Distributed Computational Guidance for High-Density Urban Air Mobility with Cooperative and Non-Cooperative Collision Avoidance
高密度城市空中出行分布式计算引导与合作与非合作避障
Conference Paper - January 2020
会议论文 - 2020 年 1 月
DOI: 10.2514/6.2020-1371
DOI: 10.2514/6.2020-1371READS 507Peng Wei George Washington University 112 PUBLICATIONS 1,608 CITATIONS SEE PROFILE
彭伟乔治华盛顿大学 112 篇论文 1,608 次引用查看个人资料

Distributed Computational Guidance for High-Density Urban Air Mobility with Cooperative and Non-Cooperative Collision Avoidance

高密度城市空中出行分布式计算引导与合作与非合作避障

Joshua R. Bertram Peng Wei
约书亚 · R · 伯特拉姆彭伟
Iowa State University
爱荷华州立大学
Ames, IA 50011
爱荷华州艾姆斯市，IA 50011
{bertram1, pwei}@iastate.edu
{bertram1, pwei}@iastate.edu

Urban Air Mobility (UAM) will result in numerous aircraft which will need to respond to a dynamically changing airspace to safely reach their destinations. We use a highly efficient Markov Decision Process (MDP) based trajectory planner to demonstrate multi-agent distributed trajectory planning that can safely avoid cooperative and non-cooperative actors in the high-density free flight airspace. We demonstrate the algorithm in an urban setting where multiple aircraft all navigate to their designated landing site while avoiding cooperative and non-cooperative aircraft. We study the algorithm performance as the aircraft density increases to demonstrate scalability and the effect of cooperative versus non-cooperative actors in the environment.

城市空中出行 (UAM) 将导致大量飞行器需要应对不断变化的空域，以安全到达目的地。我们使用了一种高效基于马尔可夫决策过程 (MDP) 的轨迹规划器，来展示多代理分布式轨迹规划，能够在高密度自由飞行空域中安全避开合作与非合作的参与者。我们在城市环境中展示了该算法，其中多架飞行器在避开合作与非合作飞行器的同时，导航至指定的着陆点。我们研究了随着飞行器密度的增加，算法的性能，以展示可扩展性以及环境中的合作与非合作参与者的效果。

I. Introduction

I. 引言

Urban Air Mobility (UAM) is a concept for future air transportation in which partially or fully autonomous air vehicles transport cargo and passengers through dense urban environments. For early adopters this technology offers the promise of bypassing ground-based freeways and hours-long commutes. As the technology matures, it will connect urban centers with outlying towns extending the reach of metropolitan areas.

城市空中出行 (UAM) 是一个关于未来空中运输的概念，部分或完全自主的航空器在密集的城市环境中运输货物和乘客。对于早期采用者来说，这项技术提供了避开地面高速公路和长时间通勤的承诺。随着技术的成熟，它将连接城市中心与外围城镇，扩大大都市区的范围。

As we contemplate this future, an important question to consider is how will this air traffic be managed? Will a structured airspace be required to enforce some order on the aircraft similar to today's air traffic management system? Or can improvements in technology allow a more dynamic, less structured airspace design to be used?

当我们思考这个未来时，一个需要考虑的重要问题是这种空中交通将如何管理？是否需要建立一个有结构的空域来对飞机实施一些秩序，类似于今天的空中交通管理系统？还是技术的改进可以允许使用更加动态、结构较少的空域设计？

This paper offers a demonstration of an algorithm that can efficiently provide guidance to aircraft operating in high-density airspace containing many aircraft. The algorithm allows each aircraft to make its own decisions in a distributed manner using simple inputs as would be available from sensors such as radar, LIDAR or systems such as ADS-B. The computational burden is very light due to the algorithm's efficiency and is capable of running on embedded processors found in UAV and UAM aircraft.

本文提供了一个算法的演示，该算法能够高效地为在高密度空域中运行的飞机提供指导，该空域包含多架飞机。该算法允许每架飞机以分布式方式自主做出决策，使用如雷达、LIDAR 或 ADS-B 系统等传感器可提供的简单输入。由于算法的高效性，计算负担非常轻，能够在无人机和城市空中出行飞机中的嵌入式处理器上运行。

The algorithm works in a cooperative setting, meaning that all aircraft in the airspace are using the algorithm from this paper, and also works in a non-cooperative setting, meaning that one or more aircraft in the airspace are not performing any avoidance. A cooperative setting is the most desirable case as both aircraft can participate in avoidance to ensure separation. However, it requires active participation from every aircraft and is vulnerable to fault conditions or rogue actors. If an aircraft's collision avoidance system fails due to a fault condition or sensor failure, it will not behave as expected or may not perform any avoidance at all. We demonstrate the algorithm's performance in both cases.

该算法在合作环境中工作，即空域中的所有飞机都在使用本文中的算法，同时在非合作环境中也有效，即空域中的一架或多架飞机没有执行避障。合作环境是最理想的情况，因为两架飞机都可以参与避障以确保分离。然而，它需要每架飞机的积极参与，并且容易受到故障条件或流氓行为者的影响。如果飞机的防撞系统因故障条件或传感器故障而失效，它将无法按预期行事，或者可能根本不执行任何避障。我们在两种情况下展示了算法的性能。

We also test the algorithms ability to scale to higher densities, while acknowledging that there is some upper limit to the number of aircraft that can be supported by a given airspace due to physical constraints. We perform simulations to evaluate the algorithm under different densities while monitoring for Near Mid-Air Collisions (NMACs). And finally we provide timing measurements of the performance of the algorithm.

我们还测试了算法在更高密度下的扩展能力，同时承认由于物理限制，任何给定空域能够支持的飞机数量都存在上限。我们在不同密度下进行模拟以评估算法，同时监控近空中相撞 (NMACs) 的情况。最后，我们提供了算法性能的时间测量数据。

II. Related Work

II. 相关工作

NASA, Uber and Airbus have been exploring the use of vertical takeoff and landing (VTOL) aircraft for Urban Air Mobility (UAM) [1-5]. In general, the UAM concept calls for UAM aircraft taking off from small-scale airports known as vertiports where VTOL aircraft depart and arrive.

美国宇航局 (NASA)、优步 (Uber) 和空客 (Airbus) 一直在探索垂直起降 (VTOL) 飞机在城市空中出行 (UAM) 中的应用 [1-5]。一般来说，UAM 概念是指 UAM 飞机从被称为垂直机场的小型机场起飞，VTOL 飞机在这里起飞和降落。

An unstructured airspace approach known as "free flight" has been proposed to cope with the ongoing congestion of the current ATC system. It was shown in [6, 7] that free flight with airborne separation is able to handle a higher traffic density, and [8] found free flight can also bring fuel and time efficiency. In a free flight framework, each aircraft is responsible for separation assurance and conflict resolution. [9] shows that free flight is potentially feasible due to enabling technologies such as Global Positioning Systems (GPS), data link communications like Automatic Dependence Surveillance-Broadcast (ADS-B) [10], Traffic Alert and Collision Avoidance Systems (TCAS) [11], but would require powerful onboard computation.

已经提出了一种称为 "自由飞行" 的无结构空域方法，以应对当前空中交通管制 (ATC) 系统的持续拥堵。文献 [6, 7] 显示，带有空中分离的自由飞行能够处理更高的交通密度，而文献 [8] 发现自由飞行还能带来燃油和时间效率。在自由飞行框架中，每架飞机都负责分离保证和冲突解决。文献 [9] 显示，由于全球定位系统 (GPS)、类似自动依赖监视广播 (ADS-B) 的数据链通信 [10]、交通警报和避撞系统 (TCAS) [11] 等启用技术，自由飞行可能可行，但需要强大的机上计算能力。

In terms of algorithms used for trajectory planning and collision avoidance, there is extensive literature

on the topic. In centralized methods, a central supervising controller resolves conflicts between aircraft. The state of each aircraft, obstacles, and trajectory constraints as well as the state of the terminal area are observable to the controller via sensors, radar, etc. The central controller precomputes trajectories for all aircraft before flight, typically by formulating the problem in and optimal control framework and solving the problem with varying methods; examples are: semidefinite programming [12], nonlinear programming [13, 14], mixed integer linear programming [15-18], mixed integer quadratic programming [19], sequential convex programming [20, 21], second-order cone programming [22], evolutionary techniques [23, 24], and particle swarm optimization [25]. One common thread among centralized approaches is that in order to pursue a global optimum, they must consider each aircraft and obstacle in the space. This necessarily leads to scalability issues with large numbers of aircraft and obstacles. Also, as new aircraft enter the scene, centralized algorithms typically need to recompute part or all of the problem in order to arrive at a new global optimum.

在轨迹规划和避障算法方面，已有大量相关文献。在集中式方法中，一个中央监督控制器解决飞机之间的冲突。通过传感器、雷达等，控制器可以观察到每架飞机的状态、障碍物、轨迹约束以及终端区的状态。中央控制器通常在飞行前为所有飞机预计算轨迹，通常通过在最优控制框架中构建问题并使用不同的方法解决问题；例如: 半定规划 [12]，非线性规划 [13, 14]，混合整数线性规划 [15-18]，混合整数二次规划 [19]，序列凸规划 [20, 21]，二阶锥规划 [22]，进化技术 [23, 24] 和粒子群优化 [25]。集中式方法的一个共同点是，为了追求全局最优，它们必须考虑空间中的每架飞机和障碍物。这必然会在飞机和障碍物数量较多时导致可扩展性问题。此外，当新的飞机进入场景时，集中式算法通常需要重新计算问题的一部分或全部，以达成新的全局最优。

In contrast, decentralized methods scale better with respect to the number of aircraft and objects in the system, but typically cannot obtain globally optimal solutions. However, decentralized methods typically do not have a single point of failure, so may be considered more robust than a centralized approach [26]. Conflicts are resolved by each aircraft locally in decentralized approaches and the underlying method can be considered as cooperative or non-cooperative. Examples of cooperative methods using some form of communication are [27-30] which typically use the communication to make smaller optimal control problems that can be solved locally. Other approaches such as [31, 32] use time slices to compute collision free paths one agent at a time.

相比之下，分布式方法在系统中的飞机和对象数量方面具有更好的可扩展性，但通常无法获得全局最优解。然而，分布式方法通常没有单一的故障点，因此可能被认为比集中式方法更健壮 [26]。在分布式方法中，冲突由每架飞机本地解决，底层方法可以是合作式的或非合作式的。使用某种形式的通信进行合作的方法的例子有 [27-30]，这些方法通常使用通信来制造更小的最优控制问题，这些问题可以本地解决。其他方法如 [31, 32] 使用时间片段逐个代理计算无碰撞路径。

Computational geometry methods such as visibility graphs [33] and Voronoi diagrams [34] can also generate paths for aircraft, but typically do not take dynamic constraints into account. Within the robotics community several sampling based dynamics aware methods have been created: probabilistic roadmaps [35], RRT [36], RRT* [37], and RT-RRT* [38] (which may be applied in centralized and decentralized contexts.) These methods attempt to generate a connected graph or tree of collision-free states that are very near each other so that a path from a start to a goal position can be determined via a path through the graph or tree. The advantage of these methods is they can discover conflict-free paths through high dimensional spaces with many obstacles, but the graph/tree must either be pre-computed or in the case of RT-RRT* can be computed on-line by fixing the amount of computation allowed during any compute cycle. While none of these methods compute a truly optimal path with a finite number of samples, with enough samples the paths can be good enough for many problems and can further be smoothed after extraction from the graph/tree.

计算几何方法，如可见性图 [33] 和 Voronoi 图 [34]，也可以为飞机生成路径，但通常不考虑动态约束。在机器人学领域，已经创建了多种基于采样的动态感知方法: 概率路线图 [35]、RRT [36]、RRT* [37] 和 RT-RRT* [38](这些方法可能适用于集中式和分布式环境。) 这些方法试图生成一个无碰撞状态的连通图或树，这些状态彼此非常接近，以便可以通过图或树的路径确定从起始位置到目标位置的路径。这些方法的优势在于，它们能够在具有许多障碍的高维空间中发现无冲突路径，但图/树必须预先计算，或者在 RT-RRT* 的情况下，可以在任何计算周期内通过固定允许的计算量在线计算。虽然这些方法中没有一种能够用有限的样本计算出真正的最优路径，但足够的样本可以使路径对于许多问题来说足够好，并且在从图/树中提取后还可以进一步平滑路径。

Collision avoidance can be solved with Model Predictive Control [39, 40] with high computational costs, and potential fields [41, 42] which are fast but do not provide guarantees of collision avoidance. Machine learning methods have also been used [43-46]. The methods perform well but require a great deal of time to train beforehand. The Monte Carlo Tree Search method used in [47] solves the problem efficiently by using a fixed window of computation. Collision avoidance can also be solved geometrically

[48-51] and the computation time only grows linearly as the number of aircraft increases. DAIDALUS (Detect and Avoid Alerting Logic for Unmanned Systems) [52] is another geometric approach developed by NASA which provides heading and altitude guidance to avoid collisions to remote pilots. ACAS-X is a collision avoidance system also based off Markov Decision Processes that provides 1-on-1 collision avoidance [53].

避碰问题可以通过模型预测控制 [39, 40] 解决，这种方法计算成本高，还有势场方法 [41, 42] ，速度快但无法保证避碰。机器学习方法也已被应用 [43-46]。这些方法表现良好，但需要大量的时间进行预先训练。文献 [47] 中使用的蒙特卡洛树搜索方法通过使用固定的计算窗口有效地解决了问题。避碰问题还可以通过几何方法解决 [48-51]，并且计算时间仅随飞机数量的增加而线性增长。DAIDALUS(无人系统检测与避障警报逻辑)[52] 是 NASA 开发的另一种几何方法，它为远程飞行员提供航向和高度指导以避免碰撞。ACAS-X 是一种基于马尔可夫决策过程的避碰系统，它提供一对一的避碰 [53]。

Within the framework of the literature, the method in this paper phrases the problem as a Markov Decision Process (MDP). MDPs are known to be intractable for large problems, but a recently discovered method for efficiently computing them was described in [54]. We harness this implementation and demonstrate its efficient performance in a UAM context with multiple UAM vehicles navigating between multiple vertiports.

在文献框架内，本文将问题构建为马尔可夫决策过程 (MDP)。已知在大规模问题中 MDP 是不可解的，但近期发现的一种有效计算 MDP 的方法在文献 [54] 中有所描述。我们利用这一实现，并展示了在多个城市空中交通 (UAM) 车辆在多个垂直港口之间导航的 UAM 环境中其高效的性能。

III. Background

III. 背景

Markov Decision Processes (MDPs) are a framework for sequential decision making with broad applications to finance, robotics, operations research and many other domains [55]. MDPs are formulated as the tuple (s_t, a_t, r_t, t) where $s_t \in S$ is the state at a given time $t, a_t \in A$ is the action taken by the agent at time t as a result of the decision process, r_t is the reward received by the agent as a result of taking the action a_t from s_t and arriving at s_{t+1} , and $T(s_t, a, s_{t+1})$ is a transition function that describes the dynamics of the environment and capture the probability $p(s_{t+1} \mid s_t, a_t)$ of transitioning to a state s_{t+1} given the action a_t taken from state s_t .

马尔可夫决策过程 (MDPs) 是一种用于顺序决策的框架，广泛应用于金融、机器人学、运筹学以及许多其他领域 [55]。MDPs 被构建为一个元组 (s_t, a_t, r_t, t) ，其中 $s_t \in S$ 是在给定时间的状态，$t, a_t \in A$ 是代理在时间 t 由于决策过程而采取的行动，r_t 是代理因采取行动 a_t 从 s_t 到达 s_{t+1} 而获得的奖励，$T(s_t, a, s_{t+1})$ 是一个转换函数，描述环境的动态并捕捉给定状态 s_t 采取行动 a_t 后转移到状态 s_{t+1} 的概率 $p(s_{t+1} \mid s_t, a_t)$ 。

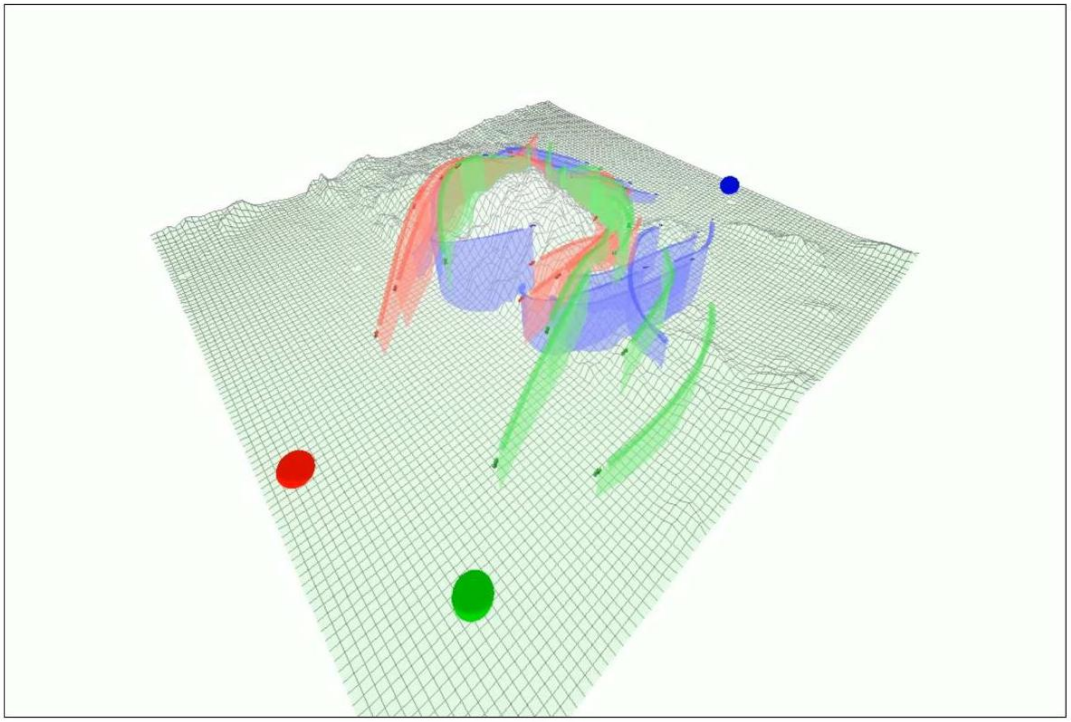

Fig. 1 Multiple aircraft with multiple goals. Colored spheres represent the goals. Airplanes and their history trails are colored according to their goal. Negative rewards are placed around terrain features so that UAVs avoid collision with terrain.

图 1 多架飞机具有多个目标。彩色球体代表目标。飞机及其历史轨迹根据其目标着色。在地面特征周围放置负奖励，以便无人机避免与地形碰撞。

A policy π can be defined that maps each state $s \in S$ to an action $a \in A$. From a given policy $\pi \in \Pi$ a value function $V^{\pi}(s)$ can be computed that computes the expected return that will be obtained within the environment by following the policy π .

可以定义一个策略 π ，它将每个状态 $s \in S$ 映射到一个行动 $a \in A$ 。从给定的策略 $\pi \in \Pi$ 可以计算出一个价值函数 $V^{\pi}(s)$ ，该函数计算在遵循策略 π 的情况下在环境中将获得的预期回报。

The solution of an MDP is termed the optimal policy π^* , which defines the optimal action $a^* \in A$ that can be taken from each state $s \in S$ to maximize the expected return. From this optimal policy π^* the optimal value function $V^*(s)$ can be computed which describes the maximum expected value that can be obtained from each state $s \in S$. And from the optimal value function $V^*(s)$, the optimal policy π^* can also easily be recovered.

MDP 的解被称为最优策略 π^* ，它定义了从每个状态 $s \in S$ 可以采取的最优行动 $a^* \in A$ ，以最大化预期回报。从最优策略 π^* 可以计算出最优价值函数 $V^*(s)$ ，该函数描述了从每个状态 $s \in S$ 可以获得的最大预期价值。并且从最优价值函数 $V^*(s)$ 也可以轻松恢复出最优策略 π^* 。

IV. Method

IV. 方法

We formulate the problem as a Markov Decision Process and use the algorithm described in [54] to efficiently solve the MDP. The algorithm in [54] was shown to efficiently perform collision avoidance while navigating to a goal in a 2D discretized state space. We extend [54] by moving to a 3D state space which is fully continuous and model the aircraft in 3D using a Dubin's aircraft model which is constrained to operate in a manner similar to a UAM air taxi in forward flight mode.

我们将问题构建为马尔可夫决策过程，并使用文献 [54] 中描述的算法高效地解决 MDP。文献 [54] 中的算法被证明在二维离散状态空间中导航至目标时能够高效地执行避障。我们通过将 [54] 扩展到完全连续的三维状态空间，并使用杜宾飞机模型在三维中对飞机进行建模，该模型被限制以类似于城市空中出租车前飞模式的方式运行。

A. Dynamic Model

字段: ## A. 动态模型

[56] proposes an extension of the Dubin's car model to aircraft using an optimal control framework relying on Pontraygin's Maximum Principle. [57] extends the model using a more realistic formulation from an aerospace engineering literature.

文献 [56] 提出了使用庞特里亚金最大原理的优化控制框架，将杜宾车模型扩展到飞机上。文献 [57] 使用来自航空航天工程文献的更现实公式扩展了该模型。

Table 1 Limits on aircraft performance to approximate an air taxi. Note that the model assumes an airspeed controller that maintains a commanded airspeed through the duration of flight. A pseudo-6DOF formulation is proposed in [58] and [59].

表 1 显示了近似空中出租车的飞机性能限制。请注意，该模型假设存在一个空速控制器，在飞行期间保持指定的空速。文献 [58] 和 [59] 提出了一个伪六自由度 (pseudo-6DOF) 公式。

$V_{\min}$ (Kts)	$V_{\max}$ (Kts)	$\psi_{\min}$ (deg/s)	$\psi_{\max}$ (deg/s)	$\alpha_{\min}$ (deg)	$\alpha_{\max}$ (deg)	$\phi_{\min}$ (deg)	$\phi_{\max}$ (deg)	$\gamma_{\min}$ (deg)	$\gamma_{\max}$ (deg)
47	133	-30	30	-5	20	-20	20	-20	20

$V_{\min}$ (Kts)	$V_{\max}$ (Kts)	$\psi_{\min}$ (deg/s)	$\psi_{\max}$ (deg/s)	$\alpha_{\min}$ (deg)	$\alpha_{\max}$ (deg)	$\phi_{\min}$ (deg)	$\phi_{\max}$ (deg)	$\gamma_{\min}$ (deg)	$\gamma_{\max}$ (deg)
47	133	-30	30	-5	20	-20	20	-20	20

The model used here is based off a pseudo-6DOF formulation, but uses performance limits that make the aircraft perform similar to a Dubin's aircraft with gentle climbs and bank angles suitable for a UAM passenger aircraft.

这里使用的模型基于伪六自由度公式，但是采用了性能限制，使得飞机的表现类似于杜宾飞机，具有适合城市空中出租车乘客飞机的温和爬升和横滚角度。

- n_x : Throttle acceleration directed out the nose of the aircraft in g ' s
- n_x : 指向飞机鼻部的油门加速度在 g 的方向上
- V : Airspeed in meters/second.
- V : 以米/秒为单位的空速。
- γ : Flight path angle in radians.
- γ : 以弧度表示的飞行路径角。
- x, y, z : position in NED coordinates in meters where altitude $h = -z$
- x, y, z : 在 NED 坐标系中的位置 (以米为单位)，其中高度 $h = -z$
- ϕ : Roll angle in radians
- ϕ : 以弧度表示的横滚角
- ψ : Horizontal azimuth angle in radians
- ψ : 水平方位角 (以弧度表示)
- α : Angle of attack in radians with respect to the flight path vector
- α : 与飞行路径向量相对应的攻角 (以弧度计)

The inputs to the model are: (1) the thrust n_x ,(2) the rate of change of angle of attack $\dot{\alpha}$ and (3) the rate of change of the roll angle ϕ .

模型的输入包括:(1) 推力 n_x ，(2) 攻角变化率 $\dot{\alpha}$ 以及 (3) 横滚角变化率 ϕ 。

The equations of motion for the aircraft are:

飞机的运动方程为:

$$\dot{V} = g\left[n_x \cos\alpha - \sin\gamma\right] \tag{1}$$

$$\dot{\gamma} = \frac{g}{V}\left[n_f \cos\phi - \cos\gamma\right] \tag{2}$$

$$\dot{\psi} = g\left[\frac{n_f \sin\phi}{V\cos\gamma}\right] \tag{3}$$

where the acceleration exerted out the top of the aircraft n_f in g s is defined as:
其中，飞机顶部施加的加速度 n_f 在 g s 中定义为:

$$n_f = n_x \sin\alpha + L, \tag{4}$$

with a lift acceleration of $L = 0.9$. Here,1"g" is a unit of acceleration equivalent to 9.8 $\mathrm{m/s^2}$.L was chosen to provide some amount of lift while in flight to counteract gravity and provide a stable flight condition with a low positive α angle of attack in the pseudo-6dof model.
其中升力加速度为 $L = 0.9$ 。在此，1"g" 是一个加速度单位，相当于 9.8 $\mathrm{m/s^2}$.L 被选为在飞行中提供一定量的升力以抵消重力，并在伪 6 自由度模型中提供一个具有低正攻角的稳定飞行条件。
The kinematic equations are:
动力学方程为:

$$\dot{x} = V\cos\gamma\cos\psi \tag{5}$$

$$\dot{y} = V\cos\gamma\sin\psi \tag{6}$$

$$\dot{z} = V\sin\gamma \tag{7}$$

While this model is not aerodynamically comprehensive, it is sufficient to describe aircraft motion suitable for examining the algorithm behavior without loss of generality. The algorithm can integrate with any aircraft dynamics model that allows the trajectory to be computed forward in time.
虽然这个模型在空气动力学上不是全面的，但它足以描述飞机运动，适合检查算法行为，而不失去一般性。该算法可以与任何允许轨迹在时间上向前计算的飞机动力学模型集成。
Performance limits are used to make the aircraft dynamics perform like a Dubin's aircraft similar to what may be expected from a UAM air taxi, as shown in Table 1.
性能限制用于使飞机动力学表现得像杜宾飞机，类似于从 UAM 空中出租车中可能预期的表现，如表 1 所示。

1. Trajectory Forward Projection

1. 轨迹前向投影

At each time step, each aircraft has a set of possible actions it can take and each of those actions will lead to different future states that the aircraft may reach. When we use the aircraft dynamics to compute the result of taking an action for a fixed amount of time, this is known as forward projection. When successive timesteps over a time period t are used to show the future trajectory resulting from an action, this is known as trajectory forward projection with window t .
在每个时间步，每架飞机都有一组可能的动作可以采取，而这些动作中的每一个都将导致飞机可能达到的不同未来状态。当我们使用飞机动力学来计算固定时间内采取一个动作的结果时，这被称为前向投影。当在一段时间 t 内连续的时间步长用来展示由一个动作产生的未来轨迹时，这被称为具有窗口 t 的轨迹前向投影。
If the actions $a \in A$ were fully continuous, we could describe the result of the forward projection of all actions as the reachability set, $R \subset S$ from our initial state $s \in S$. However, for computational efficiency we use a discrete set of actions $\widehat{a} \in A$. When we use forward projection on this discrete set of actions, we obtain an approximation of the reachabililty set $\widehat{R} \subset R$. If the discrete actions and timesteps are chosen wisely, they can however be a reasonable and useful approximation of the reachability set.
如果行动 $a \in A$ 完全连续，我们可以将所有行动的前向投影的结果描述为可达集 $R \subset S$ ，从我们的初始状态 $s \in S$ 开始。然而，为了计算效率，我们使用一组离散的行动 $\widehat{a} \in A$ 。当我们在这一组离散的行动上使用前向投影时，我们得到可达集的近似值 $\widehat{R} \subset R$ 。如果离散行动和时间步长选择得当，它们可以成为可达集的合理且有用的近似值。

For this problem, we use a discrete time step of 0.1 seconds and a window of 3.0 seconds. For the trajectory forward projection, we keep the action fixed over the window and generate one trajectory for each action $\widehat{a}$. Once the forward projection is completed, the resulting states from the trajectories form an approximation of the reachability set $\widehat{R}$. The value of each of these states $s \in \widehat{R}$ are computed within the solution of the MDP, which represents the points within the value function that are reachable within the trajectory forward projection window. If we compute the state with maximum value and then determine the action which results in that state, we then know the most valuable action to take from our current state. This action is then applied for the simulation' s0.1second time step resulting in a 10 Hz rate action selection.

对于这个问题，我们使用 0.1 秒的离散时间步长和 3.0 秒的时间窗口。对于轨迹前向投影，我们在窗口内保持行动不变，并为每个行动 $\widehat{a}$ 生成一条轨迹。一旦前向投影完成，轨迹产生的结果状态形成了可达集的近似值 $\widehat{R}$ 。这些状态的值 $s \in \widehat{R}$ 是在 MDP 的解中计算的，这代表了在轨迹前向投影窗口内可达的价值函数中的点。如果我们计算具有最大值的状态，然后确定导致该状态的动作，我们就知道了从当前状态采取的最有价值的行动。然后，这个行动被应用于模拟的 0.1 秒时间步长，从而产生 10 Hz 速率行动选择。

Thus the agent follows the optimal policy of the MDP at each time step by determining which future reachable state is most valuable, and then takes the action in the next time step that will lead it towards that state.

因此，代理在每一个时间步遵循 MDP 的最优策略，通过确定哪个未来的可达状态最有价值，然后在下一个时间步长采取将引导它朝向该状态的行动。

B.MDP Formulation

B. MDP 公式化

1. State Space

1. 状态空间

We define the environment where the aircraft operates within a 25 km by 25 km by 25 km volume which is treated as a continuous state space. The state includes all the information each aircraft needs for its decision making: the full aircraft state, the position and velocity of every other aircraft. Each aircraft is aware of its own state produced by the aircraft dynamics model. For each aircraft, the state is formed by concatenating the following:

我们定义了飞机在其中运行的環境为一个 25 km by 25 km by 25 km 的体积，该体积被视为一个连续的状态空间。状态包括每架飞机进行决策所需的所有信息: 完整的飞机状态，其他每架飞机的位置和速度。每架飞机都知道由飞机动态模型产生的自身状态。对于每架飞机，状态是由以下内容连接而成的:

- ζ the aircraft state: position x, y, z , the heading angle ψ , the flight path angle γ , and the speed V .
- ζ 飞机状态: 位置 x, y, z ， 航向角 ψ ， 飞行路径角 γ 和速度 V 。
- for each other aircraft $f_j, \forall j \in J$: the position $f_{j,x}, f_{j,y}, f_{j,z}$ and velocity $f_{j,v_x}, f_{j,v_y}, f_{j,v_z}$,
- 对于其他每架飞机 $f_j, \forall j \in J$: 位置 $f_{j,x}, f_{j,y}, f_{j,z}$ 和速度 $f_{j,v_x}, f_{j,v_y}, f_{j,v_z}$,

$$s_o = [\zeta, f_1, \cdots, f_j] \tag{8}$$

where j represents the number of other aircraft.

其中 j 表示其他飞机的数量。

2. Action Space

2. 动作空间

Inputs to the dynamics model are (1) the thrust n_x, (2) the rate of change of angle of attack $\dot{\alpha}$ and (3) the rate of change of the roll angle $\dot{\phi}$.

动力学模型的输入包括 (1) 推力 n_x,(2) ，(2) 攻角变化率 $\dot{\alpha}$ ，以及 (3) 横滚角变化率 $\dot{\phi}$ 。

The action space is a discrete set of actions that are distributed logarithmicly through the range of each input. The logarithmic distribution is designed to have a fine granularity of inputs near the zero point of the input and a coarse granularity of inputs near the extreme ends of the input range. The philosophy here is that more control is needed when trying to make small corrections to correct for small perturbations or refinements of the trajectory, and that large control inputs are required for gross course corrections.

动作空间是一组分布在对数范围内的每个输入的离散动作。对数分布设计为在输入零点附近具有精细的输入粒度，而在输入范围的两端附近具有粗糙的输入粒度。这里的理念是，在尝试对小的扰动或轨迹精炼进行小的校正时需要更多的控制，而对于粗略的航向校正则需要大的控制输入。

The action space is then defined as follows for both pitch inputs and roll inputs: 15 discrete inputs, 7 of which are positive, 7 of which are negative, and the value 0 . The 7 positive and 7 negative inputs are mirror values of each other and are based off a linear spacing of coefficients κ from .0001 to .13:

动作空间对于俯仰输入和横滚输入都定义为以下内容:15 个离散输入，其中 7 个为正，7 个为负，以及值 0。7 个正输入和 7 个负输入是彼此的镜像值，基于从 0.0001 到 0.13 的系数 κ 的线性间隔:

$$\kappa = \{.0001, .02175, .0434, .06505, .0867, .10835, .13\} \tag{9}$$

The logarithmically spaced values in radians are then computed as follows:

对数间隔的弧度值然后按以下方式计算:

$$k_r = 10^{\kappa} - 1 \tag{10}$$

$$= \{0.0003, 0.051, 0.105, 0.162, 0.221, 0.283, 0.349\} \tag{11}$$

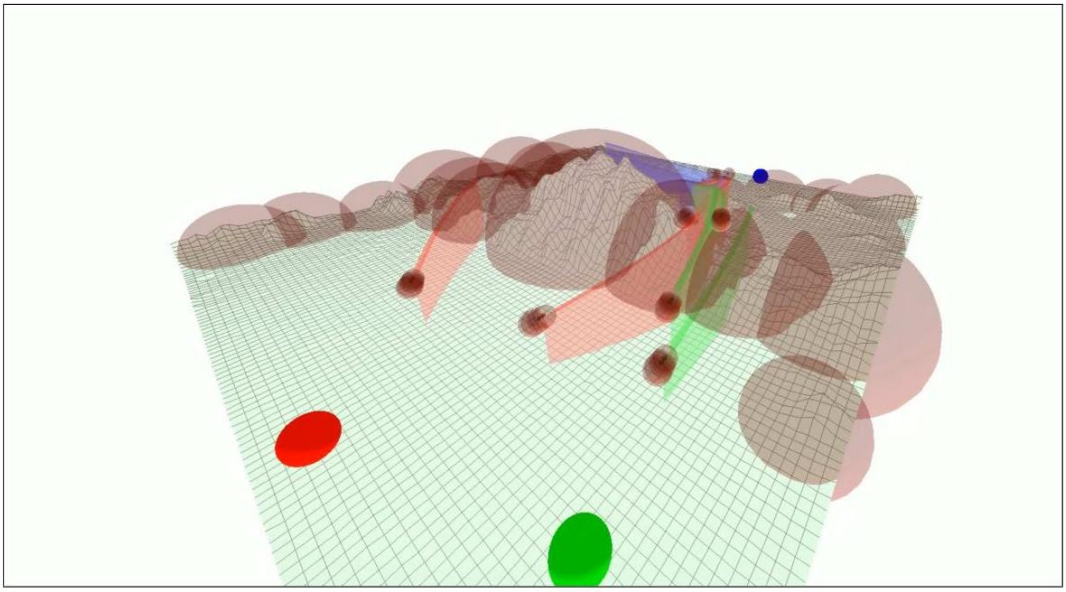

Fig. 2 Visualization of negative rewards. Each aircraft has negative rewards positioned in front of it for collision avoidance with other aircraft. Terrain features are covered by negative rewards which were manually placed and sized for terrain avoidance. A similar technique can be used for weather.

图 2 负奖励的可视化。每架飞机前方为其设置了负奖励，以避免与其他飞机碰撞。地形特征被负奖励覆盖，这些负奖励是手动放置和调整大小的，用于避开地形。类似的技术也可以用于天气。

Or in degrees:

或者用度数表示:

$$k_d = \{0.013, 2.943, 6.022, 9.258, 12.660, 16.236, 19.994\} \tag{12}$$

Making the final distribution in degrees:

将最终分布转换为度数:

$$K_d = \{-k_d, 0, k_d\} \tag{13}$$

$$= \{-19.99, -16.24, -12.66, -9.26, -6.02, -2.94, -0.01, 0, 0.01, 2.94, 6.02, 9.26, 12.66, 16.24, 19.99\}, \quad (14)$$

with the pitch and roll inputs both equal to these logarithmic distributions:
俯仰角和滚转角的输入都等于这些对数分布:

$$\dot{\alpha} = K_d \quad (15)$$

$$\dot{\phi} = K_d \quad (16)$$

(17)
n_x is a simple linear distribution: $\{-2, -1, 0, 1, 2, 3, 4\}$.
n_x 是一个简单的线性分布: $\{-2, -1, 0, 1, 2, 3, 4\}$ 。
The joint action space is then:
然后联合动作空间为:

$$A = \left\{\dot{\alpha}, \dot{\phi}, n_x\right\} \quad (18)$$

3. Reward Function

3. 奖励函数

The primary mechanism to control the behavior of an agent in a Markov Decision Process (MDP) is through the Reward Function. By providing positive and negative rewards to the agent, it is able to determine which actions lead to positive reward and the solution of an MDP maximizes the expectation of future reward.

在马尔可夫决策过程 (MDP) 中控制智能体行为的主要机制是通过奖励函数。通过给智能体提供正负奖励，它能够确定哪些动作导致正奖励，MDP 的解决方案最大化了未来奖励的期望。

Following the approach used in [54], we will treat each negative reward as a "risk well", which is a region of negative reward (i.e., a penalty) which is more intense at the center and decays outward until a fixed radius is reached, where after no penalty is applied.

按照文献 [54] 中使用的方法，我们将每个负奖励视为一个 "风险井"，即负奖励 (即惩罚) 区域，在中心更为强烈，并向外衰减，直到达到一个固定半径，在此之后不再应用惩罚。

We present our reward function in terms of the behaviors we wish to obtain in Table 2. In this table, $\widehat{p}$ represents the current position of an aircraft and $\widehat{v}$ represents that aircraft' s current linear velocity. In some cases we project the aircraft' s position forward in time with an expression $\widehat{p} + \widehat{v}t$ and then define a range of time as in $\forall t \in \{0, 1, 2\}$ to

我们在表 2 中以我们希望获得的行为来呈现我们的奖励函数。在表中，$\widehat{p}$ 表示飞机的当前位置，$\widehat{v}$ 表示该飞机的当前线性速度。在某些情况下，我们使用表达式 $\widehat{p} + \widehat{v}t$ 将飞机的位置在时间上向前投影，然后定义一个时间范围，如 $\forall t \in \{0, 1, 2\}$ 以。

Table 2 Rewards created for each aircraft
表 2 为每架飞机创建的奖励
For each other aircraft:
对于每架其他飞机:

Magnitude	Decay factor	Location	Radius	Timesteps	Comment
–1000	.97	$\widehat{p} + \widehat{v}t$	300 + 10t	$\forall t \in \{-5, 0, 5, 10, 15\}$	Collision avoidance, 5 rewards

幅度	衰减因子	位置	半径	时间步	备注
–1000	.97	$\widehat{p} + \widehat{v}t$	300 + 10t	$\forall t \in \{-5, 0, 5, 10, 15\}$	避碰，5 奖励

For each terrain feature:
对于每个地形特征:

Magnitude	Decay factor	Location	Radius	Timesteps	Comment
-1000	.99	manually placed	manually selected	N/A	Terrain avoidance

幅值	衰减因子	位置	半径	时间步	备注
-1000	.99	手动放置	手动选择	N/A	地形避障

For aircraft's goal:
对于飞机的目标:

Magnitude	Decay factor	Location	Radius	Timesteps	Comment
200	.999	manually placed	∞	N/A	Vertiport attraction

数量级	衰减因子	位置	半径	时间步	评论
200	.999	手动放置	∞	N/A	竖直机场吸引力

indicate that we create a reward at the location of the aircraft at each timestep in the future indicated by the range of t . See Figure 2 for a visualization of the rewards.
表明我们在未来每个时间步在飞机所在的位置创建一个奖励，该位置由 t 范围内的范围指示。见图 2 以可视化奖励。

All aircraft also receive a penalty below a certain altitude which prevents the aircraft from plummeting into the terrain. We define a terrain "floor" h_{floor} which is the 5th percentile of all terrain vertices. The NASA SRTM terrain data [60] contains some locations of bad data, so the 5th percentile is used to eliminate any outliers when determining a good average minimum height. For any state with an altitude of h from the hard deck up to an altitude of $h_{\text{penalty}} = h_{\text{floor}} + 200$, a penalty is applied $r_{\text{penalty}} = -(10000 - h)$ which is a very strong negative reward that will override any other positive rewards in the game.
所有飞机在低于一定高度时也会受到惩罚，以防止飞机坠入地形。我们定义了一个地形 "地板" h_{floor} ，即所有地形顶点的第 5 百分位数。NASA SRTM 地形数据 [60] 包含一些错误数据的位置，因此使用第 5 百分位数来确定足够平均最小高度时消除任何异常值。对于任何从硬甲板到 $h_{\text{penalty}} = h_{\text{floor}} + 200$ 高度的 h 高度的状态，将应用 $r_{\text{penalty}} = -(10000 - h)$ 的惩罚，这是一种非常强烈的负奖励，将覆盖游戏中的任何其他正奖励。

To avoid collision with terrain, for this paper negative rewards were manually placed in the terrain such that they covered the terrain features with a hemisphere of negative reward, providing a repulsive force that prevents collision. (In future research, this could be made more automatic for true terrain or building avoidance.)
为了避免与地形碰撞，本文中在地面手动放置了负奖励，使其覆盖地形特征，形成一个带有负奖励的半球，提供一种排斥力以防止碰撞。(在未来的研究中，这可以更加自动化，以实现真正的地形或建筑物避障。)

C. Algorithm

C. 算法

The algorithm in this paper is based off the algorithm in [54], which defines the concept of a peak in the value function that results from a reward in the space. Peaks are located in the state space at the locations of positive rewards. The algorithm takes as input a number of pre-allocated data structures that described the peaks:
本文的算法基于 [54] 中的算法，该算法定义了由空间中的奖励产生的价值函数中的峰的概念。峰值位于状态空间中正奖励的位置。算法将一些预分配的数据结构作为输入，这些数据结构描述了峰值:

- peakLocations: Locations of each peak within the state space.
- peakLocations: 状态空间中每个峰的位置。
- propLimits: A limit on how far the value function of each peak should be allowed to propagate. For our positive rewards in the state space, this is ∞ . For negative rewards, this is a fixed radius that represents the distance at which another aircraft poses no risk.
- propLimits: 限制每个峰值的价值函数传播多远。对于我们状态空间中的正奖励，这是 ∞ 。对于负奖励，这是一个固定半径，代表另一架飞机不构成风险的距离。
- propLimits: A limit on how far the value function of each peak should be allowed to propagate. For our positive rewards in the state space, this is ∞ . For negative rewards, this is a fixed radius that represents the distance at which another aircraft or terrain feature poses no risk.

- 属性限制: 限制每个峰值的价值函数传播的距离。对于状态空间中的正奖励，这是 ∞ 。对于负奖励，这是一个固定半径，表示另一架飞机或地形特征不构成风险的距离。

- discountFactors: A parameter for each reward which allows the reward to decay outward as distance from the center increases.

- 折扣因子: 每个奖励的参数，允许奖励随着距离中心的增加而衰减。

- rwdMagnitudes: The magnitude of each of the rewards.

- 奖励量级: 每个奖励的大小。

- states: The states at which the value function should be computed for this MDP. This allows flexibility of computing only neighboring states needed for following the optimal policy, or large batches for visualizing the value function for debug.

- 状态: 对于这个 MDP，应当计算价值函数的状态。这允许仅计算跟随最优策略所需的邻近状态，或者计算大批次的状态以可视化价值函数进行调试。

We reimplement the algorithm proposed in [54] to vectorize the algorithm for more efficient operation. A vectorized algorithm is one which is designed to perform identical computations at each step over a vector of data inputs and is often accelerated by SIMD (Same Instruction Multiple Data) units available in modern CPUs. Libraries are available that assist in mapping code to these accelerators. In our case, within python a library known as Numpy provides this capability. Note that for true real-time embedded performance, the code would be ported to a language such as C or C++, but this is a reasonable approximation as Numpy relies on C and C++ libraries to perform its acceleration.

我们重新实现了文献 [54] 中提出的算法，以向量化算法来提高效率。向量化算法是一种在每一步对数据输入的向量执行相同计算的算法，并且通常由现代 CPU 中可用的 SIMD(单指令多数据) 单元加速。有一些库可以帮助将代码映射到这些加速器上。在我们的案例中，Python 中有一个名为 Numpy 的库提供了这种能力。需要注意的是，为了实现真正的实时嵌入式性能，代码将被迁移到如 C 或 C++ 这样的语言，但这是一个合理的近似，因为 Numpy 依赖于 C 和 C++ 库来执行其加速。

Algorithm 1 Distributed UAM using FastMDP algorithm 1: procedure DistributEDUAM(aircraftState, worldState)

算法 1 分布式 UAM 使用快速 MDP 算法 1: 过程 DistributEDUAM(飞机状态, 世界状态)

$\mathbf{S}_0 \leftarrow$ randomized initial aircraft states

$\mathbf{S}_0 \leftarrow$ 随机初始化飞机状态

$\mathbf{A} \leftarrow$ aircraft actions (precomputed)

$\mathbf{A} \leftarrow$ 飞机动作 (预计算)

$\mathbf{L} \leftarrow$ aircraft limits (precomputed)

$\mathbf{L} \leftarrow$ 飞机限制 (预计算)

$\mathbf{S}_{t+1} \leftarrow$ allocated space

$\mathbf{S}_{t+1} \leftarrow$ 分配的空间

while aircraft remain do

当仍有飞机存在时

for each aircraft do

对每架飞机执行

$s_t \leftarrow \mathbf{S}_t$ [aircraft]

$s_t \leftarrow \mathbf{S}_t$ [aircraft]

// Build peaks per Table 2

// 根据表 2 构建峰值

$\mathbf{P}^+ \leftarrow$ build pos rewards

$\mathbf{P}^+ \leftarrow$ 构建正激励

$\mathbf{P}^- \leftarrow$ build neg rewards in Standard Positive Form

$\mathbf{P}^- \leftarrow$ 以标准正形式构建负激励

$\mathbf{P}^* \leftarrow$ build neg rewards for terrain in Standard Positive Form

$\mathbf{P}^* \leftarrow$ 以标准正形式为地形构建负激励

// Perform forward projection per Section IV.A. 1

// 按照第 IV.A.1 节进行前向投影

$_1 \leftarrow$ fwdProject $(s_t, \mathbf{A}, \mathbf{L}, 0.1 \text{ s})$

$_{10} \leftarrow$ fwdProject $(s_t, \mathbf{A}, \mathbf{L}, 1.0 \text{ s})$

// Compute the value at each reachable state
// 计算每个可达状态的价值
$\mathbf{V}^* \leftarrow$ allocate space for each reachable state
$\mathbf{V}^* \leftarrow$ 为每个可达状态分配空间
for $s_j \in \Delta_{10}$ do
对 $s_j \in \Delta_{10}$ 执行
// First for positive peaks
// 首先对正峰值
for $p_i \in \mathbf{P}^+$ do
对于 $p_i \in \mathbf{P}^+$ 执行
$d_p \leftarrow \left\| s_j - \text{location}\left(p_i\right) \right\|_2$ - distance
$d_p \leftarrow \left\| s_j - \text{location}\left(p_i\right) \right\|_2$ - 距离
$r_p \leftarrow \text{reward}\left(p_i\right)$
$\gamma_p \leftarrow \text{discount}\left(p_i\right)$
$\mathbf{V}^+\left(p_i\right) \leftarrow \left|r_p\right| \cdot \gamma_p^{d_p}$
end for
结束循环
$V_{\max}^+ \leftarrow \max_{p_i} \mathbf{V}^+$
// Next for negative peaks (in Standard Positive Form) including terrain
// 接下来处理负峰值 (以标准正形式) 包括地形
28: for $n_i \in \{\mathbf{P}^-, \mathbf{P}^*\}$ do
28: 对于 $n_i \in \{\mathbf{P}^-, \mathbf{P}^*\}$ 执行
29: $d_n \leftarrow \left\| s_j - \text{location}\left(n_i\right) \right\|_2$ D distance
29: $d_n \leftarrow \left\| s_j - \text{location}\left(n_i\right) \right\|_2$ D 距离
30: $\rho_n \leftarrow \text{negDis}\, t_i < \text{radius}\left(n_i\right)$ ▷ within radius
30: $\rho_n \leftarrow \text{negDis}\, t_i < \text{radius}\left(n_i\right)$ ▷ 在半径内
31: $r_n \leftarrow \text{reward}\left(n_i\right)$
32: $\gamma_n \leftarrow \text{discount}\left(n_i\right)$
$\mathbf{V}^-\left(p_i\right) \leftarrow \text{int}\left(\rho_n\right) \cdot \left|r_n\right| \cdot \gamma_n^{d_n}$
end for
结束循环
$V_{\max}^- \leftarrow \max_{p_i} \mathbf{V}^-$
// Hard deck penalty
// 硬甲板惩罚
if altitude $\left(s_t\right) <$ penaltyAlt then
如果高度 $\left(s_t\right) <$ 大于 penaltyAlt 则
$V_{\text{deck}} \leftarrow 1000 -$ altitude $\left(s_t\right)$
$V_{\text{deck}} \leftarrow 1000 -$ 高度 $\left(s_t\right)$
else
否则
$V_{\text{deck}} \leftarrow 0$
end if
结束 if
$\mathbf{V}^*\left[s_t\right] \leftarrow V_{\max}^+ - V_{\max}^- - V_{\text{deck}}$
end for
结束 for
// Identify the most valuable action
// 确定最有价值的动作
$i_{\max} \leftarrow \arg\max_s \left(\mathbf{V}^*\right)$
// For illustration, the corresponding value
// 作为示例，相应的价值
maxValue $\leftarrow \mathbf{V}^*\left[i_{\max}\right]$
maxValue $\leftarrow \mathbf{V}^*\left[i_{\max}\right]$
// And the next state when taking the action
// 并且在采取动作后的下一个状态
$s_{t+1} \leftarrow {}_{\mathbf{1}}\left[i_{\max}\right]$
$\mathbf{S}_{t+1}$ [aircraft] $\leftarrow s_{t+1}$

end for
结束 for
// Now that all aircraft have selected an action, apply it
// 现在所有飞机都已选择了一个动作，应用它
$\mathbf{S} \leftarrow \mathbf{S}_{t+1}$
end while
结束 while
end procedure
结束过程

V. Experiment Setup

V. 实验设置

We demonstrate this trajectory planner in a 3D aircraft simulation showing a view of the aircraft, the goals, and the terrain as shown in Figure 1. The simulation covers a 25 km by 25 km by 25 km volume which contains a configurable number of aircraft and goals. The terrain is derived from NASA SRTM radar data [60] in the Lake Tahoe, California area. The height has been exaggerated by a factor of 6 to represent a UAM operating area with terrain features which must be avoided with vertical maneuvering. While in this case the terrain features correspond to mountains, they could instead represent skyscrapers or population centers. In the simulations and videos, this forces the simulated UAM aircraft into more conflicts than would be observed with a flat environment. The location of the goals were manually selected to lie within flat, relatively featureless areas of the terrain and represent vertiports.

我们在 3D 飞机模拟中展示了这个轨迹规划器，显示了飞机视图、目标和地形，如图 1 所示。模拟覆盖了一个 25 km 乘以 25 km 乘以 25 km 的体积，其中包含可配置数量的飞机和目标。地形来源于 NASA SRTM 雷达数据 [60]，位于加利福尼亚州塔霍湖地区。高度被放大了 6 倍，以表示具有地形特征的 UAM(城市空中交通) 运行区域，这些地形特征必须通过垂直机动来避开。在这种情况下，地形特征对应于山脉，但它们也可以代表摩天大楼或人口中心。在模拟和视频中，这迫使模拟的 UAM 飞机比平坦环境下的冲突更多。目标的位置是手动选择的，位于地形中相对平坦且无特征的区域内，代表垂直机场。

As described above, all aircraft are driven by the algorithm and are assigned a goal. The color of the aircraft corresponds to the color of their goal. At each time step (0.1 seconds), the simulation updates the state for each aircraft. Each aircraft creates and solves its own MDP using the highly efficient algorithm presented in this paper, and then uses the solution of the MDP to select its next action. The actions of all aircraft are performed simultaneously in the simulation at the beginning of the next time step, simulation then advances by one time step (.1 seconds), whereupon the process repeats. Note here that because the environment changes, a new MDP is calculated at each time step, which is made possible by the performance of the algorithm in [54].

如上所述，所有飞机都由算法驱动并分配了一个目标。飞机的颜色与其目标颜色相对应。在每一个时间步 (0.1 秒)，模拟更新每架飞机的状态。每架飞机使用本文中提出的高效算法创建并解决自己的 MDP(马尔可夫决策过程)，然后使用 MDP 的解决方案选择其下一个动作。所有飞机的动作在模拟的下一个时间步开始时同时执行，模拟随后前进一个时间步 (0.1 秒)，然后过程重复。请注意，因为环境发生变化，所以在每个时间步都会计算一个新的 MDP，这是由 [54] 中算法的性能实现的。

We measure performance in two ways. First, we characterize the time it takes for the algorithm to compute the solution of the MDP. Second, we examine the algorithms ability to reach the goal while avoiding collisions. We define a Near Mid-Air Collision (NMAC) as an aircraft coming within 100 meters of another aircraft during flight. As we vary experimental parameters, we track the number of goals obtained and the number of NMACs that occur over the simulation duration.

我们用两种方式衡量性能。首先，我们描述算法计算 MDP 解决方案所需的时间。其次，我们检查算法在避免碰撞的同时达到目标的能力。我们将“几乎空中相撞”(NMAC) 定义为飞行中一架飞机接近另一架飞机 100 米以内的情况。当我们改变实验参数时，我们记录在模拟期间获得的目标数量和发生的 NMAC 数量。

VI. Results

VI. 结果

Results of simulation comparing cooperative and non-cooperative experiments are shown in Figure 3. In these simulations, there are three color-coded teams of aircraft (red, green, blue) with each team's color according to which vertiport it is navigating to (rendered as a sphere with matching color) as shown in Figure 1. The number of aircraft in the simulation $\{3, 15, 30, \ldots\}$ results from the number of aircraft per team $\{1, 5, 10, \ldots\}$.

图 3 显示了合作与非合作实验的模拟结果。在这些模拟中，有三组颜色编码的飞机 (红色、绿色、蓝色)，每组飞机的颜色对应于它正在导航到的垂直港口的颜色 (如图 1 所示，呈现为与颜色匹配的球体)。模拟中的飞机数量 $\{3, 15, 30, \ldots\}$ 来自每个队的飞机数量 $\{1, 5, 10, \ldots\}$ 。

Over the cooperative simulation runs, no collisions between any aircraft occurred. For non-cooperative simulations, one-third of the aircraft were set to be non-cooperative intruders. These intruders are still run by the algorithm, but they were not presented with any rewards related to any other aircraft (including other intruders). While they can still successfully navigate to the goal while avoiding terrain, they are blind to other aircraft. Thus, any collisions or NMACs between two intruders are ignored during the non-cooperative simulation. As expected, NMACs increased with an increase in the number of blind intruders. However, despite the large numbers of intruders the remaining two-thirds of the aircraft (red and blue aircraft) managed to avoid collisions with themselves or the intruders remakably well.

在合作模拟运行过程中，任何飞机之间都没有发生碰撞。对于非合作模拟，三分之一飞机被设置为非合作入侵者。这些入侵者仍然由算法控制，但它们没有获得与任何其他飞机 (包括其他入侵者) 相关的任何奖励。虽然它们仍然可以成功地导航到目标并避开地形，但它们对其他飞机视而不见。因此，在非合作模拟中，两个入侵者之间的任何碰撞或 NMAC 都被忽略。如预期的那样，随着盲目入侵者数量的增加，NMAC 的数量也增加了。然而，尽管入侵者数量众多，剩余的三分之二飞机 (红色和蓝色飞机) 仍然非常出色地避开了与自身或入侵者的碰撞。

Timing measurements were performed on a laptop class PC with an Intel i9-8950HK CPU running at 2.90 GHz with 32 GB RAM. For the tests the number of aircraft were increased and the time to execute the algorithm per aircraft is reported. Table 3 shows the results showing a linear increase in performance (meaning $O(n)$ performance) with increased number of aircraft. Note that our target execution time for a 10 Hz frame rate(100msperiod)is exceeded at 90 aircraft (30 aircraft per team) indicating that further optimization is needed to scale to larger numbers of aircraft. An alternate version of the algorithm is in development and early results show that it will dramatically improve the performance beyond what is reported here.

在一台配备 Intel i9-8950HK CPU、运行频率为 2.90 GHz、拥有 32 GB RAM 的笔记本电脑上进行了时间测量。测试中增加了飞机数量，并报告了每架飞机执行算法的时间。表 3 显示了结果，表明随着飞机数量的增加，性能 (即 $O(n)$ 性能) 线性增长。注意，我们的目标执行时间在 10 Hz 帧率 (100ms 周期) 下，在 90 架飞机 (每组 30 架飞机) 时被超出，这表明需要进一步优化以扩展到更大数量的飞机。算法的另一个版本正在开发中，早期结果显示它将显著提高性能，超出此处报告的水平。

We show sample videos of the simulation to provide intuition on the complexity of the task and performance of the algorithm. Two videos are provided that show 30 aircraft, one of which shows the negative rewards for terrain in order to provide insight into how the algorithm functions. A third video is provided with 150 aircraft to provide insight into the density of the airspace. Note that in all three videos, the aircraft are rendered with a wingspan of 200 meters for improved visibility, though in simulation the actual wingspan is 5 meters.

我们展示了模拟的示例视频，以提供对任务复杂性和算法性能的直观理解。提供了两个视频，展示了 30 架飞机，其中一个视频显示了地形上的负奖励，以提供对算法如何工作的洞察。还提供了一个包含 150 架飞机的视频，以提供对空域密度的洞察。注意，在所有三个视频中，飞机都被渲染为翼展 200 米以提高可见性，尽管在模拟中实际的翼展是 5 米。

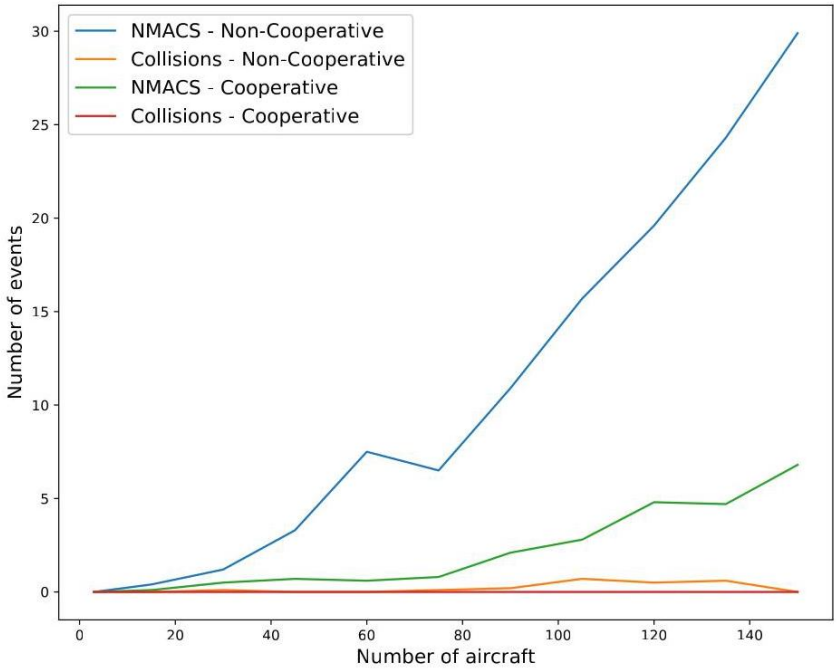

Fig. 3 Comparing results of cooperative flight versus non-cooperative flight. As expected, NMACS and Collisions increase when non-cooperative agents are present in the airspace. Despite the presence of multiple intruders, the number of collisions has been kept to a very low value. Values shown here are averaged over 10 Monte Carlo runs with random initial positions of aircraft. In the uncooperative measurement, 1/3 of the aircraft are uncooperative and do not attempt to avoid any other aircraft, acting as blind intruders. Any collisions or NMACS between any two intruders are ignored as they may blindly fly into each other.

图 3 比较了合作飞行与非合作飞行的结果。如预期，当非合作代理出现在空域中时，NMACS(最小间隔违反次数) 和碰撞次数增加。尽管存在多个入侵者，碰撞数量仍被保持在非常低的水平。此处显示的值是 10 次蒙特卡洛运行的平均值，飞机的初始位置随机。在非合作测量中，1/3 的飞机是非合作的，并不尝试避开其他飞机，充当盲目的入侵者。任何两个入侵者之间的碰撞或 NMACS 都被忽略，因为它们可能会盲目地相互飞向对方。

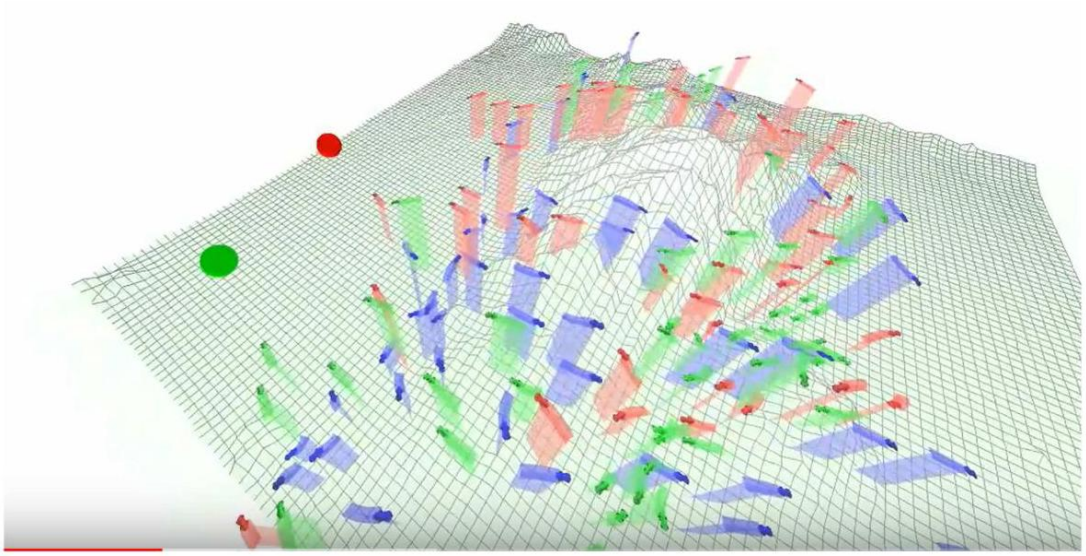

Fig. 4 Example of 150 agents to illustrate the difficulty of avoiding NMACs and collisions. Agents are rendered with 200 meter wingspan for improved visibility, though in simulation actual wingspan of air taxi is 5 meters.

图 4 150 个代理的示例，用以说明避免 NMACs(近场碰撞) 和碰撞的难度。代理以 200 米翼展渲染以提高可见性，尽管在实际仿真中，空中出租车的实际翼展为 5 米。

Table 3 Timing measurements of algorithm performance for each frame as the number of aircraft increases.

表 3 随着飞机数量的增加，算法在每个帧的性能计时测量。

Number of aircraft	Timing (ms)
3	34.9
15	43.9
30	58.1
45	67.4
60	77.1
75	95.2
90	116.4
105	123.6
120	137.7
135	149.8
150	162.6

飞机数量	时间 (毫秒)
3	34.9
15	43.9
30	58.1
45	67.4
60	77.1
75	95.2
90	116.4
105	123.6
120	137.7
135	149.8
150	162.6

Table 4 Links to videos

表 4 视频链接

Number of aircraft	URL	Comment
30	https://youtu.be/vX1BUC9bLFU	Rewards not rendered
30	https://youtu.be/7NXnl5cFWnM	Rewards are rendered
150	https://youtu.be/_B9Ath-3gQI	Rewards not rendered

飞机数量	URL	评论
30	https://youtu.be/vX1BUC9bLFU	奖励未渲染
30	https://youtu.be/7NXnl5cFWnM	奖励已渲染
150	https://youtu.be/_B9Ath-3gQI	奖励未渲染

VII. Conclusion

VII. 结论

In this paper we have demonstrated a computationally efficient distributed computational guidance algorithm suitable for high-density Urban Air Mobility applications. The algorithm can work in both cooperative and non-cooperative collision avoidance contexts. We compare operation of collision avoidance in both scenarios and show that the algorithm successfully avoids collisions even with large numbers of non-cooperative agents in the environment. We show that performance is suitable for embedded real-time applications, but to scale to large numbers of aircraft additional optimizations are required. Future research is already under way to further optimize the algorithm and early results show a dramatic improvement which will permit faster frame rates while scaling to larger numbers of aircraft and other obstacles.

在本文中，我们展示了一种计算效率高的分布式计算引导算法，适用于高密度城市空中出行应用。该算法既可以在合作碰撞避免环境中工作，也可以在非合作碰撞避免环境中工作。我们比较了两种场景下的碰撞避免操作，并展示了算法即使在环境中存在大量非合作代理时也能成功避免碰撞。我们证明，该性能适合嵌入式实时应用，但要扩展到大量飞机，还需要额外的优化。未来的研究已经在进行中，以进一步优化算法，早期结果显示了显著的改进，这将允许在扩展到更多飞机和其他障碍物时实现更快的帧率。

In other future research, a method to automatically construct negative rewards from terrain should be pursued and alternative ways of capturing the terrain efficiently should be explored. Risk wells seem very appropriate for small obstacles such as other aircraft, but a new approach will be needed to efficiently represent terrain.

在其他未来的研究中，应该追求一种从地形自动构建负奖励的方法，并探索有效捕捉地形的替代方式。风险井对于小型障碍物 (如其他飞机) 似乎非常合适，但需要一种新方法来有效地表示地形。

References

参考文献

[1] Gipson, L., "NASA Embraces Urban Air Mobility, Calls for Market Study,"https://www.nasa.gov/aero/nasa-embraces-urban-air-mobility, 2017. Accessed: 2018-01-19.

[2] "Uber Elevate | The Future of Urban Air Transport," https://www.uber.com/info/elevate/, 2017. Accessed: 2018-08-13.

[3] Holden, J., and Goel, N., "Fast-Forwarding to a Future of On-Demand Urban Air Transportation," San Francisco, CA, 2016.

[4] "Urban Air Mobility," http://publicaffairs.airbus.com/default/public-affairs/int/en/our-topics/ Urban-Air-Mobility.html, 2018. Accessed: 2018-08-13.

[5] "Future of Urban Mobility," http://www.airbus.com/newsroom/news/en/2016/12/My-Kind-Of-Flyover.html, 2017. Accessed: 2018-08-13.

[6] Hoekstra, J. M., van Gent, R. N., and Ruigrok, R. C., "Designing for safety: the 'free flight'air traffic management concept," Reliability Engineering & System Safety, Vol. 75, No. 2, 2002, pp. 215-232.

[7] Bilimoria, K. D., Grabbe, S. R., Sheth, K. S., and Lee, H. Q., "Performance evaluation of airborne separation assurance for free flight," Air Traffic Control Quarterly, Vol. 11, No. 2, 2003, pp. 85-102.

[8] Clari, M. S. V., Ruigrok, R. C., Hoekstra, J. M., and Visser, H. G., "Cost-benefit study of free flight with airborne separation assurance," Air Traffic Control Quarterly, Vol. 9, No. 4, 2001, pp. 287-309.

[9] Tomlin, C., Pappas, G. J., and Sastry, S., "Conflict resolution for air traffic management: A study in multiagent hybrid systems," IEEE Transactions on automatic control, Vol. 43, No. 4, 1998, pp. 509-521.

[10] Kahne, S., and Frolow, I., "Air traffic management: Evolution with technology," IEEE Control Systems, Vol. 16, No. 4, 1996, pp. 12-21.

[11] Harman, W. H., "TCAS- A system for preventing midair collisions," The Lincoln Laboratory Journal, Vol. 2, No. 3, 1989, pp. 437-457.

[12] Frazzoli, E., Mao, Z.-H., Oh, J.-H., and Feron, E., "Resolution of conflicts involving many aircraft via semidefinite programming," Journal of Guidance, Control, and Dynamics, Vol. 24, No. 1, 2001, pp. 79-86.

[13] Raghunathan, A. U., Gopal, V., Subramanian, D., Biegler, L. T., and Samad, T., "Dynamic optimization strategies for three-dimensional conflict resolution of multiple aircraft," Journal of guidance, control, and dynamics, Vol. 27, No. 4, 2004, pp. 586-594.

[14] Enright, P. J., and Conway, B. A., "Discrete approximations to optimal trajectories using direct transcription and nonlinear programming," Journal of Guidance, Control, and Dynamics, Vol. 15, No. 4, 1992, pp. 994-1002.

[15] Schouwenaars, T., De Moor, B., Feron, E., and How, J., "Mixed integer programming for multi-vehicle path planning," Control Conference (ECC), 2001 European, IEEE, 2001, pp. 2603-2608.

[16] Richards, A., and How, J. P., "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," American Control Conference, 2002. Proceedings of the 2002, Vol. 3, IEEE, 2002, pp. 1936-1941.

[17] Pallottino, L., Feron, E. M., and Bicchi, A., "Conflict resolution problems for air traffic management systems solved with mixed integer programming," IEEE transactions on intelligent transportation systems, Vol. 3, No. 1, 2002, pp. 3-11.

[18] Vela, A., Solak, S., Singhose, W., and Clarke, J.-P., "A mixed integer program for flight-level assignment and speed control for conflict resolution," Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on, IEEE, 2009, pp. 5219-5226.

[19] Mellinger, D., Kushleyev, A., and Kumar, V., "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE, 2012, pp. 477-483.

[20] Augugliaro, F., Schoellig, A. P., and D'Andrea, R., "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, IEEE, 2012, pp. 1917-1922.

[21] Morgan, D., Chung, S.-J., and Hadaegh, F. Y., "Model predictive control of swarms of spacecraft using sequential convex programming," Journal of Guidance, Control, and Dynamics, Vol. 37, No. 6, 2014, pp. 1725-1740.

[22] Acikmese, B., and Ploen, S. R., "Convex programming approach to powered descent guidance for mars landing," Journal of Guidance, Control, and Dynamics, Vol. 30, No. 5, 2007, pp. 1353-1366.

[23] Delahaye, D., Peyronne, C., Mongeau, M., and Puechmorel, S., "Aircraft conflict resolution by genetic algorithm and B-spline approximation," EIWAC 2010, 2nd ENRI International Workshop on ATM/CNS, 2010, pp. 71-78.

[24] Cobano, J. A., Conde, R., Alejo, D., and Ollero, A., "Path planning based on genetic algorithms and the monte-carlo method to avoid aerial vehicle collisions under uncertainties," Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 4429-4434.

[25] Pontani, M., and Conway, B. A., "Particle swarm optimization applied to space trajectories," Journal of Guidance, Control, and Dynamics, Vol. 33, No. 5, 2010, pp. 1429-1441.

[26] Pallottino, L., Scordio, V. G., Frazzoli, E., and Bicchi, A., "Probabilistic verification of a decentralized policy for conflict resolution in multi-agent systems," Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, IEEE, 2006, pp. 2448-2453.

[27] Wollkind, S., Valasek, J., and Ioerger, T., "Automated conflict resolution for air traffic management using cooperative multiagent negotiation," AIAA Guidance, Navigation, and Control Conference and Exhibit, 2004, p. 4992.

[28] Purwin, O., D'Andrea, R., and Lee, J.-W., "Theory and implementation of path planning by negotiation for decentralized agents," Robotics and Autonomous Systems, Vol. 56, No. 5, 2008, pp. 422-436.

[29] Desaraju, V. R., and How, J. P., "Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees," Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 4956-4961.

[30] Inalhan, G., Stipanovic, D. M., and Tomlin, C. J., "Decentralized optimization, with application to multiple aircraft coordination," Decision and Control, 2002, Proceedings of the 41st IEEE Conference on, Vol. 1, IEEE, 2002, pp. 1147-1155.

[31] Schouwenaars, T., How, J., and Feron, E., "Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees," AIAA Guidance, Navigation, and Control Conference and Exhibit, 2004, p. 5141.

[32] Richards, A., and How, J., "Decentralized model predictive control of cooperating UAVs," 43rd IEEE Conference on Decision and Control, Vol. 4, Citeseer, 2004, pp. 4286-4291.

[33] Hoffmann, G., Rajnarayan, D. G., Waslander, S. L., Dostal, D., Jang, J. S., and Tomlin, C. J., "The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC)," The 23rd Digital Avionics Systems Conference (IEEE Cat. No. 04CH37576), Vol. 2, IEEE, 2004, pp. 12-E.

[34] Howlet, J. K., Schulein, G., and Mansur, M. H., "A practical approach to obstacle field route planning for unmanned rotorcraft," 2004.

[35] Kavraki, L., Svestka, P., and Overmars, M. H., Probabilistic roadmaps for path planning in high-dimensional configuration spaces, Vol. 1994, Unknown Publisher, 1994.

[36] LaValle, S. M., "Rapidly-exploring random trees: A new tool for path planning," 1998.

[37] Karaman, S., and Frazzoli, E., "Sampling-based algorithms for optimal motion planning," The international journal of robotics research, Vol. 30, No. 7, 2011, pp. 846-894.

[38] Naderi, K., Rajamäki, J., and Hämälläinen, P., "RT-RRT*: A real-time path planning algorithm based on RRT," Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games, ACM, 2015, pp. 113-118.

[39] Shim, D. H., and Sastry, S., "An evasive maneuvering algorithm for UAVs in see-and-avoid situations," American Control Conference, 2007. ACC'07, IEEE, 2007, pp. 3886-3891.

[40] Shim, D. H., Kim, H. J., and Sastry, S., "Decentralized nonlinear model predictive control of multiple flying robots," Decision and control, 2003. Proceedings. 42nd IEEE conference on, Vol. 4, IEEE, 2003, pp. 3621-3626.

[41] Sigurd, K., and How, J., "UAV trajectory design using total field collision avoidance,"AIAA Guidance, Navigation, and Control Conference and Exhibit, 2003, p. 5728.

[42] Langelaan, J., and Rock, S., "Towards autonomous UAV flight in forests," AIAA Guidance, Navigation, and Control Conference and Exhibit, 2005, p. 5870.

[43] Kahn, G., Zhang, T., Levine, S., and Abbeel, P., "Plato: Policy learning using adaptive trajectory optimization," Robotics and Automation (ICRA), 2017 IEEE International Conference on, IEEE, 2017, pp. 3342-3349.

[44] Zhang, T., Kahn, G., Levine, S., and Abbeel, P., "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," Robotics and Automation (ICRA), 2016 IEEE International Conference on, IEEE, 2016, pp. 528-535.

[45] Ong, H. Y., and Kochenderfer, M. J., "Markov Decision Process-Based Distributed Conflict Resolution for Drone Air Traffic Management," Journal of Guidance, Control, and Dynamics, 2016, pp. 69-80.

[46] Chen, Y. F., Liu, M., Everett, M., and How, J. P., "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," Robotics and Automation (ICRA), 2017 IEEE International Conference on, IEEE, 2017, pp. 285-292.

[47] Yang, X., and Wei, P., "Autonomous On-Demand Free Flight Operations in Urban Air Mobility using Monte Carlo Tree Search," International Conference on Research in Air Transportation (ICRAT), Barcelona, Spain, 2018.

[48] Han, S.-C., Bang, H., and Yoo, C.-S., "Proportional navigation-based collision avoidance for UAVs," International Journal of Control, Automation and Systems, Vol. 7, No. 4, 2009, pp. 553-565.

[49] Park, J.-W., Oh, H.-D., and Tahk, M.-J., "UAV collision avoidance based on geometric approach," SICE Annual Conference, 2008, IEEE, 2008, pp. 2122-2126.

[50] Krozel, J., Peters, M., and Bilimoria, K., "A decentralized control strategy for distributed air/ground traffic separation," AIAA Guidance, Navigation, and Control Conference and Exhibit, 2000, p. 4062.

[51] Van Den Berg, J., Guy, S. J., Lin, M., and Manocha, D., "Reciprocal n-body collision avoidance," Robotics research, Springer, 2011, pp. 3-19.

[52] Muñoz, C., Narkawicz, A., Hagen, G., Upchurch, J., Dutle, A., Consiglio, M., and Chamberlain, J., "DAIDALUS: detect and avoid alerting logic for unmanned systems," 2015.

[53] Kochenderfer, M. J., Chryssanthacopoulos, J. P., Kaelbling, L. P., and Lozano-Pérez, T., "Model-based optimization of airborne collision avoidance logic," Tech. rep., MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 2010.

[54] Bertram, J. R., Yang, X., Brittain, M., and Wei, P., "Online Flight Planner with Dynamic Obstacles for Urban Air Mobility," 2019 Aviation Technology, Integration, and Operations Conference, 2019.

[55] Sutton, R. S., and Barto, A. G., Reinforcement learning: An introduction, Vol. 1, MIT press Cambridge, 1998.

[56] Chitsaz, H., and LaValle, S. M., "Time-optimal paths for a Dubins airplane," 2007 46th IEEE conference on decision and control, IEEE, 2007, pp. 2379-2384.

[57] Owen, M., Beard, R. W., and McLain, T. W., "Implementing dubins airplane paths on fixed-wing uavs," Handbook of Unmanned Aerial Vehicles, 2015, pp. 1677-1701.

[58] Park, H., Lee, B.-Y., Tahk, M.-J., and Yoo, D.-W., "Differential game based air combat maneuver generation using scoring function matrix," International Journal of Aeronautical and Space Sciences, Vol. 17, No. 2, 2016, pp. 204-213.

[59] Huynh, H., Costes, P., and Aumasson, C., "Numerical optimization of air combat maneuvers," Guidance, Navigation and Control Conference, 1987, p. 2392.

[60] Rodriguez, E., Morris, C., Belz, J., Chapin, E., Martin, J., Daffer, W., and Hensley, S., "An assessment of the SRTM topographic products," 2005.