 (on-policy) 其应用场景更加广泛，样本效率也要更高。Q-learning 算法概述如下：基于当前状态 s ，用 ϵ -贪婪法执行动作 a ，得到奖励 R ，并进入状态 s' ，选择能使 $Q(s', A)$ 最大的动作 a' 代入 TD 目标值^{2.17}，即按照如下方式更新动作价值函数：

$$Q(s, a) = Q(s, a) + \alpha \left(R + \max_a Q(s', a) - Q(s, a) \right) \quad (2.19)$$

因为有筛选最大价值对应动作的过程，所以 Q-learning 只适用于离散动作空间的问题，且经典的基于表格型的 Q-learning 只能解决规模较小的问题，当遇到复杂连续的状态空间时就难以应对了。因此，存储精确的价值函数的方法便不再可行，这种情况便需要对价值函数进行近似表示。该类近似的方法有很多种，例如线性表示、决策树、高斯过程、神经网络等方式。同时由于深度学习技术的发展，将神经网络与强化学习相结合，成为了目前解决复杂决策问题的主流方法。

Q-learning 与深度学习结合的方法被简记为 DQN 算法^[36,69]，它的基本思路是利用神经网络来代替精确计算的动作价值函数表格，Q 网络的输入就是状态向量 s ，输出是所有动作在该状态下的动作价值函数 $Q(s, A)$ 。DQN 还利用了额外的两个技巧来增加网络训练的稳定性：经验回放池 (replay buffer) 和目标网络 (target net)。经验回放池是为了解决强化学习中样本相关性问题的，神经网络的训练通常依赖于独立同分布的数据，但是强化学习过程中采样得到的样本通常不满足这样的要求，因为时序关系导致样本间会有依赖性存在，尤其是在线强化学习过程，这个现象更加明显。利用这样的样本进行训练很容易导致局部过拟合现象（在学习过程中局部样本拟合的很好，却不停地在丢失之前学到的东西），经验回放池就是将之前得到的经验序列存储到内存中，同时在训练网络时，随机从池中进行经验抽取，这样不仅可以有效避免局部过拟合同时也可以打乱样本之间的依赖性关系，学习到更具泛化能力的网络模型。目标网络是为了解决由于自举 (bootstrap) 生成的 TD 目标值所带来的梯度不稳定问题，由于 Q-learning 的学习是用 TD 误差作为损失来进行梯度回传的，这个过程与监督学习有明显的差别，其更新的目标值并不是稳定分布下人为标定的标签值，而是由

上一次更新后的网络产生的价值函数和当前奖励的加和，这个目标值显然不是从一个稳定的分布中生成的，而且在实际学习中会产生十分大的方差与不稳定性甚至导致学习失败。而目标网络是在一段时间内不参与更新的，这样利用目标网络生成的 TD 目标值在一段时间内可以保证稳定的分布形式，使得梯度下降的过程更逼近于监督学习，这样就会让学习过程变得更加稳定。DQN 的算法流程如下：

Algorithm 1: DQN 算法

Input: 每次更新样本数量 b , 学习率 α , 目标网络更新步数 n

- 1 初始化: 当前网络 Q_θ 的参数 θ , 目标网络 Q_ϕ 的参数 ϕ , 经验回访池 $B = \emptyset$,
 - 更新总次数 K , 当前更新次数 $k = 0$
 - 2 **while** $k <= K$ **do**
 - 3 将观测信息 s 输入当前网络 Q_θ , 利用 ϵ -贪婪法选取动作 a 执行, 然后将 (s, a, s', r) 添加至 B 中;
 - 4 从 B 中随机采样 b 个样本对 $\{s_j, a_j, s'_j, r_j\}_{j=1\dots b}$;
 - 5 利用目标网络计算每个样本的 TD 目标值 $y_j = r_j + \gamma \max_{a'_j} Q_{\phi'}(s'_j, a'_j)$;
 - 6 更新网络参数 θ : $\theta \leftarrow \theta - \alpha \sum_j \frac{dQ_\phi}{d\phi}(s_j, a_j) (Q_\phi(s_j, a_j) - y_j)$;
 - 7 每过 n 步即 $k \% n$ 等于 0 时, 更新目标网络参数: 将 θ 复制到 ϕ ;
 - 8 $k += 1$.
 - 9 **end**
-

2.1.6 策略梯度算法与 Actor-Critic 算法

策略梯度算法区别于值方法, 它并没有利用价值函数来提升策略, 而是直接以累积奖励的期望为目标函数, 对策略进行优化, 同时策略梯度算法是一种在轨策略算法 (on-policy), 利用了蒙特卡洛原理来近似期望目标。其推导过程如下: 强化学习的目标如式 (2.4) 所示, 将该目标作为关于网络参数 θ 的损失函数 $J(\theta)$, 对 $J(\theta)$ 直接求导可得:

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau = \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau = E_{\tau \sim \pi_\theta(\tau)} [\nabla_\theta \log \pi_\theta(\tau) r(\tau)] \quad (2.20)$$

其中 $r(\tau) = \sum_t \gamma^t r(s_t, a_t)$ 代表了累积回报, $\pi_\theta(\tau) = \pi_\theta(s_1, a_1, \dots, s_T, a_T) = p(s_1) \prod_{t=1}^T \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t)$, 且由于

$$\log \pi_\theta(\tau) = \log p(s_1) + \sum_{t=1}^T \log \pi_\theta(a_t | s_t) + \log p(s_{t+1} | s_t, a_t) \quad (2.21)$$

因此, 可以得到最终的策略梯度表达式:

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t | s_t) \right) \left(\sum_{t=1}^T r(s_t, a_t) \right) \right] \quad (2.22)$$

在得到了关于 θ 的梯度后, 就可以利用梯度上升法进行学习了, 这种原始的策略梯度算法被叫做 Reinforce 算法^[70]。在采样得到一个 episode 的样本后, 虽然可以直接通过该算法进行学习, 但是策略梯度算法有一个很大的问题就是该算法相比于值方法有更大的方差, 而大的方差就意味着算法的稳定性较差。所以在 Reinforce 算法的基础上引入了因果论中 *reward-to-go*^[71] 的概念, 并且通过另一个网络学习累积回报奖励的期望, 即 $Q(S_t, A_t)$; 并且通过减去基线(baseline)的方法利用优势函数(advantage function)来进一步缩小方差, 这样就得到了 VPG 算法(Vanilla Policy Gradient), 其算法流程如下:

Algorithm 2: VPG 算法

Input: 学习率 α , 序列样本数 b

- 1 初始化: 当前策略网络的参数 θ , 值网络网络的参数 ϕ , 更新总次数 K , 当前更新次数 $k = 0$
 - 2 **while** $k <= K$ **do**
 - 3 通过执行策略 $\pi_k = \pi(\theta_k)$ 收集完整的序列样本 $\mathcal{D}_k = \{\tau_i\}, i = 1..b$;
 - 4 计算每一步的累积期望回报 \hat{G}_t ;
 - 5 基于当前的值网络 V_{ϕ_k} 计算优势价值函数 \hat{A}_t ;
 - 6 计算当前策略网络的梯度:
$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t | s_t) \Big|_{\theta=\theta_k} \hat{A}_t;$$
 - 7 利用梯度上升法更新当前策略网络参数: $\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k$;
 - 8 更新值网络参数: $\phi_{k+1} = \arg \min_\phi \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_\phi(s_t) - \hat{R}_t \right)^2$;
 - 9 $k += 1$.
 - 10 **end**
-

通过以上分析可知, 基于策略梯度的算法可以直接优化策略模型本身, 而且其可

以解决连续动作空间问题，另外一个优点就是策略梯度的网络往往输出的是一种随机策略，相比于 Q-learning 等值方法其最优策略通常为确定性策略，随机策略往往更符合现实问题的一些要求。

上述的策略梯度算法本质上已经形成了 Actor-Critic 算法的框架，其中策略的产生由 Actor 负责，并且优化策略就是直接优化 Actor，而 Critic 负责对当前形势进行评估（估计 $reward - to - go$ ），也就是近似当前策略下的值函数，二者同步更新以学习更优的策略。目前的许多主流算法都是基于这种 Actor-Critic 框架进行改进的，这种思想也可以看成是策略梯度和值函数方法的结合，它综合了策略梯度的优点，可以直接对策略端的网络进行优化学习，并且保留了值函数网络用来进行策略评估的思想，产生的值函数也参与策略更新以减小算法方差。

2.2 本文仿真环境介绍

本文的仿真环境是基于微软的 Airsim^[72] 与 UE4(Unreal Engine) 平台所开发的，具体的搭建系统流程图如下图所示：

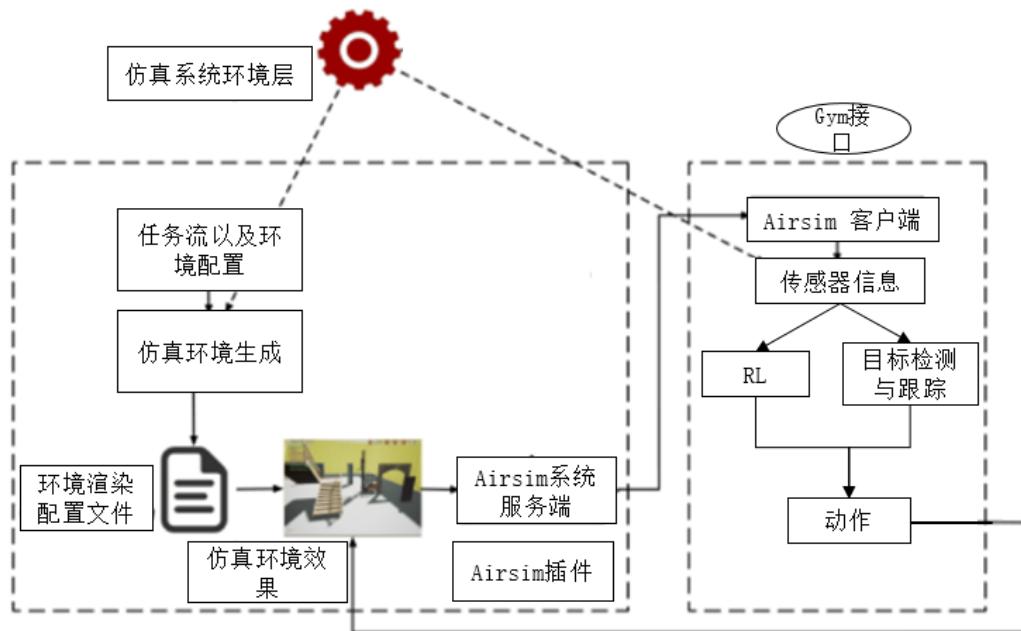


图 2.3 仿真环境开发框架图

仿真环境的底层直接利用了 Airsim 开发的无人机模型和动力学系统以及 UE4 的物理仿真环境，然后通过自行开发的环境生成组件来进行每次训练时的环境初始化以

实现强化学习过程中的域随机化^[73]。无人机的控制方式以及传感器信息获取是通过airsim的底层api进行通信的，然后通过绑定gym接口进行上层的封装，方便在算法调用时有一个统一的接口，并使得仿真环境和算法层完全脱离。

通常，对于复杂的强化学习任务来说，在测试环境和训练环境存在差异时，很容易出现域漂移现象，环境之间的差异性往往导致学习的模型无法使用。域随机化的方法属于迁移学习^[74]的领域，通过创建一系列仿真环境，环境中有着随机生成的各种属性，在这种环境中训练的模型通常会更加鲁棒以及稳定，能够学习到更加抽象的特征。尽管域随机化的范围是人为挑选的，但这种挑选的结果通常涉及到这个域内环境的先验知识以及多轮试错后得到的样本经验，这就类似于一个额外的优化过程，使得模型不仅能够学习目标任务，更能适应环境的变化。

本文开发的环境生成器支持移动障碍物的种类与数量、环境大小、墙体颜色、无人机导航目标点的随机初始化，设计以上域随机化的操作都是为了使得无人机自主导航任务中的模型更加鲁棒以及稳定。具体效果如图2.4至2.8所示。因此，本文搭建的



图 2.4 障碍物种类对比图



图 2.5 障碍物数量对比图



图 2.6 环境大小对比图

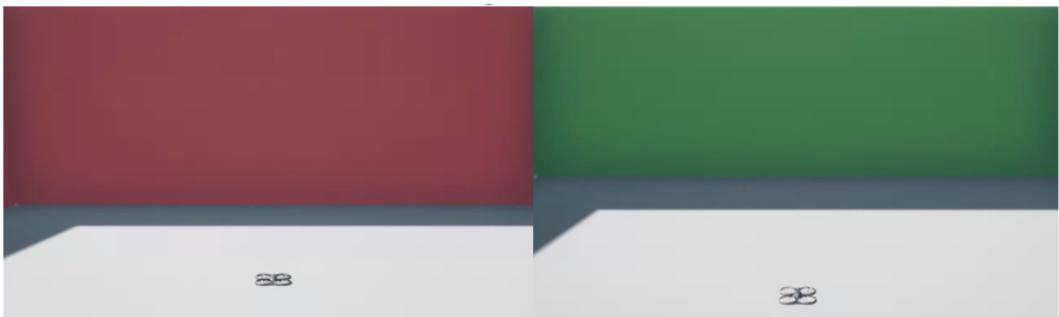


图 2.7 墙体颜色对比图

仿真环境相比其他的环境具有如下特性：支持 gym 接口，方便训练程序与环境直接交互；由于使用 UE4 平台，所以视觉信息是足够逼真的，可以实现 photo-realistic 级别的信息质量；支持环境域随机化（domain randomization），表2.1为本文仿真环境与其他同类型仿真环境的对比。

表 2.1 本文仿真环境与其他同类型仿真环境的差异

| 特性 | Ours | Airsim | Gazebo | PyRobot |
|----------------------|------|--------|--------|---------|
| Photorealism | √ | √ | ✗ | ✗ |
| Domain randomization | √ | ✗ | ✗ | ✓ |
| Gym interface | √ | √ | ✓ | ✓ |

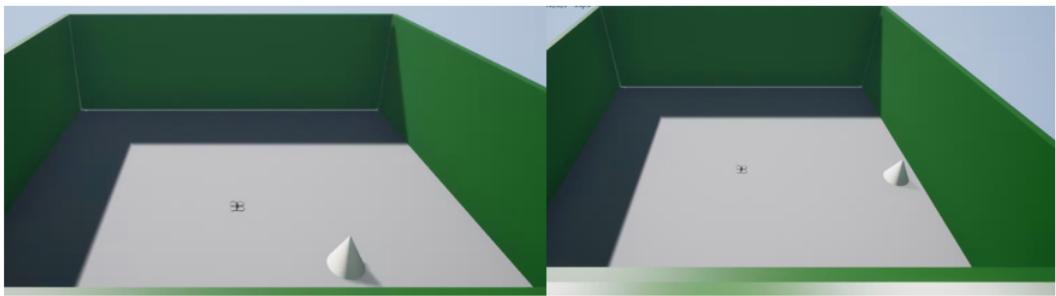


图 2.8 目标点位置对比图，白色圆锥体为目标点位置

2.3 本章小结

本章主要介绍了强化学习的算法基础，从 MDP 开始阐述强化学习的基本要素与理论发展，并详细介绍了目前几种主流的强化学习的算法，包括 on-policy 和 off-policy 的区别等细节问题，为第三章的研究内容做好理论铺垫；同时也详细介绍了本文为完成无人机自主导航与目标追踪任务，所搭建的仿真环境的系统特性与优点，为后文的实验内容提供了一个较好的仿真平台。

第3章 基于深度强化学习的无人机自主导航

非机器学习的无人机自主导航算法通常是根据定位与建图模块所输出的环境信息，计算出无障碍物的最优运动轨迹，然后控制无人机安全飞行至目标点，这类方法通常计算量较大，且非常依赖准确的地图信息（map-based）。本章首先利用强化学习思想对无人机自主导航问题进行建模分析，设计相应的动作、状态空间与奖励函数。然后针对自主导航任务设计具体的网络模型与训练过程，并对比了多种算法在该任务上的表现。然后针对现有算法的不足，提出了改进的 CLSA-SAC 算法，利用课程学习方法降低该问题的模型学习难度，并引入自注意力机制对现有的 SAC 算法进行改进，最终根据实验结果证明了本文所提出的无地图先验（mapless）的无人机自主导航算法可以有效学习到避障以及导航行为。

3.1 问题分析

据第二章中介绍的强化学习范式，无人机自主导航任务可以被视为一个 MDP 过程，无人机在环境中接受传感器的信息作为状态描述（准确来说，这里的信息应该是观测信息，其和状态信息存在隐含的映射关系，这种建模也被叫做 POMDP^[75]），根据状态信息通过自身的网络进行相应决策，输出上层动作指令来控制无人机飞行。在这个过程中，无人机所执行的每一步动作都会根据人为设定的奖励函数来获取相应奖励值，这些奖励值又会反馈到策略端帮助决策模型提升性能，整个过程如图3.1所示。因此结合上文介绍的仿真的特性，可以按照如下方式设计该问题的状态空间、动作空间等属性。



图 3.1 无人机作为 Agent 的示意图

3.1.1 状态空间设计

本文所研究的主要算法是无地图先验的强化学习算法，对视觉传感器所采集的信息并不做额外的算法处理，避免了通过视觉传感器进一步建立地图的复杂过程，从而设计一种端到端的执行算法。基于此，状态空间可以按照^[76] 中的方法进行设计，将视觉传感器信息作为感知信息直接输入到决策网络。但是本文直接利用前视相机的深度图像作为状态，并未使用 RGB 图像。虽然 RGB 图像包含了丰富的纹理和特征，但是实现避障与导航的过程并不需要学习这种复杂的图像信息，这明显区别于图像检测与分类任务；深度图则是直接反映了周围障碍物的形状尺寸以及其距离无人机的远近，本文认为将其作为状态信息经过网络处理后可以隐式地学出周围的环境语义，更有效地学习出避障功能，同时减少了模型直接输入 RGB 图像来学习该问题的难度。

直接以深度图像作为状态信息输入，只能学习到避障行为，自主导航任务同时要求无人机可以安全飞行至目标点。因此，本文将目标点信息作为额外的状态信息输入到决策网络，区别于 Xie^[77] 直接利用目标点处的图像信息，本文是将目标点与飞机的相对位置信息 (p_x, p_y, p_z) 与深度图像经过 CNN 网络后的展开的高维特征进行拼接，同时融合了飞机自身的速度信息 (v_x, v_y, v_z) 、当前偏航角 θ_{yaw} 、飞机朝向与目标点连线的夹角 ω 等状态，作为更深层决策网络的输入信息，以帮助模型更有效地学习导航功能。由于本文假设目标点位于地面上，因此目标点的高度值默认为 0，而无人机的只需要在二维 xy 平面上的坐标位置抵达目标处即算完成任务。状态空间设计如图3.2所示。

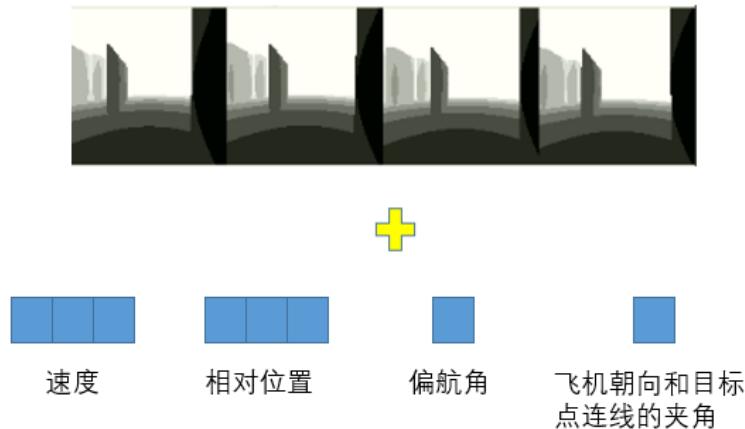


图 3.2 状态空间示意图

输入相对位置信息而非绝对位置信息，也是为方便进行后文的目标追踪任务所设计，而其中飞机朝向与目标点连线的夹角的计算方式如图3.3所示，这样的方法同样可以被视为一种基于目标驱动的视觉导航算法。同时，为了增加模型的学习性能并能更好的处理状态信息，本文将视觉传感器输出的连续4帧深度图进行堆叠（stack），形成时序上的 $4 \times 112 \times 112$ 的深度图像，将其输入网络以增强模型的记忆性^[78]和处理时序信息的能力，同时还将深度图像做了额外的归一化处理，将所有深度值大于20m的像素点的值均设为20，然后将其归一化到[0,255]的像素空间，这样做可以有效提高实现自主导航功能的成功率。

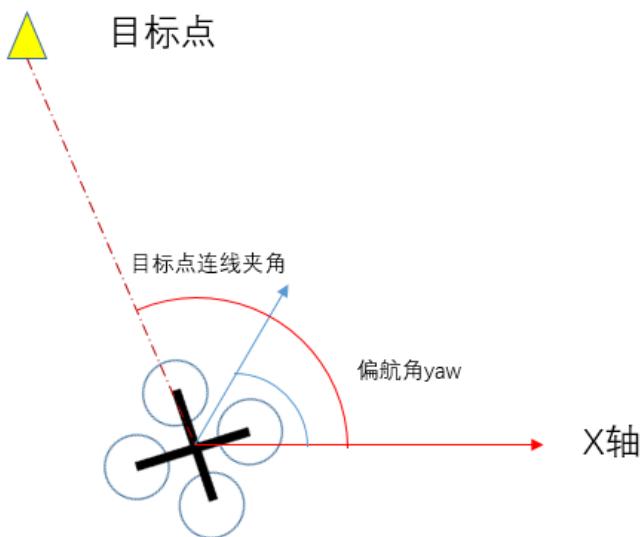


图 3.3 飞机朝向与目标点连线的夹角等于目标点连线角度减去偏航角

3.1.2 动作空间设计

基于本文的仿真系统中无人机的动作指令集，同时为保证无人机运动过程中的运动连贯性，设计了以下两种无人机自主导航任务的动作空间：10维离散动作空间以及三维连续动作空间。同时，仿真环境中动作执行频率设置在10到15 hz的范围内，这个执行频率下两种动作空间的动作组合都足以使得无人机可以有效执行各种避障飞行动作，以便完成导航任务。

无人机机体坐标系如图3.4所示，三维连续动作空间为xyz三个轴向的速度，其速度范围限制在[-1.5m/s, 1.5m/s]。为了使离散策略能够产生更合理的控制效果，相比于直接设置“向前飞行”，“向左飞行”等指令，本文的10维离散动作空间参考了

Wang^[78] 在训练无人车时所设计的上层算法的动作集，将飞机的角速度与线速度进行组合，并且针对无人机的特性设计了以下动作：

- (1) 以 $+15^{\circ}/s$ 的角速度绕 z 轴旋转
- (2) 以 $-15^{\circ}/s$ 的角速度绕 z 轴旋转
- (3) 以 $1m/s$ 的速度沿 x 轴方向向前飞行
- (4) 以 $1.5m/s$ 的速度沿 x 轴方向向前飞行
- (5) 以 $1m/s$ 的速度沿 x 轴方向向前飞行的同时，以 $+15^{\circ}/s$ 的角速度绕 z 轴旋转形成弧形轨迹
- (6) 以 $1.5m/s$ 的速度沿 x 轴方向向前飞行的同时，以 $+15^{\circ}/s$ 的角速度绕 z 轴旋转形成弧形轨迹
- (7) 以 $1m/s$ 的速度沿 x 轴方向向前飞行的同时，以 $-15^{\circ}/s$ 的角速度绕 z 轴旋转形成弧形轨迹
- (8) 以 $1.5m/s$ 的速度沿 x 轴方向向前飞行的同时，以 $-15^{\circ}/s$ 的角速度绕 z 轴旋转形成弧形轨迹
- (9) 以 $0.5m/s$ 的速度沿 z 轴方向向上飞行
- (10) 以 $-0.5m/s$ 的速度沿 z 轴方向向下飞行



图 3.4 无人机机体坐标系示意图, 蓝色代表 x 轴向, 黄色代表 y 轴向, 红色代表了 z 轴向

三维连续动作空间的学习难度较大，这是由于无人机在执行任一动作时前视相机的方向均与飞机速度方向对齐（这样的设置是为避免因获取不到障碍物信息而出现碰撞的情况），而决策模型在学习初期输出的动作具有较大随机性，导致了三维连续动作空间下，无人机在训练初期无法实现连续的运动轨迹，出现频繁的调头和抖动，导致初期难以获得有效样本进行学习，并且会使得无人机更加容易因抖动而产生碰撞造

成任务失败。因此，本文后续的各种类型算法的动作空间均设计成上述的 10 维离散动作空间。

3.1.3 奖励函数设计

奖励函数的设计是本文算法能否有效学习到自主导航功能的重点。为了使得无人机以较短时间尽快飞行至目标点，在奖励函数的设计中需要奖励高速飞行行为，同时惩罚不必要的、不及时的避障行为与碰撞行为。由于两点之间线段最短，如果无人机与目标点之间不存在障碍物，那么直线高速飞行的行为最优；如果中间存在障碍物，那么在执行避障行为时，快要接近障碍物再调头绕行的避障动作显然是耗时和低效的，高效的避障行为应该在障碍物出现在视野中的一定程度时，小范围调整转角以较小的曲率飞行实现避障，但是某些情况下调头行为也是必须存在的。因此奖励函数会严格惩罚碰撞状态，并且发生碰撞后会结束本次采样，同时轻微惩罚耗时的避障行为；奖励高效的飞行避障行为，并且当飞行至目标点时产生最大奖励。

综上所述，本文利用无人机与目标点的 xy 平面相对距离 $s_{rel} = \sqrt{p_x^2 + p_y^2}$ 、无人机朝向与目标点连线的夹角 ω 与飞机自身 x 轴向速度 v_x 来设计奖励函数，奖励函数的具体形式如下式所示：

$$r = \begin{cases} 20 & s_{rel} \leq 2m, \text{ 结束采样} \\ -20 & \text{发生碰撞, 结束采样} \\ -20 & \text{超过最大采样步数, 结束采样} \\ -20 & \text{飞机高度 } z > 5m \text{ or 飞机高度 } z < 1m, \text{ 结束采样} \\ -0.03s_{rel} + v_x \cos(\omega) & \omega \in [-\pi/2, \pi/2], \text{ 继续采样} \\ -0.03s_{rel} & \text{ohterwize, 继续采样} \end{cases} \quad (3.1)$$

由于本文假设目标点位于地面上，因此目标点的高度值默认为 0，而无人机的只需要在二维 xy 平面上的坐标位置抵达目标处即算完成任务。因此，式 (3.1) 中的第一种情况如图3.5所示。

至于最后两种情况的奖励函数设计也是符合上述分析的， $-0.03s_{rel} + v_x \cos(\omega)$ 包含两项：第一项用来惩罚无人机目前的位置，可以看到当无人机位于目标点 2m 之外时该项一直为负；第二项则是平衡避障与高速飞行的情况，当 ω 角度处于 $[-\pi/2, \pi/2]$

时，该项为正，且速度越大奖励越大，而其他角度时该项则为0，这就使得当飞机出现调头避障或者远离目标点飞行时由第一项产生轻微的负奖励进行惩罚，因此只有无人机以较小弧度进行避障行为时才会有正奖励，这就会迫使无人机在看见障碍时提前做出高效的避障行为以求最快飞行至目标点，其具体情况如图3.6和3.7所示。

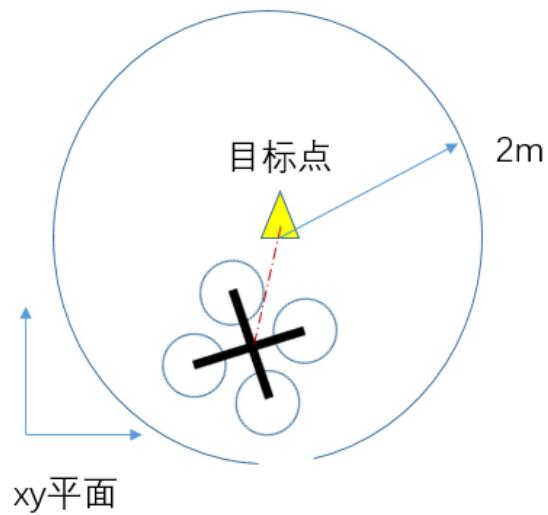


图 3.5 飞机完成任务时，xy 平面上的示意图

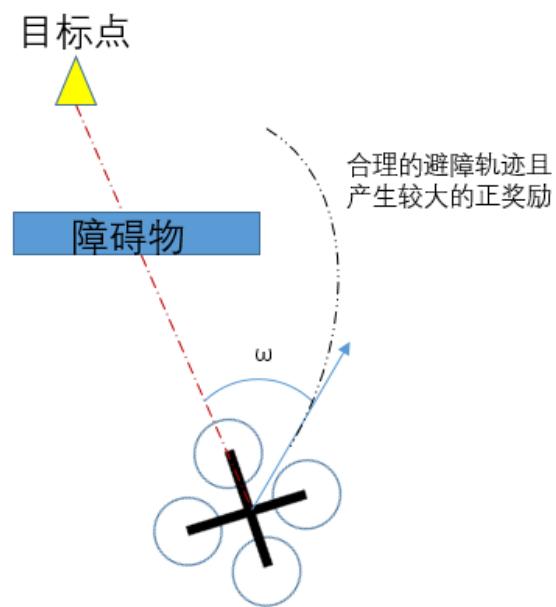


图 3.6 $\omega \in [-\pi/2, \pi/2]$ 时奖励合理的避障情况

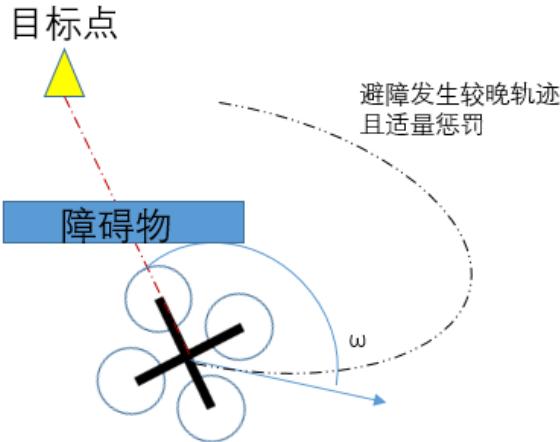


图 3.7 $\omega \notin [-\pi/2, \pi/2]$ 时避障发生较晚，应该适量惩罚

3.2 基于主流 RL 算法训练的无人机自主导航模型比较

第2章中介绍了强化学习的基础背景与 DQN 算法，这节主要介绍本章中用于训练自主导航任务的另外两种强化学习算法，以及具体训练配置与效果。

3.2.1 PPO 算法

PPO^[78] 算法与 TRPO^[79] 算法解决强化学习问题的思路是相同的：利用当前策略所在线采集的数据，在网络更新不会由于步长过大而导致策略性能崩溃的前提下，最大化提升策略性能，即解决一个受限条件下的最优化问题。TRPO 解决该问题利用了拉格朗日乘子法以及自然梯度下降法进行二阶近似求解，该过程十分复杂，PPO 则是直接利用一阶梯度信息进行优化求解，同时引入 KL 散度约束保证了新旧策略在分布上是相近的，PPO 算法的实现非常简便，同时其性能并不逊于 TRPO 算法。通常的 PPO 算法都指的是 PPO-Clip 算法，它不需要将 TRPO 的 KL 散度的约束条件代入目标函数中进行求解，而是通过巧妙的设计修剪（clip）损失函数，使得更新后的策略和旧策略在一定程度上保持近似。PPO 算法是一个在轨策略（on-policy）算法且可以同时处理离散与连续动作空间。PPO 算法通过梯度下降法优化以下损失函数来进行第 k 次参数更新：

$$\theta_{k+1} = \arg \max_{\theta} \mathbb{E}_{s, a \sim \pi_{\theta_k}} [L(s, a, \theta_k, \theta)] \quad (3.2)$$

θ_k 为网络第 k 次更新的具体值，其中损失函数 L 的形式为：

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_\theta(a | s)}{\pi_{\theta_k}(a | s)} A^{\pi_{\theta_k}}(s, a), \text{clip} \left(\frac{\pi_\theta(a | s)}{\pi_{\theta_k}(a | s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s, a) \right) \quad (3.3)$$

上式中 ϵ 是超参数，代表了新旧策略之间可接受的差异范围， A 为优势函数，可以通过价值函数拟合进行计算。那么，PPO 算法就可以分成以下两种情况进行讨论：

(1) 对于一个状态动作对 (s, a) 来说，如果优势函数 $A^{\pi_{\theta_k}}(s, a)$ 是大于 0 的，那么此时损失函数为：

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_\theta(a | s)}{\pi_{\theta_k}(a | s)}, (1 + \epsilon) \right) A^{\pi_{\theta_k}}(s, a) \quad (3.4)$$

为了增加 L 的值， θ 优化后应使得 $\pi_\theta(s, a)$ 的值增加，但是一旦 $\pi_\theta(s, a) > (1 + \epsilon)\pi_{\theta_k}(s, a)$ ，那么对于该样本对所产生的损失就变为了 $(1 + \epsilon)A^{\pi_{\theta_k}}(s, a)$ 不参与计算梯度，即这个状态动作对 (s, a) 对本次策略的梯度更新不产生影响，也就是说从优化过程上限制了不会因为一次更新而出现过于激进的策略，使得新旧策略差异过大。

(2) 同理如果优势函数 $A^{\pi_{\theta_k}}(s, a)$ 是小于 0 的，那么此时损失函数为：

$$L(s, a, \theta_k, \theta) = \max \left(\frac{\pi_\theta(a | s)}{\pi_{\theta_k}(a | s)}, (1 - \epsilon) \right) A^{\pi_{\theta_k}}(s, a) \quad (3.5)$$

为了使得 L 增加， θ 优化后应使得 $\pi_\theta(s, a)$ 变小，但是一旦 $\pi_\theta(s, a) < (1 - \epsilon)\pi_{\theta_k}(s, a)$ ，那么对于该样本对所产生的损失就变为了 $(1 - \epsilon)A^{\pi_{\theta_k}}(s, a)$ ，即这个状态动作对 (s, a) 对本次梯度更新也不产生作用。这种修剪操作从一定程度上就类似于 KL 散度所带来的正则化效果，使得策略更新过程不会过于剧烈，同时又保证了每次更新后的策略都是优于旧策略的。但是上式中的 θ 在没有进行梯度更新计算之前，是无法对其策略进行估计的，所以 PPO 算法在工程实现上用了多阶段近似，在第一阶段用当前网络的 θ_k 来近似 θ ，然后将一次梯度优化分割为多阶段，每一阶段都对 θ 进行少量样本更新，同时利用上一次更新后的 θ 来近似计算本阶段的损失，但是 θ_k 在一轮总的更新中不会变化，因此 PPO 的实现相比于 TRPO 更侧重于工程编程的技巧^[80]，具体流程如算法3所示。

Algorithm 3: PPO 算法

Input: 学习率 α , 序列样本数 b

- 1 初始化: 当前策略网络的参数 θ , 值网络网络的参数 ϕ , 更新总次数 K , 当前更新次数 $k = 0$
- 2 **while** $k <= K$ **do**
- 3 通过执行策略 $\pi_k = \pi(\theta_k)$ 收集完整的序列样本 $\mathcal{D}_k = \{\tau_i\}, i = 1..b$;
- 4 基于当前的值网络 V_{ϕ_k} , 利用 GAE 算法计算优势价值函数 \hat{A}_t ;
- 5 利用 Adam 算法最大化 PPO 目标, 更新网络参数:
- 6
$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \text{clip} \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s_t, a_t) \right)$$
- 7 更新值网络参数: $\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2$;
- 8 $k += 1$.
- 9 **end**

3.2.2 SAC(Soft Actor Critic) 算法

SAC 算法^[81]不同于 PPO 算法, 它本质上是一种离轨策略 (off-policy) 算法, 所以有着更高的样本效率, 在工程实现时, 不要求在多个并行环境中同时采样, 因此它的应用范围更加广泛。严格意义上来说, SAC 并不是从 PG 的算法框架下推导出来的, 而是对 soft-Q learning 算法^[82]的近似与扩展。SAC 算法的理论基础是最大熵强化学习, 这使其可以有效应对多模态目标 (multi-modal objective) 任务, 并且 SAC 算法结合了能量函数与最大熵理论, 学得的最优策略是最大熵约束下的随机策略, 使 SAC 算法训练的模型在面对未知环境时有更强的探索能力, 以及抗干扰能力, 而且可以作为多种任务的预训练模型。同时相比于 DDPG^[38]等算法, SAC 对超参数的设定并不敏感, 使得 SAC 算法在工程中更易于实现。

熵代表了一个随机变量的随机化程度。如果 x 是一个随机变量, 其概率分布为 P , 那么 x 的熵 H 为:

$$H(P) = \mathbb{E}_{x \sim P}[-\log P(x)] \quad (3.6)$$

在最大熵强化学习算法的设定中, 智能体所获得的奖励其中一部分与当前策略的熵成

正比，那么强化学习问题就变成了：

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{a \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t (R(s_t, a_t, s_{t+1}) + \alpha H(\pi(\cdot | s_t))) \right] \quad (3.7)$$

上式中 $\alpha > 0$ 是平衡系数。相比原本的 RL 算法，在奖励项中多加入了熵值项，使得优化过程中策略在最大化累计收益的同时，也在最大化策略的熵值累积。这个优化目标既可以从 Soft-Q learning 的能量函数与概率图模型中推出，也可以视为统计建模中的熵正则化方法。并基于此可以重新定义价值函数 $V^\pi(s)$ 和 $Q^\pi(s, a)$ ：

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t (R(s_t, a_t, s_{t+1}) + \alpha H(\pi(\cdot | s_t))) \mid s_0 = s \right] \quad (3.8)$$

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) + \alpha \sum_{t=1}^{\infty} \gamma^t H(\pi(\cdot | s_t)) \mid s_0 = s, a_0 = a \right] \quad (3.9)$$

状态价值函数与动作价值函数之间的关系为：

$$V^\pi(s) = \mathbb{E}_{a \sim \pi} [Q^\pi(s, a)] + \alpha H(\pi(\cdot | s)) = \mathbb{E}_{a \sim \pi} [Q^\pi(s, a) - \alpha \log \pi(a | s)] \quad (3.10)$$

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim P(s' | s, a)} [R(s, a, s') + \gamma V^\pi(s')] \quad (3.11)$$

那么，关于 Q^π 的贝尔曼方程为：

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim P, a' \sim \pi} [R(s, a, s') + \gamma (Q^\pi(s', a') + \alpha H(\pi(\cdot | s')))] \quad (3.12)$$

基于以上公式，可以推出用下式可代替贪婪方法有效执行策略改进：

$$\tilde{\pi}(\cdot | s) \propto \exp(Q^\pi(s, \cdot)), \forall s \quad (3.13)$$

其证明过程详见附录A，其中 $\tilde{\pi}(\cdot | s)$ 为利用 $(Q^\pi(s, \cdot))$ 改进后得到的策略，同时经过不断的策略迭代最终可收敛到如下最优策略：

$$\pi^*(a_t | s_t) = \exp \left(\frac{1}{\alpha} (Q^*(s_t, a_t) - V^*(s_t)) \right) \quad (3.14)$$

SAC 算法就是通过以上 Soft Q-learning 的理论来近似求解最优策略的。其中, Critic 部分利用神经网络分别拟合 $V^\pi(s)$ 和 $Q^\pi(s, a)$, 可以利用式 (3.10) 和 (3.11) 分别进行迭代训练。Actor 部分并没有像 Soft Q-learning 那样处理, 而是利用一个新的网络来拟合策略, 该策略可以从高斯分布或者任意噪声分布中采样, 同时在策略迭代的过程中通过优化 KL 散度保证了该策略分布是对最优策略的逼近, 对其参数 θ 的优化如下式:

$$L_\pi(\theta; s_t) = D_{\text{KL}} \left(\pi_\theta(\cdot | s_t) \| \exp \left(\frac{1}{\alpha} (Q(s_t, \cdot) - V) \right) \right) \quad (3.15)$$

本文所实现的具体 SAC 算法是其扩展到离散动作空间的版本, 同时利用了 TD3^[83] 和 Double Q-learning^[69] 算法的思想设置了双 Q 网络, 来进一步提升算法性能, 具体算法如 4 所示。

Algorithm 2: SAC 算法

Input: 学习率 α , 目标网络更新步长 ρ , 批数据大小 b

- 1 初始化: 当前策略网络的参数 θ , Q 函数网络参数 ϕ_1, ϕ_2 , 设置目标网络参数 $\phi_{\text{targ}, 1} \leftarrow \phi_1, \phi_{\text{targ}, 2} \leftarrow \phi_2$, 经验回访池 $D = \emptyset$, 更新总次数 K , 当前更新次数 $k = 0$
 - 2 **while** $k <= K$ **do**
 - 3 收集当前状态 s , 根据策略选择动作 $a \sim \pi_\theta(\cdot | s)$ 并执行;
 - 4 得到下一时刻状态 s' , 奖励值 r , 当前 episode 结束标志 d , 将 (s, a, s', r, d) 保存至 D ;
 - 5 从 D 中随机采样一批数据 $B = \{(s, a, r, s', d)\}$ 用于更新:
 - 6 计算 Q 网络的标签值:
 - 7 $y(r, s', d) = r + \gamma(1 - d) (\min_{i=1,2} Q_{\phi_{\text{targ}, i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}' | s'))$, $\tilde{a}' \sim \pi_\theta(\cdot | s')$
 - 8 梯度更新 Q 网络:
 - 9 $\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s, a, r, s', d) \in B} (Q_{\phi_i}(s, a) - y(r, s', d))^2, i = 1, 2$
 - 10 更新策略网络, 此步会用到重参数化技巧^[84]:
 - 11 $\nabla_\theta \frac{1}{|B|} \sum_{s \in B} (\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s) | s)), \tilde{a}_\theta \sim \pi_\theta(\cdot | s)$
 - 12 更新目标 Q 网络: $\phi_{\text{targ}, i} \leftarrow \rho \phi_{\text{targ}, i} + (1 - \rho) \phi_i, i = 1, 2$
 - 13 $k += 1$.
 - 14 **end**
-

3.2.3 模型设计与算法比较

基于3.1中所设计的状态以及动作空间，本文为 DQN、PPO 以及 SAC 算法设计了相似的网络结构以方便对比三种算法在自主导航任务上表现情况的优劣，其中网络模型的特征提取部分均为图3.8所示。

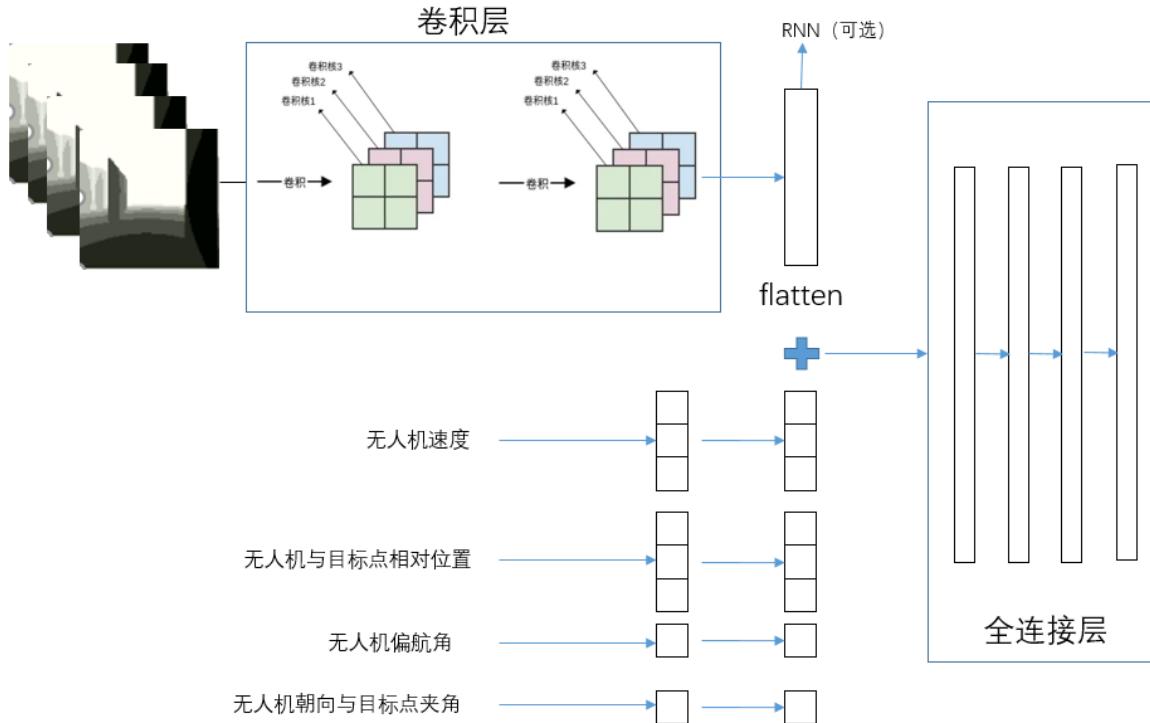


图 3.8 网络特征提取部分

特征提取部分设置了两条支路，第一条支路由 CNN 网络搭建，详细设计如图3.9所示。其中输入深度图像的尺寸为 $112 \times 112 \times 4$, 4 通道由当前时间步加上前三帧深度图组成，然后经过 4 层卷积网络后得到 $5 \times 5 \times 32$ 的特征图，然后将其展开 (flatten) 得到 800 维图像特征向量。该 CNN 部分用于学习深度图像的状态信息所包含的各种环境特征；第二条支路为融合连续特征与图像特征的全连接网络，连续特征是由图3.8中的 4 类状态信息拼接而成的 8 维向量，包括了无人机当前的三轴线速度、无人机与目标点的相对位置信息、无人机当前偏航角和无人机机头方向与目标点之间连线的夹角。特征拼接后再将该信息通过 3 层全连接网络进行特征的进一步抽象，每层全连接网络输出数目分别是 512->256->128。

以下为 DQN、PPO、SAC 算法在该任务上的细节配置：

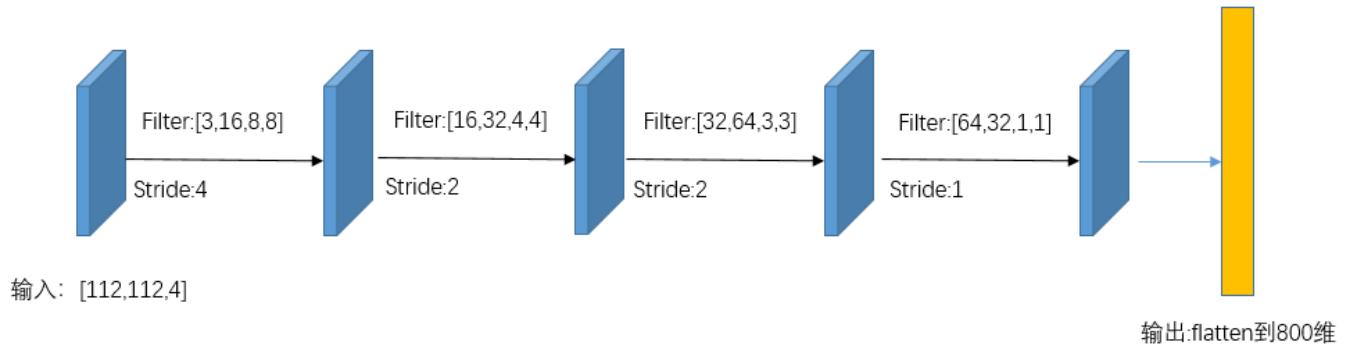


图 3.9 CNN 部分结构图

(1)DQN 算法

对于 DQN 算法来说，其 Q 函数模型的特征提取部分如图3.8所示，在全连接网络之后设计对应动作空间的 10 维动作价值输出值，如图3.10：

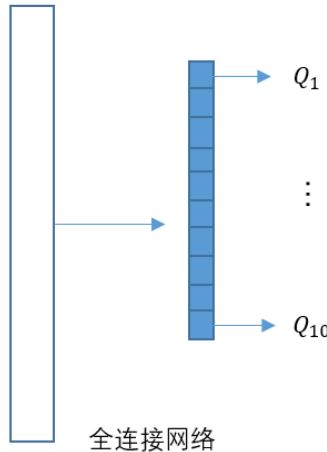


图 3.10 DQN 网络设计

输入当前状态后，每一维输出都分别对应了该离散动作所对应动作价值函数，即 $Q^\pi(s, a)$ ，参数更新形式如式 (2.17) 所示。更新时 batch_size 大小为 256；利用 Adam 算法对参数进行优化，学习率设置为 0.001，平滑系数分别为 0.9 和 0.999；用于训练的总样本数为 300000，回放池（replay buffer）大小为 30000；单步更新；目标网络（target-net）更新频率为 10000。

(2)PPO 算法

PPO 算法是基于 AC 框架的，本文的设置共享了 Actor 与 Critic 端的特征提取层，同样如上图3.8所示。然后分别设计了 Actor 产生策略的网络以及 Critic 产生状态价

值函数所对应网络，如下图3.11。Actor 策略端同样有 10 维输出，然后这些输出通过

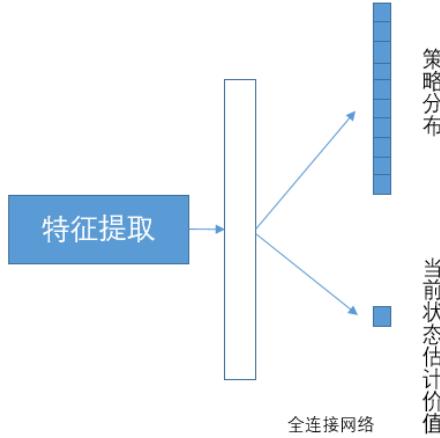


图 3.11 PPO 网络模型

softmax 技巧进行归一化，生成一个离散分布，代表当前的策略分布。这样做的原因，是智能体的动作往往是从策略分布中采样得到的，而直接采样操作通常无法求导，引入 *softmax* 操作和重参数化方法就可以方便的对该过程求导（同理连续动作空间也可以设置对应的高斯分布，然后利用重参数化方法求导）。Critic 部分直接输出一维连续值，该值就代表了对的当前状态进行估计的状态价值函数 $V^\pi(s)$ ，并且在进行值函数估计时，使用了 GAE 算法^[85] 来计算优势函数，进一步减小方差。训练损失函数形式如式 (3.3) 所示。更新时在线采集样本数 (rollout num) 为 1024，更新重复周期 (epoch) 为 4，PPO 每次更新周期的阶段数为 4，对应单次梯度更新 batch_size 大小为 256；利用 Adam 算法对参数进行优化，学习率设置为 0.001，平滑系数分别为 0.9 和 0.999；用于训练的总样本数为 300000。

(3)SAC 算法

SAC 算法同样是基于 AC 框架的，但是本文并未共享特征提取部分，而是设计了同样的结构分别学习各自的特征，其具体配置如图3.12所示。Actor 端的策略生成方式与 PPO 算法相同，Critic 部分的输出与 Q-learning 算法中值网络的设计方式一样，更新损失函数如式和式所示。更新时 batch_size 大小为 512；利用 Adam 算法对参数进行优化，学习率设置为 0.001，平滑系数分别为 0.9 和 0.999；用于训练的总样本数为 300000，回放池 (replay buffer) 大小为 30000；更新频率为 64，更新重复周期为 8；目标网络 (target-net) 更新频率为 10000。

(4) 对比实验

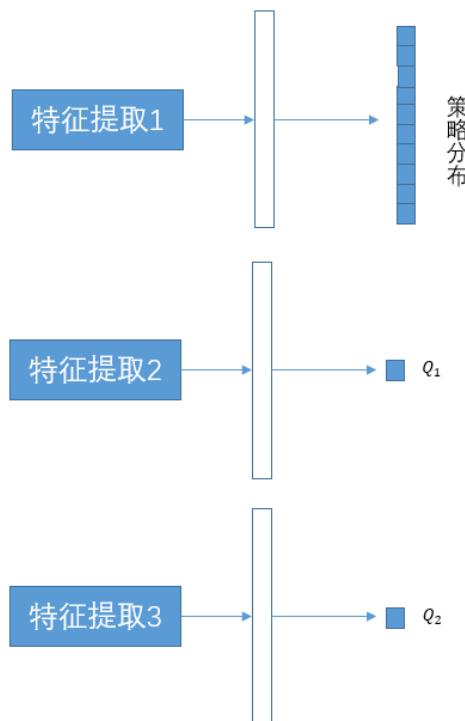


图 3.12 PPO 网络模型

三种算法分别运行在本文设计的仿真环境中，其具体系统已在2.2中介绍，本节中用于三种算法性能对比的训练环境是 $50m \times 50m$ 的室内环境，随机生成 25 个障碍物，障碍物种类设置为 4 种，且障碍物之间的最小距离为 0.5m，目标点随机初始化，环境最大采样步数为 1024，仿真环境效果如图3.13所示：

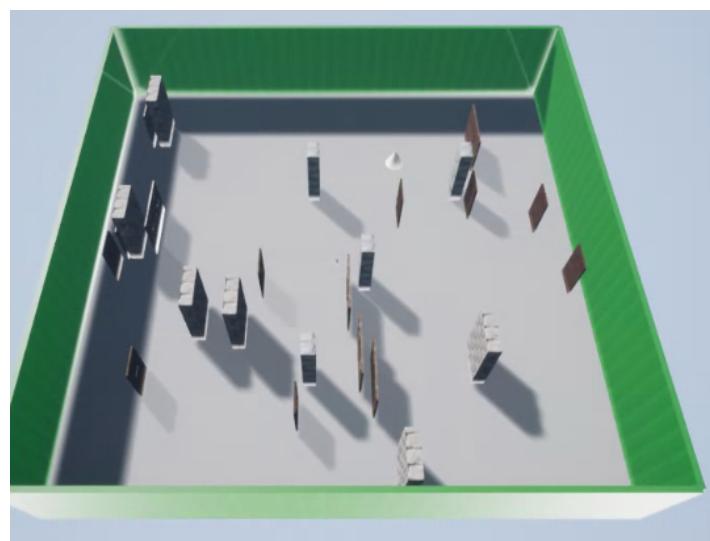


图 3.13 仿真环境示意图

基于本文设计的奖励函数，若直接利用一次采样周期的累积奖励和作为算法性能

衡量指标可能不能准确反映算法性能，因为当随机目标点离无人机初始位置较远时，初期朝向目标点飞行累积较高奖励，最终即使产生碰撞累积奖励和也会较大，这种情况衡量算法性能可能会有偏颇。因此，本文设计了导航成功率作为算法性能的衡量指标，导航成功率是连续 100 幕采样（episode）所计算出的成功率，这样设计的指标并不会受到某一次较差采样的影响，又不会出现上述累积奖励和的问题。DQN、PPO 以及 SAC 三种算法分别运行 5 个随机种子，最终的实验结果取平均值，同时计算方差，在上述环境中训练得到的成功率对比图如图3.14所示。

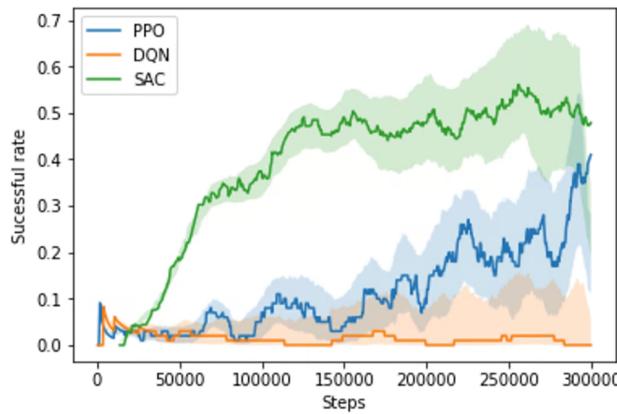


图 3.14 DQN、PPO、SAC 算法对比图，阴影部分为实验方差

从图中可以看出，DQN 算法的性能有限，在复杂的自主导航任务中并不能学到有效的策略；本文的训练设置都是在单机情况下，且由于 UE4 的特性，无法在单机环境中开启多并行环境（需要配置局域网多机实现并行环境），所以 PPO 算法在该任务中由于较低的样本效率，也无法学到较好的行为；而 SAC 算法虽然能够学到简单的避障行为，但是其最终收敛的成功率并不高，学到的策略性能较低。但是 SAC 相比于其他两种主流的强化学习算法，对于该任务有着更好的表现，因此下文也基于 SAC，并针对上述问题提出了改进的 CLSA-SAC 算法。

3.3 改进的 CLSA-SAC 算法

3.3.1 基于课程学习方法训练 SAC 模型

通过3.2节的对比实验，可以看出原始的 SAC 算法，在该任务上的学习效率较低，收敛后的性能也较为一般。经过分析发现，在训练后期大部分失败的样本都发生在目标点与初始无人机位置较远的情况下，由于障碍物较多且位置复杂更加容易产生碰撞行

为，而目标点与初始无人机位置距离较近或无障碍物时，则能通过简单的避障行为快速到达目标点，成功率较高。因此，低效的学习过程主要是由于自主导航任务难度较大，无法收集到更多有效样本帮助网络学习到关键知识，因此导致了学习过程较为缓慢。而解决这个问题的关键就在于如何在减少任务难度，提升样本效率。针对上述问题，解决方法有：课程学习方法，以及基于内在奖励的高效探索方法。基于内在奖励函数的方法^[86-88]通常是在模型训练过程中，基于启发式信息训练另一内在奖励函数模型指引策略增大前期探索行为，收集更多有效样本，增加探索效率以学习到困难任务。但是该方法通常是针对稀疏奖励环境设计，由于 SAC 函数建立在最大熵体系下，本身就有比较好的探索性质，如图3.15所示。因此本文设计了课程学习过程来帮助 SAC 算法提升学习性能，在相同训练样本数量下取得更好的学习效果。



图 3.15 此图中的情景下，SAC 算法在初期向左和向右飞行的概率是大致相同的，这就使得前期的探索性能大大增加

课程学习的思想就是先学习简单概念，然后步进式地学习到复杂问题，让模型像人一样逐步学会某一困难知识，课程学习的好处就是对于复杂任务而言，可以有效增加模型收敛速度以及最终的性能表现（几乎不会下降）。最早在 1993 年，Elman^[89]就提出了用一种递进式的方法训练神经网络，他的实验证明了通过逐渐增加样本难度可以使模型逐渐学会一种复杂的语法模型，而不设计课程学习过程模型则学不到任何知识。人为设计的课程学习过程也叫做任务确定型课程（task-specific curriculum），Bengio^[90]通过简单实验也证明了人为地逐步增加模型训练样本的难度，可以加快模型学习速度以学到更复杂的知识。Zaremba^[91]通过 LSTM 网络学习利用 Python 进行数学逻辑化编程也证明了课程学习过程对于复杂任务的学习是必要的，并且在训练过程中混合简单任务样本对于防止模型发生遗忘是必要的。目前较为流行的程序化内容

生成 (PCG)，是一种用于创建不同难度级别的视频游戏方法，也被用于训练具有较强泛化能力的强化学习模型。但是设计一个好的课程学习过程却不容易，一种不合理的课程设计可能会导致模型学习过程受阻甚至无法学习。一个合理的课程学习过程通常包含两个步骤，设计一种方法量化样本的复杂程度以及利用不同难度的任务样本去训练模型。Weinshall^[92] 通过利用在另一相关任务上预训练的模型，计算其在该任务样本上的损失值，并将损失值作为样本复杂程度的量化，这样的方法同时会将预训练模型的先验知识通过样本层级划分迁移到本任务模型训练中，加速模型收敛。OpenAI 设计了一种域随机化方法 (ADR)^[93] 逐步加大生成环境的复杂程度来完成课程学习，而随机化分布的复杂程度就可以看成一种量化指标。由于本文设计的仿真环境同样具有域随机化功能，因此本文也基于上述思想对自主导航任务人为设计了以下两种课程学习过程：

(1) 先学习向目标点飞行，后学习避障。这种方式的具体设计如图3.16。整个课程学习过程中， $50m \times 50m$ 的环境大小固定，目标点位置在该范围内随机初始化。任务初期阶段并没有障碍物出现，希望无人机可以先学会导航任务，即朝向目标点直接飞行；在任务中期逐渐增加障碍物数量，在 10 到 17 范围内随机生成，希望无人机在学会向目标点飞行后，再学习到一些简单的避障策略；在任务后期，障碍物数量在 15 到 30 范围内随机生成，希望无人机可以学习到最终的复杂任务。



图 3.16 先学习向目标点飞行，后学习避障的课程设计

(2) 同时增加环境大小与障碍物数量，这种方式的具体设计如图3.17。整个课程学习的过程中，环境范围和障碍物逐步增加。任务初期，环境大小在 $15m \times 15m$ 到 $25m \times 25m$ 范围内随机生成，障碍物数量在 8 到 13 范围内随机生成，因为环境范围较小，所以目标点生成位置都距离无人机初始位置较近，通过 SAC 的探索行为可以有效地学习到简单任务情况的导航与避障行为；任务中期，将环境扩大至 $20m \times 20m$ 到 $35m \times 35m$ 的范围，同时障碍物数量的范围增加至 13 到 22 个，这时环境会出现更加复杂的情况，无人机此时可以收集到更多有效样本进行学习；任务后期，环境大小的

范围在 $30m \times 30m$ 到 $55m \times 55m$, 障碍物数量在 20 到 30 范围内随机生成。



图 3.17 同时增加环境大小与障碍物数量的课程设计

这两种方法都人为地将整个自主导航任务的学习过程划分成三阶段，并且按照^[91]中的思想为防止模型发生遗忘，相邻两阶段的环境配置中的目标点随机生成的区域均有重叠部分。仿真环境的配置环境文件可以直接按照上述内容生成所期望的环境，同时保持了环境的域随机化。三个阶段是按照训练成功率依次进行学习的，开始后直接进入第一阶段，当成功率上升至 0.6 进入下一阶段，当成功率达到 0.7 则进入最终阶段。训练的总样本数为 300000 时，两种方法的训练成功率对比图如3.18所示，每组实验分别运行了 5 个随机种子，最终的实验结果取平均值，方差为阴影部分。

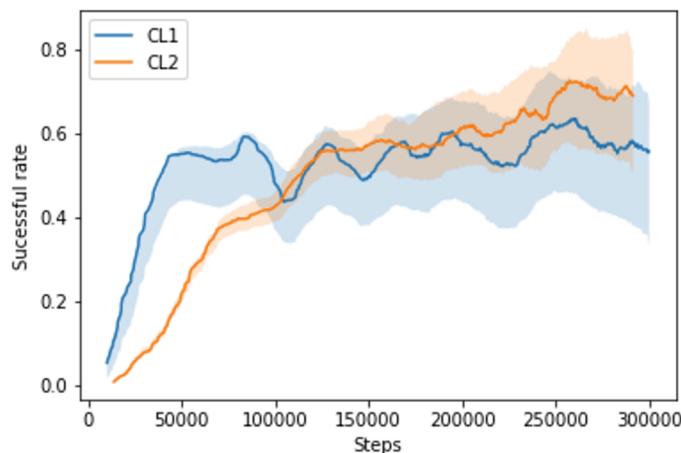


图 3.18 两种课程设计的实验对比，CL1 代表了先学习向目标点飞行，后学习避障的课程设计，CL2 代表了同时增加环境大小与障碍物数量的课程设计

从图中可以看出两种课程学习设计相比于原始的 SAC 算法，都能够有效加快模型的学习速度。但是第一种设计方法最终只进入了第二阶段就停止了，学习成功率最高也只有 0.6 左右，很难再有进一步提升，而第二种设计则最终进入到了第三阶段，同时学到了更有效的导航行为，相比于原始的 SAC 算法成功率提升到了 0.7 左右。

3.3.2 基于自注意力机制提高避障性能

利用课程学习的方法有效地提高了自主导航任务中 SAC 算法的训练效率，但是最终收敛的成功率还是略低。通过观察多组测试飞行过程可以发现，几乎所有失败的任务均是碰撞引起的，而不是由于超过最大限定步骤。这也就意味着网络虽然没有陷入局部最优解，在一定程度上学习到了导航策略，但是因为 CNN 架构是基于局部感受野提取特征的，对于视觉图像中包含的障碍物信息并没有做进一步处理，无法关注到障碍物细节，导致避障效果一般。但是人在执行避障行为时，大脑处理视觉皮层的信息会优先关注障碍物的大小和形状，甚至忽略周围并不重要的信息以免发生碰撞，也就是说人类的视觉机制会自动关注到某一重要信息上，这一过程也被称为注意力机制^[94]。因为 CNN 网络是基于局部感受野进行视觉特征提取的，同时 CNN 提取到的特征图对后续网络来说关注程度是相同的，并没有设置额外的网络层增加特定区域的权重，因此无法帮助模型能够更加关注到某一区域上的信息，而注意力机制则可以实现上述过程。注意力机制目前被广泛应用在自然语言处理与计算机视觉任务中^[95,96]，广义上的注意力机制可以被认为是一种隐含相关信息的重要性权重，在预测或推理某一元素时（例如，图像中的一块像素区域或者句子中的一个词语），可以通过注意力权重来表示该元素与其他元素之间的相关性，并可以利用其他元素的注意力加权和来对该元素进行一个大致表示。

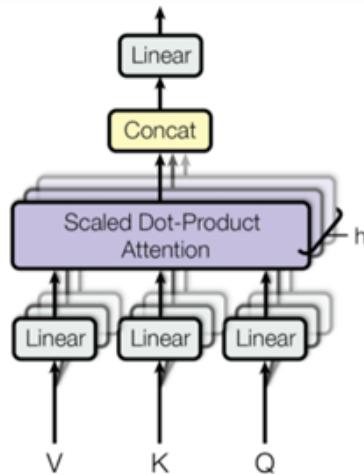
最早的注意力机制是为了克服 NLP 中“序列到序列”（seq2seq）模型无法应对长句子的缺点而设计的，相比于直接利用编码器最后输出的隐层向量作为解码器的上下文向量，注意力机制则是直接对每一个解码器的输出向量和编码器的每个隐层向量建立了直接关系，由编码器的隐层向量的加权和构成了上下文向量，而这些连接权重就是需要注意力机制生成的^[97]。生成这种权重向量的方法有许多种，如直接向量点乘计算、基于余弦相似性计算、基于附加向量与双曲正切函数计算、基于注意力层中可训练的额外权重矩阵计算等方法，这些方法通常都会将注意力权重进行额外的归一化操作。而自注意力机制（Self-Attention）是从上述思想中衍生出来的，也被称为数据内部注意力（intra-attention），它是一种特殊的注意力机制，可以将单个序列中不同位置的元素进行关联，计算序列内部各个元素之间的相关性并且可以获得一种序列表示^[98]，相关性图像如图3.19所示。自注意力机制目前被广泛应用到机器阅读，内容抽象和图像内容理解等领域。

Transformer^[99] 架构更加发挥了自注意力机制的优势，并取代了循环网络（RNN）

The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .

图 3.19 自注意力机制的权重示意图，图片引自^[98]

更好地为序列数据进行抽象建模。Transformer 计算注意力时提出了 Key,Value 和 Query 的概念，用于计算多头注意力。Key 和 Value 就相当于 RNN 建模时的上下文隐藏状态，Query 就相当于之前解码器中的输出向量，然后 Transformer 利用点乘计算自注意力，给到下层网络继续抽象特征，其计算过程如图3.20所示。

图 3.20 Transformer 计算自注意力的示意图，图片引自^[99]

那么从上述内容可以看出，注意力机制是一种全局信息抽象，可以在元素内部的每一位置都可以获得与全局元素的相关性信息，因此注意力机制就可以更好地帮助 SAC 模型关注到障碍物细节，学习更优的避障行为。基于此，本文为 SAC 算法模型设计了额外的视觉注意力机制层，该注意力机制层参考了 SAGAN^[100] 和 Transformer 的原理，在 CNN 层中加入额外的注意力机制层，可以显式地学习到当前像素位置与其他所有像素之间的相关性，比起原始的 CNN 层，注意力层可以帮助网络获得了更大的全局感受野，更加关注到障碍物形状和大小的细节部分，如图3.21所示。

本文的注意力机制层设计参考了全局注意力网络^[101]，假设 x 为卷积特征层，然

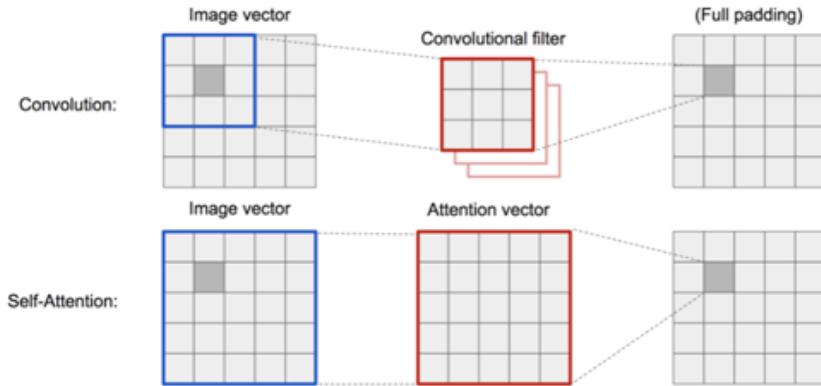


图 3.21 CNN 层配置自注意力机制可以更好地获取全局信息

后通过三个独立的 1×1 卷积分别计算 Key , $Value$, 和 $Query$, 即:

$$\begin{aligned} \text{Key: } f(x) &= W_f x \\ \text{Query: } g(x) &= W_g x \\ \text{Value: } h(x) &= W_h x \end{aligned} \quad (3.16)$$

上式中 W_f, W_g, W_h 代表了卷积核矩阵权重, 且将计算后的二维向量特征展开, 然后利用点乘计算注意力权重:

$$\alpha_{i,j} = \text{softmax} \left(f(x_i)^\top g(x_j) \right) \quad (3.17)$$

$$o_j = W_v \left(\sum_{i=1}^N \alpha_{i,j} h(x_i) \right) \quad (3.18)$$

上式中 $\alpha_{i,j}$ 是注意力权重, 代表了在第 j 个位置时应赋予第 i 个位置元素多少的关注度, 1×1 卷积核主要作用是不改变平面结构情况下对图形通道数进行降维, o_j 代表了第 j 个位置的输出向量, 计算流程如图3.22所示。注意力机制层的输出会乘上一个自适应系数后, 与原来的 CNN 特征图相加, 作为该层网络的输出, 即:

$$y = x_i + \gamma o_i \quad (3.19)$$

这种注意力机制层可以很方便地嵌入本文在3.2节所设计的网络模型中。

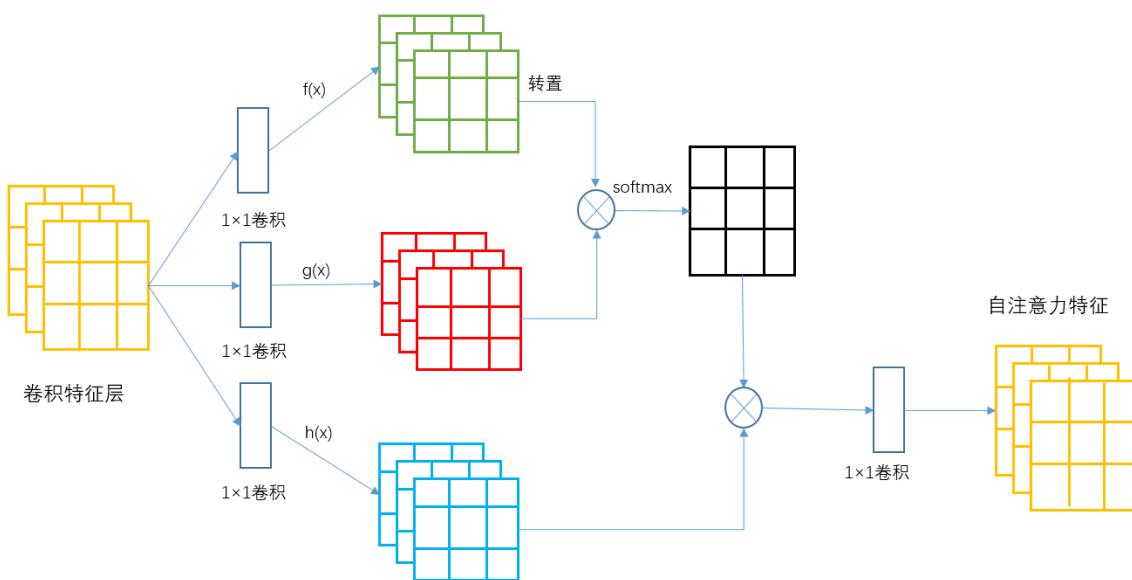


图 3.22 本文所设计的 CNN 层间的自注意力机制层

3.3.3 CLSA-SAC 算法

CLSA-SAC 算法就是利用上文所述的课程学习方法与注意力机制层来改进原有的 SAC 算法，整体的算法与模型图如图3.23所示。

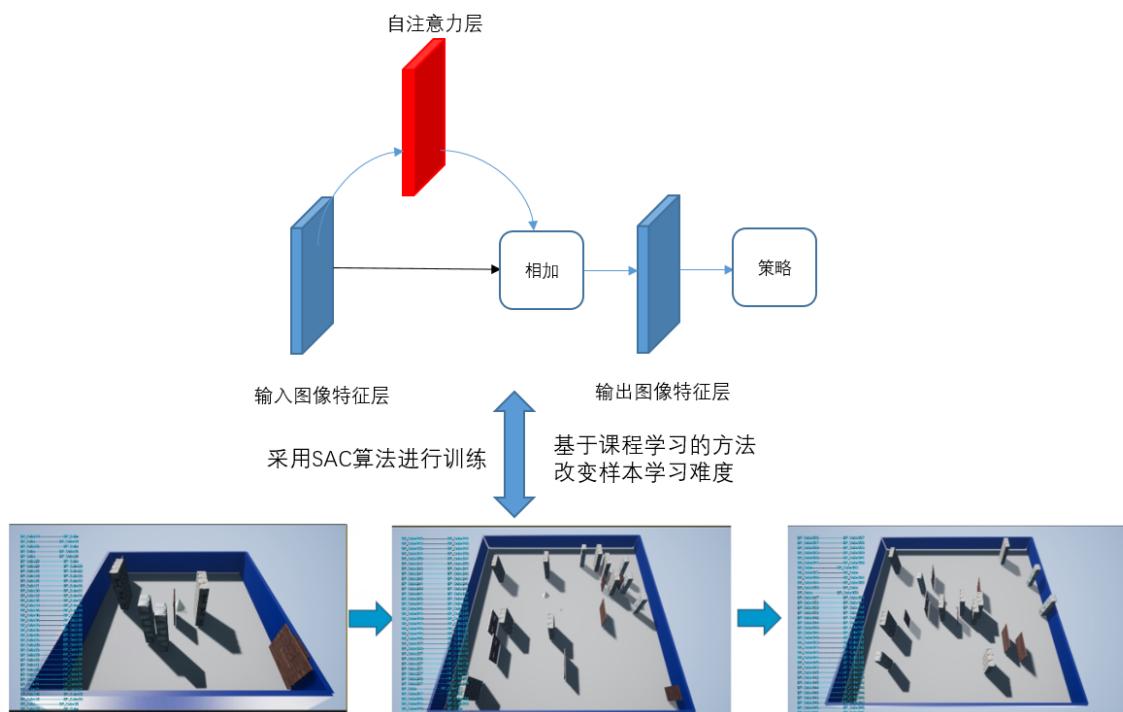


图 3.23 CLSA-SAC 算法结构图

训练时设置参数与 SAC 算法一致：更新时 batch 大小为 512；利用 Adam 算法对

参数进行优化，学习率设置为 0.001，平滑系数分别为 0.9 和 0.999；用于训练的总样本数为 300000，回放池（replay buffer）大小为 20000；更新频率为 64；注意力层参数 γ 自适应更新。CLSA-SAC 算法的消融实验对比如图3.24，分别和原始 SAC 算法以及 CL2+SAC 算法进行了对比，每组实验分别跑了 5 个随机种子，最终的实验结果取平均值，方差为阴影部分。从上图可以看出，本文所设计的 CLSA-SAC 算法能够有效地

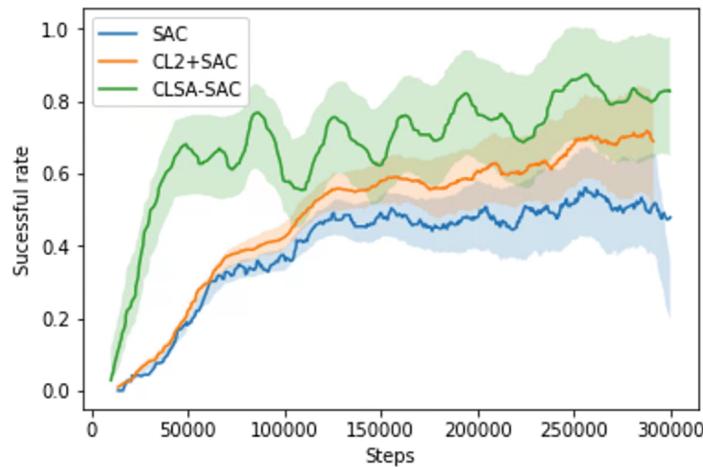


图 3.24 CLSA-SAC 算法的消融实验对比图

学习到自主导航的行为，且模型收敛后的成功率也提高到了 0.9 左右。结合图3.14可以看出，CLSA-SAC 算法在该问题上的训练成功率远超 PPO 算法以及 SAC 算法，进一步说明了本文算法的有效性。本文同样对算法进行了新环境下的测试，配置了以下两种与训练环境完全不同的新环境。两种环境均为 $50\text{m} \times 50\text{m}$ ，且障碍物形状与尺寸均与训练环境不同，如图3.25到3.27所示。



图 3.25 测试场景 1 的俯视图，黄色三角为无人机飞行起点，绿色三角为无人机目标导航点

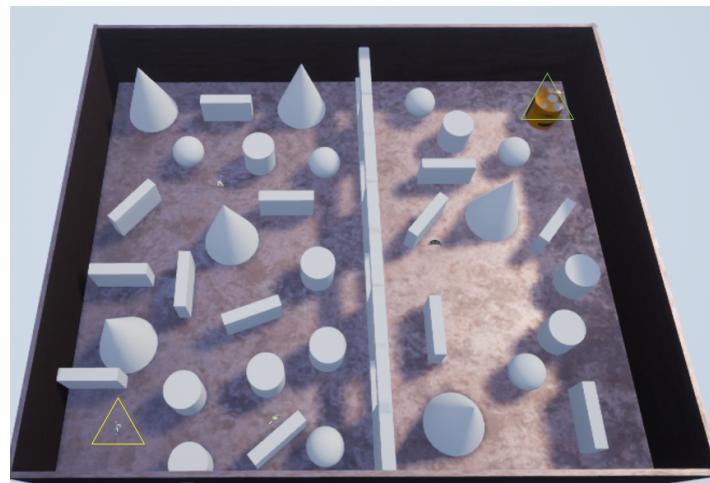


图 3.26 测试场景 2 的俯视图，黄色三角为无人机飞行起点，绿色三角为无人机目标导航



图 3.27 测试场景中的无人机飞行图以及包含深度信息的状态图

CLSA-SAC 算法训练后的自主导航模型均能在这两种新环境中控制无人机有效完成自主导航飞行任务，其飞行轨迹如图3.28和3.29所示。

另外本文也对自注意力层进行了可解释性分析，按照 Tang 等人设计^[102]的方法，将在每一像素位置处来自其余像素位置处的注意力权重加和计算总的权重值，将其视



图 3.28 测试场景 1 中的无人机飞行轨迹

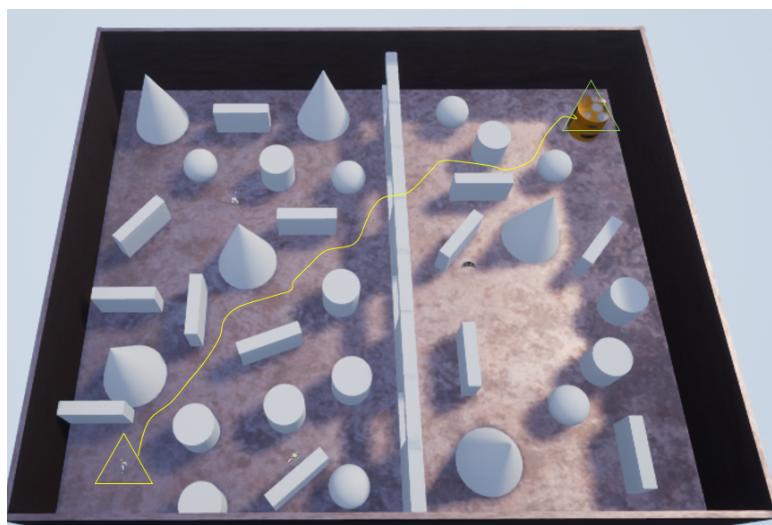


图 3.29 测试场景 2 中的无人机飞行轨迹

为每一像素在全局范围内的“影响力”：

$$\omega_i = \sum_{j=1}^N \alpha_{i,j} \quad (3.20)$$

上式中 ω_i 即为每个像素点的全局影响力权重， N 为总像素点个数， $\alpha_{i,j}$ 为式 (3.17) 中的注意力权重，然后按照特征图的尺寸重新排列后进行可视化，可以得到如图3.30所示的注意力可视化图像。

从图3.30中可以看出，在当前帧中出现的油桶型障碍物，自注意力机制层会赋予该区域内像素点很高的关注度，使得后续的模型在特征层面更加关注该部分区域，即

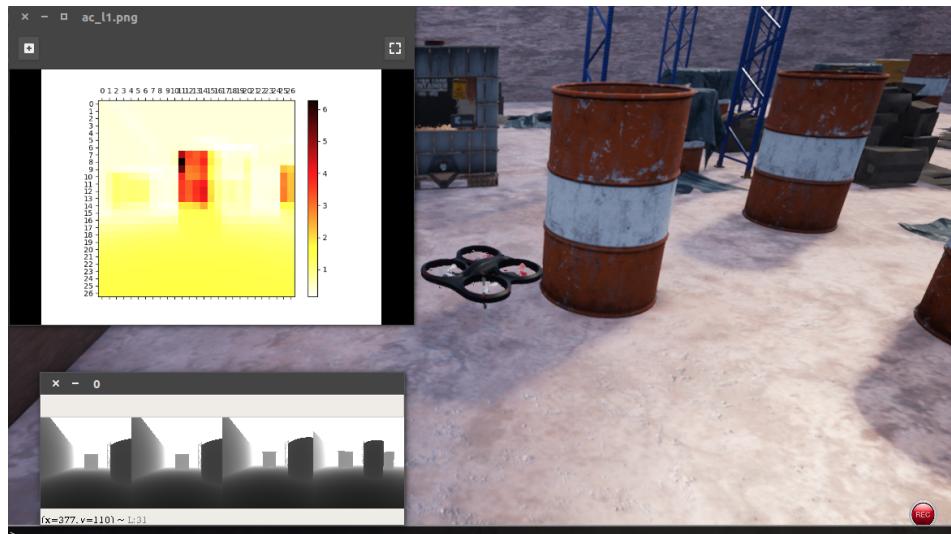


图 3.30 测试场景下的注意力可视化

自注意力层有效地“捕捉”到了障碍物，这样将会更有利于算法实现避障功能。因此本文所设计的注意力层是具有一定程度的解释性意义的。

3.4 本章小结

本章针对于传统导航算法的缺点，首先利用强化学习思想对无人机自主导航问题进行了深入的建模分析，并对动作、状态空间与奖励函数进行了合理的设计与实现。然后在本文所设计的仿真环境中实现并对比了多种主流强化学习算法在自主导航任务上的表现，均不能有效完成任务需求。最后本章针对这些算法的不足进行了合理分析，提出了改进的 CLSA-SAC 算法，利用课程学习方法降低该问题的模型学习难度并引入自注意力机制对现有的 SAC 算法进行改进，最终根据实验结果表明了本文所提出的导航算法可以有效学习到避障以及导航行为，同时对本文所设计的自注意力机制层进行了合理的可解释性分析。

第4章 基于深度学习的3D目标检测

在第3章中已经介绍了基于强化学习方法的无人机自主导航算法，其中无人机策略端的输入包括了无人机和导航目标点的相对位置信息，这样设计正是为方便后续引入3D目标检测算法来实现目标追踪过程。因此，本章首先介绍基于深度学习的2D和3D目标检测技术，然后具体介绍如何利用3D目标检测算法解算出无人机和目标点的相对位置信息，在众多解决方案中本文重点关注基于单目相机的3D目标检测算法，同时提出了一种改进的3D-CenterNet网络模型以便完成最终的目标追踪任务。

4.1 问题分析

本文所要实现的无人机目标追踪算法，是建立在第3章所设计的无人机自主导航算法的基础上，结合计算机视觉检测算法来实现的对运动目标的实时追踪，因此视觉目标检测技术是本章所研究的重点。在无人机目标追踪任务中，普通的视觉目标检测算法只能获取目标在图像平面上的二维坐标信息，由于需要实时跟踪物体，因此必须要获取目标在无人机坐标系下的三维位置信息，并将此信息输入到自主决策与导航模块，才能进行相应的决策与规划，从而实现目标追踪过程。所以无人机目标追踪任务中尤为重要的一个环节就是如何利用视觉传感器输出的图像信息解算出目标物体的三维坐标信息。而将目标在图像中的二维信息恢复到三维空间中的位置信息本身就是一个病态问题，通常丢失的一维深度信息需要根据图像特征来进行额外估计和恢复。相比于传统方法进行深度估计所造成的系统复杂性与低精确性，目前新兴的3D目标检测技术^[59,60]则恰好可以解决上述问题。

4.2 基于深度学习的2D及3D目标检测算法

本节简要介绍主流的2D和3D目标检测算法的原理以及其优劣势，为后文所设计的3D-CenterNet算法进行理论铺垫。目前基于深度学习方法解决2D目标检测问题的算法可以分为两种：单阶段目标检测算法和两阶段目标检测算法。二者的主要区别在于两阶段算法将目标检测流程拆分成了区域检测（region proposal）与分类定位（classification and localization）两个阶段，并且这两个阶段可以利用不同的方式实现；而单阶段目标检测算法则是统一了两阶段算法的过程，利用一个网络直接对图像特征

进行目标类别划分以及目标位置回归。

由于单阶段检测算法的结构简单性以及更快的检测速度，因此本节着重介绍单阶段检测算法，尤其是其中的 RetinaNet 和 CenterNet，这两种算法也是本文后续所设计的 3D-CenterNet 模型的算法基础。

4.2.1 单阶段 2D 目标检测算法

(1) RetinaNet

两阶段目标检测算法如第1章中所述，有着较高的检测精度但检测速度较慢，而 YOLO、SSD^[52] 等单阶段目标检测算法虽然有着很高的检测速度，但是其精度比起两阶段算法还是有着差距。RetinaNet^[105] 引入了 Focal Loss 进行训练，目的就是为了解决单阶段算法由于样本的类别不均衡导致的检测精度较低的问题。因为单阶段算法通常会在一张图片中产生百上千个候选框，但是其中只有相当少的一部分候选框是包含真实物体的，如果每个位置处均对最后的损失产生相同影响的话，就会产生由于负样本过多，尤其是简单负样本（easy negative）过多造成正负样本不均衡，使得模型过度优化到性能较差的位置处，最后模型更加注重发现负样本而不是正样本。因此 RetinaNet 引入了 Focal loss 来平衡正负样本对最后损失的影响并且挖掘了难分样本，通过减少简单样本的权重，增加难分样本权重使得模型向着正确方向优化。Focal loss 就是在原有的交叉熵函数上引入了两种权重因子，以二分类为例：

$$FocalLoss(p_t) = -\alpha_t (1 - p_t)^\gamma \log(p_t) \quad (4.1)$$

上式中 p_t 代表了网络对该目标类别的预测值，具体形式为：

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise} \end{cases} \quad (4.2)$$

α_t 为平衡正负样本的权重因子，对于正样本来说 $\alpha_t = \alpha$ ，对于负样本来说 $\alpha_t = 1 - \alpha$ ； $1 - p_t$ 项负责平衡难易样本，当 p_t 较小时，该项较大，则该样本为难分样本（hard example），对最终损失的贡献也较大， γ 为一个调节因子。

RetinaNet 的网络结构利用了 ResNet+FPN 模型作为 backbone 与 bottleneck，FPN^[106] 是一种金字塔式的特征提取网络，其每一层都对应了不同尺度的特征图，FPN 网络能

够在增加较少计算量的前提下，融合低分辨率语义信息较强的特征图和高分辨率语义信息较弱但空间信息丰富的特征图，使得金字塔中的每一层都具有丰富的空间与语义信息。head 部分利用了 RPN 网络结构，在每一个尺度层都设置了一个共享的回归和分类子网络。其中，分类子网络在每个尺度层都包含 A 个 anchor 和 K 个类别，预测物体出现的概率，同时分类子网络的参数在整个 FPN 金字塔的层间共享。与 RPN 相比，分类子网络更深，只用 3×3 卷积做全卷积处理，并且不和回归子网络共享参数，这种设计决定也使得 RetinaNet 的性能得到了更大的提升。回归子网络与分类子网络平行，作者在 FPN 金字塔每个层都接到一个 FCN 上，意图回归每个 anchor 对邻近 ground truth 的位置和尺寸偏移量。回归子网络的设计和分类相同，不同的是它为每个空间位置输出 $4A$ 个线性输出（分类网络为 sigmoid 函数输出）。其整体结构如图4.1所示。

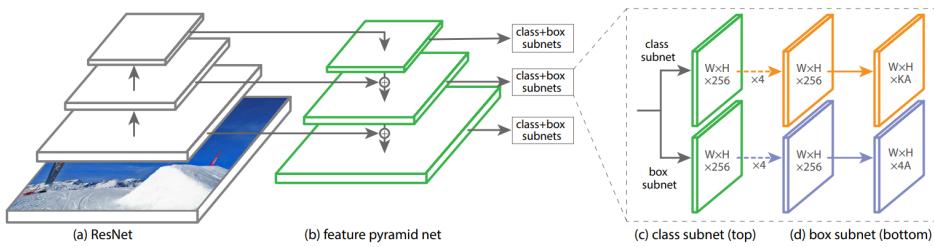


图 4.1 RetinaNet 流程示意图

(2)CenterNet

CenterNet^[107] 是本文重点关注的算法，其是一种去预设框（anchor-free）的单阶段目标检测算法，去 anchor 操作就意味着在计算损失时候不需要复杂的真值框与 anchor 之间互相匹配的计算，这就使得 CenterNet 的算法流程十分优雅简单，并且融合了多种单阶段检测模型的设计优点，可以在特征图上直接检测目标的中心点位置和其大小，其效果如图4.2所示。CenterNet 算法具体训练过程如下：

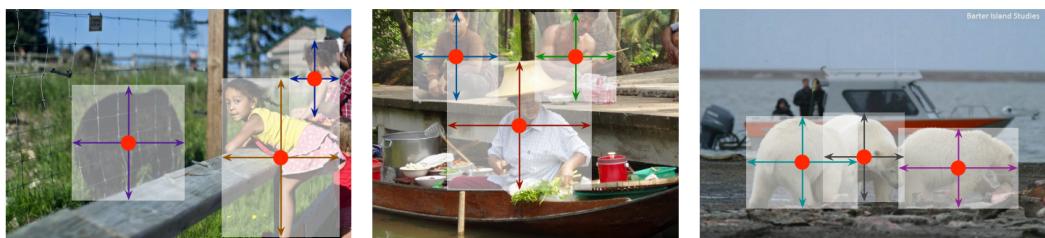


图 4.2 CenterNet 检测效果图

a) 假设输入的图片为 $I \in R^{W \times H \times 3}$, W 和 H 分别代表了图像的宽与高，通

过 backbone 特征提取网络进行特征抽取与下采样，使得输出的特征图大小为 $I' \in R^{W/R \times H/R \times N}$, R 为降采样率, N 为特征通道数。

b) 分类子网络是一个全卷积网络,可以根据特征图 I' 生成分类预测热力图(heatmap) $\hat{Y} \in [0, 1]^{\frac{W}{R} \times \frac{H}{R} \times C}$, C 代表了目标类别总数,这个 heatmap 代表了特征图中的每个像素点处是否是某一类物体中心的预测概率值。对分类子网络进行训练时,参考了 CornerNet^[108] 的方法,对于是 ground truth 的某一类物体,首先利用双线性插值计算真实物体中心点的位置,然后降采样 R 倍后向下取整作为真值的关键点位置,标记为 1,同时生成 $Y \in [0, 1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ 这张关键点真值图。基于 Y ,利用高斯核函数 $Y_{xyc} = \exp\left(-\frac{(x-\tilde{p}_x)^2 + (y-\tilde{p}_y)^2}{2\sigma_p^2}\right)$ 将额外的权重值分配到关键点周围的坐标位置处,其中 σ 代表了和目标尺寸大小相关的标准差, p 代表了 ground truth 的目标位置,同时如果两个高斯分布重叠在一起,则取二者最大值作为该位置处的目标值,即在 $Y_{xyc} = 1$ 处的周围生成如4.3的效果图。利用这个生成的关键点热力图可以很方便地计算损失,且

| | | | | | | |
|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|
| 0.02425801345428226 | 0.05103688810314776 | 0.07974465034866318 | 0.092535281158422 | 0.07974465034866318 | 0.05103688810314776 | 0.02425801345428226 |
| 0.06872199640635958 | 0.14458549326087305 | 0.225913452682986 | 0.26214880584576306 | 0.225913452682986 | 0.14458549326087305 | 0.06872199640635958 |
| 0.14458549326087305 | 0.30419612285238284 | 0.4753035374189698 | 0.5515397744971643 | 0.4753035374189698 | 0.30419612285238284 | 0.14458549326087305 |
| 0.225913452682986 | 0.4753035374189698 | 0.7426572389044386 | 0.8617756314171564 | 0.7426572389044386 | 0.4753035374189698 | 0.225913452682986 |
| 0.26214880584576306 | 0.5515397744971643 | 0.8617756314171564 | 1.0 | 0.8617756314171564 | 0.5515397744971643 | 0.26214880584576306 |
| 0.225913452682986 | 0.4753035374189698 | 0.7426572389044386 | 0.8617756314171564 | 0.7426572389044386 | 0.4753035374189698 | 0.225913452682986 |
| 0.14458549326087305 | 0.30419612285238284 | 0.4753035374189698 | 0.5515397744971643 | 0.4753035374189698 | 0.30419612285238284 | 0.14458549326087305 |
| 0.06872199640635958 | 0.14458549326087305 | 0.225913452682986 | 0.26214880584576306 | 0.225913452682986 | 0.14458549326087305 | 0.06872199640635958 |
| 0.02425801345428226 | 0.05103688810314776 | 0.07974465034866318 | 0.092535281158422 | 0.07974465034866318 | 0.05103688810314776 | 0.02425801345428226 |

图 4.3 真值点附近的 heatmap 图

这个过程并没有 anchor 和 ground 进行相互匹配区分正负样本的过程,热力图中为 1 的地方自然就是正样本,周围的即是负样本,这样可以非常方便地直接利用预测值和真值的 heatmap 进行比较计算损失值。

c) 回归子网络则是根据特征图 I' 生成每个像素处的 4 个线性值,分别代表了长宽值以及目标中心偏移值,所有 C 类预测对象均共享该回归值。

d) 损失由三部分构成:分类损失引入了 Focal loss 平衡正负样本,如下式 (4.3):

$$L_k = \frac{-1}{N} \sum_{xyc} \begin{cases} \left(1 - \hat{Y}_{xyc}\right)^\alpha \log \left(\hat{Y}_{xyc}\right) & \text{if } Y_{xyc} = 1 \\ \left(1 - Y_{xyc}\right)^\beta \left(\hat{Y}_{xyc}\right)^\alpha \log \left(1 - \hat{Y}_{xyc}\right) & \text{otherwise} \end{cases} \quad (4.3)$$

α 和 β 都是超参数, N 是图片中真值关键点数量,这个损失的设计是基于 Focal Loss 的。对于 $Y_{xyc} = 1$ 的关键点位置,如果是易分样本的中心点,则适当减少其损失比重,而难分样本的中心点则增加其比重。而对于非关键点位置,预测的 \hat{Y}_{xyc} 值本应是 0,如果此时预测值越大计算损失的权重就越大,说明这个点是一个难分负样本,而

$(1 - Y_{xyc})^\beta$ 是刻画该点距离中心点远近的权重系数，这一项使距离中心点越远的点的损失比重占的越大，而越近的点损失比重则越小，这相当于弱化了实际中心点周围的其他负样本的损失比重，即处理正负样本不平衡与难分样本挖掘的问题。

目标中心的偏移预测损失为式(4.4):

$$L_{off} = \frac{1}{N} \sum_p \left| \hat{O}_{\bar{p}} - \left(\frac{p}{R} - \tilde{p} \right) \right| \quad (4.4)$$

上式中 $\frac{p}{R}$ 代表了中心点位置降采样后的真值， \tilde{p} 代表了向下取整值， $\hat{O}_{\bar{p}}$ 为预测值。由于上文对原始图片进行了 R 倍降采样，这样的特征图重新映射到原始图片就会产生精度误差，因此对于原始图片上的每一个中心点都会在降采样后的特征图上产生一个额外的位置偏移量，网络通过另一个卷积子网络对该偏移进行回归，并采用 $L1$ 损失进行训练。同理对于目标大小损失的计算如式(4.5):

$$L_{size} = \frac{1}{N} \sum_{k=1}^N \left| \hat{S}p_k - s_k \right| \quad (4.5)$$

上式中 $\hat{S}p_k$ 为网络预测值， s_k 为真实物体尺寸，采用 $L1$ 损失进行训练。那么总的训练损失即为:

$$L_{det} = L_k + \lambda_{size} L_{size} + \lambda_{off} L_{off} \quad (4.6)$$

CenterNet 的具体模型结构图如图4.4所示。

4.2.2 基于深度学习的 3D 目标检测算法

如第1章所述，基于深度学习的 3D 目标检测算法可以按照输入信息类型分为以下三种：视觉输入，雷达输入，多传感器融合输入的算法。基于雷达信息的算法通常有较高的准确度，但是雷达的高价格限制了其在工业领域的应用，而视觉传感器通常价格低廉而且可以获得物体表面丰富的纹理信息，目前在工业界得到大量应用。由于双目相机的深度信息感知视野较小（距离较远时，深度信息会有较大误差），同时需要较多算力计算深度信息因而并不适用于本文的追踪场景，所以本文重点关注基于单目相机视觉信息的 3D 目标检测算法。从技术手段上看，基于单目相机的 3D 目标检测算法可以分为以下几种：基于先验信息融合的算法、基于模板信息匹配的算法、基于额外深度估计的算法、基于单阶段 2D 检测算法改进的 3D 检测算法、基于伪点云

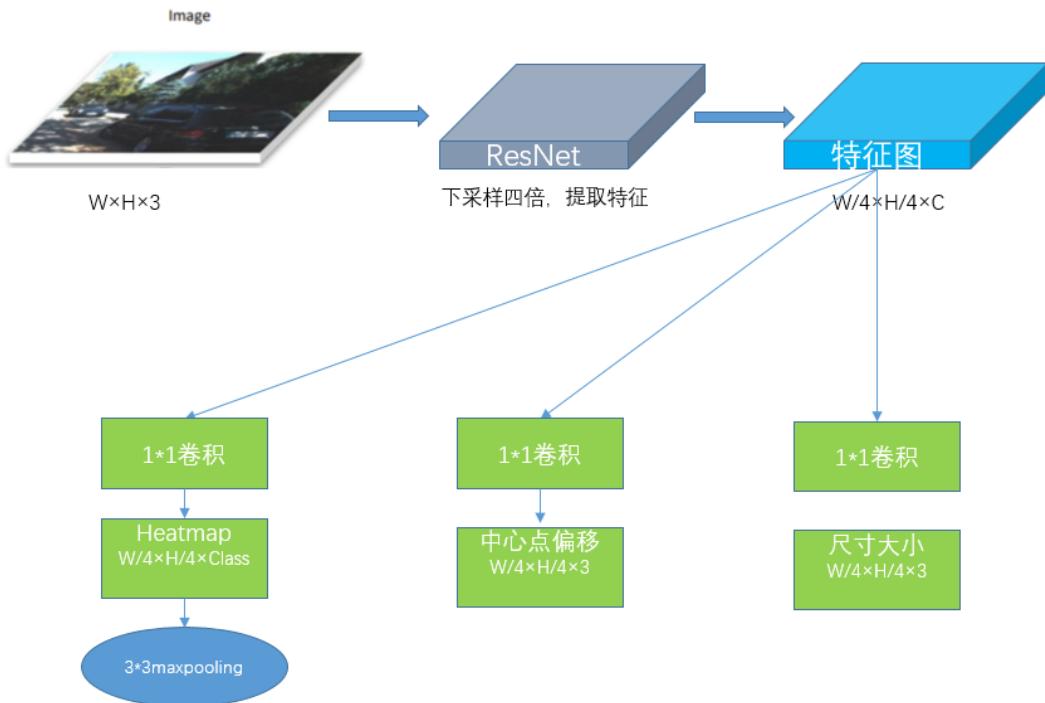
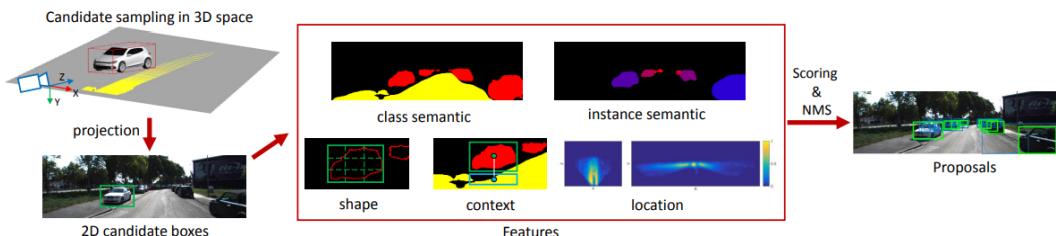


图 4.4 CenterNet 的具体模型结构图

方法的算法等等。本节重点介绍几种结合 2D 检测算法的 3D 目标检测算法。

(1) Mono3D

Mono3D^[109] 是基于先验假设信息的 3D 目标检测算法，它首先根据先验假设条件生成 3D 候选框，然后将该候选框位置根据相机参数从三维平面映射到图像平面，得到图片平面中的 2D 候选框。然后在该候选框位置处结合 Faster R-CNN 网络，利用语义分割信息、上下文信息、形状信息以及位置先验信息等作为特征来计算检测框的能量函数，根据能量函数得分筛选最终检测结果，其具体流程如图 4.5 所示。Mono3D 是

图 4.5 Mono3D 流程图, 图片引自^[109]

第一个使用先验信息来提取 3D 候选框的检测算法，但是在计算能量损失函数的过程中会出现误差累积效应，导致了其精度较差。

(2) Deep3Dbox

Deep3Dbox^[110]首先假设了3D的目标框的各个顶点经过重映射(re-project)后在图像平面内一定会被2D目标检测框紧紧包围住，这就意味着2D检测框的每条边都至少要包含一个3D目标框的顶点，因此每个2D检测框可能有4096个3D检测框的配置分布。由于场景中的物体的俯仰和横滚角度大部分为0，所以2D框上边和下边框最多各各包含3D框中的上下面4个点，而3D框的左右边框最多各包含3D框中垂直边的两个点，因此这时3D框的数量减少到64个。Deep3Dbox的算法框架可以分为三个部分：2D图像目标检测、目标三维尺寸以及角度姿态估计以及目标3D中心点解算。

2D目标检测部分可以利用多种算法实现，例如Faster-RCNN、SSD以及YOLO等算法，在获得2D目标检测结果后，将该目标框从原始图中裁剪出来并缩放到一定大小。得到2D检测图像后将该图片输入到目标三维尺寸以及角度姿态估计模块的VGG网络中，并直接回归尺寸大小以及朝向角。对于目标尺寸，网络输出的是各类模型与先验平均尺寸的相对大小，其损失计算如下：

$$L_{dims} = \frac{1}{n} \sum (D^* - \bar{D} - \delta)^2 \quad (4.7)$$

D^* 为尺寸真值， \bar{D} 为各类模型的先验尺寸平均值， δ 为网络输出值。对于姿态角，Deep3Dbox设置了multibin的结构进行角度输出，其中multi-bin将 $-\pi$ 到 π 的角度进行了离散化，然后网络对于每一个离散化的bin输出一个置信值分数，和角度相对偏移量，损失计算如下式：

$$L_{loc} = -\frac{1}{n_{\theta^*}} \sum \cos(\theta^* - c_i - \Delta\theta_i) \quad (4.8)$$

θ^* 为角度真值， c_i 为当前bin对应的角度值， $\Delta\theta_i$ 为网络输出的角度相对偏移量预测值。在得到三维尺寸和朝向后，利用下式来解算目标三维中心点位置：

$$x_{min} = \left(K \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} d_x/2 \\ -d_y/2 \\ d_z/2 \\ 1 \end{bmatrix} \right)_x \quad (4.9)$$

其中 K 是相机内参矩阵， $R \in SO(3)$ 代表了旋转矩阵，由角度预测值计算， $T =$

$[t_x, t_y, t_z]^T$ 代表了目标中心点在相机参考系下的位置，而上式可以对 x_{\min} , x_{\max} , y_{\min} , y_{\max} 联立四个方程组，对 T 的三个坐标值进行求解。Deep3Dbox 直接扩展了 2D 目标检测算法，并引入了额外的网络支路来回归目标三维信息，是一种典型的基于 2D 目标检测的 3D 目标检测算法，但其后处理过程较为复杂，检测速度较慢。

(3)M3D-RPN

M3D-RPN^[11] 是一种单阶段 3D 目标检测算法，相比于之前的算法没有复杂的子网络处理以及多种先验数据的引入，而是直接利用 Faster R-CNN 中的 RPN 网络进行 3D 目标检测，整体架构类似于 YOLO 和 SSD 系列的单阶段目标检测算法。M3D-RPN 直接改进了原有的 RPN 网络，使得 RPN 网络的预设框 (anchor) 从只包含图像平面信息的 2D 框变为了包含空间和图像两种尺度的 2D-3D 框，anchor 的具体形式如图4.6所示。在特征图上的每个点均预设了 12 种 anchor，这些三维先验 anchor 的引入能够很大程度上帮助网络处理和回归三维空间中的各种信息，减轻网络对 3D 参数估计的负担。

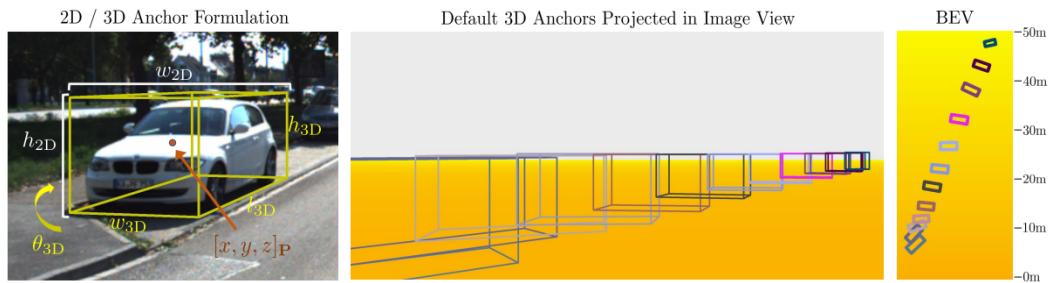
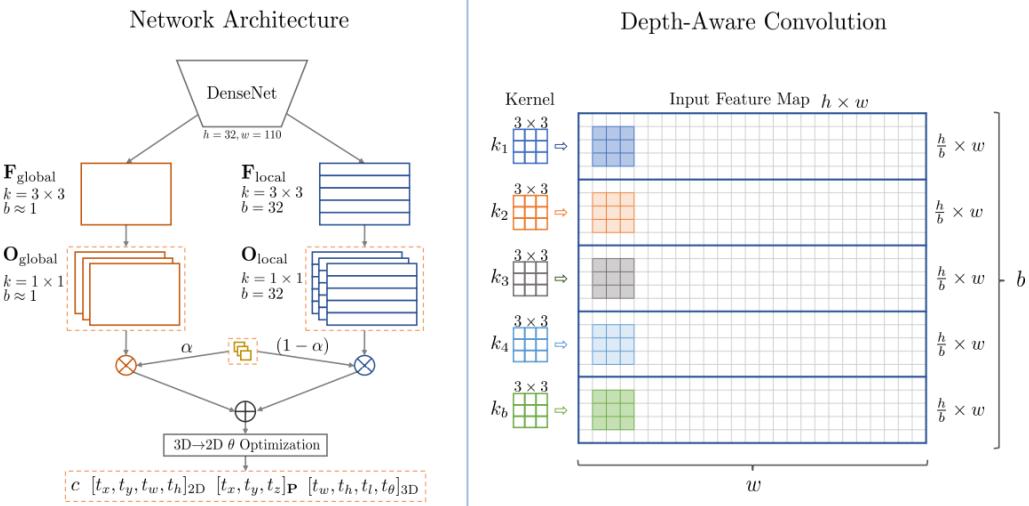


图 4.6 M3D-RPN 的 anchor 示意图, 图片引自^[11]

同时 M3D-RPN 还提出了一种深度感知网络，将图片横向分割为多个区域，每个区域中都有一个独立的卷积核负责提取该区域内的局部特征，如图4.7所示。深度感知网络能够很好的处理局部信息，而局部信息正是 3D 检测模型能否有较高性能的关键，M3D-RPN 通过这种设计在高层信息处理中将局部和全局信息特征加权求和，使得网络能够更好地关注到 3D 检测所需的特征。M3D-RPN 为每一个预设 anchor 均输出 2D 以及 3D 的参数估计，2D 输出采用 IoU loss 计算损失，3D 输出使用了 L1 loss 计算损失，同时对于目标偏角进行了后处理，利用 3D box 投影到图像平面的信息对偏角进行了矫正优化，使得最后的检测性能进一步提高，并最终实现了 SOTA 的表现。

图 4.7 M3D-RPN 的深度感知网络, 图片引自^[111]

4.3 改进的 3D 目标检测算法 3D-CenterNet

上述 3D 目标检测算法都很大程度上依赖于人为设计的附加先验信息，虽然大多数算法都利用了 2D 目标检测模型，但更多地是利用其进行特征提取或者图像平面的 2D 目标框选取，而且算法整体分成多阶段进行了复杂的前处理和后处理。针对上述问题，本文基于 CenterNet 模型，提出了一种改进的基于关键点估计的单阶段 3D 目标检测算法：3D-CenterNet。该算法的核心是按照 CenterNet 的设计直接对 3D 关键点位置和三维参数进行估计，避免引入复杂的先验知识、前后处理操作以及 anchor 的设计，并且利用了一种多参数解耦方式计算 3D 参数的损失，有效提高了算法性能和检测速度，以下为该算法和模型的详细介绍。

4.3.1 问题描述

3D-CenterNet 算法对单目图像的 3D 目标检测问题进行如下建模：在相机内参矩阵 K 已知的情况下，输入一张给定的 RGB 图像 $I \in R^{W \times H \times 3}$ ，其中 W, H 代表图像的宽和高，对于图像中出现的物体准确地预测其类别标签（一共预设 C 类物体），并输出对应的 3D 包围框（bounding box）。其中 3D 包围框由 $(x, y, z, h, w, l, \theta)$ 这七个参数共同描述， (h, w, l) 代表了在三维空间中目标物体的尺寸大小， (x, y, z) 代表了物体在相机坐标系下的相对位置， θ 对应的是目标物体的偏航角，本文按照 KITTI 数据集中的情况假设目标的俯仰角和横滚角都是 0。

4.3.2 模型结构

3D-CenterNet 继承了 CenterNet 的模型结构，和 M3D-RPN 整体架构相似，都是单阶段的 3D 目标检测方法，但是与其不同的是去除了 2D 参数的回归支路，而是直接在图像平面估计 3D 参数，这就使得 3D-CenterNet 的算法并不需要 3D-2D 的复杂后处理以及多阶段优化过程，从而避免了由 2D-3D 的转换过程产生的额外噪声，其模型结构如图4.8所示。整体模型包含了两条支路，一条用于对 3D 关键点进行估计，另

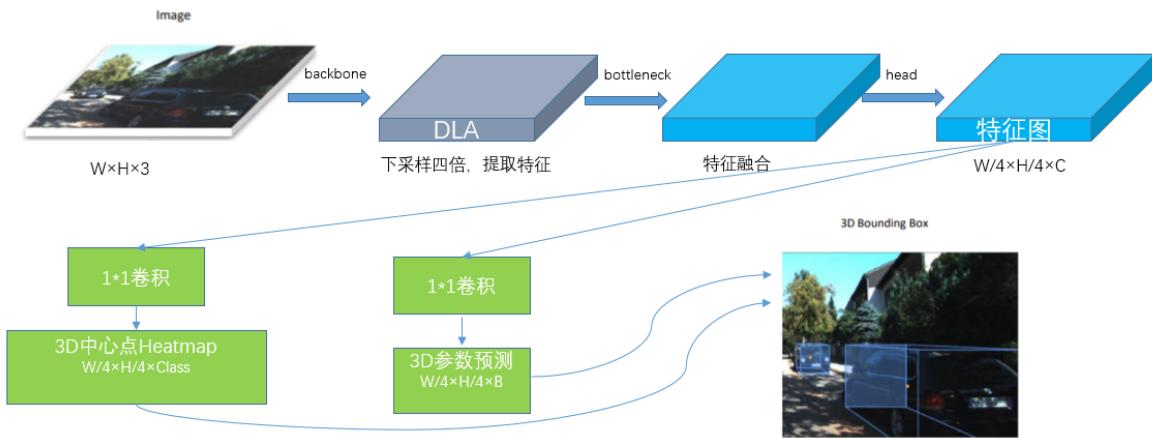


图 4.8 3D-CenterNet 网络结构，类似于 CenterNet

一条支路用于对 3D 参数进行回归，这样的简单设计使得网络模型在训练时更易收敛。

下面依次介绍图4.8中网络模型的各个关键部分：

(1)backbone 部分

网络模型的 backbone 部分，比起使用主流的 Resnet 以及 darknet，本文的 3D-CenterNet 使用了 DLA-34^[112] 进行特征提取，DLA 网络包含两种结构：迭代深度融合结构 (IDA) 和分层深度融合结构 (HDA)，这两种结构都可以看做是对 Resnet 中 skip connection 的进一步扩展，能够更好的融合多层次的特征得到更丰富的空间和语义特征图，IDA 和 HDA 的结构如图4.9和4.10所示。将 IDA 和 HDA 结构相结合就构成了

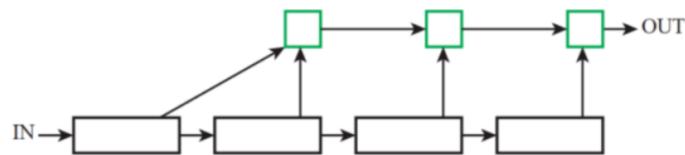


图 4.9 IDA 结构，绿色方块代表“Aggregation Node”，负责特征从浅到深传播的同时进行特征聚集

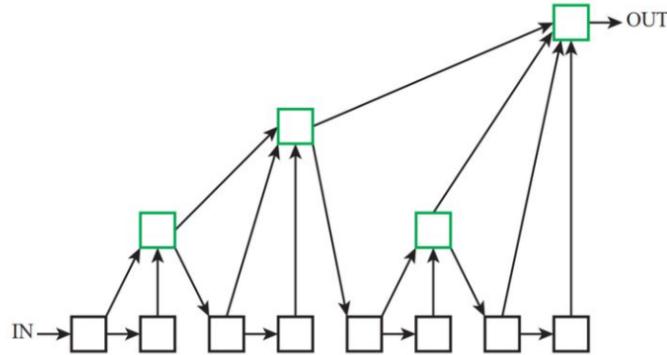
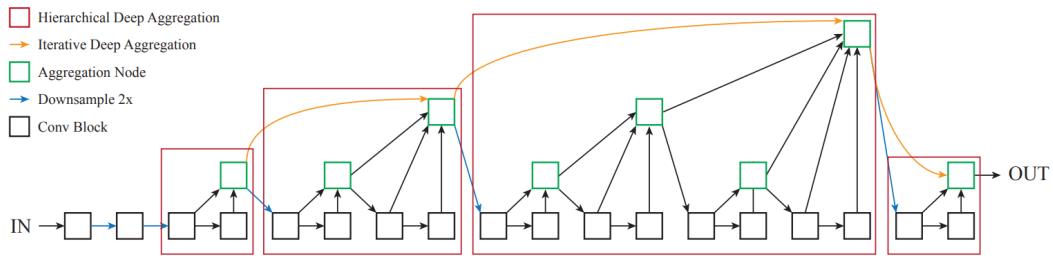


图 4.10 HDA 结构

图 4.11 DLA 结构, 图片引自^[112]

DLA 网络, 如图4.11所示。上图中每个红色框可以看作一个特征提取阶段 (stage), 从整体上看使用 IDA 连接多个不同的阶段, 在每个阶段内使用 HDA 融合层级特征。与 ResNet 类似, DLA 在每个 stage 之间都会进行一次降采样操作。

(2)bottleneck 部分

bottleneck 部分主要用于做多尺度特征融合, 得到最终用于检测阶段的特征图。相比于之前的 RetinaNet 所使用的主流 FPN 金字塔特征融合结构, 本文利用了 DLA 网络的 IDA 结构进行特征融合, 并对 FPN 进行了如下改进, 改进后的结构图如图4.12所示。图中 1 代表了原始图像, 2、4、8、16、32 都是 DLA 网络输出的不同降采样倍数的特征图, up2 和 up4 分别代表了上采样 2 倍与 4 倍, “+” 代表了特征直接相加, 最终得到了降采样 4 倍的融合特征图。相比于 FPN 结构, 这样的 bottleneck 设计能够在每一次上采样前, 都利用 IDA 结构得到本阶段之前的多层特征间更好的融合特征, 这使得网络的特征融合性能更强, 能够得到更好的空间语义信息, 对于后续检测任务来说有很大的帮助。同时在上采样阶段加入了可形变卷积 (DCN)^[113], 可形变卷积相比于普通的 CNN 可以很好的融合目标偏移位置处的信息, 并且具有抵抗形变的良好性能。

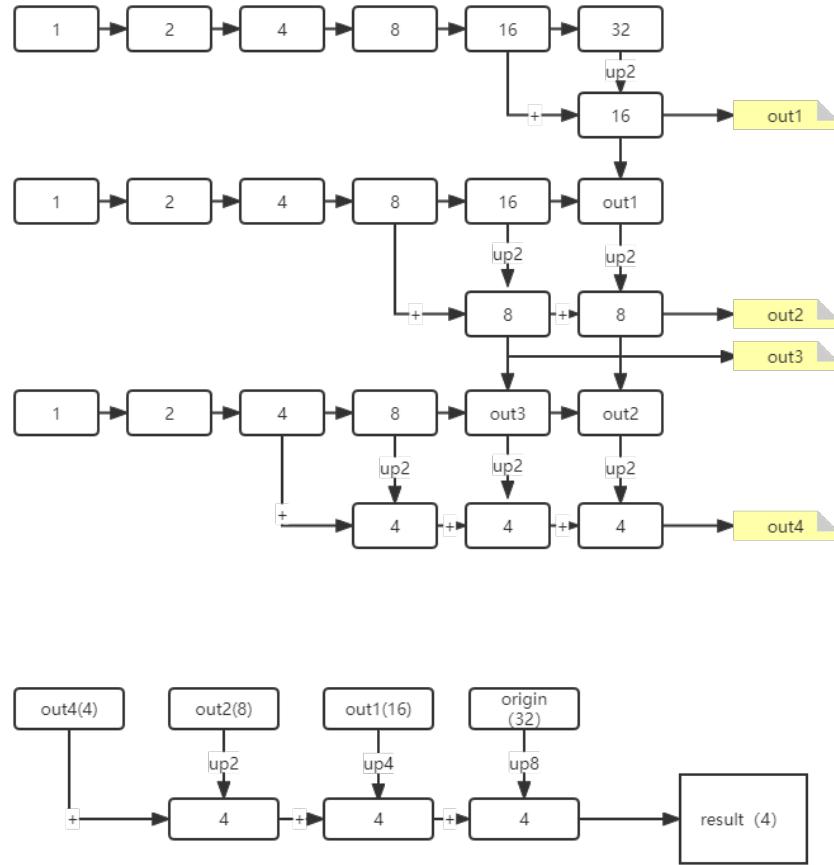


图 4.12 改进的特征融合结构

(3)head 部分

head 主要包含两部分：第一部分是利用降采样 4 倍的特征图生成关键点预测图。关键点预测支路和 CenterNet 的配置相同，均将目标的中心点视作关键点，不同的是 3D-CenterNet 是将三维空间中映射到图像平面的中心点视作关键点，而非 2D 包围框的中心点，二者一般都会存在一些差距。3D 中心点映射到 2D 图像平面的公式如下：

$$\begin{bmatrix} z \cdot x_c \\ z \cdot y_c \\ z \end{bmatrix} = K_{3 \times 3} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (4.10)$$

上式中 (x, y, z) 是目标在三维空间下的中心点位置， (x_c, y_c) 是映射到图像平面中心点的位置， K 为相机内参矩阵。而真值关键点图的生成方式和 CenterNet 部分一样，在每一个关键点为 1 处均按高斯核函数扩展邻近关键点真值。

另一部分是在每个特征点处进行三维参数预测，这条支路输出以下参数：

$$\tau = \begin{bmatrix} \delta_z & \delta_{x_c} & \delta_{y_c} & \delta_h & \delta_w & \delta_l & \sin \theta_l & \cos \theta_l \end{bmatrix}^\top \quad (4.11)$$

其中 $\delta_{x_c}, \delta_{y_c}$ 是关键点预测偏移量， δ_z 是深度值偏移量， $\delta_h, \delta_w, \delta_l$ 是尺寸偏移量， $\sin \theta_l$ 和 $\cos \theta_l$ 是相对旋转角预测值。深度和尺寸预测值都是相对于先验长度的偏移量，对于 KITTI 数据集中的各类数据分别计算其平均先验尺寸和深度信息，其中车对应的先验尺寸为 $\begin{bmatrix} \bar{h}_{car} & \bar{w}_{car} & \bar{l}_{car} \end{bmatrix}^\top = \begin{bmatrix} 1.63 & 1.53 & 3.88 \end{bmatrix}^\top$ ，骑手为 $\begin{bmatrix} \bar{h}_{cyclist} & \bar{w}_{cyclist} & \bar{l}_{cyclist} \end{bmatrix}^\top = \begin{bmatrix} 1.70 & 0.58 & 1.78 \end{bmatrix}^\top$ ，行为人 $\begin{bmatrix} \bar{h}_{pedestrain} & \bar{w}_{pedestrain} & \bar{l}_{pedestrain} \end{bmatrix}^\top = \begin{bmatrix} 1.73 & 0.67 & 0.88 \end{bmatrix}^\top$ 。数据集中目标的平均深度与方差为 $\begin{bmatrix} \mu_z & \sigma_z \end{bmatrix}^\top = \begin{bmatrix} 28.01 & 16.32 \end{bmatrix}^\top$ ，部分数据参考到^[114]。所以，3D 参数回归支路的输出为 $S_r \in \mathbb{R}^{\frac{H}{R} \times \frac{W}{R} \times 8}$ 。其中，尺寸相关的网络支路的输出值会经过 *sigmoid* 函数后得到最终的偏移量预测值 $\delta_h, \delta_w, \delta_l$ ，即：

$$\begin{bmatrix} \delta_h \\ \delta_w \\ \delta_l \end{bmatrix} = \sigma \left(\begin{bmatrix} o_h \\ o_w \\ o_l \end{bmatrix} \right) - \frac{1}{2} \quad (4.12)$$

而相对旋转角预测值 θ_l 的计算采用了同 M3D-RPN 一样的 L2 正则化：

$$\begin{bmatrix} \sin \theta_l \\ \cos \theta_l \end{bmatrix} = \begin{bmatrix} o_{\sin} / \sqrt{o_{\sin}^2 + o_{\cos}^2} \\ o_{\cos} / \sqrt{o_{\sin}^2 + o_{\cos}^2} \end{bmatrix} \quad (4.13)$$

得到上述 8 维预测参数后，即可恢复到 3D 空间中。对于深度预测值，利用下式恢复到三维尺度空间，其中 $\delta_z = o_z$ ：

$$z = \mu_z + \delta_z \sigma_z \quad (4.14)$$

得到预测的深度信息后，结合中心点偏移预测值，计算物体中心点在三维空间中的预测位置：

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = K_{3 \times 3}^{-1} \begin{bmatrix} z \cdot (x_c + \delta_{x_c}) \\ z \cdot (y_c + \delta_{y_c}) \\ z \end{bmatrix} \quad (4.15)$$

三维空间中预测尺寸按照下式计算：

$$\begin{bmatrix} h \\ w \\ l \end{bmatrix} = \begin{bmatrix} \bar{h} \cdot e^{\delta_h} \\ \bar{w} \cdot e^{\delta_w} \\ \bar{l} \cdot e^{\delta_l} \end{bmatrix} \quad (4.16)$$

预测偏航角利用了 Deep3Dbox 的方法，网络并非直接预测目标物体的偏航角 θ 而是预测了相对旋转角 θ_l ，其对应关系如图4.13所示。因此偏航角 θ 的计算如下式：

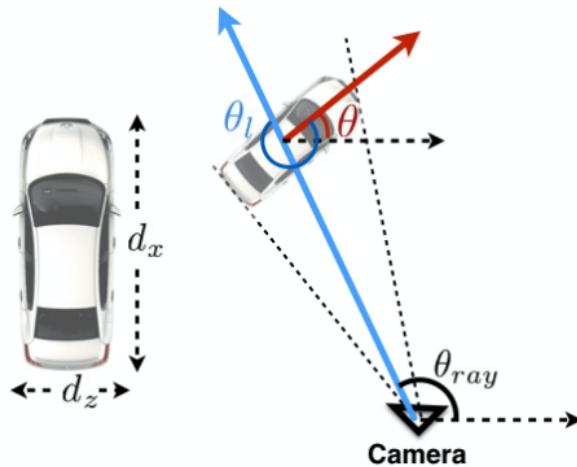


图 4.13 偏航角 θ 、相对旋转角 θ_l 和目标位置偏角 θ_{ray} 的关系，图片引自^[110]

$$\theta = \theta_{ray} + \arctan \left(\frac{\sin \theta_l}{\cos \theta_l} \right) \quad (4.17)$$

上式中 θ_{ray} 可由已经得到的目标三维空间位置 (x, y, z) 计算得出。最终可以根据以上恢复到三维空间的信息，计算得到物体的 3D 包围框（bounding box）的 8 个顶点在相机坐标系下的具体坐标：

$$B = R_\theta \begin{bmatrix} \pm h/2 \\ \pm w/2 \\ \pm l/2 \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (4.18)$$

式中 R_θ 为利用预测偏航角 θ 计算出的旋转矩阵， B 为 8 个坐标顶点集合。

4.3.3 损失设计

对于关键点预测支路，3D-CenterNet 计算损失的方式与 CenterNet 相同，均利用了 Focal Loss 来平衡正负样本，如下式：

$$L_{\text{cls}} = -\frac{1}{N} \sum_{i,j=1}^{h,w} (1 - \check{y}_{i,j})^\beta (1 - \check{s}_{i,j})^\alpha \log(\check{s}_{i,j}) \quad (4.19)$$

$$\check{y}_{i,j} = \begin{cases} 0 & \text{if } y_{i,j} = 1 \\ y_{i,j} & \text{otherwise} \end{cases}, \quad \check{s}_{i,j} = \begin{cases} s_{i,j} & \text{if } y_{i,j} = 1 \\ 1 - s_{i,j} & \text{otherwise} \end{cases} \quad (4.20)$$

α 和 β 都是控制 Focal loss 的超参数， $y_{i,j}$ 代表了真实值， $s_{i,j}$ 是网络输出的预测值。

对于回归支路，本文的 3D-CenterNet 并没有采用像 M3D-RPN 那样对每个输出的参数直接利用 L1 损失计算其与真值之间的差别，而是利用了 Simonelli^[114] 所提出的多参数损失解耦的方式进行计算。具体过程如下：对于中心点偏移的预测值 $(\delta_{x_c}, \delta_{y_c})$ 和深度值预测偏移量 δ_z ，利用式 (4.14) 和式 (4.15) 可以计算得到物体的预测三维中心点 (x, y, z) ，然后结合真值对应的偏航角 $\hat{\alpha}$ 和尺寸 $(\hat{h}, \hat{w}, \hat{l})$ ，利用式 (4.18) 可以生成一个新的 3D-box 元素 B_{loc} ；同理利用预测的角度信息 α 和真实值 $(\hat{x}, \hat{y}, \hat{z})$ 与 $(\hat{h}, \hat{w}, \hat{l})$ 可以生成 B_{angle} ；利用尺寸预测信息 (h, w, l) 和真实值 $(\hat{x}, \hat{y}, \hat{z})$ 与 $\hat{\alpha}$ 可以生成 B_{dim} 。然后分别计算三个解耦的 3D-box 元素与真值对应的 3D-box 元素 \hat{B} 之间的 L1 损失：

$$L_{\text{reg}} = \frac{\lambda_{loc}}{N} \|\hat{B} - B_{loc}\|_1 + \frac{\lambda_{angle}}{N} \|\hat{B} - B_{angle}\|_1 + \frac{\lambda_{dim}}{N} \|\hat{B} - B_{dim}\|_1 \quad (4.21)$$

该损失是在三维空间内计算的，更有利于网络输出合理的三维空间预测参数。最终的损失可以写成下式：

$$L = L_{\text{cls}} + L_{\text{reg}} \quad (4.22)$$

这样的设计使得多参数对损失的影响按照不同参数在三维空间进行了解耦，使得回归的 3D 参数融合计算出的 3D-box 会更加合理，算法性能得到进一步提升。

4.3.4 KITTI 数据集

KITTI 数据集由德国卡尔斯鲁厄理工学院和丰田技术研究院联合制作的，是目前国际上最大的自动驾驶场景下的计算机视觉算法评测数据集。本文主要利用 KITTI 数

据集中的 3D 物体检测数据集，简称 KITTI-3D。该数据集包含了市区，乡村和高速公路等多种场景下的真实图像数据，标注的真值包含了物体类别、截断情况、遮挡情况、偏航角、三维尺寸、3D 位置、2D 包围框等详细信息，KITTI-3D 也是目前被广泛应用的 3D 目标检测数据集。KITTI-3D 数据集包含 7481 张训练数据以及 7518 张测试数据。其中评估难度可以按照图片中物体的遮挡以及截断情况分为简单，中等和困难三种。因为测试数据现在并不开源，所以本文将开源的 KITTI-3D 训练数据集分成两部分，其中 3740 张用于训练，3741 张用于测试评估。

4.3.5 训练过程与实验结果

(1) 数据预处理

为更好地训练模型，在数据预处理的过程中去除了目标中心点在图像平面外的目标数据。在去除无效数据之后，将所有图片的尺寸缩放为 1280×384 。然后对图片分别进行了随机水平翻转、尺寸缩放和平移，缩放尺寸比例为 $[0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3]$ 中的随机值，平移相对尺寸为 $[-0.2, -0.1, 0, 0.1, 0.2]$ 中的随机值，对于随机缩放与平移后的样本只计算关键点损失，而不参与 3D 参数的训练，这样做的目的是让网络的学习过程更加稳定。

(2) 训练配置

3D-CenterNet 采用随机梯度下降法进行训练，梯度更新方式采用 Adam 算法，其中 Adam 的平滑系数 β_1 和 β_2 分别为 0.9 和 0.999，损失函数式 (4.19) 中 α 和 β 分别为 2 和 4，损失函数式 4.21 中 $\lambda_{loc} = \lambda_{angle} = \lambda_{dim} = 1$ 。因为整体设计是基于 CenterNet 的，所以不包含 anchor 框，也就没有 anchor 和 ground truth 匹配的过程，其中关键点为 1 的点均视为正例。`batch_size` 设为 32，初始学习率为 0.0005，训练迭代 15000 步，学习率在第 25 和 40 次迭代时分别缩小 10 倍。网络模型均基于 Pytorch 框架下实现，所有训练与测试过程均在 Ubuntu 16.04 Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz NVIDIA Titan xp 上进行。

(3) 测试结果

在测试时，只需将关键点预测降序排列前 50 且出分数大于 0.25 的检测点位置选出即可作为检测结果，无需复杂的后处理和 NMS 操作。测试集评估指标采用的是 KITTI 官方所使用的 $AP_{3d}|_{R11}$ (0.7 的 3D bounding-box IoU 作为阈值，同时采用 11 点插值计算 AP 值)，与其他算法的对比结果如表 4.1 所示。从表中能够得知相比于之前

表 4.1 3D-CenterNet 与其他算法在 KITTI 数据集的“Car”类别的测试数据对比

| Method | Easy | Moderate | Hard |
|------------------|--------------|--------------|--------------|
| Mono3D | 2.53 | 2.31 | 2.30 |
| MonoGR | 13.88 | 10.19 | 7.62 |
| M3D-RPN | 20.40 | 15.78 | 13.34 |
| 3D-Center | 22.69 | 15.73 | 13.41 |

的单阶段目标检测算法，本文所改进的 3D-CenterNet 在困难和简单类型的数据集上均能够有效地提高单目相机 3D 目标检测任务的精度。

本文中所提出的改进方法，对于 3D-CenterNet 算法的影响程度的消融对比实验如表4.2所示。从表中可以看出本文在原有的 CenterNet 模型基础上，所引入的以 DLA

表 4.2 3D-CenterNet 中所使用的的各种方法的 ablation 实验

| Method | Easy | Moderate | Hard |
|--------------------------------|--------------|--------------|--------------|
| CenterNet(只有 3D 回归层没有其他 trick) | 17.37 | 13.16 | 10.27 |
| 使用 ResNet 作为 backbone | 21.13 | 14.14 | 12.26 |
| 使用 FPN 作为 bottleneck | 22.46 | 13.24 | 11.27 |
| 不使用 DCN 网络 | 21.22 | 13.82 | 11.29 |
| 不使用解耦损失，直接利用 L1 损失 | 21.43 | 15.48 | 13.10 |
| 3D-Center | 22.69 | 15.73 | 13.41 |

网络做为“backone”，“bottleneck”层的新特征融合结构、上采样前使用 DCN 进行特征提取以及使用解耦损失训练网络的操作，都能够有效提升 3D-CenterNet 模型的检测精度，同时也进一步说明了本文所设计的 3D-CenterNet 算法的有效性。

3D-CenterNet 在 KITTI 测试集上的实际检测效果如图4.14和4.15所示，图中绿色框代表“car”，黄色框代表“cyclist”，青色框代表了“pedestrian”：



图 4.14 3D-CenterNet 在 KITTI 测试集上的检测效果



图 4.15 3D-CenterNet 在 KITTI 测试集上的检测效果

3D-CenterNet 算法在校园内随机采集的数据图片上的实际检测效果如图4.16到4.18所示。

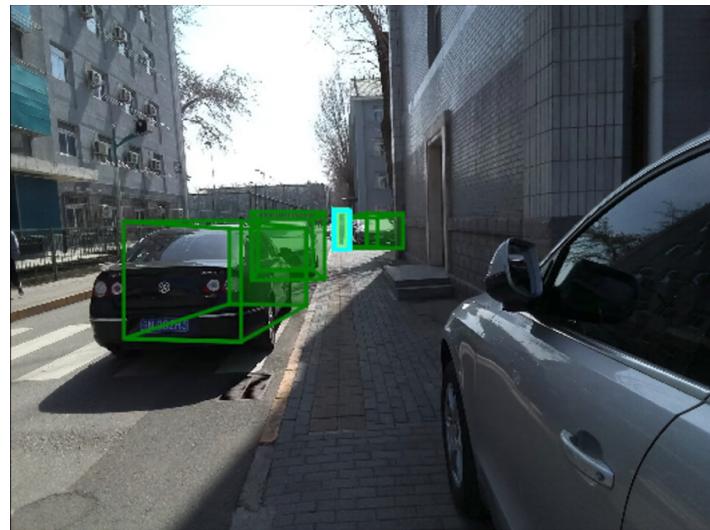


图 4.16 3D-CenterNet 在校园内随机采集的数据图片上的检测效果

虽然 KITTI 数据集中大部分图片是在街道路面场景下采集的，但是由于深度网络模型的泛化能力，以及预训练好的 DLA 网络在更大的数据集中学习到的对前景和背景特征的抽象能力，使得最终的检测模型同样可以适用于非公路场景以及仿真环境中的室内图片。3D-CenterNet 算法在非公路场景和在本文所设计的室内仿真环境中采集数据的检测效果如图4.19到4.21所示。

4.4 本章小结

本章首先介绍了基于深度学习的 2D 目标检测算法，2D 目标检测算法是 3D 目标检测的基础，然后介绍了几种主流的 3D 目标检测算法，并详细分析了其各自的优缺点。

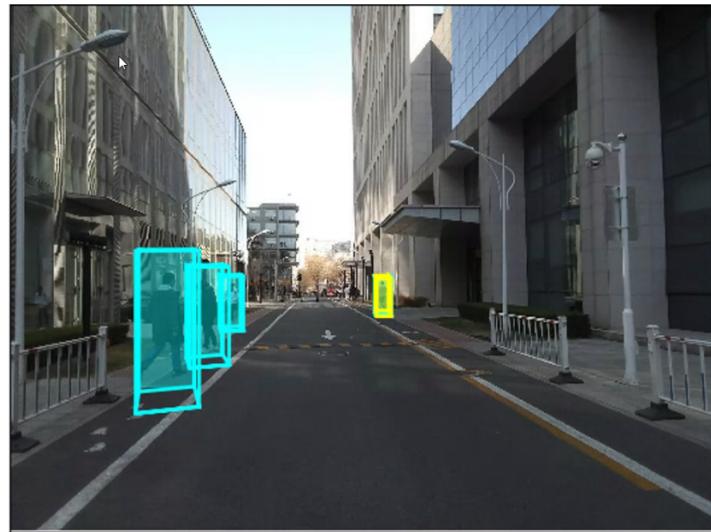


图 4.17 3D-CenterNet 在校园内随机采集的数据图片上的检测效果

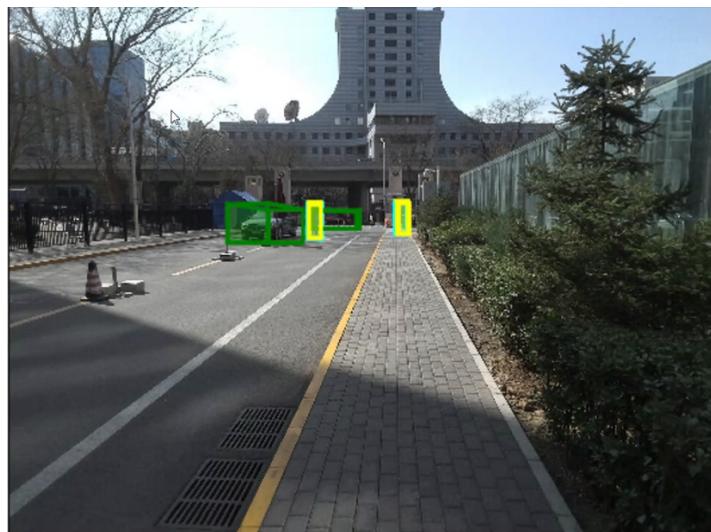


图 4.18 3D-CenterNet 在校园内随机采集的数据图片上的检测效果

点，最后针对主流算法的优劣势提出了基于 CenterNet 网络的一种改进的单阶段 3D 目标检测模型 3D-CenterNet，该模型有效地利用了 CenterNet 算法的优点，保证了检测速度。并且，3D-CenterNet 利用关键点的方法进行目标检测的同时直接回归三维参数并利用了解耦损失函数进行训练。最终通过在 Kitti 数据集上的实验，表明了本文所提出的 3D-CenterNet 算法在 3D 目标检测任务上的有效性。



图 4.19 3D-CenterNet 在非公路场景中的检测效果



图 4.20 3D-CenterNet 在仿真环境中采集的数据图片上的检测效果

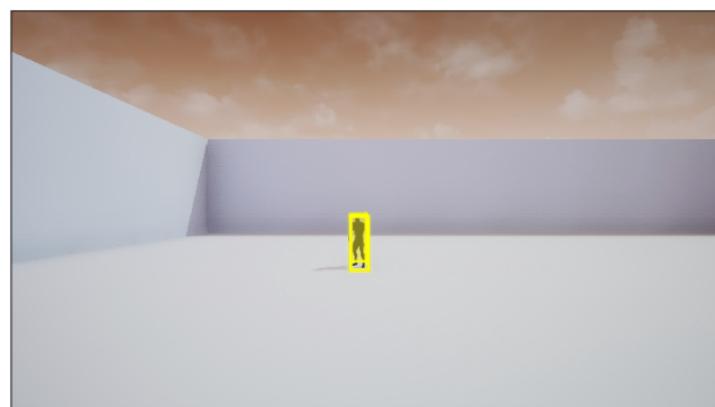


图 4.21 3D-CenterNet 在仿真环境中采集的数据图片上的检测效果

第5章 无人机目标追踪算法的软件系统实现

在第3章和第4章中分别介绍了如何利用强化算法实现无人机自主导航功能，以及如何利用单目视觉信息进行3D目标检测。基于以上两章的内容，本章主要设计如何有效利用上述两种算法实现无人机目标追踪任务。其思路为：首先利用3D目标检测算法对感兴趣物体进行逐帧检测，同时利用目标物体信息进行匹配筛选得到待跟踪目标，然后将检测算法输出的目标三维坐标信息结合卡尔曼滤波技术对目标的准确位置进行估计，最后将该目标位置信息传入到自主导航算法模块实现对目标的实时追踪任务。

5.1 无人机目标追踪算法框架

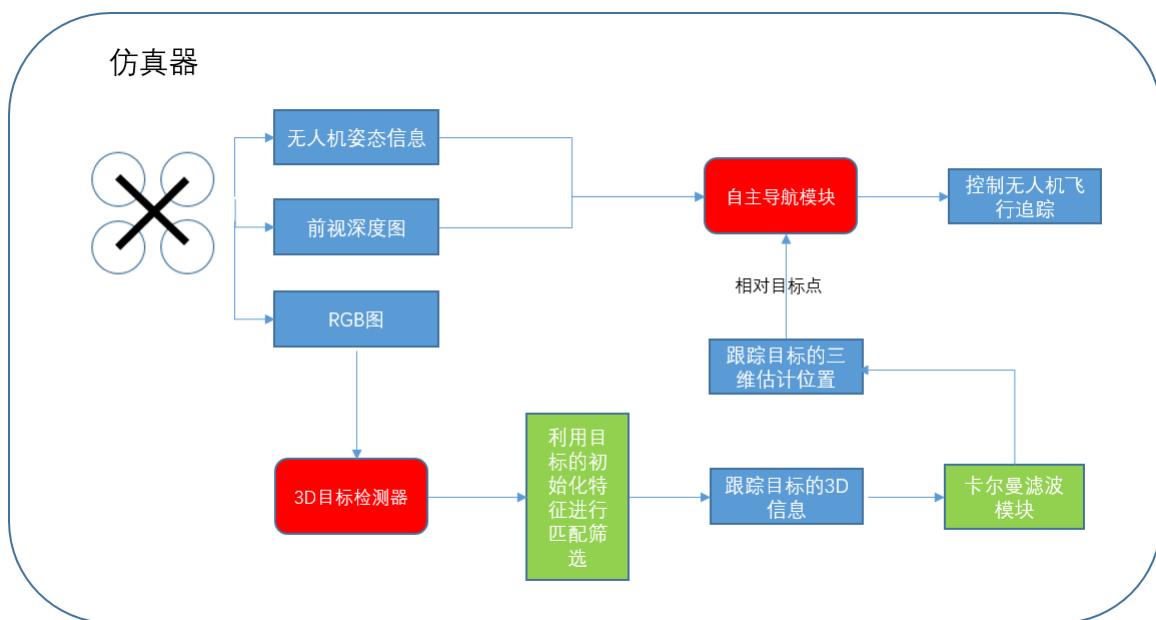


图5.1 无人机目标追踪算法框架

图5.1所示内容为本章所设计的无人机目标追踪算法框架，其具体流程可以分为以下三个部分。

5.1.1 3D 目标检测与目标匹配过程

单目相机虽然价格低廉，便于安装，但是其不能像雷达传感器一样能够 360° 感知周围环境信息，相机的视野范围通常是固定的。因此为了更好地去获取运动目标的视觉信息，本文为仿真环境中的无人机配置了 4 台不同角度安装的单目相机，以确保可以最大程度的进行 360° 范围内的目标检测，同时能够有效避免在追踪过程中由于前视方向的相机视野范围受限导致目标丢失的情况。具体的相机角度配置如图 5.2 所示，绕机身 z 轴在 0° 、 90° 、 -90° 和 180° 的方向分别配备一个单目相机，来获取 450×253 尺寸的图像信息，然后将 4 个方向的图片送入 3D 目标检测网络进行检测，可以分别得到各自对应的目标检测信息，如图 5.3 所示。同时为了压缩检测时间，通过多进程方式来处理图像 I/O 流程，以达到最快的检测速度。

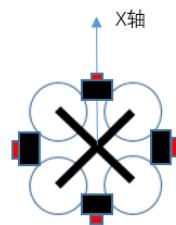


图 5.2 仿真环境中无人机相机配置示意图

检测算法能够很好地得到图像中出现的各类预检物体的信息，但是通常跟踪目标只有一个，因此视觉跟踪类算法都会预先初始化一个目标区域来提取跟踪物体的初始化状态特征。本文也同样设计了该操作，在第一帧所有检测出的物体中选出一个待跟踪目标，然后将其在图像中的 2D 检测区域的特征向量作为目标的初始化特征，该特征负责在之后的检测过程进行目标匹配。提取目标特征的方式采用了 Fast R-CNN 设计的 ROIpooling 的方法，可以实现对不同尺寸的目标图片提取相同维度的特征。ROIpooling 首先根据输入图像以及最终的检测信息，得到物体的 2 维检测框，然后将 2 维检测框映射到 DLA 特征提取网络输出的特征图上的对应位置，将映射后得到的特征图上的区域，划分成数量相同的块（块的数量和输出维度特征维度相同），最终对每个块的区域进行 max pooling 的操作，得到最后的特征向量，其计算过程的示意图如 5.4 所示。之后依次计算初始化目标区域的特征和其余检测区域的特征余弦相似度，取相似度最大的目标作为最终的追踪目标，其具体实现流程如图 5.5 所示。图中初始化所要追踪的目标为行人，然后根据第一帧检测出的行人目标图片计算其对应的初始化目标特征，然后在后续帧中虽然检测出多种物体，但是分别计算了每种物体的深度特

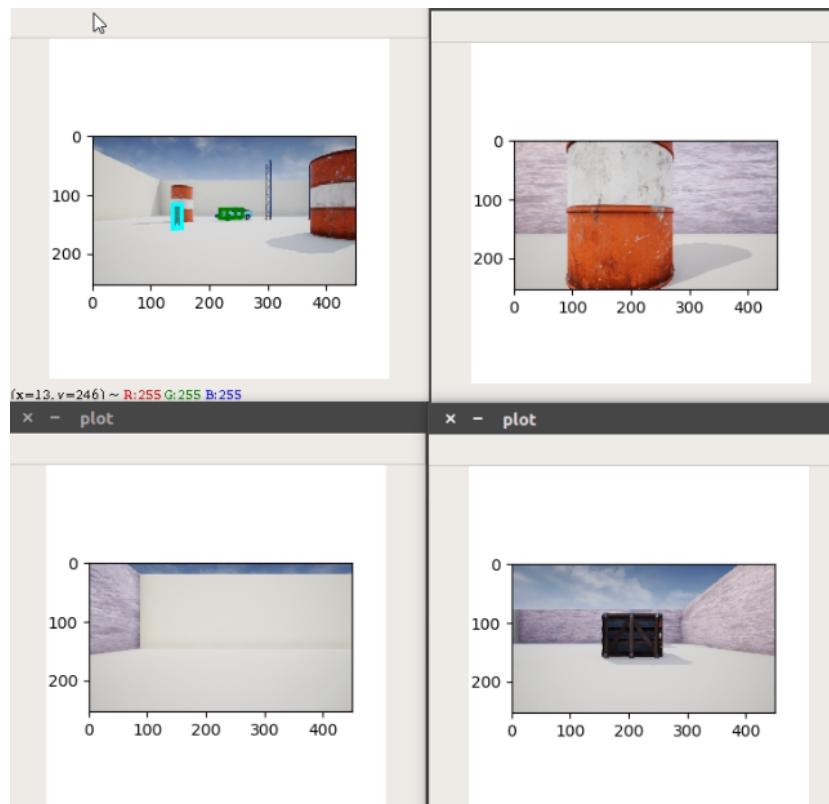


图 5.3 多角度相机的 3D 目标检测示意图，图中含有一个行人和一个车辆均被检测出

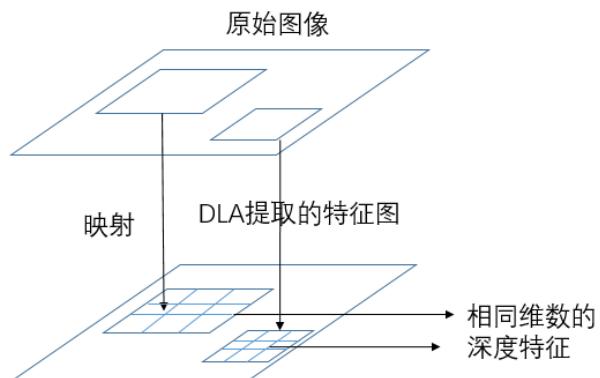


图 5.4 ROIpooling 计算目标区域的深度特征

征和初始化目标特征之间的余弦相似度，如图5.5中蓝色框中的行人和初始化目标的余弦相似度为 0.77，而绿色框中的汽车和初始化目标的余弦相似度仅为 0.09，因此筛选出蓝色框中的目标作为后续跟踪对象，同时更新该目标对应的特征为初始化特征。

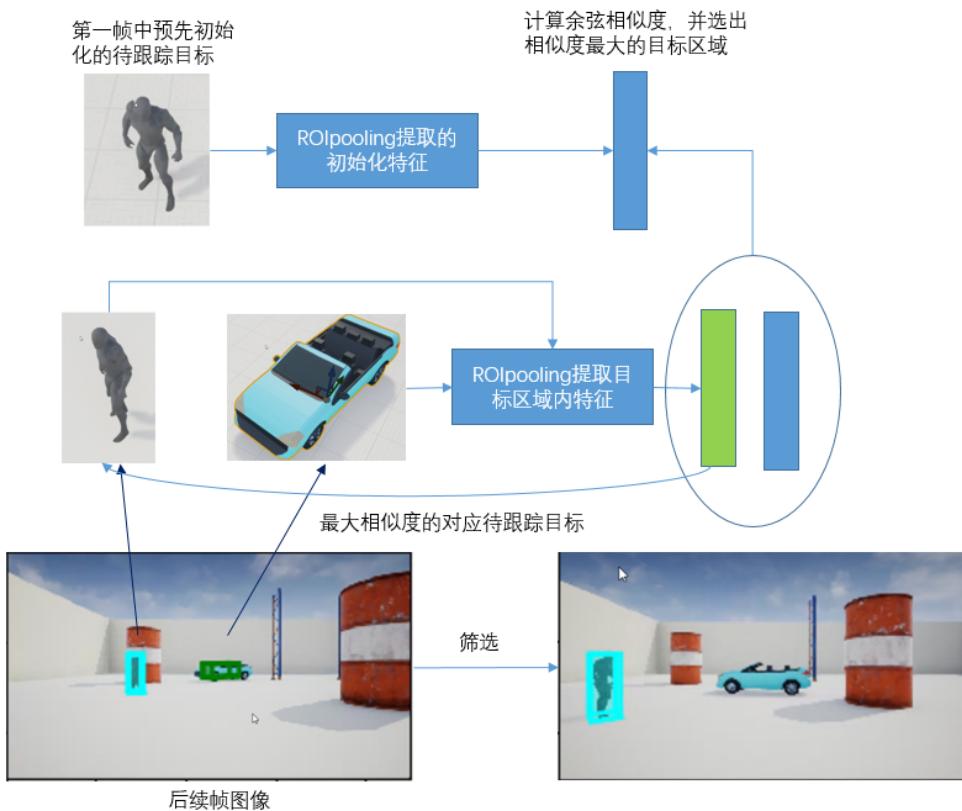


图 5.5 基于余弦相似度的目标匹配过程

5.1.2 卡尔曼滤波

经过上述目标检测与匹配过程，可以得到检测算法输出的待跟踪物体的三维坐标信息，该信息为目标在机体坐标系下的相对位置信息 x_{3d}, y_{3d}, z_{3d} ，同时加上无人机自身在世界坐标系下的位置即可得到目标物体在世界坐标系下位置的测量值 x_m, y_m, z_m 。同时由于实验中的跟踪目标在平面中进行运动，因此可以简化目标高度为固定值并将其运动过程描述为一阶线性方程。这样物体的运动状态为 $s = [x, y, v_x, v_y]$ ，其中 x, y, v_x, v_y 为分别为 x 轴和 y 轴的位置和速度信息，于是就有了以下的预测方程：

$$\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_t = A \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_{t-1} + Q \quad (5.1)$$

式(5.1)中 A 为状态转移矩阵,其具体形式为:

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.2)$$

Q 为过程协方差矩阵,在本文中设为下式:

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \quad (5.3)$$

而测量方程为:

$$\begin{bmatrix} x_m \\ y_m \end{bmatrix}_t = H \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} + R \quad (5.4)$$

式(5.4)中 H 为测量参数矩阵, R 为测量噪声协方差,本文中其具体形式为:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} R = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.25 \end{bmatrix} \quad (5.5)$$

经过以上建模后,就可以直接利用卡尔曼滤波公式对系统状态 $s = [x, y, v_x, v_y]$ 进行有效估计,同时由于卡尔曼滤波算法中包含下一时刻的预测值,因此即使由于遮挡导致检测失败,系统仍然会根据预测值给出目标当前的估计位置,图5.6为本文所设计的卡尔曼滤波过程在仿真环境中的实际效果。如图5.6所示,即使有时会由于遮挡以及3D检测算法的漏检导致当前时刻没有红色的测量值,但是卡尔曼滤波器仍然会根据预测方程给出当前时刻的估计值以完成追踪任务。

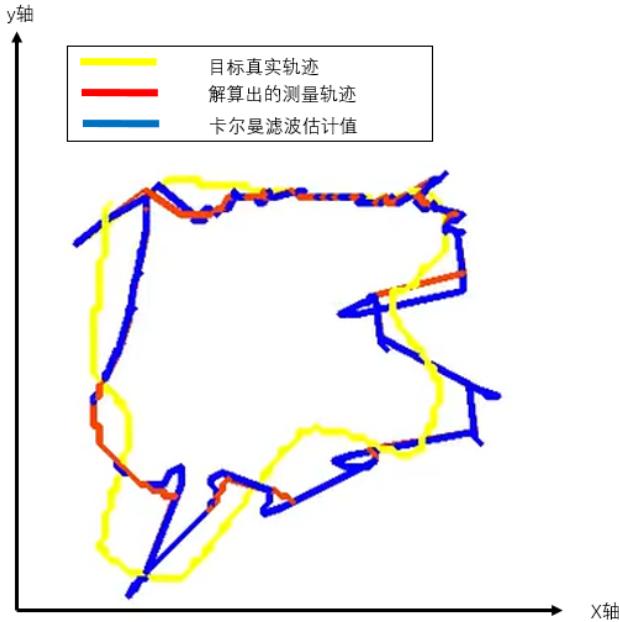


图 5.6 卡尔曼滤波效果示意图。其中黄色轨迹为物体真实运动轨迹，红色轨迹为物体通过检测算法解算出的位置测量值，蓝色曲线为卡尔曼滤波后的估计值

5.1.3 配合自主导航模块完成目标追踪

利用卡尔曼滤波估计出的目标状态 $s = [x, y, v_x, v_y]$ ，减掉无人机当前自身位置，即可得到卡尔曼滤波后的相对位置信息 (x_r, y_r, z_{3d}) 。因为自主导航算法的输入已经被提前设计为目标与无人机的相对位置信息，所以将每一时刻的相对位置估计值输入到自主导航算法模块即可完成目标追踪任务。

本文目标追踪算法的具体代码是基于 Pytorch 的多进程封装实现的，由于 Airsim 仿真环境中读取图像需要时间，因此相机的图像读取以及 3D 目标检测算法分别设置在 4 个进程中完成，然后利用共享内存机制将目标位置传回主进程中的卡尔曼滤波算法，最后将卡尔曼滤波算法输出的目标估计位置信息给到自主导航模块完成追踪任务。由于 3D 目标检测与目标匹配算法主要运行在并行多进程环境中，因此最终的任务实现频率主要取决于卡尔曼滤波算法和自主导航算法所用时间，算法整体运行频率在 10Hz，各个模块所用时间如表5.1所示。传统高精度地图的建立频率通常在 1Hz 左右，因此可以看出无地图的目标追踪算法整体用时要远远小于传统方法中高精度地图的建立时间。

表 5.1 本文目标追踪算法各个模块所用时间

| 算法模块 | 3D 目标检测算法 | 目标匹配算法 | 卡尔曼滤波算法 | 自主导航算法 |
|--------|-----------|--------|---------|--------|
| 测试时间/s | 0.072 | 0.004 | 0.002 | 0.091 |

5.2 具体实验效果

为验证上述目标追踪算法的有效性,在本文所设计的仿真平台中搭建了如图5.7所示的测试环境。测试环境中存在一待跟踪目标(human),该目标从0时刻开始会沿着

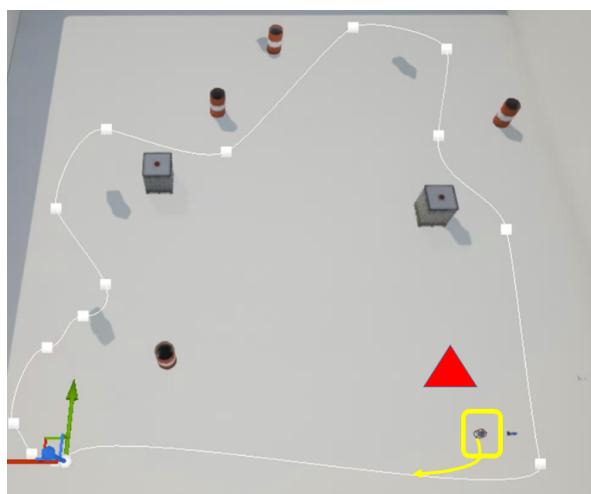


图 5.7 无人机目标追踪算法测试环境,白色曲线为目标物体(human)的运动轨迹,红色三角区域为无人机起飞地点,黄色方框标记位置为目标(human)

白色轨迹进行移动(该白色轨迹与5.6中目标真实运动轨迹相同)。无人机同样从0时刻开始启动上述目标追踪算法系统,其效果如图5.8所示,启动后4个位置的相机会分别捕捉各自方向的图片。在算法开始运行后,3D 目标检测算法会检测出物体位置,并根据匹配算法筛选得到当前待跟踪物体,然后标注在图像中。同时卡尔曼滤波算法会进行更新与迭代得到当期目标位置的最优估计值,如图5.9所示。即使在无人机追踪过程中由于目标运动到有障碍物遮挡位置,造成无法视觉算法无法有效检测目标物体,但由于卡尔曼滤波算法会输出预测信息给到自主导航算法,因此仍能够很好地控制无人机绕过障碍物并继续运动到目标位置附近实现后续的检测和追踪,其具体效果如图5.10所示。

目标追踪算法在每一时刻均会运行上述过程,从而最终实现了对运动目标的有效追踪,其最终完成了对目标物体进行一圈有效追踪的效果如图5.11所示。

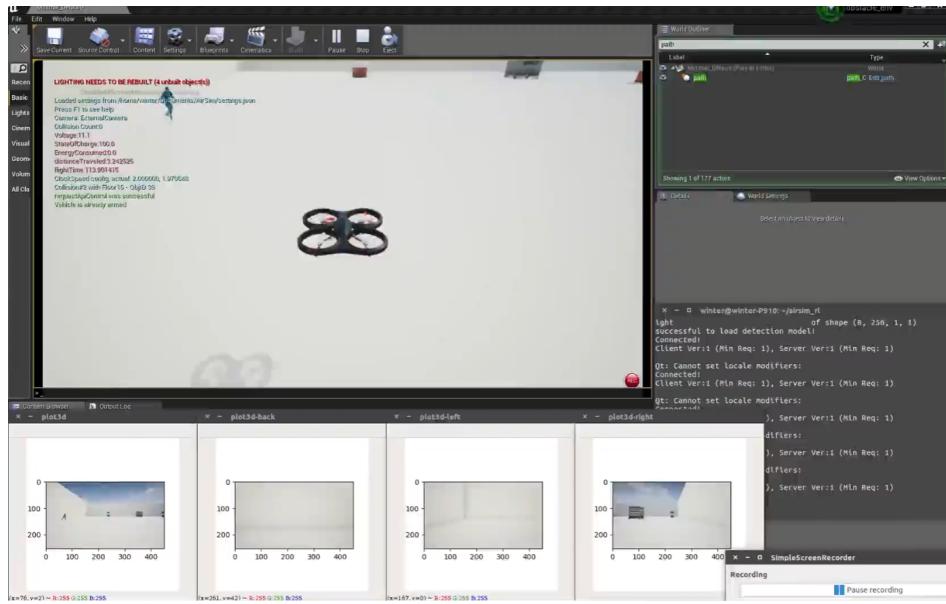


图 5.8 无人机目标追踪算法启动示意图

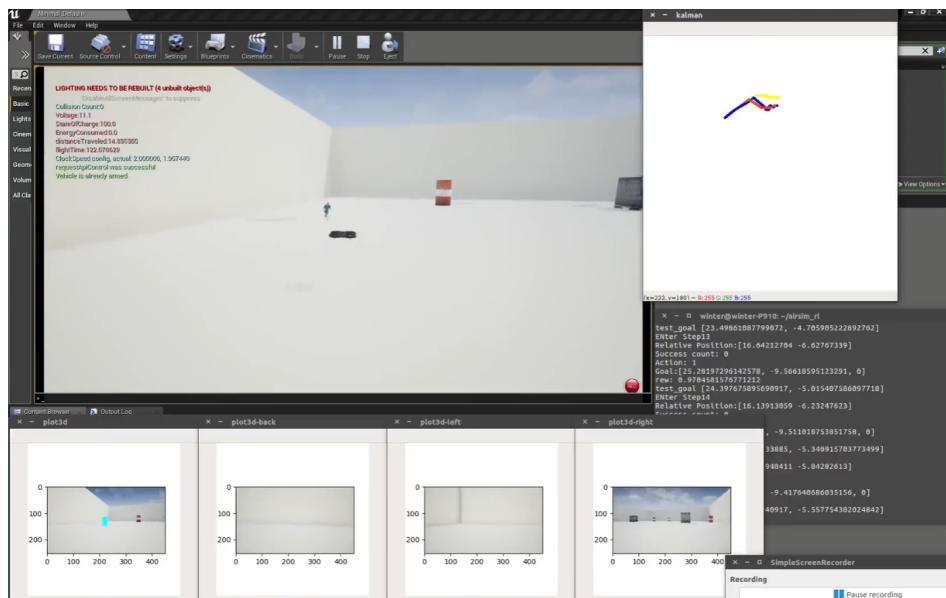


图 5.9 无人机目标追踪算法 3D 目标检测与卡尔曼滤波效果示意图，左下图中蓝色方框标注出了目标的检测位置

5.3 本章小结

本章主要利用了前文设计的基于强化学习的自主导航算法以及 3D 目标检测算法，在仿真环境中配合卡尔曼滤波器共同完成无人机对运动目标的追踪任务。本章主要介绍了该部分内容的软件实现细节和具体效果，并测试了各个模块运行的时间，通过实验证明了本文所提出的目标追踪算法的有效性。

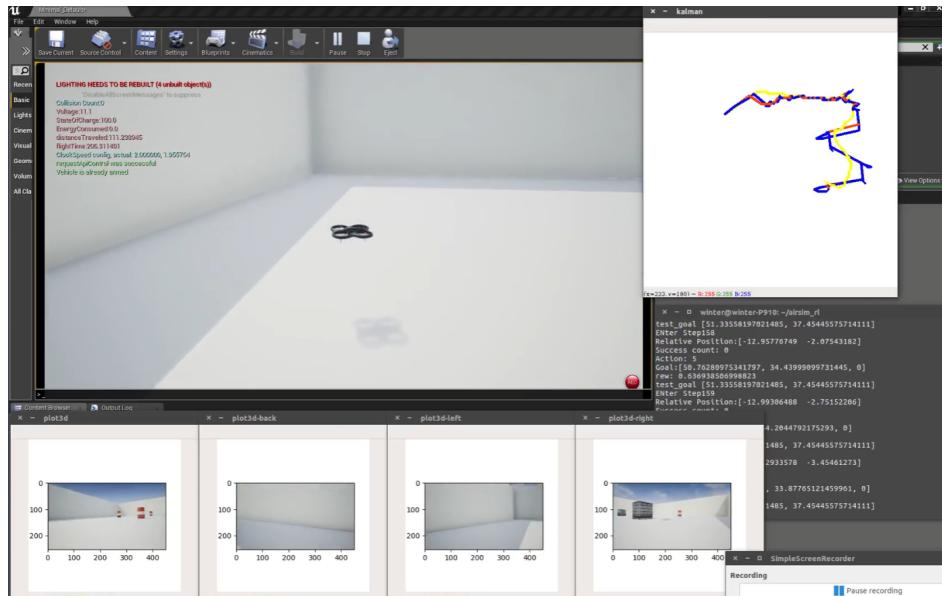


图 5.10 目标物体运动到左下图中的油箱后面，此时检测算法并没有有效标出物体，但是卡尔曼滤波算法的蓝色曲线仍然给出了物体预测位置，并控制无人机飞向预测位置从而继续完成追踪任务

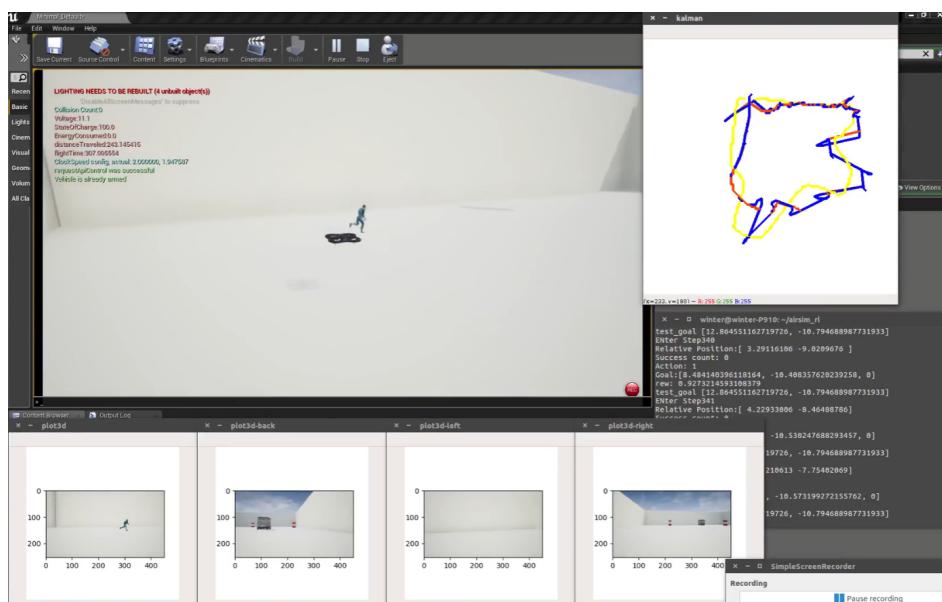


图 5.11 本文设计的无人机目标追踪算法完成追踪任务示意图

结论

基于机器学习方法实现无人机系统的自主导航与目标追踪任务，与传统方法相比难度和挑战性较大。本文综合参考了目前相关领域的诸多工作，深入地研究了主流强化学习方法实现自主导航功能的问题和弊端，提出了可行的解决方案。同时结合本文提出的轻量级 3D 目标检测算法，最终设计出了可行的目标追踪系统流程，并且在仿真环境中成功实现了无人机目标追踪功能，本文的重点研究内容与成果可以分为以下几个部分：

(1) 针对现有强化学习方法在无人机自主导航任务上所存在的问题，基于 SAC 算法设计了适用于本文场景的课程学习过程以及增强避障能力的自注意力网络层，进而提出了改进的 CLSA-SAC 算法。其中课程学习过程可以帮助强化学习模型阶段性探索有效样本并减轻模型的学习压力，逐渐习得复杂的导航任务；额外设计的自注意力机制层，能够帮助算法学到更有效的避障行为，同时本文还对自注意力机制层做了可解释性分析，进一步验证了自注意力层的作用。

(2) 本文提出了一种基于 CenterNet 网络所改进的轻量级 3D 目标检测算法 3D-CenterNet。该算法在 CenterNet 的网络架构基础上设计了新的 3D 参数回归层，同时为了更好地提取多尺度特征，本文还设计了一种新的特征融合层，并且在特征融合的采样阶段引入了可形变卷积，可以帮助网络更有效地提取多尺度特征。另外在训练过程中，本文还引入了多参数解耦损失，将网络的输出进行 3 维空间内的解耦，使得预测的 3D 参数更加合理，这些设计最终有效地增加了模型的训练稳定性与检测性能。

(3) 基于 Airsim 与 UE4 平台，本文设计并实现了高性能的无人机自主导航与目标追踪系统的仿真训练环境，该环境支持域随机化等操作，使得实现课程学习、减少域漂移现象以及增加训练稳定性都有所保障。基于本文所提出的自主导航算法与 3D 目标检测算法，以及额外设计的细粒度目标匹配和卡尔曼滤波过程，本文最终在上述仿真系统中成功实现了无人机目标追踪功能。相比于传统实现方法，该系统的设计大大减少了算法计算复杂度，降低了算力需求。

本文所研究的基于机器学习的无人机相关算法是未来无人机技术发展的重要方向，目前该领域越来越多的算法开始有效结合机器学习方法来提高性能表现，本文的研究可以为之后的工作提供一个简单的思路和指导。但是由于时间有限，且本文所研究的自主导航和目标追踪两个主要内容涉及的技术领域较多，本文的研究结果还存在

一些不足。未来的研究工作可以围绕以下几个方面进行开展：

(1) 在引入强化学习解决自主导航任务时，避障效果主要取决于输入的样本数据的质量，因此碰撞情形有时无法避免。针对这个缺点如何结合现有的优化与路径规划方法，设计一种更加可靠的自主导航算法，在无需地图的前提下实现可解释的理论上完全可行的避障算法。

(2) 本文所设计的 3D 目标检测模型虽然在高性能 GPU 上的前向运行时间仅仅只有 0.07s，但是如何压缩网络模型部署到低性能的可移动图形处理设备上仍有待解决，这个过程中模型压缩和蒸馏设计是十分重要的，值得进一步研究。

(3) 如何更加有效的进行目标行为预测，本文对目标行为的预测仅仅利用了线性一阶卡尔曼滤波模型，但是现实生活中车辆和行人的运动模式显然更为复杂，如何结合深度学习技术在不同场景下对目标行为作出合理预测也是十分重要的一个研究方向。

参考文献

- [1] Aguilar W G, Salcedo V S, Sandoval D S, et al. Developing of a video-based model for uav autonomous navigation[C]. Latin American Workshop on Computational Neuroscience. Springer, 2017: 94–105.
- [2] Wise R, Rysdyk R. Uav coordination for autonomous target tracking[C]. AIAA Guidance, Navigation, and Control Conference and Exhibit. 2006: 6453.
- [3] Goerzen C, Kong Z, Mettler B. A survey of motion planning algorithms from the perspective of autonomous uav guidance[J]. Journal of Intelligent and Robotic Systems, 2010, 57(1): 65–100.
- [4] Weiss S, Scaramuzza D, Siegwart R. Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments[J]. Journal of Field Robotics, 2011, 28(6): 854–874.
- [5] Brockers R, Hummenberger M, Weiss S, et al. Towards autonomous navigation of miniature uav [C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2014: 631–637.
- [6] Imanberdiyev N, Fu C, Kayacan E, et al. Autonomous navigation of uav by using real-time model-based reinforcement learning[C]. 2016 14th international conference on control, automation, robotics and vision (ICARCV). IEEE, 2016: 1–6.
- [7] Wang C, Wang J, Zhang X, et al. Autonomous navigation of uav in large-scale unknown complex environment with deep reinforcement learning[C]. 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP). Ieee, 2017: 858–862.
- [8] Mur-Artal R, Montiel J M M, Tardos J D. Orb-slam: a versatile and accurate monocular slam system [J]. IEEE transactions on robotics, 2015, 31(5): 1147–1163.
- [9] Mur-Artal R, Tardós J D. Orb-slam2: An open-source slam system for monocular, stereo, and rgbd cameras[J]. IEEE transactions on robotics, 2017, 33(5): 1255–1262.
- [10] Pritsker A A B. Introduction to simulation and slam ii[M]. Halsted Press, 1984.
- [11] Tisdale J, Kim Z, Hedrick J K. Autonomous uav path planning and estimation[J]. IEEE Robotics & Automation Magazine, 2009, 16(2): 35–42.
- [12] Zhao Y, Zheng Z, Liu Y. Survey on computational-intelligence-based uav path planning[J]. Knowledge-Based Systems, 2018, 158: 54–64.
- [13] Meier L, Honegger D, Pollefeys M. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms[C]. 2015 IEEE international conference on robotics and automation (ICRA). IEEE, 2015: 6235–6240.

- [14] Salih A L, Moghavvemi M, Mohamed H A, et al. Flight pid controller design for a uav quadrotor [J]. *Scientific research and essays*, 2010, 5(23): 3660–3667.
- [15] Li Y. Deep reinforcement learning: An overview[J]. *arXiv preprint arXiv:1701.07274*, 2017.
- [16] Arulkumaran K, Deisenroth M P, Brundage M, et al. Deep reinforcement learning: A brief survey [J]. *IEEE Signal Processing Magazine*, 2017, 34(6): 26–38.
- [17] Henderson P, Islam R, Bachman P, et al. Deep reinforcement learning that matters[C]. *Proceedings of the AAAI conference on artificial intelligence: volume 32*. 2018.
- [18] LeCun Y, Bengio Y, Hinton G. Deep learning[J]. *nature*, 2015, 521(7553): 436–444.
- [19] Goodfellow I, Bengio Y, Courville A. Deep learning[M]. MIT press, 2016.
- [20] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. MIT press, 2018.
- [21] Çalışır S, Pehlivanoğlu M K. Model-free reinforcement learning algorithms: A survey[C]. 2019 27th Signal Processing and Communications Applications Conference (SIU). IEEE, 2019: 1–4.
- [22] Tai L, Paolo G, Liu M. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation[C]. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017: 31–36.
- [23] Pfeiffer M, Shukla S, Turchetta M, et al. Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations[J]. *IEEE Robotics and Automation Letters*, 2018, 3(4): 4423–4430.
- [24] Zou Z, Shi Z, Guo Y, et al. Object detection in 20 years: A survey[J]. *arXiv preprint arXiv:1905.05055*, 2019.
- [25] Ciaparrone G, Sánchez F L, Tabik S, et al. Deep learning in video multi-object tracking: A survey [J]. *Neurocomputing*, 2020, 381: 61–88.
- [26] Torralba A, Oliva A. Depth estimation from image structure[J]. *IEEE Transactions on pattern analysis and machine intelligence*, 2002, 24(9): 1226–1238.
- [27] Chen X, Ma H, Wan J, et al. Multi-view 3d object detection network for autonomous driving[C]. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017: 1907–1915.
- [28] Tai L, Liu M. A robot exploration strategy based on q-learning network[C]. 2016 ieee international conference on real-time computing and robotics (rcar). IEEE, 2016: 57–62.
- [29] Pham H X, La H M, Feil-Seifer D, et al. Autonomous uav navigation using reinforcement learning [J]. *arXiv preprint arXiv:1801.05086*, 2018.
- [30] Ko B, Choi H J, Hong C, et al. Neural network-based autonomous navigation for a homecare mobile

- robot[C]. 2017 IEEE International Conference on Big Data and Smart Computing (BigComp). IEEE, 2017: 403–406.
- [31] Wooden D, Malchano M, Blankespoor K, et al. Autonomous navigation for bigdog[C]. 2010 IEEE international conference on robotics and automation. Ieee, 2010: 4736–4741.
- [32] Lowe D G. Distinctive image features from scale-invariant keypoints[J]. International journal of computer vision, 2004, 60(2): 91–110.
- [33] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[J]. arXiv preprint arXiv:1312.5602, 2013.
- [34] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of go without human knowledge [J]. nature, 2017, 550(7676): 354–359.
- [35] Sallab A E, Abdou M, Perot E, et al. Deep reinforcement learning framework for autonomous driving[J]. Electronic Imaging, 2017, 2017(19): 70–76.
- [36] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning [J]. nature, 2015, 518(7540): 529–533.
- [37] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning[C]. International conference on machine learning. PMLR, 2016: 1928–1937.
- [38] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. arXiv preprint arXiv:1509.02971, 2015.
- [39] Chen T, Gupta S, Gupta A. Learning exploration policies for navigation[J]. arXiv preprint arXiv:1903.01959, 2019.
- [40] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012, 25.
- [41] Watkins C J, Dayan P. Q-learning[J]. Machine learning, 1992, 8(3): 279–292.
- [42] Chaplot D S, Gandhi D, Gupta S, et al. Learning to explore using active neural slam[J]. arXiv preprint arXiv:2004.05155, 2020.
- [43] Botvinick M M. Hierarchical reinforcement learning and decision making[J]. Current opinion in neurobiology, 2012, 22(6): 956–962.
- [44] Sudowe P, Leibe B. Efficient use of geometric constraints for sliding-window object detection in video[C]. International Conference on Computer Vision Systems. Springer, 2011: 11–20.
- [45] Uijlings J R, Van De Sande K E, Gevers T, et al. Selective search for object recognition[J]. International journal of computer vision, 2013, 104(2): 154–171.
- [46] Cherkassky V, Ma Y. Practical selection of svm parameters and noise estimation for svm regression

- [J]. Neural networks, 2004, 17(1): 113–126.
- [47] Yu S H, Kim D H, Lee S L, et al. Sift based image similarity search using an edge image pyramid and an interesting region detection[J]. Journal of KIISE: Databases, 2008, 35(4): 345–355.
- [48] Wang X, Han T X, Yan S. An hog-lbp human detector with partial occlusion handling[C]. 2009 IEEE 12th international conference on computer vision. IEEE, 2009: 32–39.
- [49] Viola P, Jones M. Rapid object detection using a boosted cascade of simple features[C]. Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001: volume 1. Ieee, 2001: I–I.
- [50] Papageorgiou C P, Oren M, Poggio T. A general framework for object detection[C]. Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271). IEEE, 1998: 555–562.
- [51] Felzenszwalb P, McAllester D, Ramanan D. A discriminatively trained, multiscale, deformable part model[C]. 2008 IEEE conference on computer vision and pattern recognition. Ieee, 2008: 1–8.
- [52] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]. European conference on computer vision. Springer, 2016: 21–37.
- [53] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779–788.
- [54] Girshick R. Fast r-cnn[C]. Proceedings of the IEEE international conference on computer vision. 2015: 1440–1448.
- [55] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3431–3440.
- [56] Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps[J]. arXiv preprint arXiv:1312.6034, 2013.
- [57] Liu F, Shen C, Lin G. Deep convolutional neural fields for depth estimation from a single image[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 5162–5170.
- [58] Wang C, Buenaposada J M, Zhu R, et al. Learning depth from monocular videos using direct methods[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 2022–2030.
- [59] Arnold E, Al-Jarrah O Y, Dianati M, et al. A survey on 3d object detection methods for autonomous driving applications[J]. IEEE Transactions on Intelligent Transportation Systems, 2019, 20(10): 3782–3795.
- [60] Liang W, Xu P, Guo L, et al. A survey of 3d object detection[J]. Multimedia Tools and Applications, 2021, 80(19): 29617–29641.

- [61] Zhou Y, Tuzel O. Voxelnet: End-to-end learning for point cloud based 3d object detection[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 4490–4499.
- [62] Ku J, Mozifian M, Lee J, et al. Joint 3d proposal generation and object detection from view aggregation[C]. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018: 1–8.
- [63] Puterman M L. Markov decision processes[J]. Handbooks in operations research and management science, 1990, 2: 331–434.
- [64] Mahadevan S. To discount or not to discount in reinforcement learning: A case study comparing r learning and q learning[M]. Machine Learning Proceedings 1994. Elsevier, 1994: 164–172.
- [65] Greensmith E, Bartlett P L, Baxter J. Variance reduction techniques for gradient estimates in reinforcement learning.[J]. Journal of Machine Learning Research, 2004, 5(9).
- [66] Baird L C. Reinforcement learning in continuous time: Advantage updating[C]. Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94): volume 4. IEEE, 1994: 2448–2453.
- [67] Barron E, Ishii H. The bellman equation for minimizing the maximum cost[J]. Nonlinear Analysis: Theory, Methods & Applications, 1989, 13(9): 1067–1090.
- [68] Meyn S P. The policy iteration algorithm for average reward markov decision processes with general state space[J]. IEEE Transactions on Automatic Control, 1997, 42(12): 1663–1680.
- [69] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning[C]. Proceedings of the AAAI conference on artificial intelligence: volume 30. 2016.
- [70] Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Machine learning, 1992, 8(3): 229–256.
- [71] Tamar A, Di Castro D, Mannor S. Learning the variance of the reward-to-go[J]. The Journal of Machine Learning Research, 2016, 17(1): 361–396.
- [72] Shah S, Dey D, Lovett C, et al. Airsim: High-fidelity visual and physical simulation for autonomous vehicles[C]. Field and service robotics. Springer, 2018: 621–635.
- [73] Tobin J, Fong R, Ray A, et al. Domain randomization for transferring deep neural networks from simulation to the real world[C]. 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2017: 23–30.
- [74] Taylor M E, Stone P. Transfer learning for reinforcement learning domains: A survey.[J]. Journal of Machine Learning Research, 2009, 10(7).
- [75] Cassandra A R. A survey of pomdp applications[C]. Working notes of AAAI 1998 fall symposium

- on planning with partially observable Markov decision processes: volume 1724. 1998.
- [76] Krishnan S, Boroujerdian B, Fu W, et al. Air learning: a deep reinforcement learning gym for autonomous aerial robot visual navigation[J]. *Machine Learning*, 2021, 110(9): 2501–2540.
- [77] Zhu Y, Mottaghi R, Kolve E, et al. Target-driven visual navigation in indoor scenes using deep reinforcement learning[C]. 2017 IEEE international conference on robotics and automation (ICRA). IEEE, 2017: 3357–3364.
- [78] Xie L, Wang S, Markham A, et al. Towards monocular vision based obstacle avoidance through deep reinforcement learning[J]. arXiv preprint arXiv:1706.09829, 2017.
- [79] Schulman J, Levine S, Abbeel P, et al. Trust region policy optimization[C]. International conference on machine learning. PMLR, 2015: 1889–1897.
- [80] Engstrom L, Ilyas A, Santurkar S, et al. Implementation matters in deep policy gradients: A case study on ppo and trpo[J]. arXiv preprint arXiv:2005.12729, 2020.
- [81] Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor[C]. International conference on machine learning. PMLR, 2018: 1861–1870.
- [82] Haarnoja T, Tang H, Abbeel P, et al. Reinforcement learning with deep energy-based policies[C]. International Conference on Machine Learning. PMLR, 2017: 1352–1361.
- [83] Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods[C]. International conference on machine learning. PMLR, 2018: 1587–1596.
- [84] Kingma D P, Salimans T, Welling M. Variational dropout and the local reparameterization trick[J]. Advances in neural information processing systems, 2015, 28.
- [85] Schulman J, Moritz P, Levine S, et al. High-dimensional continuous control using generalized advantage estimation[J]. arXiv preprint arXiv:1506.02438, 2015.
- [86] Burda Y, Edwards H, Storkey A, et al. Exploration by random network distillation[J]. arXiv preprint arXiv:1810.12894, 2018.
- [87] Bellemare M, Srinivasan S, Ostrovski G, et al. Unifying count-based exploration and intrinsic motivation[J]. Advances in neural information processing systems, 2016, 29.
- [88] Burda Y, Edwards H, Pathak D, et al. Large-scale study of curiosity-driven learning[J]. arXiv preprint arXiv:1808.04355, 2018.
- [89] Elman J L. Learning and development in neural networks: The importance of starting small[J]. *Cognition*, 1993, 48(1): 71–99.
- [90] Bengio Y, Louradour J, Collobert R, et al. Curriculum learning[C]. Proceedings of the 26th annual

- international conference on machine learning. 2009: 41–48.
- [91] Zaremba W, Sutskever I. Learning to execute[J]. arXiv preprint arXiv:1410.4615, 2014.
- [92] Weinshall D, Cohen G, Amir D. Curriculum learning by transfer learning: Theory and experiments with deep networks[C]. International Conference on Machine Learning. PMLR, 2018: 5238–5246.
- [93] Akkaya I, Andrychowicz M, Chociej M, et al. Solving rubik’s cube with a robot hand[J]. arXiv preprint arXiv:1910.07113, 2019.
- [94] van der Heijden A H. Selective attention in vision[M]. Routledge, 2003.
- [95] Han K, Wang Y, Chen H, et al. A survey on visual transformer[J]. arXiv e-prints, 2020: arXiv–2012.
- [96] Galassi A, Lippi M, Torroni P. Attention in natural language processing[J]. IEEE Transactions on Neural Networks and Learning Systems, 2020, 32(10): 4291–4308.
- [97] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate [J]. arXiv preprint arXiv:1409.0473, 2014.
- [98] Cheng J, Dong L, Lapata M. Long short-term memory-networks for machine reading[J]. arXiv preprint arXiv:1601.06733, 2016.
- [99] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.
- [100] Zhang H, Goodfellow I, Metaxas D, et al. Self-attention generative adversarial networks[C]. International conference on machine learning. PMLR, 2019: 7354–7363.
- [101] Wang X, Girshick R, Gupta A, et al. Non-local neural networks[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 7794–7803.
- [102] Tang Y, Nguyen D, Ha D. Neuroevolution of self-interpretable agents[C]. Proceedings of the 2020 Genetic and Evolutionary Computation Conference. 2020: 414–424.
- [103] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 580–587.
- [104] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[J]. Advances in neural information processing systems, 2015, 28.
- [105] Lin T Y, Goyal P, Girshick R, et al. Focal loss for dense object detection[C]. Proceedings of the IEEE international conference on computer vision. 2017: 2980–2988.
- [106] Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2117–2125.
- [107] Zhou X, Wang D, Krähenbühl P. Objects as points[J]. arXiv preprint arXiv:1904.07850, 2019.

- [108] Law H, Deng J. Cornernet: Detecting objects as paired keypoints[C]. Proceedings of the European conference on computer vision (ECCV). 2018: 734–750.
- [109] Chen X, Kundu K, Zhang Z, et al. Monocular 3d object detection for autonomous driving[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2147–2156.
- [110] Mousavian A, Anguelov D, Flynn J, et al. 3d bounding box estimation using deep learning and geometry[C]. Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2017: 7074–7082.
- [111] Brazil G, Liu X. M3d-rpn: Monocular 3d region proposal network for object detection[C]. Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 9287–9296.
- [112] Yu F, Wang D, Shelhamer E, et al. Deep layer aggregation[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 2403–2412.
- [113] Dai J, Qi H, Xiong Y, et al. Deformable convolutional networks[C]. Proceedings of the IEEE international conference on computer vision. 2017: 764–773.
- [114] Simonelli A, Bulo S R, Porzi L, et al. Disentangling monocular 3d object detection[C]. Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 1991–1999.

附录 A 相关证明

1. 策略评估过程贝尔曼方程收敛性证明

定义关于策略 π 的贝尔曼算子 \mathcal{B}_π (Bellman Expectation Backup Operator) 为：

$$\mathcal{B}_\pi U(s) := \sum_{a \in \mathcal{A}} \pi(a | s) \sum_{s' \in \mathcal{S}} p(s' | s, a) [r(s, a, s') + \gamma U(s')], \quad \forall s \in \mathcal{S} \quad (\text{A-1})$$

上式中贝尔曼算子是对 $U(s)$ 的操作。下面证明贝尔曼算子 \mathcal{B}_π 是一个收缩映射。根据贝尔曼算子的定义，有：

$$\begin{aligned} |\mathcal{B}_\pi U_1(s) - \mathcal{B}_\pi U_2(s)| &= \left| \sum_{a \in \mathcal{A}} \pi(a | s) \sum_{s' \in \mathcal{S}} p(s' | s, a) [\gamma (U_1(s') - U_2(s'))] \right| \\ &\leq \gamma \sum_{a \in \mathcal{A}} \pi(a | s) \sum_{s' \in \mathcal{S}} p(s' | s, a) |(U_1(s') - U_2(s'))| \\ &\leq \gamma \sum_{a \in \mathcal{A}} \pi(a | s) \sum_{s' \in \mathcal{S}} p(s' | s, a) \left(\max_{s'' \in \mathcal{S}} |U_1(s'') - U_2(s'')| \right) \quad (\text{A-2}) \\ &= \gamma \max_{s'' \in \mathcal{S}} |U_1(s'') - U_2(s'')| \\ &= \gamma \|U_1 - U_2\|_\infty \end{aligned}$$

因此，对于任意的 s 都有A-2成立，所以式A-2可以写成：

$$\|\mathcal{B}_\pi U_1 - \mathcal{B}_\pi U_2\|_\infty \leq \gamma \|U_1 - U_2\|_\infty \quad (\text{A-3})$$

当 $\gamma < 1$ 时贝尔曼算子是一个严格的收缩映射，下面证明序列 $\{U, \mathcal{B}_\pi U, \mathcal{B}_\pi^2 U, \dots\}$ 是收敛的：

$$\begin{aligned} \|\mathcal{B}_\pi^{m+1} U - \mathcal{B}_\pi^m U\|_\infty &\leq \gamma \|\mathcal{B}_\pi^m U - \mathcal{B}_\pi^{m-1} U\|_\infty \\ &\leq \gamma^2 \|\mathcal{B}_\pi^{m-1} U - \mathcal{B}_\pi^{m-2} U\|_\infty \\ &\quad \dots \\ &\leq \gamma^m \|\mathcal{B}_\pi U - U\|_\infty \end{aligned} \quad (\text{A-4})$$

根据A-4可得，当 m 趋近于无穷时，序列列 $\{U, \mathcal{B}_\pi U, \mathcal{B}_\pi^2 U, \dots\}$ 会收敛到一个值，不断利用贝尔曼算子对 $V(S)$ 进行迭代，所得序列会收敛到唯一不动点，即收敛性证明完

毕。

2. 策略改进中的策略的价值函数 $V'_\pi(s)$ 一定会优于原有策略的价值函数 $V_\pi(s)$ 的证明

一般来说, 如果 π 和 π' 是任意的两个确定的策略, 对任意 $s \in \mathcal{S}$, 有 $Q_\pi(s, \pi'(s)) \geq V_\pi(s)$, 那么我们称 π' 比 π 更好或者一样好, 这也就意味着:

$$V_{\pi'}(s) \geq V_\pi(s), \quad s \in \mathcal{S} \quad (\text{A-5})$$

上式证明过程为:

$$\begin{aligned} V_\pi(s) &\leq Q_\pi(s, \pi'(s)) \\ &= \mathbb{E}[r_{t+1} + \gamma V_\pi(s_{t+1}) \mid s_t = s, a_t = \pi'(s)] \\ &= \mathbb{E}_{\pi'}[r_{t+1} + \gamma V_\pi(s_{t+1}) \mid s_t = s] \\ &\leq \mathbb{E}_{\pi'}[r_{t+1} + \gamma q_\pi(s_{t+1}, \pi'(s_{t+1})) \mid s_t = s] \\ &= \mathbb{E}_{\pi'}[r_{t+1} + \gamma \mathbb{E}[r_{t+2} + \gamma V_\pi(s_{t+2}) \mid s_{t+1}, a_{t+1} = \pi'(s_{t+1})] \mid s_t = s] \quad (\text{A-6}) \\ &= \mathbb{E}_{\pi'}[r_{t+1} + \gamma r_{t+2} + \gamma^2 V_\pi(s_{t+2}) \mid s_t = s] \\ &\leq \mathbb{E}_{\pi'}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V_\pi(s_{t+3}) \mid s_t = s] \\ &\vdots \\ &\leq \mathbb{E}_{\pi'}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \dots \mid s_t = s] \\ &= V_{\pi'}(s) \end{aligned}$$

上式意味着对于当前状态 s , 只要根据 $Q_\pi(s, \pi(s))$, 找到一个最优的动作 (即最大动作价值函数对应的动作), 所得到的新策略的 $V'_\pi(s)$ 一定会优于或者等于原始策略。这样构造出来的贪心策略 (当前取最优) 满足策略改进的条件, 所以新的策略一定比原来的策略要好或者一样好, 得到的新策略为:

$$\begin{aligned} \pi'(s) &:= \arg \max_{a \in \mathcal{A}} Q_\pi(s, a) \\ &= \arg \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} p(s' \mid s, a) [r(s, a, s') + \gamma V_\pi(s')] \quad (\text{A-7}) \end{aligned}$$

式A-7一般称之为策略改进过程。

3. 价值迭代的收敛性与唯一性证明

首先定义贝尔曼最优算子为 \mathcal{B}_* (Bellman optimality backup operator) :

$$\mathcal{B}_*U(s) := \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} p(s' | s, a) [r(s, a, s') + \gamma U(s')], \quad \forall s \in \mathcal{S} \quad (\text{A-8})$$

那么有:

$$\begin{aligned} |\mathcal{B}_*U_1(s) - \mathcal{B}_*U_2(s)| &= \left| \max_{a_1 \in \mathcal{A}} \sum_{s' \in \mathcal{S}} p(s' | s, a_1) [r(s, a_1, s') + \gamma U_1(s')] - \right. \\ &\quad \left. \max_{a_2 \in \mathcal{A}} \sum_{s' \in \mathcal{S}} p(s' | s, a_2) [r(s, a_2, s') + \gamma U_2(s')] \right| \\ &\leq \left| \max_{a_1 \in \mathcal{A}} \left(\sum_{s' \in \mathcal{S}} p(s' | s, a_1) [r(s, a_1, s') + \gamma U_1(s')] - \right. \right. \\ &\quad \left. \left. \sum_{s' \in \mathcal{S}} p(s' | s, a_1) [r(s, a_1, s') + \gamma U_2(s')] \right) \right| \\ &= \gamma \left| \max_{a \in \mathcal{A}} \left(\sum_{s' \in \mathcal{S}} p(s' | s, a) [U_1(s') - U_2(s')] \right) \right| \quad (\text{A-9}) \\ &\leq \gamma \max_{a \in \mathcal{A}} \left| \sum_{s' \in \mathcal{S}} p(s' | s, a) [U_1(s') - U_2(s')] \right| \\ &\leq \gamma \max_{a \in \mathcal{A}} \left[\sum_{s' \in \mathcal{S}} p(s' | s, a) |U_1(s') - U_2(s')| \right] \\ &\leq \gamma \max_{a \in \mathcal{A}} \left[\sum_{s' \in \mathcal{S}} p(s' | s, a) \|U_1(s) - U_2(s)\|_\infty \right] \\ &= \gamma \max_{a \in \mathcal{A}} \|U_1(s) - U_2(s)\|_\infty = \gamma \|U_1(s) - U_2(s)\|_\infty \end{aligned}$$

由此可知 \mathcal{B}_* 是一个收缩映射, 那么:

$$\begin{aligned} \|\mathcal{B}_*^{m+1}U - \mathcal{B}_*^mU\|_\infty &\leq \gamma \|\mathcal{B}_*^mU - \mathcal{B}_*^{m-1}U\|_\infty \\ &\leq \gamma^2 \|\mathcal{B}_*^{m-1}U - \mathcal{B}_*^{m-2}U\|_\infty \\ &\quad \cdots \\ &\leq \gamma^m \|\mathcal{B}_*U - U\|_\infty \end{aligned} \quad (\text{A-10})$$

当 m 趋近于无穷时, 序列 $\{U, \mathcal{B}_*U, \mathcal{B}_*^2U, \dots\}$ 会收敛到一个值, 我们不断利用贝尔曼算子对 $V_*(s)$ 进行迭代, 所得序列会收敛到一个不动点。以上则为值迭代的收敛性证明, 这表明值迭代过程一定存在一个不动点, 下面再来证明这个不动点的唯一性。

首先，假设 \mathcal{B}_* 含有两个不动点 U, V 且 $U \neq V$ 。那么一定有 $\|U - V\|_\infty > 0$ ，由于 U, V 都是不动点，也有 $\|\mathcal{B}_*U - \mathcal{B}_*V\|_\infty = \|U - V\|_\infty$ ，但是由于设 \mathcal{B}_* 是一个压缩映射，根据上面的证明有 $\|\mathcal{B}_*U - \mathcal{B}_*V\|_\infty \leq \gamma\|U - V\|_\infty < \|U - V\|_\infty$ ，因此假设不成立。所以，不动点一定是唯一的。

4.SAC 算法策略改进的证明

首先证明，在满足如下策略改进时：

$$\tilde{\pi}(\cdot | s) \propto \exp(Q_{soft}^\pi(s, \cdot)), \quad \forall s \quad (\text{A-11})$$

策略对应的 soft-q 价值函数一定是提高的，即：

$$Q_{soft}^{\tilde{\pi}}(s, a) \geq Q_{soft}^\pi(s, a) \quad \forall s, a \quad (\text{A-12})$$

通过 SAC 的建模，可以得到下式：

$$\mathcal{H}(\pi(\cdot | s)) + \mathbb{E}_{a \sim \pi}[Q_{soft}^\pi(s, a)] = -D_{KL}(\pi(\cdot | s) \| \tilde{\pi}(\cdot | s)) + \log \int \exp(Q_{soft}^\pi(s, a)) da \quad (\text{A-13})$$

其证明过程为：

$$\begin{aligned} \text{right} &= - \int \pi \log \frac{\pi}{\tilde{\pi}} + \log \int \exp(Q_{soft}^\pi(s, a)) da \\ \text{right} &= - \int \pi \log \pi + \int \pi \log \tilde{\pi} + \int \pi \log \int \exp(Q_{soft}^\pi(s, a)) da, \int \exp(Q_{soft}^\pi(s, a)) da \text{ 与 } a \text{ 无关} \\ \text{right} &= - \int \pi \log \pi + \int \pi \log \frac{\exp(Q_{soft}^\pi(s, a))}{\int \exp(Q_{soft}^\pi(s, a)) da} + \int \pi \log \int \exp(Q_{soft}^\pi(s, a)) da \\ \text{right} &= - \int \pi \log \pi + \int \pi Q_{soft}^\pi(s, a) da = \text{左边} \end{aligned} \quad (\text{A-14})$$

那么，也可以得到：

$$H(\tilde{\pi}(s)) + E_{a \sim \tilde{\pi}}[Q_{soft}^\pi(s, a)] = \log \int \exp(Q_{soft}^\pi(s, a)) da \quad (\text{A-15})$$

接下来证明策略一定会提升：

$$\begin{aligned}
Q_{\text{soft}}^{\pi}(s, a) &= \mathbb{E}_{s_1} [r_0 + \gamma (\mathcal{H}(\pi(\cdot | s_1)) + \mathbb{E}_{a_1 \sim \pi} [Q_{\text{soft}}^{\pi}(s_1, a_1)])] \\
&\leq \mathbb{E}_{s_1} [r_0 + \gamma (\mathcal{H}(\tilde{\pi}(\cdot | s_1)) + \mathbb{E}_{a_1 \sim \pi} [Q_{\text{soft}}^{\pi}(s_1, a_1)])] \\
&= \mathbb{E}_{s_1} [r_0 + \gamma (\mathcal{H}(\tilde{\pi}(\cdot | s_1)) + r_1)] + \gamma^2 \mathbb{E}_{s_2} [\mathcal{H}(\pi(\cdot | s_2)) + \mathbb{E}_{a_2 \sim \pi} [Q_{\text{soft}}^{\pi}(s_2, a_2)]] \\
&\leq \mathbb{E}_{s_1} [r_0 + \gamma (\mathcal{H}(\tilde{\pi}(\cdot | s_1)) + r_1)] + \gamma^2 \mathbb{E}_{s_2} [\mathcal{H}(\tilde{\pi}(\cdot | s_2)) + \mathbb{E}_{a_2 \sim \pi} [Q_{\text{soft}}^{\pi}(s_2, a_2)]] \\
&= \mathbb{E}_{s_1 a_2 \sim \tilde{\pi}, s_2} [r_0 + \gamma (\mathcal{H}(\tilde{\pi}(\cdot | s_1)) + r_1) + \gamma^2 (\mathcal{H}(\tilde{\pi}(\cdot | s_2)) + r_2)] \\
&\quad + \gamma^3 \mathbb{E}_{s_3} [\mathcal{H}(\tilde{\pi}(\cdot | s_3)) + \mathbb{E}_{a_3 \sim \tilde{\pi}} [Q_{\text{soft}}^{\pi}(s_3, a_3)]] \\
&\vdots \\
&\leq \mathbb{E}_{\tau \sim \tilde{\pi}} [r_0 + \sum_{t=1}^{\infty} \gamma^t (\mathcal{H}(\tilde{\pi}(\cdot | s_t)) + r_t)] \\
&= Q_{\text{soft}}^{\tilde{\pi}}(s, a).
\end{aligned} \tag{A-16}$$

由于价值函数一定会更优，因此按照这种方法进行策略改进一定会提升策略。