Deep-Reinforcement-Learning-Based Collision Avoidance in UAV Environment

基于深度强化学习的无人机环境中碰撞避免

Sihem Ouahouah, Miloud Bagaa ® , Senior Member, IEEE, Jonathan Prados-Garzon ® , and Tarik Taleb ®

Sihem Ouahouah, Miloud Bagaa ® ，高级会员，IEEE，Jonathan Prados-Garzon ® ，和 Tarik Taleb ®

Abstract-Unmanned aerial vehicles (UAVs) have recently attracted both academia and industry representatives due to their utilization in tremendous emerging applications. Most UAV applications adopt visual line of sight (VLOS) due to ongoing regulations. There is a consensus between industry for extending UAVs' commercial operations to cover the urban and populated area-controlled airspace beyond VLOS (BVLOS). There is ongoing regulation for enabling BVLOS UAV management. Regrettably, this comes with unavoidable challenges related to UAVs' autonomy for detecting and avoiding static and mobile objects. An intelligent component should either be deployed onboard the UAV or at a multiaccess-edge computing (MEC) that can read the gathered data from different UAV's sensors, process them, and then make the right decision to detect and avoid the physical collision. The sensing data should be collected using various sensors but not limited to Lidar, depth camera, video, or ultrasonic. This article proposes probabilistic and deep-reinforcement-learning (DRL)-based algorithms for avoiding collisions while saving energy consumption. The proposed algorithms can be either run on top of the UAV or at the MEC according to the UAV capacity and the task overhead. We have designed and developed our algorithms to work for any environment without a need for any prior knowledge. The proposed solutions have been evaluated in a harsh environment that consists of many UAVs moving randomly in a small area without any correlation. The obtained results demonstrated the efficiency of these solutions for avoiding the collision while saving energy consumption in familiar and unfamiliar environments.

摘要-无人机 (UAVs) 最近吸引了学术界和工业界的代表，因为它们在大量新兴应用中的使用。大多数无人机应用采用视距内飞行 (VLOS)，因为现行的规定。业界普遍同意将无人机的商业运作范围扩展到超出视距 (BVLOS) 的城市和人口密集区域的受控空域。目前正在进行制定法规以实现 BVLOS 无人机管理。遗憾的是，这带来了与无人机自主检测和避开静态和移动物体相关的不可避免的挑战。应该在无人机上部署智能组件，或者在多接入边缘计算 (MEC) 上读取来自不同无人机传感器的收集数据，处理这些数据，然后做出正确决定以检测和避免物理碰撞。感知数据应该使用各种传感器收集，但不限于激光雷达、深度相机、视频或超声波。本文提出了基于概率和深度强化学习 (DRL) 的算法，用于避免碰撞同时节省能耗。根据无人机的容量和任务开销，提出的算法可以运行在无人机之上或 MEC 上。我们设计和开发了我们的算法，使其能够在任何环境中工作，无需任何先验知识。所提出的解决方案已经在恶劣环境中进行了评估，该环境由许多在小型区域内随机移动且没有任何相关性的无人机组成。获得的结果证明了这些解决方案在熟悉和不熟悉的环境中避免碰撞同时节省能耗的效率。

Index Terms-Collision avoidance, deep reinforcement learning, machine learning, multiaccess-edge computing (MEC), unmanned aerial vehicles (UAVs).

索引术语-碰撞避免，深度强化学习，机器学习，多接入边缘计算 (MEC)，无人机 (UAVs)。

I. INTRODUCTION

I. 引言

UNMANNED aerial vehicles (UAVs), commonly rec- ognized as drones, are small, fast, and mobile cyber-physical entities employed in different industrial verticals, including power supply inspection, parcel and package delivery, disaster management, and traffic monitoring [1]. The utilization of UAVs goes beyond industrial and academic purposes to daily personal use. A UAV operator must always be capable of maintaining the visual line of sight (VLOS) of its UAV that is piloting due to ongoing regulations, unaided by any technology other than prescription glasses or contact lenses. While UAVs are used mostly within VLOS, there is enthusiasm toward their utilization beyond VLOS (BVLOS) to enable new emerging applications. Therefore, there is a consensus between industry for attenuation of the regulation by extending UAVs' commercial operations to cover the urban and populated area controlled airspace BVLOS. The latter will be enabled by leveraging a cellular wireless network. 5G system and beyond considers the UAV management BVLOS as one of the essential demonstrators. On the other side, emerging networking paradigms, such as Edge Computing can substitute UAVs to handle high processing flight control applications. Furthermore, GPU vendors allow for realizing different microarchitectures (e.g., Fermi, Maxwell, and Pascal) that might enable real-time and high resourced applications for the UAV's flight control [2].

无人驾驶飞行器 (UAVs)，通常被称为无人机，是小型、快速、移动的赛博物理实体，被应用于不同的工业垂直领域，包括电力供应检查、包裹和快递投送、灾害管理和交通监控 [1]。无人机的作用已经超出了工业和学术目的，进入了日常个人使用。根据现行规定，无人机操作员必须始终能够保持对其操控的无人机的视觉直线 sight(VLOS)，不能借助任何技术，除了处方眼镜或隐形眼镜。虽然无人机大多在 VLOS 内使用，但人们对其在 VLOS 之外 (BVLOS) 的应用充满热情，以启用新的新兴应用。因此，行业之间就通过扩展无人机商业运营范围以覆盖城市和人口密集区域的受控空域 BVLOS 来减轻法规达成共识。后者将通过利用蜂窝无线网络实现。5G 系统及其后续系统将无人机管理 BVLOS 视为关键演示者之一。另一方面，新兴的网络范例，如边缘计算可以替代无人机来处理高处理的飞行控制应用。此外，GPU 供应商允许实现不同的微架构 (例如，Fermi、Maxwell 和 Pascal)，这可能会为无人机的飞行控制提供实时和高资源应用 [2]。

The UAVs' commercial revenue sees considerable growth by the near future [3].The expected increase in the number of UAVs involves new challenges related to their control and management. Efficient solutions for UAV's collision avoidance is one of the challenges that have been widely tackled in the literature in both ground vehicles [4], [5] context, as well as in the context of UAVs [3], [6]-[24]. Different sensors have been leveraged for scanning and detecting objects surroundings UAVs. Some solutions use cameras for detecting mobile and static obstacles around UAVs [6], [14]. Nevertheless, the information provided by video cameras requires intensive processing to be translated into useful information to control UAVs [25].

无人机的商业收入在不久的将来将看到显著增长 [3]。无人机数量的预期增加涉及到了它们控制和管理的新的挑战。无人机避障的有效解决方案是文献中广泛解决的挑战之一，无论是在地面车辆 [4]、[5] 的背景下，还是在无人机 [3]、[6]-[24] 的背景下。不同的传感器已被利用来扫描和探测无人机周围的物体。一些解决方案使用摄像头来探测无人机周围的移动和静态障碍物 [6]、[14]。然而，视频相机提供的信息需要经过密集处理才能转化为控制无人机的有用信息 [25]。

Several works [3], [11], [17], [18] have proposed path planning solutions where the UAVs are provided with their whole trajectories before starting their missions to overcome the limitation mentioned above. The path planning fits well in applications with invariable environment scenarios. However, mostly, UAVs fly in unsettled indoor, urban, and confined areas. Indeed, sensing and path planning approaches' success is highly related to the computational capacity of the UAV, the accuracy of the sensor, and the knowledge's degree on the environment. On another side, reinforcement-learning (RL)-based approaches got much success in emerging topics, including robotic prediction, vehicular ad hoc networks (VANET), and UAVs. For instance, Xiao et al. [26] have provided a heuristic to enhance communication and prevent jamming attacks in VANET by leveraging UAV. An extended version of this work has been suggested in [27] to prevent the jamming attacks in VANET by leveraging both UAV and RL approaches. This success attracted the researchers to use RL to ensure a self-decision-making system for a safe UAV's autonomous flight.

许多研究 [3]、[11]、[17]、[18] 提出了路径规划解决方案，在这些方案中，无人机在开始执行任务之前获得了它们的完整轨迹，以克服上述限制。路径规划非常适合环境场景不变的应用。然而，大多数情况下，无人机在未解决的室内、城市和受限区域飞行。实际上，感知和路径规划方法的成功与无人机的计算能力、传感器的准确性和对环境的了解程度密切相关。另一方面，基于强化学习 (RL) 的方法在新兴话题中取得了很大的成功，包括机器人预测、车载自组织网络 (VANET) 和无人机。例如，Xiao 等人 [26] 提出了一种启发式方法，通过利用无人机增强通信并防止 VANET 中的干扰攻击。这项工作的扩展版本在 [27] 中提出，通过利用无人机和 RL 方法来防止 VANET 中的干扰攻击。这种成功吸引了研究人员使用 RL 来确保无人机自主飞行的自我决策系统安全。

Manuscript received February 1, 2021; revised April 23, 2021; accepted sion March 7, 2022. This work was supported in part by the Spanish National Project TRUE-5G under Grant PID2019-108713RB-C53, and in part by the European Union's Horizon 2020 Research and Innovation Program through the 5G!Drones Project under Grant 857031. (Corresponding author: Miloud Bagaa.)

手稿于 2021 年 2 月 1 日收到，于 2021 年 4 月 23 日修订，于 2022 年 3 月 7 日接受。这项工作得到了西班牙国家项目 TRUE-5G 的部分资助，资助编号为 PID2019-108713RB-C53，以及部分得到了欧盟“地平线 2020”研究和创新计划通过 5G!Drones 项目下的资助编号 857031 的支持。(通讯作者:Miloud Bagaa。)

Sihem Ouahouah and Miloud Bagaa are with the Department of Communications and Networking, School of Electrical Engineering, Aalto University, 00076 Espoo, Finland (e-mail: sihem.ouahouah@aalto.fi; miloud.bagaa@aalto.fi).

Sihem Ouahouah 和 Miloud Bagaa 任职于芬兰阿尔托大学电气工程学院通信和网络系，00076 艾斯波 (电子邮件:sihem.ouahouah@aalto.fi; miloud.bagaa@aalto.fi)。

Jonathan Prados-Garzon is with the Department of Signal Theory, Telematics and Communications, University of Granada, 18014 Granada, Spain (e-mail: jpg@ugr.es).

Jonathan Prados-Garzon 任职于西班牙格拉纳达大学信号理论、远程通信和通信系，18014 格拉纳达 (电子邮件:jpg@ugr.es)。

RL consists in providing a kind of knowledge on the environment based on the previous UAV's experiences. Thus, RL builds the knowledge by interacting with the environment based on a Markov decision process (MDP) model and following one of the RL methods (e.g., Q _learning, deep Q -networks (DQN), Policy_gradient, or Actor_critic). The RL-based UAVs control solutions proposed so far [19]-[24] need large data sets that referrer to the abstraction level used to model the RL environment system (e.g., velocity, wind velocity, etc.). The data sets used in RL-based solutions might return the same limitations pointed in the classical approaches stated above [19]. To overcome these limitations, this article suggests two strategies for avoiding the collision in a UAV environment. The first solution, named a probability-distribution-based collision-avoidance framework (PICA), leverages the probabilistic model for avoiding collisions. In contrast, the second solution, dubbed-RL-based collision-avoidance framework (RELIANCE), leverages DQNs for avoiding collisions.

强化学习 (RL) 涉及根据之前无人机的经验在环境中提供一种知识。因此，强化学习通过与环境的交互并根据马尔可夫决策过程 (MDP) 模型，采用一种强化学习方法 (例如，Q 学习，深度 Q 网络 (DQN)，策略梯度或演员-评论家方法) 来构建知识。迄今为止提出的基于强化学习的无人机控制解决方案 [19]-[24] 需要大量的数据集，这些数据集与用于建模强化学习环境系统的抽象级别相关 (例如，速度、风速等)。基于强化学习的解决方案中使用的数据集可能会带来与上述经典方法中指出的相同的局限性 [19]。为了克服这些局限性，本文提出了两种避免无人机环境中碰撞的策略。第一种解决方案，名为基于概率分布的避碰框架 (PICA)，利用概率模型来避免碰撞。相比之下，第二种解决方案，称为基于强化学习的避碰框架 (RELIANCE)，利用 DQNs 来避免碰撞。

To deal with the disparate UAVs processing capacities and to ease the deployment of the proposed solutions, we suggest two deployment approaches: 1) the UAV's flight controller is deployed onboard or 2) at the multiaccess edge computing (MEC). The first approach convenes UAVs with the new GPU microarchitecture technology is where the agent, either of RELIANCE or PICA, can smoothly make decision and select the best actions. On the other hand, the second way assembles UAVs with limited computing capacities. In order to ensure close management, services running should be migrated among MECs using Follow Me Edge-Cloud concept. Bekkouche et al. [28] suggested a MEC architecture that ensures UAVs' resource provisioning. In case the UAV has a limited resource capacity, the same architecture as proposed in [28] can be adopted. In this case, the RELIANCE/PICA agent is responsible for making decisions at the MEC, and then sending the respective actions to the UAV for controlling its motion.

为了应对不同无人机 (UAVs) 的处理能力差异以及简化所提出解决方案的部署，我们建议了两种部署方法:1) 将无人机的飞行控制器部署在机载上，或者 2) 在多接入边缘计算 (MEC) 上。第一种方法适用于配备了新型 GPU 微架构技术的无人机，在这种情况下，无论是 RELIANCE 还是 PICA 的代理都能顺利做出决策并选择最佳行动。另一方面，第二种方法适用于计算能力有限的无人机。为了确保紧密的管理，运行中的服务应当使用"跟随我边缘-云"概念在 MECs 之间迁移。Bekkouche 等人 [28] 提出了一种 MEC 架构，确保了无人机资源的供应。如果无人机资源容量有限，可以采用 [28] 中提出的相同架构。在这种情况下，RELIANCE/PICA 代理负责在 MEC 上做出决策，然后将相应的行动发送到无人机以控制其运动。

Both algorithms aim to enable autonomous decision making for a UAV while a safe and short flight is ensured to save energy consumption. To ensure a fast convergence of RELIANCE and PICA, we have used a detail-less and generalized state by focusing on the closest part of the environment to the agent. In RELIANCE states' design, we have leveraged partially observable MDP (POMDP) to ensure the generalization and fast convergence. We have used a partially observable state that focuses only on the UAV surrounding to avoid the collisions instead of the whole deployment area. The limitation of the observation at the UAV vicinity helps to reduce the state space by aggregating many observations to a single state. Thanks to this strategy, RELIANCE and PICA avoid overloading computation processing by ignoring useless knowledge. Moreover, this strategy helps the RELIANCE solution to converge quickly by treating many observations as the same state. Furthermore, to ensure the generalization and that both algorithms can work in unseen environments, we have used relative target positions. The benefits of this strategy are twofold: it facilitates and speeds up the convergence of the neural networks (NNs) and, most importantly, improves the generalization of the agent, which is agnostic to the scenario scale.

Tarik Taleb is with the Department of Communications and Networking, School of Electrical Engineering, Aalto University, 00076 Espoo, Finland, and also with the Department of Computer and Information Security, Sejong University, Seoul 05006, South Korea.

Tarik Taleb 任职于芬兰阿尔托大学电气工程学院通信和网络系，00076 艾斯波，同时也在韩国首尔 05006 清静大学计算机与信息安全系。

Digital Object Identifier 10.1109/JIOT.2021.3118949

数字对象标识符 10.1109/JIOT.2021.3118949

We have evaluated and compared both algorithms in terms of collision avoidance and energy saving in familiar and unfamiliar environments. The obtained results demonstrate the ability of both algorithms in the generalization by performing well in unknown environments. Also, the simulation results clearly show the superiority of RELIANCE comparing to PICA.

两种算法均旨在实现无人机 (UAV) 的自主决策，同时确保安全且短暂的飞行以节省能耗。为了保证 RELIANCE 和 PICA 算法的快速收敛，我们采用了无细节的通用状态，专注于距离代理最近的周围环境。在 RELIANCE 状态设计中，我们利用了部分可观测马尔可夫决策过程 (POMDP) 以确保泛化性和快速收敛。我们使用了一个仅关注无人机周围的部分可观测状态来避免碰撞，而不是整个部署区域。对无人机周边观测的限制有助于通过将许多观测聚合到单个状态来减少状态空间。得益于这一策略，RELIANCE 和 PICA 避免了计算处理的过载，忽略了无用的知识。此外，这一策略还有助于 RELIANCE 解决方案通过将许多观测视为相同状态而快速收敛。为了确保泛化性以及两种算法都能在未见过的环境中工作，我们使用了相对目标位置。这一策略的双重好处是: 它促进了神经网络的 (NNs) 收敛速度，并最重要的是，提高了代理的泛化性，使其与场景规模无关。我们在熟悉和陌生的环境中评估和比较了两种算法在避障和节能方面的表现。获得的结果证明了两种算法在未知环境中的泛化能力。同时，仿真结果清楚地显示了 RELIANCE 相对于 PICA 的优越性。

The remainder of this article is organized as follows. Section II reviews the related works. Section III includes our system model and problem statement. In Section IV, the PICA solution is described. An overview on DQN and a RELIANCE solution are detailed in Section V. Section VI presents and discusses the simulation results of PICA and RELIANCE evaluations. Finally, the primary conclusion is drawn in Section VII.

本文其余部分的组织如下。第二节回顾了相关工作。第三节包括我们的系统模型和问题陈述。第四节描述了 PICA 解决方案。第五节详细介绍了 DQN 的概述以及 RELIANCE 解决方案。第六节展示了 PICA 和 RELIANCE 评估的仿真结果并进行了讨论。最后，第七节得出了主要结论。

II. RELATED WORK

II. 相关工作

There is a vast literature to address the collision-avoidance problem in the context of both unmanned ground vehicles [4], [5] and UAVs [3], [6]-[24].

有大量文献致力于解决无人地面车辆 [4], [5] 和无人机 [3], [6]-[24] 的避障问题。

Most of the solutions rely on exact methods [3], [6]-[18], i.e., analytical modeling and optimization techniques, to tackle the UAVs collision-avoidance problem (UCAP). The existing works, based on exact methods, usually considers part of the UCAP aspects to handle its modeling and computational complexity. However, in order to provide a realistic and practical model of the UCAP, many issues have to be taken into account.

大多数解决方案依赖于精确方法 [3], [6]-[18]，即分析建模和优化技术，来解决无人机避障问题 (UCAP)。基于精确方法的现有研究通常只考虑 UCAP 的某些方面，以处理其建模和计算复杂性。然而，为了提供一个现实且实用的 UCAP 模型，必须考虑许多问题。

1) Obstacle Detection: In order to detect the static and mobile objects, the UAVs need to be equipped with onboard sensors. The number of these sensors and their precision might be affected and limited due to several external factors, e.g., specific scenario and UAVs' autonomy. For instance, GPS might not work for indoor scenarios like Industry. Other sensors like radars [8] might be too heavy, energy consuming, and expensive. Then, the concrete set of onboard sensors in UAVs depends on the application and scenario.

1) 障碍物检测: 为了检测静态和移动物体，无人机需要配备机上传感器。这些传感器的数量和精度可能会受到多种外部因素的影响和限制，例如特定场景和无人机的自主性。例如，GPS 在室内场景如工业环境中可能无法工作。其他传感器如雷达 [8] 可能过重、能耗高且昂贵。因此，无人机上的具体传感器集合取决于应用和场景。

2) Sensor Errors: Onboard sensors to detect objects are not error-free. All of them have precision errors, which might be affected by external conditions. For instance, GPS error is affected by weather conditions and follows a Gaussian distribution [3], [29].

2) 传感器误差: 用于检测物体的机上传感器并非无误差。它们都有精度误差，这些误差可能会受到外部条件的影响。例如，GPS 误差会受到天气条件的影响，并遵循高斯分布 [3], [29]。

3) Complex Control: There are several variables to control the UAVs movement, e.g., direction, velocity, and acceleration. Furthermore, these variables strongly depend on external factors, such as wind velocity.

3) 复杂控制: 控制无人机运动有许多变量，例如方向、速度和加速度。此外，这些变量强烈依赖于外部因素，如风速。

4) Different Approaches: There are two different approaches to solve UCAP, namely, path planning and sensing and avoiding [25] methods. Path planning solutions compute the trajectories of the UAVs offline, whereas sensing and avoiding (online) methods determine the movement of the UAVs for small time steps depending on the environment conditions. Sensing and avoiding methods offer higher flexibility and are suitable for a wider range of scenarios. Path planning fits well only for scenarios where the environment remains relatively static during the whole UAVs' mission. The main drawback of the online methods is that typically, the UAV has to run the algorithm (e.g., due to latency constraints), which might exhibit high computational complexity and consume energy.

4) 不同方法: 解决 UCAP 问题有两种不同的方法，分别是路径规划和感测与避障 [25] 方法。路径规划解决方案是在离线状态下计算无人机 (UAVs) 的轨迹，而感测与避障 (在线) 方法则根据环境条件在短时间内确定无人机的移动。感测与避障方法提供了更高的灵活性，适用于更广泛的场景。路径规划仅适用于环境在整个无人机任务期间相对静态的场景。在线方法的主要缺点是，通常无人机必须运行算法 (例如，由于延迟约束)，这可能会显示出高计算复杂性和能源消耗。

In the light of the above, the UCAP requires a high-domain knowledge and its modeling leads to complex or even intractable optimization programs. To overcome these problems, machine learning techniques are particularly attractive for addressing UCAP, so they have been recently received a lot of attention by the research community [19]-[24].

鉴于上述情况，UCAP 需要高度的专业知识，其建模可能导致复杂甚至不可处理的优化程序。为了克服这些问题，机器学习技术在解决 UCAP 问题上特别有吸引力，因此它们最近受到了研究界的大量关注 [19]-[24]。

Choi and Cha [19] provided a comprehensive survey of ML-assisted solutions for autonomous flight. Specifically, they focus on object recognition and UAV's control strategy. The authors conclude that ML is a promising approach to enable stable flight under uncertain environments, though there are still some open issues that need to be carefully addressed. Among them, the existing works do not apply ML in all the UCAP's issues together for autonomous flight. Then, holistic solutions, which cover most of the real-world problems and are suitable for a wider spectrum of scenarios, are required. Furthermore, the existing solutions need large data sets for training. In this regard, they encourage new less data-hungry proposals with a more lightweight training. Finally, they motivate the need for real world tests for a stronger validation of the ML-based UAVs control strategy. Similarly, Fraga-Lamas et al. overview the latest advances on IoT UAV systems controlled by deep learning techniques. They analyze the object detection and collision-avoidance problems and present a survey on the state of the art of deep learning techniques to solve them. Also, they detail the most relevant existing data sets and UAVs' communication architectures. Finally, they identify the open challenges for UCAP. Interestingly, they extract some similar conclusions to the ones drawn in [19]. For instance, the necessity for large amount of data to generate robust models and the difficulty to produce those data.

Choi 和 Cha[19] 对机器学习辅助的自主飞行解决方案进行了全面调研。具体来说，他们专注于目标识别和无人机的控制策略。作者得出结论，尽管仍有一些开放性问题需要仔细解决，但机器学习是使无人机在不确定环境下实现稳定飞行的一个有前景的方法。其中，现有研究并未将机器学习应用于 UCAP 的所有问题以实现自主飞行。因此，需要能够覆盖大多数现实世界问题并适用于更广泛场景的全局解决方案。此外，现有解决方案需要大量数据集进行训练。在这方面，他们鼓励提出新的、对数据需求较小的建议，并采用更轻量级的训练。最后，他们强调了对基于机器学习的无人机控制策略进行现实世界测试以进行更强验证的必要性。同样，Fraga-Lamas 等人概述了通过深度学习技术控制的物联网无人机系统的最新进展。他们分析了目标检测和避障问题，并对解决这些问题的深度学习技术的现状进行了调研。此外，他们详细介绍了最相关的现有数据集和无人机的通信架构。最后，他们确定了 UCAP 的开放挑战。有趣的是，他们得出了一些与 [19] 相似的结论。例如，生成健壮模型需要大量数据的需求以及生成这些数据的困难。

In this article, we propose a simple, yet powerful deep RL (DRL) and probability-distribution-based solutions. Our proposals are suitable for many scenarios, while they need reduced data sets to converge and produce robust models.

在本文中，我们提出了一种简单而强大的深度强化学习 (DRL) 和基于概率分布的解决方案。我们的提议适用于许多场景，同时它们需要较小的数据集来收敛并产生健壮的模型。

III. System Model And Problem Statement

III. 系统模型与问题陈述

In this article, we aim to control the movement of a UAV, hereinafter referred to as UoI (UAV of Interest), while avoiding the collisions with static and mobile objects. Table I summarizes the different notations used in the paper. UoI needs to move within a confined area of dimension $X \times Y$ while avoiding the collisions. The UoI motion begins from a predefined initial position $P_S = (x_S, y_S)$ and stops once achieves a predefined targeted destination $P_T = (x_T, y_T)$. Let $\mathcal{A}$ denote the possible action directions of UoI. As mentioned in [28], the possible movement of a UAV is limited and related to the environment that it works in. For the sake of simplicity and without loss of generality, we consider $\mathcal{A}$ has eight possible directions, $\mathcal{A} = \{Up$, Down, Right, Left, Up_right, Up_left, Down_right, Down_left}, as depicted in Fig. 1. For simplicity, we assume a constant velocity and altitude for the UoI. Furthermore, we consider that UoI operates autonomously without either any remote ground control or predefined way-points plan. On another side, we consider that the 2-D flying area can include either static (e.g., buildings) and mobile (e.g., birds and other UAVs) obstacles. Therefore, the UoI has to be equipped with an accurate sensor (e.g., Lidar) to precisely detect the surrounding objects' positions. Other ultrasonic-based, video-based, and radio-based techniques for detecting obstacles and mobile objects have been investigated in [30] and [31]. Hereafter, we refer to the static and mobile objects as static_intruders and mobile_intruders, respectively. We define $\mathcal{I}$ as the set of intruders, where $\mathcal{I} = \{\{$ static_intruder $\} \bigcup \{$mobile_intruder$\}\}$.

在本文中，我们的目标是控制一架无人机的运动，以下简称 UoI(感兴趣无人机)，同时避免与静态和移动物体发生碰撞。表 I 概括了本文中使用的不同符号。UoI 需要在尺寸为 $X \times Y$ 的限定区域内移动，同时避免碰撞。UoI 的运动从预定义的初始位置 $P_S = (x_S, y_S)$ 开始，一旦达到预定义的目标位置 $P_T = (x_T, y_T)$ 就停止。令 $\mathcal{A}$ 表示 UoI 可能的动作方向。如 [28] 中所述，无人机的可能运动是有限的，并与它工作的环境相关。为了简单起见，并且在不失一般性的情况下，我们假设 $\mathcal{A}$ 有八个可能的方向，$\mathcal{A} = \{Up$ ，下、右、左、右上、左上、右下、左下，如图 1 所示。为了简化问题，我们假设 UoI 具有恒定的速度和高度。此外，我们认为 UoI 能够自主运行，无需远程地面控制或预定的航点计划。另一方面，我们认为 2-D 飞行区域可能包含静态 (例如，建筑物) 和移动 (例如，鸟类和其他无人机) 障碍物。因此，UoI 必须配备精确的传感器 (例如，激光雷达)，以准确检测周围物体的位置。其他基于超声波、基于视频和基于无线电的技术用于检测障碍物和移动物体已在 [30] 和 [31] 中进行了研究。此后，我们分别将静态和移动物体称为 static_intruders 和 mobile_intruders。我们定义 $\mathcal{I}$ 为入侵者集合，其中 $\mathcal{I} = \{\{$ static_intruder $\} \bigcup \{$mobile_intruder$\}\}$。

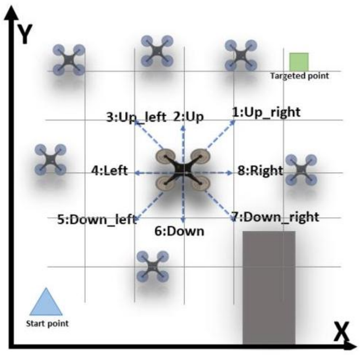


Fig. 1. System model of collision avoidance in the UAV environment.
图 1. 无人机环境中碰撞避障系统模型。
TABLE I
SUMMARY NOTATIONS
汇总符号

Symbol	Description
UoI	The UAV of Interest.
$X \times Y$	The 2-dimensional geographical area.
P_S	The UoI’ s started position.
$P_{\mathcal{T}}$	The UoI’ s targeted position.
A	The set of directions controlling UoI motion.
$\mathcal{I}$	The set of mobile and static intruders.
$U(t)$	The position of UoI at time step(t).
$J(t)$	The position of intruder j at time step(t).
γ	The euclidean distance between the positions.
A	The cardinal of the set A .
Z	The set of zones.
z_i	The zone surrounding the position i .
ρ	The radius of every z_i .
$\eta(i)$	The set of intruders neighboring the position i .
$\mathcal{P}(z_i)$	The probability of collision at the zone (z_i) .
p_i^j	The probability of collision between the point i and its neighbor j .
	The priority factor rate.
$\triangle$	The squared shaped area surrounds the UoI.
L_{Δ}	The size of Δ side.
α	The learning rate of DQN.
$\mathcal{L}$	The loss function.
ϵ	A threshold distance before two UAVs collide.
batch size	The size of each batch.
Q^{π}	The policy network.
Q^{T}	The target network.
$\mathcal{M}$	A number of episodes to update Q^{T} with Q^{π} .
ω	The wight and bias of neural network.
P_{rel}	The related address of P_T according to $U(t)$.
ξ	The decay parameter.

符号	描述
UoI	关注的无人机。
$X \times Y$	二维地理区域。
P_S	UoI 的起始位置。
$P_{\mathcal{T}}$	UoI 的目标位置。
A	控制 UoI 运动的方向集合。
$\mathcal{I}$	移动和静态入侵者的集合。
$U(t)$	UoI 在时间步 (t) 的位置。
$J(t)$	侵入者 j 在时间步 (t) 的位置。
γ	位置之间的欧几里得距离。
A	集合 A 的基数。
Z	区域集合。
z_i	围绕位置 i 的区域。
ρ	每个 z_i 的半径。
$\eta(i)$	靠近位置 i 的入侵者集合。
$\mathcal{P}(z_i)$	在区域 (z_i) 发生碰撞的概率。
p_i^j	点 i 与其邻居 j 发生碰撞的概率。
	优先级因子率。
$\triangle$	正方形区域围绕感兴趣区域 (UoI)。
L_{Δ}	Δ 边的长度。
α	DQN 的学习率。
$\mathcal{L}$	损失函数。
ϵ	两个无人机相撞前的阈值距离。
批次大小。	每个批次的大小。
Q^{π}	策略网络。
Q^{T}	目标网络。
$\mathcal{M}$	更新 Q^{T} 与 Q^{π} 的剧集数。
ω	神经网络的权重和偏置。
P_{rel}	根据 $U(t)$ 的相关地址 P_T 。
ξ	衰减参数。

The UoI might collide with one of the mobile and static obstacles. We assume an arbitrary trajectory for the mobile intruders, which is unknown by the UoI. A collision occurs whenever the Euclidean distance between the UoI and any intruders is lower than a predefined threshold distance ϵ . The safety distance ϵ varies from few centimeters to few meters according to different parameters related to the environment and used sensors. The distance ϵ can vary according to the sensor technology used to measure the distances, such as Lidar, depth camera, video, or ultrasonic. Also, the accuracy of the same type of sensors can vary from a manufacturer to another. The UoI can detect this collision by harnessing its onboard sensors at any time t . Let $P_u(t) = (x_u(t), y_u(t))$ and $P_j(t) = (x_j(t), y_j(t))$ denote the position of the UoI and the position of the intruder j at a given instant t , respectively. Then, a collision instance is formally defined as follows:

无人机 (UoI) 可能会与移动障碍物或静止障碍物发生碰撞。我们假设移动入侵者具有任意轨迹，该轨迹对无人机 (UoI) 来说是未知的。当无人机 (UoI) 与任何入侵者之间的欧几里得距离小于预定义的阈值距离 ϵ 时，就会发生碰撞。安全距离 ϵ 根据与环境和使用的传感器相关的不同参数，从几厘米到几米不等。距离 ϵ 可以根据用于测量距离的传感器技术 (如激光雷达、深度相机、视频或超声波) 而变化。同样，同类型传感器的精度也可能因制造商而异。无人机 (UoI) 可以通过利用其机载传感器在任何时间检测到这种碰撞 t 。令 $P_u(t) = (x_u(t), y_u(t))$ 和 $P_j(t) = (x_j(t), y_j(t))$ 分别表示无人机 (UoI) 和入侵者 j 在某一时刻 t 的位置。那么，碰撞实例正式定义如下:

$$\text{Collision } = \begin{cases} 1, & \text{if } \exists j \in \mathcal{I} \text{ where } \delta_u^j \leq \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where

其中

$$\delta_u^j = \sqrt{(x_u(t) - x_j(t))^2 + (y_u(t) - y_j(t))^2}. \tag{2}$$

In this article, we also take into consideration the limitation on the battery capacity of the UoI. To that end, the ultimate goal of the algorithm in charge of controlling the UoI movement is to achieve the target P_T following the shortest path while avoiding collisions with mobile and static objects. The shortest path minimizes the distance traveled by the UoI, thus contributing to energy saving.

在本文中，我们还考虑了无人机 (UoI) 电池容量的限制。为此，负责控制无人机 (UoI) 移动的算法的最终目标是实现目标 P_T 的跟踪，同时沿着最短路径避开移动和静止物体的碰撞。最短路径可以最小化无人机 (UoI) 的行驶距离，从而有助于节能。

IV. PROBABILITY-DISTRIBUTION-BASED COLLISION-AVOIDANCE FRAMEWORK

IV. 基于概率分布的碰撞避障框架

In this section, we propose a heuristic, dubbed PICA, to control the movement of the UoI to reach a target position while avoiding collisions. It considers the mission time, i.e., the time from the UoI starts its mission until it reaches its target P_T , is slotted. At each time step t , the UoI collects data from different sensors to sense the objects' presence in its vicinity. As depicted in Fig. 2, UoI is aware of the surrounding objects in a circular area z_i , whose extension is limited by the sensors' ranges. Specifically, the circular area z_i has a radius ρ . The state of the circular area z_i , i.e., the spatial distribution of the intruders within it, is updated at every time step after the UoI changes its position according to an action $a \in \mathcal{A}$ taken by PICA. Let $\mathcal{Z}$ denote the set of circular shaped areas z_i , where $\mathcal{Z} = \{z_i : \forall i \in \mathcal{A}\}$.

在本节中，我们提出了一个启发式方法，名为 PICA，用于控制单位兴趣区域 (UoI) 移动至目标位置的同时避免碰撞。该方法考虑任务时间，即从 UoI 开始任务到达到目标位置 P_T 的时段。在每一个时间步 t ，UoI 从不同的传感器收集数据，以感知其周围物体的存在。如图 2 所示，UoI 知道在其周围的圆形区域 z_i 内的物体，该区域的扩展受限于传感器的范围。具体来说，这个圆形区域 z_i 有一个半径 ρ 。圆形区域 z_i 的状态，即其中的入侵者的空间分布，在 UoI 根据 PICA 采取的动作 $a \in \mathcal{A}$ 改变位置后，在每个时间步更新。令 $\mathcal{Z}$ 表示圆形区域 z_i 的集合，其中 $\mathcal{Z} = \{z_i : \forall i \in \mathcal{A}\}$ 。

Inside every z_i it might exist intruders neighboring every z_i ' s center i . Let $\eta(i)$ denote the set of static_intruders and mobile_intruders inside the area z_i . We denote by δ_i^j the Euclidean distance between j th intruder belonging $\eta(i)$ and the position of i . The distance between each intruder and the center of z_i can be computed by PICA using the triangulation method.

在每个 z_i 内部可能存在邻近每个 z_i 中心的入侵者 i 。令 $\eta(i)$ 表示区域 z_i 内的静态入侵者和移动入侵者的集合。我们用 δ_i^j 表示 j th 个属于 $\eta(i)$ 的入侵者与 i 位置之间的欧几里得距离。每个入侵者与 z_i 中心的距离可以通过 PICA 使用三角测量法计算得出。

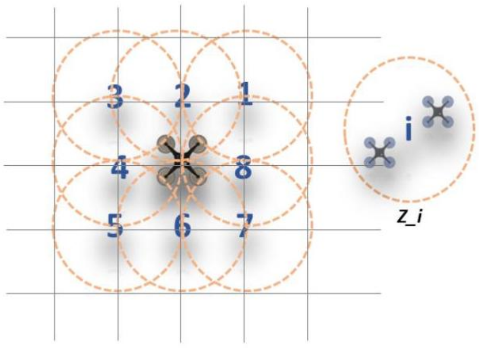


Fig. 2. PICA: zone concept for selecting directions.
图 2. PICA: 选择方向的区域概念。

The density distribution of the intruders inside z_i could refer to the likelihood of a collision if the UoI moves to position i . In other words, the denser z_i , the higher the probability that the UoI experiences a collision. Let us define $\mathcal{P}(z_i)$ as the probability of collision inside z_i , formally defined as follows.

侵入者内部 z_i 的密度分布可以指代如果感兴趣区域 (UoI) 移动到位置 i 时发生碰撞的可能性。换句话说，z_i 越密集，感兴趣区域 (UoI) 遭遇碰撞的概率就越高。让我们将 $\mathcal{P}(z_i)$ 定义为 z_i 内部发生碰撞的概率，正式定义如下。

$$\mathcal{P}\left(z_i\right)=\frac{\sum\limits_{\forall j \in \eta(i)} p_i^j}{\sum\limits_i \sum\limits_{\forall j \in \eta(i)} p_i^j} \; \forall i \in\{1,2, \ldots,|\mathcal{A}|\} \tag{3}$$

where
其中

$$p_i^j=1-\frac{\delta_i^j}{\rho} \tag{4}$$

where p_i^j represents the likelihood that UoI collides with j if action $i \in \mathcal{A}$ is chosen. Formally, the closer j is to UoI, the higher the probability of collision. Note that the value of δ_i^j/ρ is within the interval $[0,1]$. In order to avoid the collision, the position with the lower $\mathcal{P}\left(z_i\right)$ should be chosen.

其中 p_i^j 表示如果选择行动 $i \in \mathcal{A}$ ，感兴趣区域 (UoI) 与 j 发生碰撞的可能性。正式地说，j 越接近感兴趣区域 (UoI)，碰撞的概率就越高。注意，δ_i^j/ρ 的值在区间 $[0,1]$ 内。为了避免碰撞，应该选择 $\mathcal{P}\left(z_i\right)$ 较低的位置。

In addition to the safety factor, PICA aims to go through the shortest path by seeking at each time step t the direction that brings the UoI closest to the target. To achieve this goal, PICA measures the remaining distance to the target from every i . Indeed, to decide the UoI’ s next direction, PICA ranks every Z_i ’ s center, i , using the following equation:

除了安全因素之外，PICA 还旨在通过在每个时间步寻求 t 将感兴趣区域 (UoI) 最接近目标的方向来穿越最短路径。为了达到这个目标，PICA 从每个 i 测量到目标的剩余距离。实际上，为了决定感兴趣区域 (UoI) 的下一个方向，PICA 使用以下方程式对每个 Z_i 的中心，i ，进行排序:

$$R_i=\theta P\left(z_i\right)+(1-\theta) \frac{\delta_i^{P_T}}{\sqrt{X^2+Y^2}} \tag{5}$$

where $\theta \in[0,1]$ is a parameter used to favor either safety or energy. $\delta_i^{P_T}$ denotes the distance between the current position and the target. The concept of application integrating the UoI has an immediate impact on selecting either θ or its complement $(1-\theta)$. In our case, we give more priority to the safety of the UoI agent. For this reason, we have selected higher values of θ . Furthermore, $\delta_i^{P_T}$ refers to the Euclidean distance between the Z_i ’ s center and the targeted point P_T . Since the unity of both the probabilities and the distances values are in different scales. To prevent distance domination, we have normalized the value of $\delta_i^{P_T}$ to be between 0 and 1 by diving it by the maximum possible distance $\sqrt{X^2+Y^2}$. Indeed, the UoI will choose to move in the direction a that has the lowest rank using the following formula:

其中 $\theta \in[0,1]$ 是一个参数，用于偏好安全性或能源效率。$\delta_i^{P_T}$ 表示当前位置与目标之间的距离。将 UoI 集成的应用概念对选择 θ 或其补数 $(1-\theta)$ 产生直接影响。在我们的案例中，我们更重视 UoI 代理的安全性。因此，我们选择了较高的 θ 值。此外，$\delta_i^{P_T}$ 指的是 Z_i 中心与目标点 P_T 之间的欧几里得距离。由于概率和距离值的单位处于不同的量级。为了防止距离的支配作用，我们将 $\delta_i^{P_T}$ 的值通过除以最大可能的距离 $\sqrt{X^2+Y^2}$ 归一化到 0 和 1 之间。实际上，UoI 将选择使用以下公式沿 a 方向移动，该方向具有最低的排名:

$$a=\underset{i \in \mathcal{A}}{\arg \min }\left\{R_i\right\} \tag{6}$$

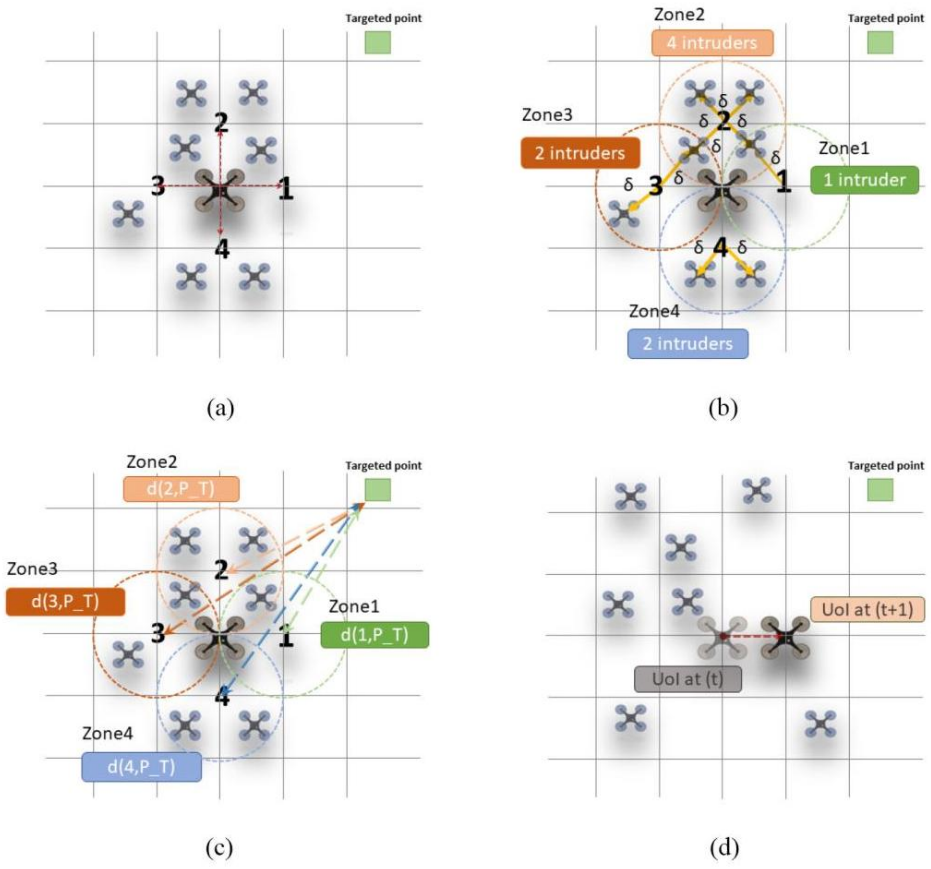


Fig. 3. PICA descriptive example. (a) UAV UoI at instant (t). (b) Density distribution of the intruders inside each zone. (c) Remaining distance to the target from the center of each zone. (d) UAV UoI at instant $(t+1)$.

图 3. PICA 描述性示例。(a) 在时刻 (t) 的 UAV UoI。(b) 每个区域内部人侵者的密度分布。(c) 从每个区域的中心到目标的剩余距离。(d) 在时刻 $(t+1)$ 的 UAV UoI。

Fig. 3 illustrates the PICA functionality, which is detailed in Algorithm 1. For simplicity, in this example, we consider that UoI can move only on four directions $\{Up$, Down, Left, and Right $\}$ corresponding to the positions $\{1,2,3,4\}$, respectively. We also consider that all the distances δ_i^J are the same and equal to δ . Hereafter, based on the density of intruders in each zone, PICA computes the probability of collisions $\mathcal{P}(z_i)$ for every z_i (Algorithm 1: line 10) using (3) and (4). For example, the probability of collision $\mathcal{P}(z_2)$ at zone 2 is the summation of the probabilities that UoI collides at Z_i ' s center 2 with every $j \in \eta(2)$. Indeed, as shown in Fig. 3(b), $\mathcal{P}(z_2)=\left[\left(\sum\limits_{\forall j \in \eta(2)} p_2^j\right) / \left(\sum\limits_{\forall i \in \{1,\ldots,4\}} \sum\limits_{\forall j \in \eta(i)} p_i^j\right)\right]$, where $p_2^1=p_2^2=p_2^3=p_2^4=(1-[\delta/\rho])$. In this case, the probability collision distribution of the positions1,2,3, and 4 is $1/9, 4/9, 2/9$, and $2/9$, respectively. As depicted in Fig. 3(b), the zone 2 is the most dense in term of intruders compared to the other ones. In contrast, zone 1 is the less dense since it contains only one intruder with probability collision $1/9$. To rank the candidates' directions $i \in \{1,\ldots,4\}$ of PICA, based on (5) and (6), it chooses the best action that has the lowest probability collision and the lowest remaining distance to the target (Algorithm 1: lines 12 and 15). Following the same example and as shown in Fig. 3(b) and (c), the candidate direction (1) has the smallest rank since it has the lowest probability and the lowest missing distance to the target (Algorithm 1: line 15). Finally, at the time step $(t+1)$, the UoI moves into the position (1) as depicted in Fig. 3(d).

图 3 展示了 PICA 功能性，这在算法 1 中有详细说明。为了简化，在这个示例中，我们认为 UoI 只能在四个方向上移动 $\{Up$ ，下、左和右 $\}$ 分别对应于位置 $\{1,2,3,4\}$ 。我们还认为所有距离 δ_i^J 都是相同的，等于 δ 。此后，基于每个区域人侵者的密度，PICA 计算每个 z_i 的碰撞概率 $\mathcal{P}(z_i)$ (算法 1: 第 10 行) 使用 (3) 和 (4)。例如，区域 2 的碰撞概率 $\mathcal{P}(z_2)$ 是 UoI 在 Z_i 中心 2 与每个 $j \in \eta(2)$ 碰撞的概率之和。实际上，如图 3(b) 所示，$\mathcal{P}(z_2)=\left[\left(\sum\limits_{\forall j \in \eta(2)} p_2^j\right) / \left(\sum\limits_{\forall i \in \{1,\ldots,4\}} \sum\limits_{\forall j \in \eta(i)} p_i^j\right)\right]$ ，其中 $p_2^1=p_2^2=p_2^3=p_2^4=(1-[\delta/\rho])$

。在这种情况下，位置 1、2、3 和 4 的碰撞概率分布分别为 $1/9, 4/9, 2/9$ 和 $2/9$ 。如图 3(b) 所示，区域 2 在入侵者数量方面比其他区域更为密集。相反，区域 1 是最不密集的，因为它只包含一个碰撞概率为 1/9 的入侵者。为了对 PICA 候选方向 $i \in \{1, \ldots, 4\}$ 进行排名，基于 (5) 和 (6)，它选择具有最低碰撞概率和最低剩余距离至目标的最佳动作 (算法 1: 第 12 和 15 行)。遵循相同的示例，如图 3(b) 和 (c) 所示，候选方向 (1) 具有最小的排名，因为它具有最低的概率和最低的缺失距离至目标 (算法 1: 第 15 行)。最后，在时间步 $(t+1)$ ，UoI 移动到位置 (1)，如图 3(d) 所示。

V. RELIANCE: REINFORCEMENT-LEARNING-BASED COLLISION-AVOIDANCE SOLUTION

V. 依赖性: 基于强化学习的避障解决方案

In this section, we provide an RL-based solution for avoiding the collision. In contrast to the model-based approach, MDP, which requires full knowledge about the environment (i.e., transition probabilities), RL does not require any prior knowledge. This makes RL a more suitable framework for dealing with unsuspected and uncorrelated mobility of objects around UoI. Thanks to sampling and bootstrapping in RL, RELIANCE can forecast the next movement of each mobile object and then avoid the collision. Fig. 4 depicts the main overview of the RELIANCE solution. In this section's balance, we will give first some background on RL, and, more precisely, DQN employed in this article. Then, we will give a detailed description of RELIANCE.

在本节中，我们提供了一个基于强化学习 (RL) 的避障解决方案。与需要了解环境全貌 (即转移概率) 的模型基础方法 MDP 相比，强化学习不需要任何先验知识。这使得强化学习成为一个更适合处理兴趣区域 (UoI) 周围物体不可预知和无关联移动的框架。得益于强化学习中的采样和引导，RELIANCE 能够预测每个移动对象的下一步动作，从而避免碰撞。图 4 展示了 RELIANCE 解决方案的主要概览。在本节的其余部分，我们将首先介绍强化学习以及本文中使用的深度 Q 网络 (DQN) 的背景知识，然后我们将详细描述 RELIANCE。

A. Background on RL and DQN

A. 强化学习和 DQN 的背景

The RL technique has been widely used in the literature in various applications and services, such as robotics and industry 5.0. RL's ultimate goal is to endow vertical industry with the ability to learn, improve, and adapt according to the environment's changes. With the new trend toward the self-optimized and the cognitive network, industry, and academia shifted their attention to employing RL. An RL system mainly consists of five elements, as depicted in Fig. 4, which are: 1) environment $\mathcal{E}$; 2) states $\mathcal{S}$; 3) agent, in our case, is the motion controller

强化学习技术已在文献中被广泛应用于各种应用和服务中，如机器人技术和工业 5.0。强化学习的最终目标是赋予垂直行业学习、改进和根据环境变化进行适应的能力。随着向自优化和认知网络的新趋势发展，工业界和学术界将注意力转向了应用强化学习。一个强化学习系统主要由五个元素组成，如图 4 所示，分别是:1) 环境 $\mathcal{E}$ ； 2) 状态 $\mathcal{S}$ ； 3) 在我们的案例中，代理是运动控制器

Algorithm 1: PICA

算法 1:PICA

```
Input
        $X$ :The x _axis limit of the geographical area $X \times Y$ .
        $Y$ :The y_axis limit of the geographical area $X \times Y$ .
        $\rho$ : The radius.
        $\theta$ : The priority rate.
        $\mathcal{A}$ :The set of actions.
        $P_T$ :The started point of UoI in the environment.
        $P_S$ :The targeted point of UoI in the environment.
Output:
        done: The UoI reaches $P_T$ or collides with one of the intruders.
```

```
done ← False;
while (done = False) do
    $Z \leftarrow \varnothing$ ;
    $R \leftarrow \varnothing$ ;
    foreach ($a \in A$) do
        $z \leftarrow \text{Circle}\,(a, \rho)$ ;
        $\text{Z} \leftarrow \text{Z} \cup z$
    end
    foreach ($z \in Z$) do
        Compute $P\,(z)$ ;
          Compute $\delta_i^{P_T}$ ;
        $r \leftarrow \theta P\,(z) + \left(1 - \dot{\theta}\right) \frac{\delta_i^{P_T}}{\sqrt{X^2+Y^2}}$;
        $R \leftarrow R \cup r$
    end
    $a \leftarrow \underset{i \in A}{\arg\min}\,\{R_i\}$
    UoI applies action a;
    if ($U\,(t) = P_T$ or Collision) then
        done ← True;
    end
end
```

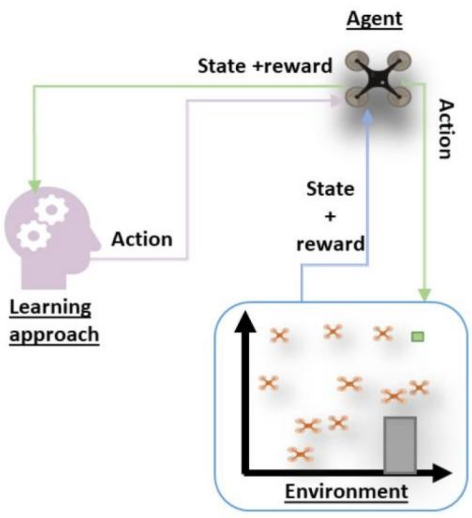


Fig. 4. RL-based collision-avoidance system overview.
图 4. 基于强化学习的避障系统概览。
of the UoI; 4) Actions $\mathcal{A}$; and 5) rewards r received after the execution of each action $a \in \mathcal{A}$.
的兴趣区域 (UoI)；4) 动作 $\mathcal{A}$ ； 以及 5) 在执行每个动作 $a \in \mathcal{A}$ 后接收到的奖励 r 。

While RL works either for episodic or continuous tasks, in this work, our environment is episodic. Each episode presents the UoI mission that starts at $P_S = (x_S, y_S)$ and ends either when UoI attends its destination $P_T = (x_T, y_T)$ or collides with a mobile or static obstacle. As depicted in Fig. 4, UoI discretely interacts with the environment by taking different actions, and then accordingly receiving an observation and rewards that reflect the action taken. The agent UoI keeps interacting with the environment $\mathcal{E}$ and receiving reward r_t on steps $t \in \{1, 2, \ldots, T\}$. While $T = \infty$ for continuous tasks, it

虽然强化学习 (RL) 适用于回合任务或连续任务，但在本研究中，我们的环境是回合制的。每个回合呈现一个 UoI 任务，该任务从 $P_S = (x_S, y_S)$ 开始，当 UoI 到达目的地 $P_T = (x_T, y_T)$ 或者与移动或静止障碍物发生碰撞时结束。如图 4 所示，UoI 通过采取不同的动作以离散的方式与环境交互，并相应地接收到反映所采取动作的观察值和奖励。UoI 代理持续与环境 $\mathcal{E}$ 交互，并在步骤 $t \in \{1, 2, \ldots, T\}$ 上接收奖励 r_t 。虽然在连续任务中 $T = \infty$ ，但是

is finite in the case of an episodic task. The objective of the UoI agent is to increase cumulative reward G_t received after time step t until the end of the episode

在回合制任务中是有限的。UoI 代理的目标是在时间步 t 之后增加直到回合结束所获得的累积奖励 G_t 。

$$G_t \doteq \sum_{k=0}^{T} \gamma^k r_{t+k+1} = r_{t+1} + \gamma G_{t+1} \tag{7}$$

such that $\gamma \in [0,1]$ is the discount factor and r_t denotes the immediate reward received at the instant t .

其中 $\gamma \in [0,1]$ 是折扣因子，r_t 表示在瞬间 t 收到的即时奖励。

Many RL techniques have been proposed in the literature, including policy-based (e.g., REINFORCE), actor-critic (e.g., A3C and DDPG), and value-based approaches (e.g., QN, DQN, and DDQN). While the two formal methods aim to provide the policy that estimates the state's action probabilities, the latter approach estimates the state-action value. Then, this value is used to deliver the optimal policy. Considering that the space of actions is discrete and limited, in this work, we opt for a value-based approach and, more precisely, DQN. Particularly, we have chosen DQN due to the size of the action-state space, as explained later.

文献中已经提出了许多强化学习技术，包括基于策略的方法 (例如，REINFORCE)、演员-评论家方法 (例如，A3C 和 DDPG) 以及基于价值的方法 (例如，QN、DQN 和 DDQN)。虽然前两种正式方法旨在提供估计状态动作概率的策略，但后者估计的是状态-动作价值。然后，这个价值被用来导出最优策略。考虑到动作空间是离散和有限的，在本文中，我们选择基于价值的方法，更具体地说，是 DQN。特别地，我们选择 DQN 是因为动作-状态空间的大小，如下文所述。

The state-action value of state $s \in \mathcal{S}$ using action $a \in \mathcal{A}$ under the policy π is defined as follows [32]:

在策略 π 下，状态 $s \in \mathcal{S}$ 使用动作 $a \in \mathcal{A}$ 的状态-动作价值定义如下 [32]:

$$Q_\pi(s,a) \doteq \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a\right]$$

$$\leftarrow \mathbb{E}_\pi \left[\sum_{k=0}^{T} \gamma^k r_{t+k+1} \mid S_t = s, A_t = a\right]. \tag{8}$$

The optimal action-state value $Q^*(s,a)$ can be delivered from $Q_\pi(s,a)$ by choosing the optimal policy. Formally, $Q*(s,a) = \max_\pi Q_\pi(s,a)$ for $a \in \mathcal{A}$ and $s \in \mathcal{S}.Q^*(s,a)$ can be also delivered using the Bellman optimality equation using the following formula [32]:

最优动作状态值 $Q^*(s,a)$ 可以通过选择最优策略从 $Q_\pi(s,a)$ 中得到。正式地，$Q*(s,a) = \max_\pi Q_\pi(s,a)$ 对于 $a \in \mathcal{A}$ 和 $s \in \mathcal{S}.Q^*(s,a)$ 也可以使用贝尔曼最优性方程通过以下公式得到 [32]:

$$Q_*(s,a) \leftarrow \mathbb{E}\left[r_{t+1} + \gamma \max_{a \in \mathcal{A}} Q_*(S_{t+1}, a') \mid S_t = s, A_t = a\right].$$

(9)

The basic idea behind many value-based algorithms is sampling and bootstrapping. The sampling is leveraged for enabling the algorithm to learn by exploring the environment, thanks to the trial and error approach. Meanwhile, bootstrapping is a technique used to estimate the state-action value in order to speed up the algorithm convergence [32]. Q-Learning Algorithm is one of the widely used algorithm in the literature. The Q -learning algorithm leverages sampling and bootstrapping methods to converge to the optimal policy. During the learning step, the Q -learning algorithm updates the state value action using the following formula:

许多基于价值的算法背后的基本思想是采样和引导。采样利用算法通过探索环境进行学习，得益于试错法。同时，引导是一种用来估计状态动作值的技术，以加快算法的收敛 [32]。Q-学习算法是文献中广泛使用的算法之一。Q-学习算法利用采样和引导方法收敛到最优策略。在学习步骤中，Q-学习算法使用以下公式更新状态动作值:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha$$

$$\times \left[r_{t+1} + \gamma \times \max_{a \in \mathcal{A}} Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{10}$$

such that α is the learning rate.

其中 α 是学习率。

Thanks to bootstrapping, Q -learning repeatedly updates $Q(s_t, a_t)$ by shifting it toward the optimal value using TD error $\left(r_{t+1} + \gamma \times \max_{a\in\mathcal{A}} Q(s_{t+1}, a) - Q(s_t, a_t)\right)$ and learning rate α . This approach enables to gradually increasing $Q(s_t, a_t)$ toward the optimal value. The optimal policy can be delivered from the optimal state action using the following formula:

得益于引导，Q -学习通过使用 TD 误差 $\left(r_{t+1} + \gamma \times \max_{a\in\mathcal{A}} Q(s_{t+1}, a) - Q(s_t, a_t)\right)$ 和学习率 α 不断更新 $Q(s_t, a_t)$ ，使其向最优值移动。这种方法使得 $Q(s_t, a_t)$ 逐渐增加至最优值。最优策略可以通过以下公式从最优状态动作中得到:

$$\forall s \in S : \pi^*(s) \leftarrow \arg\max_a Q(s, a) . \tag{11}$$

In the Q -learning algorithm, the state-action value Q is presented as a table, where the states are the lines and actions are the columns. Unfortunately, Q learning is unfeasible for large action-state spaces as ours. Fortunately, DQN has been suggested to overcome that limitation [33] by creating an estimator of the Q table by leveraging the NN. In fact, the Q table is approximated with an NN with parameter ω .

在 Q -学习算法中，状态动作值 Q 以表格的形式呈现，其中状态是行，动作是列。不幸的是，对于像我们这样的大型动作状态空间，Q 学习是不可行的。幸运的是，DQN 被提出以克服这一限制 [33]，通过利用神经网络 (NN) 创建 Q 表的估计器。实际上，Q 表通过具有参数 ω 的神经网络进行近似。

Unfortunately, the basic DQN algorithm suffers from overestimations of action value due to using the same NN in the update and estimation of the next Q value used to compute the TD error. This approach creates lots of noise and makes it hard to find the action with the maximum expected/estimated Q -value. To prevent this issue, Mnih et al. [34] have suggested DQN that uses two different Q NNs. The first one, called policy $Q\text{NN}Q^\pi$, is used for estimating the action. Meanwhile, the second one, called target Q network Q^T , is used to generate the target action values. To mitigate the noises in the update, while Q^π is updated at each iteration, Q^T is updated from Q^π only after a specific number of episodes.

不幸的是，基本的 DQN 算法由于在更新和估计下一个 Q 值时使用相同的神经网络，导致了动作价值的过度估计。这种方法产生了大量噪声，使得很难找到具有最大预期/估计 Q -价值的动作。为了防止这个问题，Mnih 等人 [34] 建议使用两个不同 Q 神经网络的 DQN。第一个，称为策略 $Q\text{NN}Q^\pi$ ，用于估计动作。同时，第二个，称为目标 Q 网络 Q^T ，用于生成目标动作价值。为了减轻更新中的噪声，虽然 Q^π 在每次迭代中都会更新，但 Q^T 只在特定数量的剧集之后从 Q^π 中更新。

In this case, the policy π during the exploitation, either during the training or inference modes, is generated from the approximation $Q^\pi(s_t, a_t, \omega)$ NN using the following equation:

在这种情况下，无论是在训练还是推理模式下进行利用时，策略 π 都是通过以下方程使用近似 $Q^\pi(s_t, a_t, \omega)$ NN 生成的:

$$\pi_{s_t} \leftarrow \arg\max_{a\in\mathcal{A}} Q^\pi(s_t, a, \omega) \tag{12}$$

Meanwhile, the parameter ω (bias and weights) of the estimator Q^π is updated periodically during the training step using the following formula:

同时，评估器 Q^π 的参数 ω (偏差和权重) 在训练步骤期间定期更新，使用以下公式:

$$\begin{aligned} \omega_{t+1} = \omega_t + \alpha \times \Big[& r_{t+1} + \gamma \times \max_a Q^T(s_{t+1}, a; w') \\ & - Q^\pi(s_t, a_t; w) \Big] \times \nabla_w Q^\pi(s_t, a_t; w) . \end{aligned} \tag{13}$$

By leveraging different gradient descent methods (e.g., stochastic gradient descent, RMSprop, or ADAM), the DQN Algorithm keeps updating ω during the training step. ω is updated from replay memory(B)that consists of transitions observed during the exploration or exploitation. Each transition $< s_t, a_t, r, s_{t+1} >$ consists of the current state s_t , the taken action a_t , the immediate received reward r and the next state s_{t+1} . To break the correlation between transitions and to allow a stable learning curve, a batch of transitions (i.e., batch_size) is randomly selected from the reply memory $\mathcal{B}$ [32].

通过利用不同的梯度下降方法 (例如，随机梯度下降，RMSprop 或 ADAM)，DQN 算法在训练步骤中不断更新 ω 。ω 从重放记忆 (B) 中更新，该记忆由探索或利用期间观察到的转换组成。每个转换 $< s_t, a_t, r, s_{t+1} >$ 包括当前状态 s_t 、采取的动作 a_t 、立即收到的奖励 r 以及下一个状态 s_{t+1} 。为了打破转换之间的相关性并允许稳定的学习曲线，从回复记忆 $\mathcal{B}$ 中随机选择一批转换 (即 batch_size)[32]。

To ensure a balance between the exploration and exploitation during the training to update ω , an epsilon greedy method is used. The algorithm keeps randomly switching between the exploration and exploitation modes. At the end of the training, the DQN algorithm should favor exploitation than exploration to assist its convergence. For this purpose, an epsilon decay strategy has been adopted by decreasing the epsilon decay ξ parameter during the training. ξ initially starts by 1, and it should converge to 0 at the end of the training. To switch between the exploration and exploitation, a random number (i.e., $[0,1]$) is generated and compared to ξ . If the generated number is lower than ξ , the exploration procedure is executed. Otherwise, the exploitation procedure is considered.

为了在训练过程中更新 ω 时确保探索和利用之间的平衡，使用了一种　贪婪方法。算法在探索和利用模式之间随机切换。在训练结束时，DQN 算法应更倾向于利用而不是探索以帮助其收敛。为此，在训练过程中采用了　衰减策略，通过减少　衰减 ξ 参数。ξ 初始值为 1，在训练结束时应该收敛到 0。为了在探索和利用之间切换，生成一个随机数 (即 $[0,1]$) 并将其与 ξ 比较。如果生成的数字小于 ξ ，则执行探索过程。否则，考虑利用过程。

B. RELIANCE Model Overview

B. RELIANCE 模型概述

The autonomous flight control system of the UoI is realized as an RL agent. To model the energy-aware collision-avoidance problem using the RL framework, as mentioned in the previous section, the following elements need to be formally defined: 1) environment; 2) state; 3) agent; 4) actions; and 5) reward.

UoI 的自主飞行控制系统被实现为一个强化学习代理。如前所述，为了使用 RL 框架模拟能量感知避障问题，以下元素需要被正式定义:1) 环境；2) 状态；3) 代理；4) 动作；5) 奖励。

1) Agent: The RL agent is instantiated and run within the UoI to control its trajectory in order to avoid collisions while minimizing the energy consumption by taking the shortest path until its destination.

1) 代理: 在 UoI 内实例化和运行 RL 代理，以控制其轨迹以避免碰撞，同时通过走最短路径直到目的地来最小化能耗。

2) Environment: Geographical are of dimensions $X \times Y$ that include a set of mobile (e.g., other UAVs) and static objects (e.g., walls). The agent moves within this 2-D confined area.

2) 环境: 尺寸为 $X \times Y$ 的地理区域，包括一组移动对象 (例如，其他无人机) 和静态对象 (例如，墙壁)。代理在这个二维受限区域内移动。

3) Actions: The action space comprises a set of eight directions, i.e., $\mathcal{A} = \{Up, Down, Right, Left,$ Up_right, Up_left, Down_right, Down_left$\}$, previously mentioned.

3) 动作: 动作空间包括一组八个方向，即 $\mathcal{A} = \{Up, Down, Right, Left,$ 向上右，向上左，向下右，向下左$\}$，如前所述。

4) Reward: If the agent succeeds and reaches its targeted destination, it is positively rewarded with 100 . If the UoI experiences a collision during its trajectory to the destination, the agent is penalized with a negative reward of -100 . Finally, in order to encourage the agent to take the shortest path, there is a penalty of -0.1 for each step taken by the agent until reaching its destination.

4) 奖励: 如果代理成功并到达其目标目的地，它将获得正奖励 100。如果在 UoI 向目的地移动的轨迹中发生碰撞，代理将受到-100 的负奖励。最后，为了鼓励代理选择最短路径，代理在到达目的地之前的每一步都会受到-0.1 的惩罚。

5) State: The state (agent's observations) consists of two parts.

5) 状态: 状态 (代理的观察) 由两部分组成。

a) The distance vector $P_{\text{rel}} = (x_{\text{rel}}, y_{\text{rel}})$ is defined as the vector from the current UoI' s position (x_C, y_C) to the UoI' s destination (x_T, y_T) . That is

a) 距离向量 $P_{\text{rel}} = (x_{\text{rel}}, y_{\text{rel}})$ 定义为从当前 UoI 的位置 (x_C, y_C) 到 UoI 目的地的向量 (x_T, y_T) 。也就是说

$$x_{\text{rel}} = \frac{x_T - x_C}{X}$$

$$y_{\text{rel}} = \frac{y_T - y_C}{Y}$$

Observe that $(x_{\text{rel}}, y_{\text{rel}}) = (0,0)$ means the UoI is at the destination. Also, note that x_{rel} and y_{rel} have been normalized by X and Y (flight area dimensions), respectively. In this way, the agent is agnostic to the scenario scale, which makes the solution more general, i.e., the same trained model can be used in many environments.

注意 $(x_{\text{rel}}, y_{\text{rel}}) = (0, 0)$ 表示 UoI 位于目的地。同时，注意 x_{rel} 和 y_{rel} 已经分别通过 X 和 Y (飞行区域尺寸) 进行了标准化。这样，代理对场景规模是不可知的，这使得解决方案更具普遍性，即相同的训练模型可以在许多环境中使用。

b) The number of mobile and static objects distribution across a grid square centered around the UoI. Specifically, a $\Delta = \|L_\Delta \times L_\Delta\|$ grid square is considered. The UoI dimensions give the size of each tile of the grid. The grid is formally described as a binary matrix. Each element of this matrix indicates whether there is any static or mobile object (intruder) within the respective cell (tile) $(= 1)$ or not $(= 0)$ (see Fig. 5). This approach enables us to consider the UoI vicinity, which is the most relevant to avoid collisions and reduce the state space by aggregating many observations to a single state. Thus, faster learning and convergence will be perceived. As depicted in Fig. 5, thanks to the aggregation method adopted by RELIANCE, different observations depicted in Fig. 5(a)-(c) can be presented by the same state shown in Fig. 5(d).

b) 移动和静态对象在以用户感兴趣区域 (UoI) 为中心的网格单元中的分布数量。具体来说，考虑一个 $\Delta = \|L_\Delta \times L_\Delta\|$ 网格单元。UoI 的尺寸决定了网格中每个瓦片的尺寸。该网格被正式描述为一个二进制矩阵。该矩阵的每个元素指示在相应的单元格 (瓦片) $(= 1)$ 内是否存在任何静态或移动对象 (入侵者) $(= 0)$ (见图 5)。这种方法使我们能够考虑与避免碰撞和减少状态空间相关的 UoI 邻域，通过将许多观察结果聚合成单个状态。因此，将感知到更快的学习和收敛。如图 5 所示，得益于 RELIANCE 采取的聚合方法，图 5(a)-(c) 中所示的不同观察结果可以通过图 5(d) 中所示相同的状态来表示。

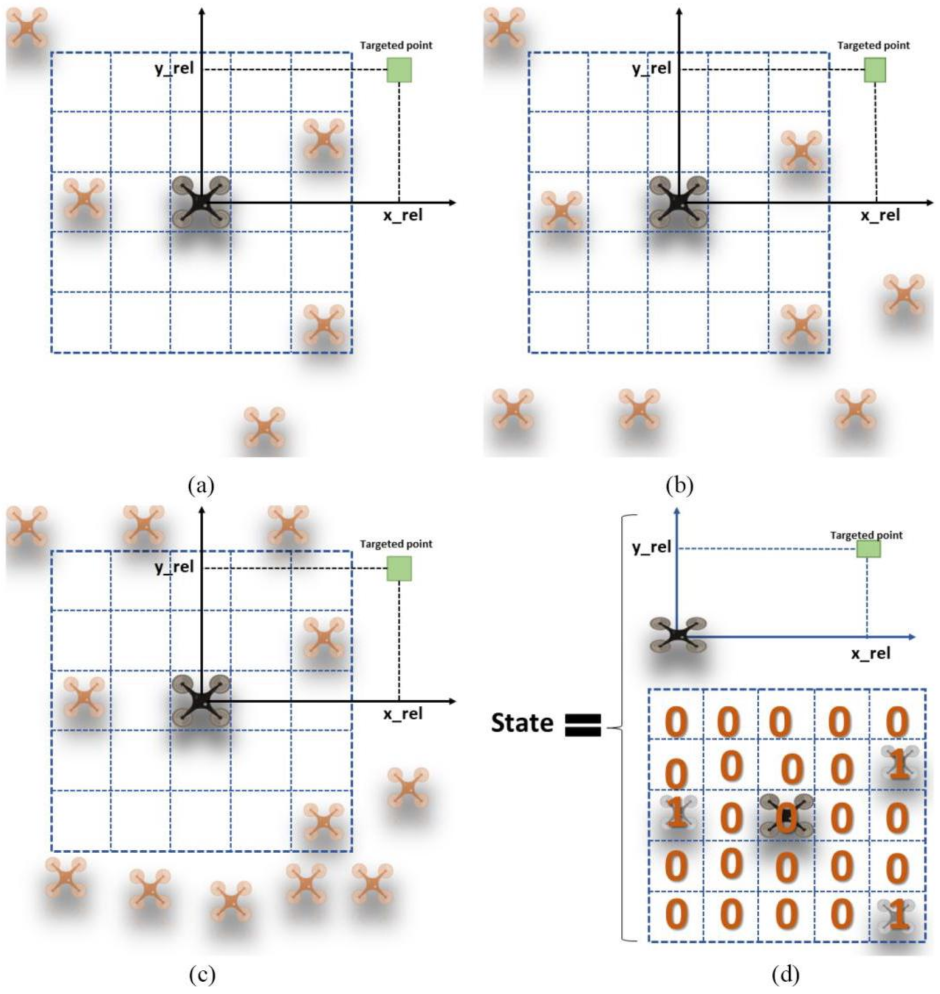


Fig. 5. State aggregation process adopted by RELIANCE. (a) Scenario1 with two intruders beyond the square. (b) Scenario2 with five intruders beyond the square. (c) Scenario3 with nine intruders beyond the square. (d) Same state refers to all the previous scenarios.

图 5。RELIANCE 采纳的状态聚合过程。(a) 场景 1，两个入侵者在正方形之外。(b) 场景 2，五个入侵者在正方形之外。(c) 场景 3，九个入侵者在正方形之外。(d) 指所有先前场景的相同状态。

It is remarkable that both components of the state considered are agnostic of the scenario, which makes the solution more general.

值得一提的是，考虑的状态的两个组成部分都与场景无关，这使得解决方案更具普遍性。

C. RELIANCE Example Description

C. RELIANCE 示例描述

As stated previously, we provide the agent with an RL-Algorithm that adopts one of the existing RL approaches. The principle role of this algorithm consists of giving the agent the ability to self-decide in which direction has to move following the defined goals. Before starting the flight, we provide the agent with the coordinates of its started and targeted points, P_S and P_T , respectively. Thus, at each step, the agent needs to do the following.

如前所述，我们为代理提供了一个采用现有 RL 方法之一的 RL 算法。该算法的主要作用是赋予代理根据定义的目标自我决定移动方向的能力。在开始飞行之前，我们向代理提供其起始点和目标点的坐标，分别为 P_S 和 P_T 。因此，在每一步，代理需要执行以下操作。

1) It receives the status of the area from the sensing equipment.

1) 它从感应设备接收区域状态。**

2) It traces the square-shaped area surrounding the agent where the current agent position centers the square.

2) 它追踪围绕代理的正方形区域，当前代理位置位于正方形的中心。

3) It ignores the area beyond the square and uses the received sensing status to update every cell inside the square by (1) if it contains an intruder and (0) otherwise.

3) 它忽略正方形外的区域，并使用接收到的感知状态更新正方形内的每个单元格，如果包含入侵者则更新为 (1)，否则为 (0)。

4) It computes the relative address P_{rel} of the targeted point P_T in proportion to the current position of the agent.

4) 它计算目标点 P_{rel} 的相对地址 P_T ，与代理当前位置成比例。

5) Normalize the value of the targeted point relative position.

5) 规范化目标点的相对位置值。

6) It generates the current state S_t where $S_t = \{P_{\text{rel}} = (x_{\text{rel}}, y_{\text{rel}}), \Delta\}$.

6) 它生成当前状态 S_t ，其中 $S_t = \{P_{\text{rel}} = (x_{\text{rel}}, y_{\text{rel}}), \Delta\}$ 。

7) It uses the prior learned knowledge to choose the best action (direction) that might allow the agent to not collide with any nearby intruder and get closer to its target.

7) 它使用先前学到的知识来选择最佳动作 (方向)，以使代理不与附近的任何入侵者相撞并靠近目标。

8) Apply the selected action and observe the impact of the agent's dynamic on the environment by recording the new knowledge in terms of reward earned and new agent position.

8) 应用所选动作，并通过记录关于获得的奖励和新代理位置的新知识来观察代理动态对环境的影响。

9) The agent repeats the previous steps until reaching the targeted point P_T or an instance of collision occurs.

9) 代理重复前面的步骤，直到达到目标点 P_T 或发生碰撞实例。

Indeed, as shown in Fig. 5, state S_t introduced at each step highly impacts the learning process and the action selection. Thus, the state's definition needs to be done based on clear and logical arguments. In what follows, we detail our logic behind the definition of state S_t .

实际上，如图 5 所示，每一步引入的状态 S_t 对学习过程和动作选择有重大影响。因此，状态的定义需要基于清晰和逻辑的论证。接下来，我们详细说明我们定义状态 S_t 的逻辑。

First, instead of using the 2-D coordinate of the target, the use of the relative address allows the agent to move in the direction of the targeted position wherever its position is in the environment. Furthermore, the use of the relative address allows the agent to involve the remaining distance to the target in the learning process of the agent. Thus, we keep the agent seeking the shortest way to move on.

首先，代理不是使用目标的二维坐标，而是使用相对地址，使其能够无论在环境的哪个位置，都能向目标位置移动。此外，使用相对地址使得代理能够将剩余距离纳入到学习过程中。因此，我们保持代理寻找最短路径移动。

On the other side, the focus on the square surrounding the agent position instead of considering the whole environment area aims to aggregate many environment states to one state at the agent. Let Δ denote the surrounding area of the agent. The size of the surrounding area is a hyper-parameter that can be tuned during the training step. As shown in Fig. 5, three different environment states can be presented by the same state. However, they are the intruders' position beyond the square, and the state is the same since it considers only the intruders positioned inside the square and the relative targeted position. Indeed, the agent will know a limited number of states. Furthermore, the nonuse of related intruders features (e.g., intruders coordinates) aims to have a generalized view that might be used in similar status even with different intruders' positions. Finally, the use of such a state might help improve the learning convergence time of the agent. Moreover, this approach helps to train the agent on a limited number of intruders and can also be functional in an environment with a high number of intruders.

另一方面，关注代理位置周围的正方形而不是考虑整个环境区域，目的是在代理处将许多环境状态聚合为一个状态。令 Δ 表示代理周围的区域。周围区域的大小是一个超参数，可以在训练步骤中调整。如图 5 所示，三个不同的环境状态可以通过相同的状态来表示。然而，这些状态是入侵者位于正方形之外的位置，由于状态仅考虑位于正方形内部的入侵者和相对目标位置，因此状态是相同的。实际上，代理将知道有限数量的状态。此外，不使用相关入侵者特征 (例如，入侵者的坐标) 是为了获得一种泛化的视角，这种视角即使在不同的入侵者位置下也可能用于类似的状态。最后，使用这种状态可能有助于改善代理的学习收敛时间。此外，这种方法可以帮助在有限数量的入侵者上训练代理，并且在入侵者数量众多的环境中也可以使用。

D. RELIANCE DQN Algorithm

D. 依赖 DQN 算法

Throughout this section, we detail the RL approach used by our based RL agent. Several RL approaches exist in the literature, such as the Q -learning, deep neuron network (DQN), policy gradient, and actor-critic. However, the elements state-space and the action-space from the RL modeled system are either deterministic or continuous, highly impacting the selection of the RL approach. In our case, eight deterministic actions compose the action space. On the other side, the state-space contains two deterministic elements: 1) the relative address and 2) the square-shaped area. The Q -learning approach requires that the agent is within a deterministic limited space. Indeed, the Q -learning approach seems to be the most suitable for our problem. However, the size of the square-shaped area might be considerable. Then, the agent could take a long time to converge, and consequently, the computation process will consume more resources. Furthermore, the learning process can be less efficient by getting a useless action.

在本节中，我们详细介绍了我们的基础强化学习 (RL) 代理所使用的 RL 方法。文献中存在多种 RL 方法，例如 Q 学习、深度神经元网络 (DQN)、策略梯度和演员-评论家方法。然而，来自 RL 建模系统的状态空间和动作空间要么是确定性的，要么是连续的，这对 RL 方法的选择产生了重大影响。在我们的案例中，动作空间由八个确定性动作组成。另一方面，状态空间包含两个确定性元素:1) 相对地址和 2) 正方形区域。Q 学习方法要求代理位于一个确定性的有限空间内。实际上，Q 学习方法似乎最适合我们的问题。但是，正方形区域的大小可能相当大。因此，代理可能需要很长时间才能收敛，从而计算过程将消耗更多资源。此外，如果获得一个无用的动作，学习过程可能会效率较低。

To mitigate this problem, we opted to use the DRL (DQN) approach. Thus, more details about our based DQN autonomous UAV collision-avoidance and energy-aware agent are summarized in Algorithm 2. First, the agent starts by instantiating two different NNs, named the policy network Q^{π} , and target network Q^{T} . To ensure the fast convergence of the algorithm, we have used Xavier initialization to initialize the weights of both NNs Q^{π} and Q^{T} . The Xavier initialization helps to converge fast and prevent the exploding and vanishing gradients during the training process. To give the agent

为了缓解这个问题，我们选择使用深度强化学习 (DRL，即 DQN) 方法。因此，关于我们的基于 DQN 的自主无人机避障和节能代理的更多细节在算法 2 中进行了总结。首先，代理通过实例化两个不同的神经网络 (NN) 开始，分别命名为策略网络 Q^{π} 和目标网络 Q^{T} 。为了确保算法的快速收敛，我们使用了 Xavier 初始化来初始化两个 NN 的权重 Q^{π} 和 Q^{T} 。Xavier 初始化有助于快速收敛，并在训练过程中防止梯度爆炸和消失。为了给代理

Algorithm 2: RELIANCE: RL-Based Collision-

算法 2:RELIANCE: 基于 RL 的碰撞-

```
Avoidance Solution
  Input
       $Q^{\pi}$ and $Q^{T}$ : The initialized policy and target networks using
  Xavier-uniform.
       $\mathcal{B}$ : The batch replay memory size to size $N$ .
       batch_size: The size of each batch.
       $\mathcal{M}$ : A number of episodes to update $Q^{T}$ with $Q^{\pi}$ .
```

```
            $\xi_0$ : The initial value of epsilon greedy.
            $\mathcal{N}$ : Number of episodes.
    Output
            done: The UoI reaches $P_T$ or collides with one of the
    intruders.
    episode = 1 ;
    while episode $\leq N$ do
        done $\leftarrow$ False;
        $\xi \leftarrow \frac{\xi_0}{\xi_0 + \text{ episode}}$;
        $S_0 = \mathcal{E}$ .init( );
        while done = False do
            $S_t = \{P_{\text{rel}} = (x_{\text{rel}}, y_{\text{rel}}), \Delta\}$ ;
            if random () $\leq \xi$ then
                $a \leftarrow$ randint $(\mathcal{A})$ ;
            else
                $a \leftarrow \arg\max Q^\pi (S_t, a)$ ;
            end
            $S_{t+1}$ , reward, done $\leftarrow \mathcal{E}(S_t, a)$ ;
            $\mathcal{B} \leftarrow (S_t, a,$ reward $, S_{t+1},$ done $)$ ;
            $t \leftarrow t + 1$
            if size $(\mathcal{B}) \geq$ batch_size then
                mini_batch $\leftarrow$ random ($\mathcal{B}$ , batch_size ) ;
                foreach $(S_i, a_i,$ reward $_i,$ done $_i, S_i')$ $\in$ mini_batch do
                    if (done $e_i$ = True) then
                        $y_i \leftarrow$ reward $_i$
                    else
                        $y_i \leftarrow \text{reward}_i + \gamma \max_{a' \in \mathcal{A}} Q^T (S_i', a_i')$ ;
                    end
                end
                $\mathcal{L} = \frac{1}{N} \sum_{i=0}^{i=N-1} (Q^\pi (S_i, a_i), y_i)^2$;
                Update $\omega$ of $Q^\pi$ using $\mathcal{L}$ ;
            end
        end
        if episode $\%\mathcal{M} = 0$ then
            $Q^T \leftarrow Q^\pi$;
        end
        episode = episode +1;
end
```

more time to explore the behavior of the actions set, we opted to use the decayed ϵ -greedy strategy. The algorithm starts from the first episode and ends at the last episode N (Algorithm 2: lines 1 and 2). For each episode (Algorithm 2: lines 2-33), RELIANCE does the next steps. Initially, the episode sets to an undone state (Algorithm 2: line 3). Then, ξ is initialized to enable either the exploration or exploitation (Algorithm 2: line 4). Later, the environment is initialized by creating a new mission to train the agent (Algorithm 2: line 5).

为了有更多时间探索动作集的行为，我们选择使用衰减的 ϵ -贪婪策略。算法从第一个情节开始，到最后一个情节 N 结束 (算法 2: 第 1 和第 2 行)。对于每个情节 (算法 2: 第 2-33 行)，RELIANCE 执行以下步骤。最初，情节被设置为未完成状态 (算法 2: 第 3 行)。然后，ξ 被初始化以启用探索或利用 (算法 2: 第 4 行)。之后，通过创建一个新的任务来训练智能体，初始化环境 (算法 2: 第 5 行)。

While the episode is not completed (UoI achieves the target or collides), we do the following steps (Algorithm 2: lines $6-28$): first, the agent generates and normalize the current state (Algorithm 2: line 7). Then, according to the decayed value of ξ and a randomly generated number, we select either exploration or exploitation (Algorithm 2: lines 8-12). If the exploration is selected, a random action is issued from $\mathcal{A}$ (Algorithm 2: lines 8-10). Otherwise, the agent of the UoI chooses the action with the maximum reward previously earned using the policy network (Algorithm 2: lines 10-12). After the agent applies the selected action and saves the transition to $\mathcal{B}$ (Algorithm 2: lines 13 and 14), the agent moves to the new observed state (Algorithm 2: line 15). However, when the number of experiences exceeds the batch_size, the agent selects a random batch of transitions from $\mathcal{B}$ to update the Q^{π} following $TD(0)$ (Algorithm 2: lines 16-27). The agent keeps updating the Q^{T} using Q^{π} every $\mathcal{M}$ steps. The agent repeats the previous steps until the end of all the episodes in the training.

当情节尚未完成 (UoI 达到目标或发生碰撞) 时，我们会执行以下步骤 (算法 2: 行 $6-28$): 首先，代理生成并标准化当前状态 (算法 2: 第 7 行)。然后，根据 ξ 的衰减值和一个随机生成的数字，我们选择探索或利用 (算法 2: 行 8-12)。如果选择了探索，则从 $\mathcal{A}$ 发出一个随机动作 (算法 2: 行 8-10)。否则，UoI 的代理选择之前使用策略网络获得的最大奖励的动作 (算法 2: 行 10-12)。代理应用所选动作并将转换保存到 $\mathcal{B}$ (算法 2: 行 13 和 14) 后，代理移动到新的观察状态 (算法 2: 第 15 行)。但是，当经验数量超过 batch_size 时，代理从 $\mathcal{B}$ 中选择一个随机批次转换来更新 Q^{π} ，遵循 $TD(0)$ (算法 2: 行 16-27)。代理每 $\mathcal{M}$ 步使用 Q^{π} 更新 Q^{T} 。代理重复之前的步骤，直到训练中所有情节结束。

VI. EXPERIMENTATION AND RESULTS

VI. 实验与结果

In this section, we evaluate the performances of our two solutions PICA and RELIANCE. In the balance of this section, we first present the simulation setup; then, we present the convergence of RELIANCE during the training mode. Last but not least, we conclude this section by evaluating the performances of RELIANCE in inference mode to PICA.

在本节中，我们评估了我们两种解决方案 PICA 和 RELIANCE 的性能。在本节的其余部分，我们首先介绍仿真设置；然后，我们介绍 RELIANCE 在训练模式下的收敛性。最后但同样重要的是，我们通过将 RELIANCE 在推理模式下与 PICA 的性能进行比较来结束本节。

A. Simulation Setup

A. 仿真设置

Existing UAV simulators, such as Air Sim and software in the loop (SITL), use telemetry data to control the motion of a single UAV in a closed and well-controlled environment. These simulators mainly focus on telemetry data to maintain a single UAV for landing and flying. Still, they did not consider mobile objects, which is a handicap facing their utilization to evaluate PICA and RELIANCE solutions. In order to overcome these limitations, we have developed an OpenAI Gym [35] compliant simulator [1] with graphical rendering capability using the Python language and OpenCV library. This simulator provides a customizable environment that considers both static (e.g., building) and dynamic (e.g., UAVs and birds) obstacles. The static obstacles can be included in a JSON format to the simulator. In the simulator, we have adopted a discrete-time implementation of the events (e.g., UAVs mobility). This strategy helps to reduce the simulation time significantly by considering only the counted events rather than using real execution time. To make the proposed framework orthogonal on agents (PICA, RELIANCE...) implementation, we have developed a complete framework that consists of an abstraction layer of the agent and environment. We have also designed RELIANCE and PICA to be transparent in the environment and easily adapted to other simulators or real experiments later. Thus, we believe the suggestion of this simulator will have an added value to the scientific community. While PICA has been implemented using Python and Numpy, the NN model of RELIANCE is implemented with Python and Pytorch library.

现有的无人机模拟器，如 Air Sim 和软件在环 (SITL)，使用遥测数据来控制单个无人机在封闭且受控的环境中的运动。这些模拟器主要关注遥测数据，以维护单个无人机的着陆和飞行。然而，它们并未考虑移动对象，这是它们在评估 PICA 和 RELIANCE 解决方案时面临的一个障碍。为了克服这些限制，我们开发了一个符合 OpenAI Gym [35] 规范的模拟器 [1] ，该模拟器具有图形渲染能力，使用 Python 语言和 OpenCV 库。该模拟器提供了一个可定制的环境，考虑了静态 (例如，建筑物) 和动态 (例如，无人机和鸟类) 障碍物。静态障碍物可以以 JSON 格式包含在模拟器中。在模拟器中，我们采用了事件 (例如，无人机的移动性) 的离散时间实现。这种策略通过仅考虑计数事件而不是使用实际执行时间，大大减少了模拟时间。为了使所提出的框架与代理 (PICA、RELIANCE...) 的实现正交，我们开发了一个完整的框架，包括代理和环境的一个抽象层。我们还设计了 RELIANCE 和 PICA，使其在环境中透明，并且可以轻松适应其他模拟器或稍后的实际实验。因此，我们相信这个模拟器的提议将为科学社区带来额外的价值。虽然 PICA 使用 Python 和 Numpy 实现，但 RELIANCE 的 NN 模型使用 Python 和 Pytorch 库实现。

Besides the UoI, the environment also consists of a set of customizable number of static_intruders and mobile_intruders. As aforementioned, both UoI and mobile_intruders move in a 2-D plan using eight possible actions. The mobile_intruders move in the simulator using a random walk technique. In contrast to mobile_intruders, the UoI moves under the control of either PICA or RELIANCE agents. The rendering environment consists of a gray screen with black rectangles and blue, green, and red circles. As depicted in Fig. 6, the black rectangles and red circles refer to the static obstacles and the intruder(s), respectively. Meanwhile, the blue and green circles refer to the UoI and its targeted position, respectively. The simulation runs in episodes, such that each of which ends when UoI collides or reaches the target.

除了用户操作界面 (UoI) 之外，环境还包括一组可定制的静态入侵者和移动入侵者的数量。如前所述，用户操作界面和移动入侵者都在二维平面上使用八种可能的动作进行移动。移动入侵者在模拟器中使用随机游走技术进行移动。与移动入侵者相比，用户操作界面在 PICA 或 RELIANCE 代理的控制下移动。渲染环境由带有黑色矩形和蓝色、绿色、红色圆圈的灰色屏幕组成。如图 6 所示，黑色矩形和红色圆圈分别指代静态障碍物和入侵者。同时，蓝色和绿色圆圈分别指代用户操作界面和其目标位置。模拟以剧集的形式运行，每一集在用户操作界面发生碰撞或到达目标时结束。

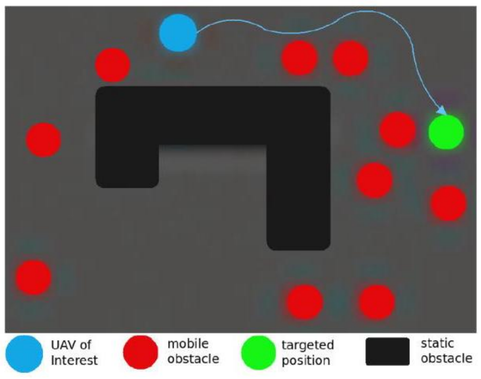


Fig. 6. OpenAI Gym compliant simulator.
图 6. 符合 OpenAI Gym 规范的模拟器。

B. RELIANCE Training Mode

B. RELIANCE 训练模式

The training of the RELIANCE model happens using 14 Dual Intel Xeon E5-2680 v3 @ 2.5 GHz, with 117 GB of RAM, one Nvidia P100 GPU, and running CentOS 7. During the training process, we have fixed the size of the simulation area by 20×20 and considered ten mobile_intruders besides three static obstacles with different shapes and sizes. We have fixed the hyperparameter surrounding area Δ of the agent by 5×5 after performing a set of different tests. To ensure fast convergence without underfitting or overfitting, we have tuned the NN hyperparameters used by RELIANCE. We have performed many experimental tests before fixing the hyper-parameters. We have fixed the discount factor γ by 0.95 and the learning rate α by 10^{-4} . We have also used two fully connected hidden layers in which the number of units (i.e., activation functions) is 40 . We have also tested with 400 units in each layer. However, a similar convergence rate is perceived. We have also observed similar performances during the inference mode. We adopted the rectified linear unit (ReLU) activation function for both hidden and output layers. An Xavier initialization has been adopted to initialize the NN units in the model. This initialization helps to converge fast and prevent the exploding and vanishing gradients during the training process. During the training, we have used batch_size = 1024 , replay buffer size = 500000 , and target update = 8 to update the weight of target network Q^T from the policy network Q^π .

RELIANCE 模型的训练使用 14 台双英特尔至强 E5-2680 v3 @ 2.5 GHz，配备 117 GB 的 RAM，一块 Nvidia P100 GPU，并运行 CentOS 7。在训练过程中，我们通过 20×20 固定了模拟区域的尺寸，并考虑了十个移动入侵者以及三个形状和大小不同的静态障碍物。在一组不同的测试之后，我们通过 5×5 固定了代理周围环境的超参数 Δ 。为了确保快速收敛且不过拟合或欠拟合，我们调整了 RELIANCE 使用的神经网络超参数。在确定超参数之前，我们进行了许多实验测试。我们将折扣因子 γ 设为 0.95，并将学习率 α 设为 10^{-4} 。我们还使用了两个全连接的隐藏层，其中单元数 (即激活函数) 为 40。我们还测试了每层 400 个单元的情况，但观察到的收敛率相似。在推理模式下，我们也观察到了相似的性能。我

们为隐藏层和输出层都采用了修正线性单元 (ReLU) 激活函数。采用了 Xavier 初始化方法来初始化模型中的 NN 单元。这种初始化方法有助于快速收敛，并在训练过程中防止梯度爆炸和消失。在训练过程中，我们使用了批量大小 $= 1024$ 、回放缓冲区大小 $= 500000$ 和目标更新 $= 8$ 来从策略网络 Q^{π} 更新目标网络的权重 Q^{T} 。

As depicted in Fig. 7, we have conducted two sets of experiments. Initially, we have trained one RELIANCE agent as depicted in Fig. 7(a) for a period of 2000 episodes. In this figure, while the blue curve shows the cumulative reward gained

如图 7 所示，我们进行了两组实验。最初，我们如图 7(a) 所示训练了一个 RELIANCE 代理，持续了 2000 个环节。在这个图中，蓝色曲线显示了累积奖励的获得情况

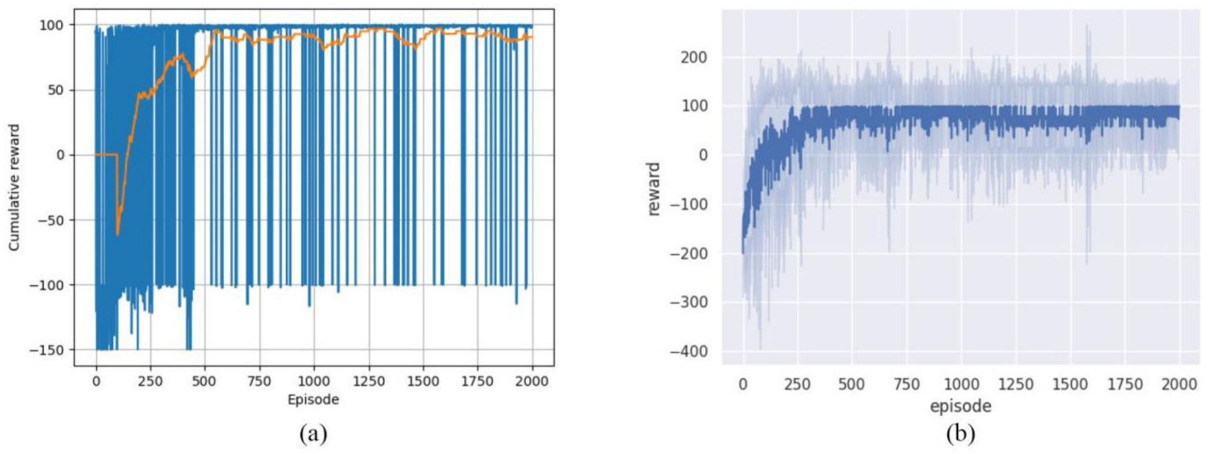


Fig. 7. Convergence evaluation of RELIANCE during the training mode. (a) Immediate and average rewards. (b) Average rewards of 40 agents simultaneously.

图 7. 在训练模式下对 RELIANCE 的收敛性评估。(a) 立即和平均奖励。(b) 40 个代理同时的平均奖励。

at the end of each episode, the red one shows the average of the last 50 cumulative rewards. From this figure, we observe that the RELIANCE agent converges at 600 episodes. Starting from that point, the RELIANCE agent succeeds in most of the cases to achieve the target without any collision. A live video has been recorded that shows the convergence of RELIANCE. [2]

在每个回合结束时，红色表示最后 50 个累积奖励的平均值。从这张图中，我们观察到 RELIANCE 代理在 600 个回合时收敛。从那时起，RELIANCE 代理在大多数情况下都能成功实现目标，且没有发生任何碰撞。已经录制了一个实时视频，展示了 RELIANCE 的收敛过程。[2]

Meanwhile, in Fig. 7(b), we have evaluated RELIANCE agent's stability. The NN's bias and weights are randomly initialized in the RELIANCE agent, affecting the training convergence. Moreover, at each episode, the starting and target point of UoI, and the mobility of mobile_intruders are randomly generated. In Fig. 7(b), we have trained 40 RELIANCE agents, simultaneously. In this figure, we have evaluated both the average and the cumulative variance reward achieved. We observe that all the agents succeeded in converging by getting almost the total possible reward after only 400 episodes. Also, we observe that the variance between the trained agents is close to 0, which confirms the algorithm's convergence.

同时，在图 7(b) 中，我们评估了 RELIANCE 代理的稳定性。在 RELIANCE 代理中，NN 的偏置和权重是随机初始化的，这影响了训练的收敛。此外，在每一回合，UoI 的起点和目标点以及移动干扰者的移动性都是随机生成的。在图 7(b) 中，我们同时训练了 40 个 RELIANCE 代理。在这个图中，我们评估了平均奖励和累积方差奖励。我们观察到所有代理在仅 400 个回合后便成功收敛，获得了几乎全部可能的奖励。我们还观察到，训练过的代理之间的方差接近 0，这证实了算法的收敛性。

[1] https://youtu.be/7UcRxfaAREw
[1] https://youtu.be/7UcRxfaAREw

C. PICA and RELIANCE Performance Evaluation During the Inference Mode

C. 在推理模式下 PICA 和 RELIANCE 的性能评估

In this section, we evaluate the performances of RELIANCE in the inference mode against the PICA solution. We simulate 10^3 episodes and compare the two solutions in terms of the following metrics.

在本节中，我们评估了 RELIANCE 在推理模式下与 PICA 解决方案的性能。我们模拟了 10^3 个回合，并按照以下指标比较了两种解决方案。

1) Percentage of Collision: It is defined as the percentage of times that the UAV agent collides with static_intruders or mobile_intruders. This metric shows the percentage of time that the UAV agent fails to achieve its final destination.

1) 碰撞百分比: 它定义为无人机代理与静态干扰者或移动干扰者发生碰撞的百分比。这个指标显示了无人机代理未能到达最终目的地的百分比。

2) PDF of Extra Traveled Distance: It shows the extra distance needed by a UAV to prevent collisions. This metric shows the probability of a distribution function (PDF) of the extra distance traveled to avoid collisions. In fact, the energy consumption in the UAVs is proportional to the traveled distance before attending the target location. Overall, the more the traveled distance, the higher energy consumption becomes.

2) 额外行驶距离的概率密度函数 (PDF): 它显示了无人机为避免碰撞所需的额外距离。这个指标反映了避免碰撞所需额外行驶距离的概率密度函数 (PDF)。实际上，无人机在到达目标位置之前的能量消耗与行驶距离成正比。总的来说，行驶距离越长，能量消耗就越高。

3) PDF of the Number of Success Before a Failure: It shows the PDF of the number of successes arriving at the target before the failure, i.e., the UoI collides with any object. In other words, this metric shows the capability of each solution for traveling consecutive missions without any collision.

3) 失败前成功次数的概率密度函数 (PDF): 它显示了在失败之前到达目标的成功次数的 PDF，即 UoI 与任何物体发生碰撞之前。换句话说，这个指标显示了每个解决方案在连续执行任务且不发生任何碰撞的能力。

To assess the generalization capability of RELIANCE, we have considered three different scenarios during the inference mode. As aforementioned, we have trained RELIANCE agent against static obstacles and ten mobile_intruders. In contrast, during the inference mode, besides the three static obstacles, we have evaluated the performance of PICA and RELIANCE agents against 5,10, and 20 mobile_intruders, respectively. The idea behinds these three scenarios is to show the capability of RELIANCE to outlive in unfamiliar environments by leveraging the effectiveness of surrounding area Δ and aggregated state.

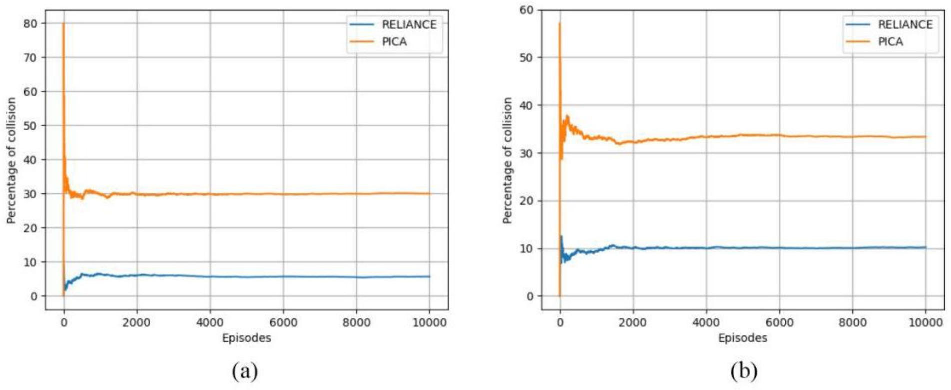
为了评估 RELIANCE 的泛化能力，我们在推理模式下考虑了三种不同的场景。如前所述，我们训练了 RELIANCE 代理以对抗静态障碍物和十个移动干扰者。相比之下，在推理模式下，除了三个静态障碍物之外，我们还分别评估了 PICA 和 RELIANCE 代理在面对 5、10 和 20 个移动干扰者时的性能。这三个场景背后的想法是展示 RELIANCE 通过利用周围区域 Δ 和聚合状态的有效性，在陌生环境中生存下来的能力。

1) Percentage of Collision: Fig. 8 shows the percentage of collisions as a function of the number of episodes. Both solutions have been evaluated in harsh conditions by including static obstacles and many mobile_intruders in a small area with dimensions 20×20 . Moreover, the mobile_intruders move randomly without any correlation, which makes hard to predict their next movement. [1] The first observation that we can draw from this figure is that RELIANCE ensures the generalization by behaving well in unseen environments (e.g., 5 and 20 mobile_intruders). We also observe that whatever the scenarios (5,10, or 20 mobile_intruders), the RELIANCE offers better performances than PICA. As expected, we also observe that the number of mobile_intruders has a negative impact on the number of collisions as shown in Fig. 8(a)-(c), respectively.

1) 碰撞百分比: 图 8 显示了碰撞百分比作为回合数量的函数。两种解决方案都在恶劣条件下进行了评估，包括在尺寸为 20×20 的小区域内加入静态障碍物和许多移动干扰者。此外，移动干扰者随机移动，没有任何相关性，这使得预测它们的下一步移动变得困难。[1] 我们可以从这张图中得出的第一个观察结果是，RELIANCE 通过在未见过的环境中表现良好 (例如，5 和 20 个移动干扰者) 来确保泛化。我们还观察到，无论场景如何 (5、10 或 20 个移动干扰者)，RELIANCE 都比 PICA 提供更好的性能。如预期，我们还可以观察到，移动干扰者的数量对碰撞数量有负面影响，如图 8(a)-(c) 所示。

For five mobile_intruders as depicted in Fig. 8(a), regardless of the number of episodes, the PICA agent has arrived at the target without collision with 70% of success. Whereas, the RELIANCE agent has succeeded with 95% to reach the target while avoiding the collisions. Increasing the number of mobile_intruders to 10 hurts the collision percentage in the network as depicted in Fig. 8(b). We observe that the percentage of cases that the UAV agent arrives at the target without collisions drooped out from 70% and 95% to 65% and 90% for PICA and RELIANCE, respectively. Finally, as depicted in Fig. 8(c), we observe that the increase of the number of mobile_intruders to 20 leads to reduce the percentage of success to arrive at the destination without collisions to 60% and 80% for PICA and RELIANCE, respectively.

对于图 8(a) 所示的 5 个移动干扰者，无论回合数量如何，PICA 代理都能在没有碰撞的情况下到达目标，成功率为 70% 。而 RELIANCE 代理成功率达到 95% 时，在避开碰撞的同时到达目标。将移动干扰者的数量增加到 10 个，会损害网络中的碰撞百分比，如图 8(b) 所示。我们观察到，无人机代理在没有碰撞的情况下到达目标的百分比从 70% 和 95% 下降到 PICA 和 RELIANCE 分别为 65% 和 90% 。最后，如图 8(c) 所示，我们观察到移动干扰者数量增加到 20 个导致无碰撞到达目的地的成功率分别降低到 PICA 的 60% 和 RELIANCE 的 80% 。



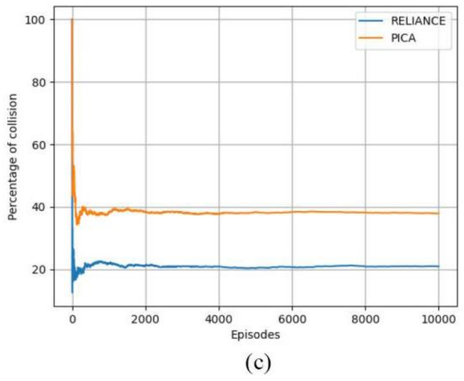


Fig. 8. Percentage of collision in PICA and RELIANCE solutions. (a) RELIANCE and PICA against five mobile_intruders. (b) RELIANCE and PICA against ten mobile_intruders. (c) RELIANCE and PICA against 20 mobile_intruders.

图 8. PICA 和 RELIANCE 解决方案中的碰撞百分比。(a) RELIANCE 和 PICA 对抗五个移动入侵者。(b) RELIANCE 和 PICA 对抗十个移动入侵者。(c) RELIANCE 和 PICA 对抗 20 个移动入侵者。

The better performances achieved by RELIANCE compared to PICA can be explained as follows. In both solutions, the algorithm controlling the UoI makes the decisions relying on the snapshot from the environment to avoid collisions. The environment snapshot refers to the surrounding area of UoI that is defined by Δ and $\mathcal{Z}$ in RELIANCE and PICA, respectively. On the one hand, based on this snapshot, PICA takes the action that minimizes the likelihood of collisions in $\mathcal{Z}$. Nonetheless, PICA does not consider the dynamics of the mobile intruders within $\mathcal{Z}$. In contrast, RELIANCE using the DRL approach can learn the temporal correlation between different snapshots Δ and therefore make more effective decisions to avoid mobile intruders. This fact explains why PICA exhibits a higher number of collisions compared to

RELIANCE 相比 PICA 实现的更好性能可以解释如下。在两种解决方案中，控制 UoI 的算法都是依赖环境快照来做出避免碰撞的决定。环境快照指的是由 Δ 和 $\mathcal{Z}$ 分别定义的 UoI 周围区域。一方面，基于这个快照，PICA 采取的行动是最大化减少 $\mathcal{Z}$ 中碰撞的可能性。然而，PICA 没有考虑 $\mathcal{Z}$ 中移动入侵者的动态。相比之下，RELIANCE 使用 DRL 方法可以学习不同快照 Δ 之间的时间相关性，因此能够做出更有效的决定来避免移动入侵者。这一事实解释了为什么 PICA 相比之下显示出更多的碰撞次数。
RELIANCE.

2) PDF of Extra Traveled Distance: Fig. 9 shows the performances of PICA and RELIANCE related to energy saving. Unfortunately, avoiding the collision comes with an unavoidable overhead in terms of the extra distance traveled by the UoI. This figure shows this extra distance compared to the traveled distance in the straight travel, i.e., the Euclidean distance between the UoI starting and target points. We have estimated the PDF of the PICA and RELIANCE extra distances from the results from 10^3 episodes. To that end, we employed the KernelDensity function from sklearn.neighbors. We observe that regardless of the number of mobile intruders (5,10, or 20), the percentage of extra distance does not exceed 60% , i.e., the UoI travels 1.6 times the distance of the optimal path.

2) 额外行驶距离的概率密度函数: 图 9 展示了 PICA 和 RELIANCE 在节能方面的性能。不幸的是，避免碰撞不可避免地带来了在行驶额外距离方面的开销，即 UoI 行驶的额外距离。该图展示了与直线行驶距离 (即 UoI 起点和目标点之间的欧几里得距离) 相比的额外距离。我们从 10^3 个场景的结果中估计了 PICA 和 RELIANCE 额外距离的概率密度函数。为此，我们使用了 sklearn.neighbors 中的 KernelDensity 函数。我们观察到，无论移动干扰者的数量 (5、10 或 20)，额外距离的百分比都不会超过 60% ，即 UoI 行驶的距离是最优路径的 1.6 倍。

Fig. 9(a) and (b) shows the PDF of the extra traveled distance for five mobile intruders. From Fig. 9(a), RELIANCE succeeded in almost 90% of cases to add only 35% of extra distance compared to the optimal one (i.e., 1.35 times). With more than 0.08 probability, RELIANCE succeeded in traveling the distance with less than 10% extra distance. We also observe from 9(b) that PICA succeeded in reaching the target in 70% of cases without adding any extra distance. Also, most of the extra distance of PICA does not exceed 40% . Observe that PICA offers shorter extra traveled distances than RELIANCE, which, overall, translates into energy saving. However, this reduction in the traveled extra distance offered by PICA is at the cost of a higher probability of collision, as discussed previously.

图 9(a) 和 (b) 展示了五个移动干扰者的额外行驶距离的概率密度函数。从图 9(a) 中，我们可以看到 RELIANCE 在几乎所有情况下仅增加了 35% 的额外距离，与最优距离相比 (即 1.35 倍)。RELIANCE 在超过 0.08 的概率下成功行驶了小于 10% 额外距离。我们还可以从 9(b) 中观察到 PICA 在 70% 的情况下成功到达目标点时没有增加任何额外距离。此外，PICA 的大部分额外距离没有超过 40% 。注意到 PICA 提供的额外行驶距离比 RELIANCE 短，总体上，这转化为节能。然而，正如之前讨论的，PICA 在减少额外行驶距离的同时，增加了碰撞的概率。

Fig. 9(c)-(f) shows the PDF of extra traveled distance for 10 and 20 mobile intruders, respectively. Similar to the case of five mobile intruders, we observe that the PICA algorithm succeeded in most of the cases without adding any extra distance. Interestingly, we observe that increasing the number of mobile intruders reduces the extra traveled distance offered by the PICA solution. This can be explained as follows, in the simulation, the extra distance of incomplete mission are filtered (not considered). At each episode, the starting and target point (i.e., mission) of UoI are randomly generated. In fact, increasing the number of intruders will create more collisions on the long distance missions comparing to the short ones. Hence, more short distance mission will participate for generating the PDF of extra distance. Usually, the probability of adding extra distance in shorter mission is lower than the longer ones, which positively affects the PDF of extra distance metric. Meanwhile, from Fig. 9(a), (c), and (e), we observe that similar behavior in terms of extra traveled distance. The RELIANCE solution succeeded to save long distance mission, however with unavoidable extra distance.

图 9(c)-(f) 展示了 10 个和 20 个移动入侵者额外行驶距离的概率密度函数 (PDF)。与五个移动入侵者的情形类似，我们观察到 PICA 算法在大多数情况下成功完成任务，没有增加任何额外距离。有趣的是，我们发现增加移动入侵者的数量会减少 PICA 解决方案提供的额外行驶距离。这可以这样解释: 在模拟中，未完成任务的额外距离被过滤 (不考虑)。在每一轮中，UoI 的起点和目标点 (即任务) 是随机生成的。实际上，增加入侵者的数量会在长距离任务上产生更多的碰撞，与短距离任务相比。因此，更多的短距离任务会参与生成额外距离的 PDF。通常，在较短任务中增加额外距离的概率低于长距离任务，这对额外距离指标的 PDF 产生积极影响。同时，从图 9(a)，(c) 和 (e) 中，我们观察到在额外行驶距离方面有类似的行为。RELIANCE 解决方案成功保存了长距离任务，但不可避免地增加了额外距离。

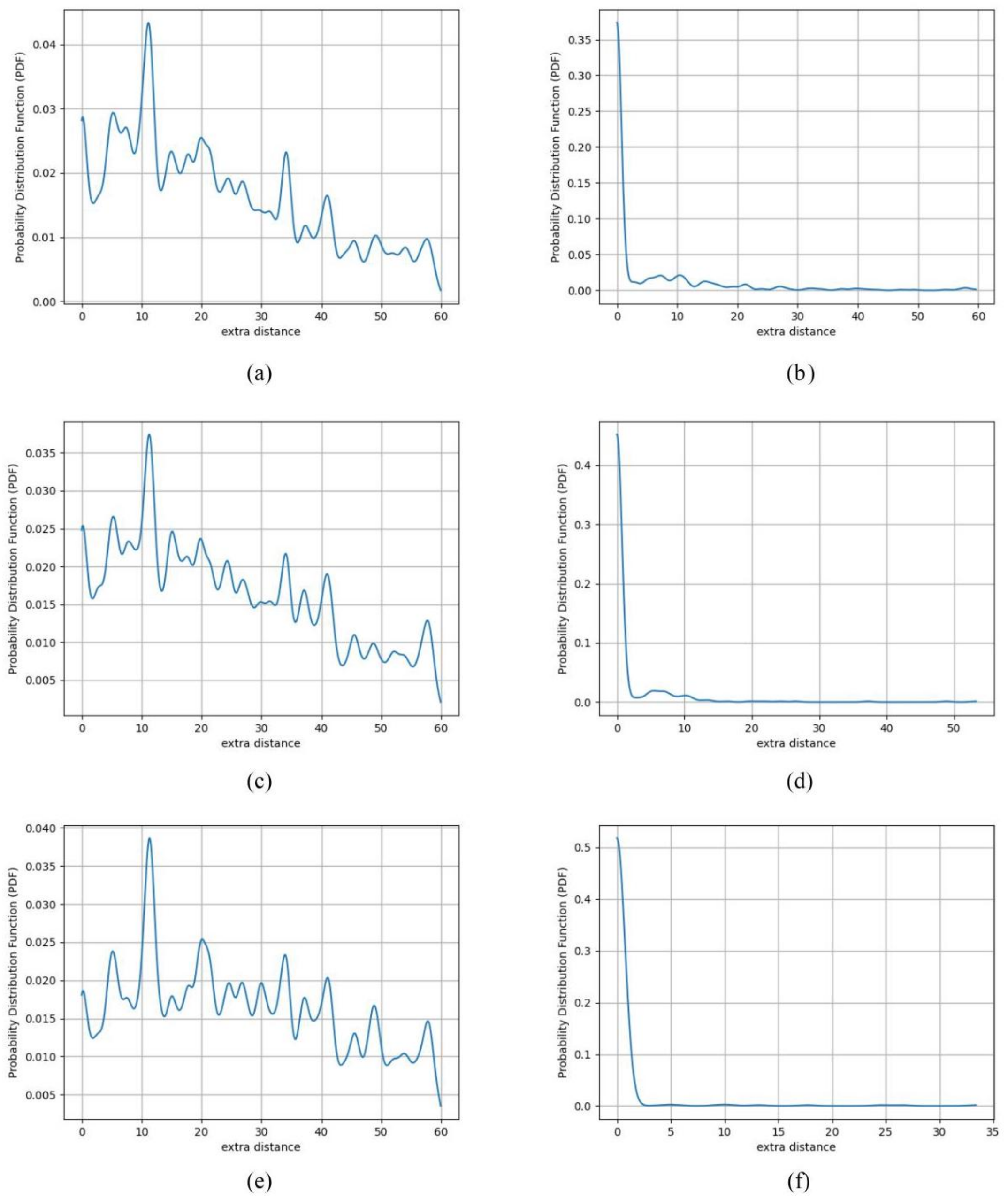


Fig. 9. PDF of extra traveled distance. (a) PDF of extra traveled distance in RELIANCE (five mobile_intruders). (b) PDF of extra traveled distance in PICA (five mobile_intruders). (c) PDF of extra traveled distance in RELIANCE (ten mobile_intruders). (d) PDF of extra traveled distance in PICA (ten mobile_intruders). (e) PDF of extra traveled distance in RELIANCE (20 mobile_intruders). (f) PDF of extra traveled distance in PICA (20 mobile_intruders).

图 9. 额外行驶距离的概率密度函数 (PDF)。(a) RELIANCE 中五个移动入侵者的额外行驶距离 PDF。(b) PICA 中五个移动入侵者的额外行驶距离 PDF。(c) RELIANCE 中十个移动入侵者的额外行驶距离 PDF。(d) PICA 中十个移动入侵者的额外行驶距离 PDF。(e) RELIANCE 中 20 个移动入侵者的额外行驶距离 PDF。(f) PICA 中 20 个移动入侵者的额外行驶距离 PDF。

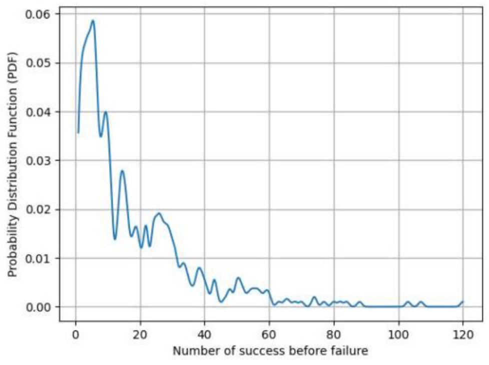
The extra traveled distance and percentage of collision are two contradictory objectives. The lower percentage of collision is, the higher likelihood of extra traveled distance becomes. While the PICA solution leverages a probabilistic approach by considering only one snapshot of the environment, RELIANCE employs DRL to make the correlation between snapshots and then takes the decisions that consider the mobility of intruders. The safety level, i.e., low probability of collision with surrounding intruders, offered by RELIANCE is at the expense of traveling longer extra distances.

额外行驶距离和碰撞百分比是两个相互矛盾的目标。碰撞百分比越低，额外行驶距离的可能性就越高。PICA 解决方案采用概率方法，仅考虑环境的一个快照；而 RELIANCE 则采用深度强化学习 (DRL)

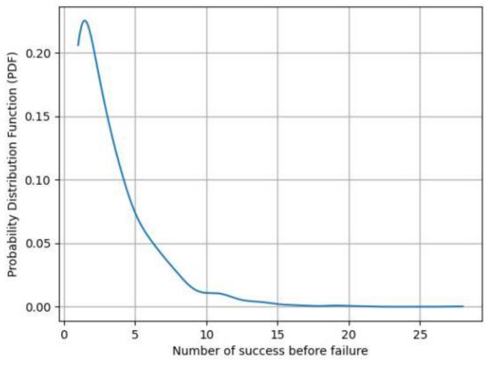
来建立快照之间的相关性，然后做出考虑入侵者移动性的决策。RELIANCE 提供的安全水平，即与周围入侵者发生碰撞的概率较低，是以行驶更长的额外距离为代价的。

3) PDF of the Number of Success Before Failure: Fig. 10 depicts the PDF of the number of success before a failure happens. It shows the PDF of the number of hits arriving at the target before the collapse. This metric shows the capability of each solution for traveling consecutive missions without any collision. We have conducted three sets of experiments by varying the number of mobile_intruders from 5,10, and 20, respectively. The first observation that we can draw from this figure is that the RELIANCE solution performs better than the PICA solution. Also, we observe that the number of mobile_intruders harms the number of successes before failure.

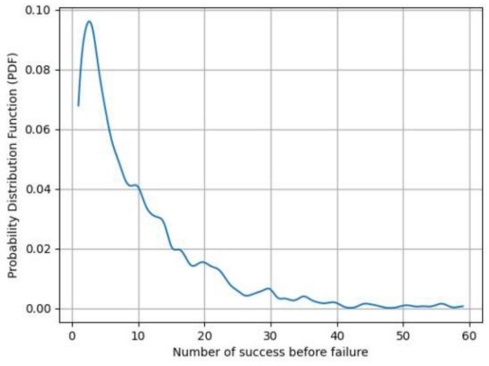
3) 失败前的成功次数的概率密度函数 (PDF): 图 10 描述了失败前成功次数的 PDF。它显示了在崩溃前到达目标的成功次数的 PDF。这个指标显示了每种解决方案在连续执行任务而不发生任何碰撞的能力。我们进行了三组实验，分别改变移动干扰者的数量为 5、10 和 20。从这张图中我们可以得出的第一个观察结果是 RELIANCE 解决方案的表现优于 PICA 解决方案。我们还观察到移动干扰者的数量对失败前的成功次数产生了影响。



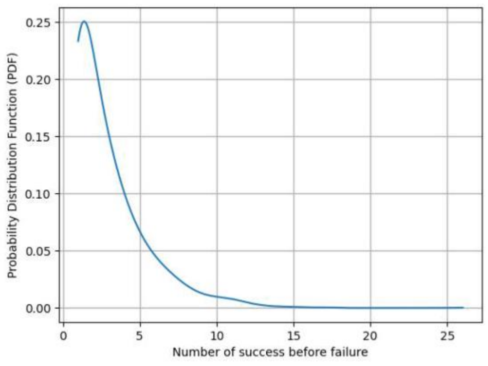
(a)



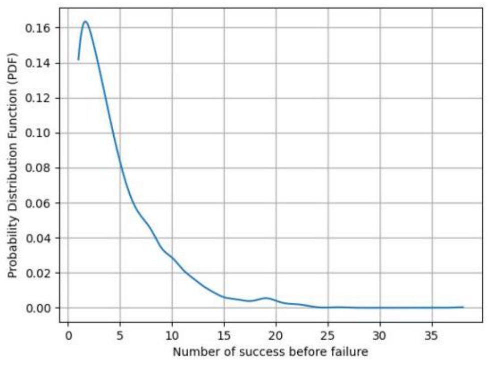
(b)



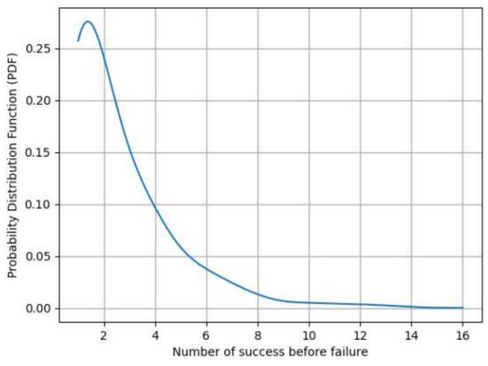
(c)



(d)



(e)



(f)

Fig. 10. PDF of extra traveled distance. (a) PDF of the number of success before a failure in RELIANCE (five intruders). (b) PDF of the number of success before a failure in PICA (five intruders).

(c) PDF of the number of success before a failure in RELIANCE (ten intruders). (d) PDF of number of success before a failure in PICA (ten intruders). (e) PDF of the number of success before a failure in RELIANCE (20 intruders). (f) PDF of the number of success before a failure in PICA (20 intruders).

图 10. 额外行驶距离的 PDF。(a)RELIANCE 中失败前成功次数的 PDF(五个干扰者)。(b)PICA 中失败前成功次数的 PDF(五个干扰者)。(c)RELIANCE 中失败前成功次数的 PDF(十个干扰者)。(d)PICA 中失败前成功次数的 PDF(十个干扰者)。(e)RELIANCE 中失败前成功次数的 PDF(20 个干扰者)。(f)PICA 中失败前成功次数的 PDF(20 个干扰者)。

Fig. 10(a) and (b) shows the performances of PICA and RELIANCE when five mobile_intruders is considered. As depicted in these figures, while RELIANCE succeeded in getting 120 successful episodes achieving the target safely, PICA succeeded in achieving the target in 30 episodes without any single failure. We also observe that RELIANCE's probability of fewer than five times consecutively arriving at the target without interruption does not exceed 12% . Meanwhile, in PICA, UoI with a probability of almost 1 does not exceed the 15 episodes consecutively. Fig. 10(c) and (d) shows the impact of ten mobile_intruders on PICA and RELIANCE. We can observe that the increase in the number of mobile_intruders hurts the number of successes before failure. In RELIANCE, the number of successful episodes before a failure is drooped from 120 to 60 . Also, the probability of five consecutive times arrive at the target without interruption does not exceed 20% . Finally, Fig. 10(e) and (f) shows the impact of ten mobile_intruders on the two solutions. We observe that in 1/3 of cases RELIANCE, the number of success before a failure does not exceed the threshold 5. Meanwhile, for PICA, the agent with almost probability 1 does not succeed to exceed 15 episodes.

图 10(a) 和 (b) 展示了当考虑五个移动入侵者时 PICA 和 RELIANCE 的性能。如图所示，RELIANCE 成功地在 120 个成功的片段中安全地达到目标，而 PICA 在 30 个片段中成功达到目标，并且没有一次失败。我们还观察到 RELIANCE 连续不到五次无干扰到达目标的概率不超过 12% 。同时，在 PICA 中，几乎概率为 1 的 UoI 连续不超过 15 个片段。图 10(c) 和 (d) 展示了十个移动入侵者对 PICA 和 RELIANCE 的影响。我们可以观察到移动入侵者数量的增加影响了失败前的成功次数。在 RELIANCE 中，失败前的成功片段数从 120 下降到 60。此外，连续五次到达目标的概率不超过 20% 。最后，图 10(e) 和 (f) 展示了十个移动入侵者对两种解决方案的影响。我们观察到在 1/3 的情况下，RELIANCE 失败前的成功次数不超过阈值 5。同时，对于 PICA，几乎概率为 1 的代理无法超过 15 个片段。

VII. CONCLUSION

VII. 结论

The new enthusiasm for extending UAV commercial operations to cover the urban and populated area controlled airspace BVLOS comes with unavoidable challenges related to object detection and collision avoidance. In this article, we suggested two solutions named: 1) PICA framework and 2) RELIANCE. While the PICA solution leverages the probability density for avoiding collisions, RELIANCE uses the DQN technique to prevent collisions while saving energy consumption. We have also developed an OpenAI Gym [35] compliant environment [1] with graphical rendering capability using Python language and OpenCV library to evaluate these two solutions. We have developed a complete framework that includes an abstraction of the environment and agent. Our plan to make the platform's code source, including PICA and RELIANCE agents, public for the research community.

对扩展无人机商业运营至城市和人口密集区域控制的空域 BVLOS 的新热情带来了与目标检测和避障相关的不可避免挑战。在本文中，我们提出了两种解决方案，分别命名为:1)PICA 框架和 2)RELIANCE。PICA 解决方案利用概率密度来避免碰撞，而 RELIANCE 使用 DQN 技术来防止碰撞的同时节省能耗。我们还开发了一个符合 OpenAI Gym [35] 规范的、具有图形渲染能力的环境，使用 Python 语言和 OpenCV 库来评估这两种解决方案。我们开发了一个完整的框架，包括环境和代理的抽象。我们计划将平台的源代码公开，包括 PICA 和 RELIANCE 代理，供研究界使用。

We have simulated the agent in the context of both PICA and RELIANCE under similar circumstances. The agent behaves successively following PICA or RELIANCE to prevent the collision and save energy consumption. We have evaluated both protocols in known and unknown environments to assist their generalization capability. The obtained results demonstrate their capacity for generalization. Also, they show the superiority of RELIANCE over PICA in terms of collision avoidance. Also, the simulation results demonstrated the convergence of RELIANCE during the training process [2] .

我们在 PICA 和 RELIANCE 的背景下模拟了代理，在相似情况下其行为依次遵循 PICA 或 RELIANCE 来防止碰撞和节省能耗。我们在已知和未知环境中评估了这两种协议，以帮助它们具备泛化能力。获得的结果证明了它们的泛化能力。同时，它们也显示了 RELIANCE 在避障方面的优越性。此外，

模拟结果还证明了 RELIANCE 在训练过程中的收敛性 ^{2} 。

As a future research direction, we plan to consider other RL Algorithms, including 1) the Policy gradient method, such as RELIANCE; Actor-Critic approach, including but not limited to A3C, deep deterministic policy gradient (DDPG), trust region policy optimization (TRPO), and proximal policy optimization (PPO). Also, we plan to consider more complex scenarios by considering the velocity and the acceleration of UAVs. A real deployment implementation is envisaged of RELIANCE by leveraging the UAVs available in our lab.

作为未来的研究方向，我们计划考虑其他强化学习算法，包括 1) 策略梯度方法，如 RELIANCE；演员-评论家方法，包括但不限于 A3C、深度确定性策略梯度 (DDPG)、信任域策略优化 (TRPO) 和近端策略优化 (PPO)。我们还计划通过考虑无人机的速度和加速度来考虑更复杂的场景。我们计划通过利用我们实验室中可用的无人机来实现 RELIANCE 的实地部署。

REFERENCES

参考文献

[1] N. H. Motlagh, T. Taleb, and O. Arouk, ”Low-altitude unmanned aerial vehicles-based Internet of Things services: Comprehensive survey and future perspectives,” IEEE Internet Things J., vol. 3, no. 6, pp. 899-922, Dec. 2016.

[2] D. Jaiswal and P. Kumar, ”Real-time implementation of moving object detection in UAV videos using GPUs,” J. Real Time Image Process., vol. 17, no. 5, pp. 1301-1317, 2020.

[3] S. Ouahouah, J. Prados-Garzon, T. Taleb, and C. Benzaid, ”Energy-aware collision avoidance stochastic optimizer for a UAVs set,” in Proc. Int. Wireless Commun. Mobile Comput. (IWCMC), 2020, pp. 1636-1641.

[4] T. Taleb, K. Ooi, and K. Hashimoto, ”An efficient collision avoidance strategy in ITS systems,” in Proc. IEEE Wireless Commun. Netw. Conf., Mar. 2008, pp. 2212-2217.

[5] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, ”Toward an effective risk-conscious and collaborative vehicular collision avoidance system,” IEEE Commun. Mag., vol. 55, no. 3, pp. 38-43, Mar. 2010.

[6] F. Giancarmine et al., ”Multi-sensor-based fully autonomous noncooperative collision avoidance system for unmanned air vehicles,” *J* . Aerosp. Comput. Inf. Commun., vol. 5, no. 10, pp. 338-360, 2008.

[7] R. Sharma and D. Ghose, ”Collision avoidance between UAV clusters using swarm intelligence techniques,” Int. J. Syst. Sci., vol. 40, no. 5,

[8] Y. K. Kwag and C. H. Chung, ”UAV based collision avoidance radar sensor,” in Proc. IEEE Int. Geo-Sci. Remote Sens. Symp. (IGARSS), Jul. 2007, pp. 639-642.

[9] J. W. Park, H. D. Oh, and M. J. Tahk, ”UAV collision avoidance based on geometric approach,” in Proc. SICE Annu. Conf., Aug. 2008, pp. 2122-2126.

[10] J. P. K. Kim and M. Tahk, ”UAV collision avoidance using probabilistic method in 3-D,” in Proc. Int. Conf. Control Autom. Syst., Aug. 2007, pp. 826-829.

[11] M. Shanmugavel, A. Tsourdos, and B. A. White, ”Collision avoidance and path planning of multiple UAVs using flyable paths in 3D,” in Proc. 15th Int. Conf. Methods Models Autom. Robot., Aug. 2010, pp. 218-222.

[12] Z. Chao, L. Ming, Z. Shaolei, and Z. Wenguang, ”Collision-free UAV formation flight control based on nonlinear MPC,” in Proc. Int. Conf. Electron. Commun. Control (ICECC), Sep. 2011, pp. 1951-1956.

[13] J. G. Manathara and D. Ghose, ”Reactive collision avoidance of multiple realistic UAVs,” Aircraft Eng. Aerosp. Technol., vol. 83, no. 6, pp. 388-396, 2011. [Online]. Available: https://doi.org/10.1108/00022661111173261

[14] M. C. P. Santos, C. D. Rosales, M. Sarcinelli-Filho, and R. Carelli, ”A novel null-space-based UAV trajectory tracking controller with collision avoidance,” IEEE/ASME Trans. Mechatronics, vol. 22, no. 6, pp. 2543-2553, Dec. 2017.

[15] L. A. Tony, D. Ghose, and A. Chakravarthy, ”Avoidance maps: A new concept in UAV collision avoidance,” in Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS), Jun. 2017, pp. 1483-1492.

[16] R. He, R. Wei, and Q. Zhang, ”UAV autonomous collision avoidance approach,” Automatika, vol. 58, no. 2, pp. 195-204, 2017. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00051144.2017.1388646

[17] Y. Lin and S. Saripalli, ”Sampling-based path planning for UAV collision avoidance,” IEEE Trans. Intell. Transp. Syst., vol. 18, no. 11, pp. 3179-3192, Nov. 2017.

[18] S. Ouahouah, J. Prados-Garzon, T. Taleb, and C. Benzaid, ”Energy and delay aware physical collision avoidance in unmanned aerial vehicles,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), 2018, pp. 1-7.

[19] S. Y. Choi and D. Cha, ”Unmanned aerial vehicles using machine learning for autonomous flight; state-of-the-art,” Adv. Robot., vol. 33, no. 6, pp. 265-277, 2019.

[20] A. Singla, S. Padakandla, and S. Bhatnagar, ”Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 1, pp. 107-118, Jan. 2021.

[21] S.-Y. Shin, Y.-W. Kang, and Y.-G. Kim, ”Obstacle avoidance drone by deep reinforcement learning and its racing with human pilot,” Appl. Sci., vol. 9, no. 24, p. 5571, Dec. 2019. [Online]. Available: http://dx.doi.org/10.3390/app9245571

[22] P. Fraga-Lamas, L. Ramos, V. Mondéjar-Guerra, and T. M. Fernández-Caramés, ”A review on IoT deep learning UAV systems for autonomous obstacle detection and collision avoidance,” Remote Sens., vol. 11, no. 18, p. 2144, 2019. [Online]. Available: http://dx.doi.org/10.3390/ rs11182144

[23] D. Wang, T. Fan, T. Han, and J. Pan, ”A two-stage reinforcement learning approach for multi-UAV collision avoidance under imperfect sensing,” IEEE Robot. Autom. Lett., vol. 5, no. 2, pp. 3098-3105, Apr. 2020.

[24] K. Wan, X. Gao, Z. Hu, and G. Wu, ”Robust motion control for UAV in dynamic uncertain environments using deep reinforcement learning,” Remote Sens., vol. 12, no. 4, p. 640, 2020. [Online]. Available: http://dx.doi.org/10.3390/rs12040640

[25] B. M. Albaker and N. A. Rahim, ”A survey of collision avoidance approaches for unmanned aerial vehicles,” in Proc. Int. Conf. Tech. Postgraduates (TECHPOS), Dec. 2009, pp. 1-7.

[26] L. Xiao, W. Zhuang, S. Zhou, and C. Chen, UAV Relay in VANETs Against Smart Jamming With Reinforcement Learning. Cham, Switzerland: Springer, 2019, pp. 105-129. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-01731-6_5

[27] L. Xiao, X. Lu, D. Xu, Y. Tang, L. Wang, and W. Zhuang, ”UAV relay in VANETs against smart jamming with reinforcement learning,” IEEE Trans. Veh. Technol., vol. 67, no. 5, pp. 4087-4097, May 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8246580

[28] O. Bekkouche, T. Taleb, and M. Bagaa, ”UAVs traffic control based on multi-access edge computing,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2018, pp. 1-6.

[29] T.-H. Yi, H.-N. Li, and M. Gu, ”Experimental assessment of high-rate GPS receivers for deformation monitoring of bridge,” Measurement, vol. 46, no. 1, pp. 420-432, Aug. 2012.

[30] S. V. Amarasinghe, H. S. Hewawasam, W. B. D. K. Fernando, J. V. Wijayakulasooriya, G. M. R. I. Godaliyadda, and M. P. B. Ekanayake, ”Vision based obstacle detection and map generation for reconnaissance,” in Proc. 9th Int. Conf. Ind. Inf. Syst. (ICIIS), 2014, pp. 1-6.

[31] D. Kim and H. Ryu, ”Obstacle recognition system using ultrasonic sensor and duplex radio-frequency camera for the visually impaired person,” in Proc. 13th Int. Conf. Adv. Commun. Technol. (ICACT), 2011, pp. 326-329.

[32] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[33] V. Mnih et al., ”Playing Atari with deep reinforcement learning,” 2013. [Online]. Available: http://arxiv.org/abs/1312.5602.

[34] V. Mnih et al., ”Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529-533, 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[35] Gym. Accessed: May 26, 2021. [Online]. Available: https://gym.openai.com/envs/#classic_control

Sihem Ouahouah received the Engineering degree from the University of Science and Technology Houari Boumediene, Bab Ezzouar, Algeria, in 2005, and the master's degree in computer science from Ecole Nationale Supérieure d'Informatique, Oued Smar, Algeria, in 2015. She is currently pursuing the Doctoral degree with Aalto University, Espoo, Finland.

Sihem Ouahouah 于 2005 年从阿尔及利亚巴布祖阿尔的科学与技术大学获得工学学位，并于 2015 年从阿尔及利亚乌斯马尔国家高等信息学校获得计算机科学硕士学位。她目前正于芬兰埃斯波的阿尔托大学攻读博士学位。

Her research interests include unmanned aerial vehicles, machine learning, and the Internet of Things.

她的研究兴趣包括无人航空器、机器学习和物联网。

Miloud Bagaa (Senior Member, IEEE) received the engineer's, master's, and Ph.D. degrees from the University of Science and Technology Houari Boumediene, Bab Ezzouar, Algeria, in 2005, 2008, and 2014, respectively.

Miloud Bagaa(IEEE 高级会员) 分别于 2005 年、2008 年和 2014 年在阿尔及利亚巴布祖阿尔的科学与技术大学获得工学学位、硕士学位和博士学位。

From 2009 to 2015, he was a Researcher with the Research Center on Scientific and Technical Information, Ben Aknoun, Algeria. From 2015 to 2016, he was granted a Postdoctoral Fellowship from the European Research Consortium for Informatics and Mathematics, and worked with the Norwegian University of Science and Technology, Trondheim, Norway. From 2016 to 2019, he was a Postdoctoral Researcher with Aalto University, Espoo, Finland, then a Senior Researcher from 2019 to October 2020, where he is currently a Senior Cloud Specialist with the IT Center For Science LTD. His research interests include machine learning, optimization, networking modeling, and network slicing.

从 2009 年至 2015 年，他在阿尔及利亚本阿克努的科学研究与技术信息中心担任研究员。从 2015 年至 2016 年，他获得欧洲信息与数学研究联盟的博士后奖学金，并在挪威特隆赫姆的挪威科技大学工作。从 2016 年至 2019 年，他是芬兰埃斯波的阿尔托大学的博士后研究员，然后从 2019 年至 2020 年 10 月担任高级研究员，目前他是科学 LTD IT 中心的资深云专家。他的研究兴趣包括机器学习、优化、网络建模和网络切片。

Jonathan Prados-Garzon received the B.Sc., M.Sc., and Ph.D. degrees from the University of Granada, Granada, Spain, in 2011, 2012, and 2018, respectively.

Jonathan Prados-Garzon 分别于 2011 年、2012 年和 2018 年在西班牙格拉纳达大学获得理学学士、理学硕士和博士学位。

From 2018 to 2020, he worked as a Postdoctoral Researcher with MOSA!C Lab, led by Prof. T. Taleb, and the Department of Communications and Networking, Aalto University, Espoo, Finland. He is currently a Postdoctoral Researcher with WiMuNet Lab, headed by Prof. J. M. L. Soler, and the

Department of Signal Theory, Telematics and Communications, University of Granada. His research interests include mobile broadband networks, network softwariza-tion, deterministic networking, and network performance modeling and optimization.

从 2018 年到 2020 年，他在芬兰埃斯波阿尔托大学的通信与网络系以及由 T. Taleb 教授领导的 MOSA!C 实验室担任博士后研究员。他目前是西班牙格拉纳达大学信号理论、远程通信和通信系的 WiMuNet 实验室博士后研究员，该实验室由 J. M. L. Soler 教授领导。他的研究兴趣包括移动宽带网络、网络软件化、确定性网络以及网络性能建模与优化。

Tarik Taleb received the B.E. degree (with Distinction) in information engineering and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively.

Tarik Taleb 分别于 2001 年、2003 年和 2005 年在日本仙台东北大学获得信息工程学士学位 (优异)、信息科学硕士学位和博士学位。

He is currently a Professor with the School of Electrical Engineering, Aalto University, Espoo, Finland, where he is the Founder and Director of the MOSA!C Lab. He is also working as a Part Time Professor with the Center of Wireless Communications, University of Oulu, Oulu, Finland. Prior to his current academic position, he was working as a Senior Researcher and a 3GPP Standards Expert with NEC Europe Ltd., Heidelberg, Germany. He is also affiliated with Sejong University, Seoul, South Korea. He was then leading the NEC Europe Labs Team working on Research and Development projects on carrier cloud platforms, an important vision of 5G systems. Before joining NEC and till March 2009, he worked as an Assistant Professor with the Graduate School of Information Sciences, Tohoku University, Sendai, in a lab fully funded by KDDI. From October 2005 to March 2006, he worked as a Research Fellow with the Intelligent Cosmos Research Institute, Sendai. He has been also directly engaged in the development and standardization of the Evolved Packet System as a member of 3GPP's System Architecture working group. His research interests lie in the field of architectural enhancements to mobile core networks (particularly, 3GPP's), network softwarization and slicing, mobile cloud networking, network function virtualization, software defined networking, mobile multimedia streaming, intervehicular communications, and social media networking.

他目前是芬兰埃斯波阿尔托大学电气工程学院的教授，同时也是 MOSA!C 实验室的创立者和负责人。他还担任芬兰奥卢大学无线通信中心的兼职教授。在目前的学术职位之前，他曾在德国海德堡的 NEC 欧洲有限公司担任高级研究员和 3GPP 标准专家。He 还与韩国首尔的首尔大学有隶属关系。当时，他领导 NEC 欧洲实验室团队，从事关于运营商云平台的研究和开发项目，这是 5G 系统的一个重要愿景。在加入 NEC 之前至 2009 年 3 月，他在仙台东北大学信息科学研究生院担任助理教授，所在实验室完全由 KDDI 资助。2005 年 10 月至 2006 年 3 月，他在仙台的智能宇宙研究所担任研究员。他还直接参与了 Evolved Packet System 的发展和标准化工作，作为 3GPP 系统架构工作组的成员。他的研究兴趣在于移动核心网络 (特别是 3GPP 的) 架构增强、网络软化与切片、移动云网络、网络功能虚拟化、软件定义网络、移动多媒体流、车际通信和社交媒体网络。

Prof. Taleb was a recipient of the 2017 IEEE ComSoc Communications Software Technical Achievement Award in December 2017, for his outstanding contributions to network softwarization. He is also the (co)-recipient of the 2017 IEEE Communications Society Fred W. Ellersick Prize in May 2017, the 2009 IEEE ComSoc Asia-Pacific Best Young Researcher Award in June 2009, the 2008 TELECOM System Technology Award from the Telecommunications Advancement Foundation in March 2008, the 2007 Funai Foundation Science Promotion Award in April 2007, the 2006 IEEE Computer Society Japan Chapter Young Author Award in December 2006, the Niwa Yasujirou Memorial Award in February 2005, and the Young Researcher's Encouragement Award from the Japan chapter of the IEEE Vehicular Technology Society in October 2003. Some of his research works have been also awarded best paper awards at prestigious IEEE-flagged conferences. He is a member of the IEEE Communications Society Standardization

Program Development Board. He is/was on the Editorial Board of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE Wireless Communications Magazine, IEEE JOURNAL ON INTERNET OF THINGS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and a number of Wiley journals.

Taleb 教授是 2017 年 12 月 IEEE ComSoc 通信软件技术成就奖的获得者，以表彰他在网络软化方面的杰出贡献。他也是 2017 年 5 月 IEEE 通信学会 Fred W. Ellersick 奖的 (共同) 获得者，2009 年 6 月 IEEE ComSoc 亚太最佳青年研究者奖的获得者，2008 年 3 月电信发展基金会颁发的电信系统技术奖的获得者，2007 年 4 月 Funai 基金会科学促进奖的获得者，2006 年 12 月 IEEE 计算机学会日本分会青年作者奖的获得者，2005 年 2 月 Niwa Yasujirou 纪念奖的获得者，以及 2003 年 10 月 IEEE 车辆技术学会日本分会青年研究者鼓励奖的获得者。他的一些研究作品在享有盛誉的 IEEE 标志性会议上也获得了最佳论文奖。他是 IEEE 通信学会标准化程序发展委员会的成员。他是/曾是《IEEE 无线通信交易》、《IEEE 无线通信杂志》、《IEEE 物联网杂志》、《IEEE 车辆技术交易》、《IEEE 通信调查与教程》以及多家 Wiley 期刊的编辑委员会成员。