无人机防碰撞系统的建模和实现

XXXX

2020 年 12 月 21 日

摘要

随着民用航空领域无人机的应用越来越广泛，无人机的防碰撞系统就显得更加重要和有价值。理想的无人机防碰撞系统会检测到周围无人机的位置，并且根据无人机的位置来选择能够减小碰撞风险的最优的飞行动作。目前在生成防碰撞的策略时,一般使用部分可观测的马尔科夫决策过程 (POMDP) 对无人机的行为进行建模，并在该模型的基础上进行最优策略的生成。目前已有的防碰撞系统需要敌机的精确位置信息来生成防碰撞的行为，并且忽视飞行路径变化，这样就会出现防碰撞系统的鲁棒性不高，并且计算过程消耗过多的飞行资源的问题。本文根据原始的飞行路径中的有限的信息构建了一个防碰撞系统。我们使用 POMDP 来进行防碰撞系统的建模，系统仅包含无人机的目标位置信息以及敌机位置变化的模糊信息，并且利用这些信息来生成防碰撞的逻辑。我们实现了一个防碰撞的模块，并将该模块集成到了 PX4 飞行控制平台上，进行了飞行模拟实验来验证了我们实现的系统的有效性。

1 引言

安全性对于航空领域至关重要。为了提升安全性，现代的飞行控制包括多个部分，其中包含有对应飞机驾驶资质的良好受训的飞行员，根据飞行状况和地面信号通知飞行员进行飞行行为的空中交通控制系统，以及飞机配备的飞行控制辅助系统。在飞行中最严重的情况就是飞机出现空中的碰撞。通常防止飞机之间相撞的方法可能是通过交通管制系统给出飞机的飞行指令，飞行员按照指令进行飞机操纵，改变航线和高度，在跨洋航行时，飞行员会用机载自动驾驶系统进行辅助飞行。

1.1 机载防碰撞系统

随着空中交通的普及，发生空中飞机碰撞的风险也相应的增加了，因此机载防碰撞系统的研发备受航空界关注。在上个世纪 60 年代到 70 年代，多个飞机制造商开发了飞机的防碰撞系统，例如 BCAS 信标防碰撞系统。BCAS 通过接收地面的信标防碰撞系统发来的信号数据来确定敌机的距离和高度。1978 年，美国联邦航空局开始开发交通预警防碰撞系统 TCAS [?,?]，该系统现今被广泛应用于航空领域。TCAS 是对 BCAS 的改进版本，TCAS 不仅能够获得敌机的位置信息，还能根据这些信息给出飞机操控行为的建议，例如给出应该改变爬升率和下降率给飞行员，从而减少碰撞的风险。经过几十年的对防碰撞系统的开发与改进，TCAS 目前是大型客运以及货运机型上的必备系统。TCAS 使用一个机载的信标雷达来监控空中的交通状态并通过防碰撞的逻辑对可能接近的空中碰撞风险给出飞行员相关的预警信号以及操作建议。TCAS 防碰撞逻辑是根据飞行控制的几十年的经验来构建的，在实现之初，就已经通过了模拟的检验并保证了飞行控制系统的安全性和操作上的可接受性。但是基于经验构建的防碰撞系统在鲁棒性上可能就存在不足。比如该系统不会考虑如果飞行员不按照防碰撞系统的指令进行操作时的情况，并且 TCAS 代码实现较为复杂，完全理解 TCAS 的整个控制逻辑比较困难，更不用说对逻辑进行修改。文献 [?] 中就给出了 TCAS 相关的开发，并且介绍了机防碰撞系统的相关细节。

1.2 无人机防碰撞系统的进展

近些年，随着无人机的不断发展和应用，许多研究开始对无人机的防碰撞进行探讨并开发无人机的防碰撞系统。其中比较有名的是 ACAS X 机载防碰撞系统 [?]。ACAS X 当前将飞机附近的飞行环境用一个部分可观测的马尔科夫决策过程来进行建模，并却用动态规划算法来计算 POMDP 的最优策略，来尽可能的减少碰撞发生的可能性。在计算出最优策略后，防碰撞系统将策略存储在表中，飞行器可以根据当前的状态和表，查到当前应该采取的最优的行动，以最大程度地避免碰撞的发生。在 [?] 中，ACAS X 生成的防碰撞处理逻辑在鲁棒性、效率以及灵活性等层面上都超过超过了之前使用的 TCAS 系统。ACAS Xu，是 ACAS X 系统针对于无人机的版本，其中 u 表示无人机，该系统目前仍在研发中。实际上，无人机的利用和普及使得目前无论是无人还是有人操纵的飞行器，都需要考虑无人机和一般飞机的飞行行为。无人机在以下场景中有着丰富的应用，无人机在军事应用领域已

经开始显得愈发重要，由于无人机的轻量化、良好的隐蔽性以及对于飞行人员来说的安全性，受到军事应用的青睐。军用无人机在集群作战、设备运输以及设施保护等多个方面都有这越来越重要的作用。在民用航空领域，无人机在航空影像、快递运输等方面也扮演重要角色。公共领域无人飞行器的普及，对无人航空器的安全性就有着更高的要求，特别是对于空中防碰撞的要求，这需要无人机有着在空中感知其他飞行单位，并根据其他飞行单位的位置作出防碰撞飞行动作的能力。和一般的民航客机和货机相比，无人机的防碰撞系统面临着更多的挑战，比如无人机有着更复杂的飞行环境，更弱的环境感知能力以及更短的防碰撞动作触发时间，这些都使得保证无人机防碰撞系统的正确性变得更加困难。和大型固定翼客运和货运飞机相比 [**?**]，无人机的防碰撞有着以下的难点：(1) 无人机需要使用更便宜、轻量化以及抗噪能力更强的传感器。(2) 固定翼飞机的飞行行为有着固定的动作控制，而无人机的飞行行为更加复杂。(3) ACAS X 仅仅提供垂直方向的控制，而无人机在水平方向和垂直方向上都有防碰撞的要求。在无人机防碰撞领域已经有许多相关的研究 [**?**, **?**, **?**]。在 [**?**] 中，该文章用 POMDP 来对只能对环境有有限的观测能力的防碰撞系统进行建模，并在离散的状态空间中利用迭代算法计算出策略，并且在全球鹰无人机模拟器上进行了实验验证 [**?**]。[**?**] 直接解决了连续状态空间中的 POMDP 来解决高维空间中的状态空间的请哭那个。[**?**] 利用深度神经网络来学习一个近似的查找表，但是他们和我们的工作的差距在于，他们没有在真实的系统上来实现这个防碰撞的逻辑，并且他们的仿真并没有考虑过多的飞行动作会对飞机本身的飞行轨迹产生影响，在我们的工作中我们考虑了这一点。以上所提到的工作一般更加依赖于从传感器获得的敌机的精确位置信息（比如位置、与本机距离等）来生成防碰撞的行为。在实际情况下，由于传感器的能力限制，我们并不能获得如此详尽的敌机和当前飞机的信息，因此要么这些防碰撞系统的鲁棒性不高，要么获得精确信息占用了太多的无人机的计算资源，对计算资源造成了浪费。

1.3 本文的贡献

为了解决以上提出的问题，本文提出了针对于仅能获得敌机有限信息的防碰撞系统，并且该防碰撞系统能尽量简化对于无人机的操作，以减小对无人机原有的飞行行为的变化，以节省飞行资源。我们仍然用 POMDP 来对只有目的地信息，敌机和本机的粗略飞行信息的防碰撞系统进行建模，并利用该 POMDP 模型来生成防碰撞逻辑。通过在 Pixhawk 模拟器上进行

了模拟实验，并且我们还将该防碰撞模块集成到了 Pixhawk 无人机系统上，并将防碰撞模块的实现集成到了真实的飞行控制平台 PX4 上，能够在真实环境中进行飞行。从本文在 Pixhawk 无人机平台上的实验可以看出，我们生成的防碰撞逻辑在真实飞行情况下也有着不错的表现。

2 部分可观测马尔科夫决策过程

这一节主要回顾 POMDP 的相关定义以及 [?] 中如何给出的无人机防碰撞的解决方案。

对于一个在随机环境下的智能体（例如本文无人机碰撞系统中的无人机）。该智能体和环境一同构成了我们研究的整个系统，该系统会记录智能体和环境所在的状态。在每个单位时间内，智能体通过观察整个系统所处的状态，来决定下一步需要执行的动作，该动作又会产生智能体本身的状态变化，从而影响到整个系统的状态。在概率模型中，我们通常使用一个状态的概率分布，来表示智能体在执行一个动作后的结果的不确定性。当系统的状态空间对智能体完全可知的时候，我们可以用马尔科夫决策过程（MDP）直接对系统建模 [?]。但如果只能提仅仅知道系统状态的部分信息时，用部分可观测马尔科夫决策过程就能更好的对这样的情况进行建模。在无人机这个场景下，部分可观测具体可能是无人机配备了带有噪音的传感器，不能获得或者不能区分获得的数据是否精确，也就不能确定敌机的精确的位置信息，而是仅能知道一个粗糙的对于位置信息的描述（例如敌机在左侧、右侧等）。

POMDP 对 MDP 模型进行扩展，POMDP 隐藏了 MDP 中可见的当前所在的实际状态，并提供了一个观测进程根据当前所在的状态随机的生成观测。POMDP 对智能体的决策过程进行建模，智能体决策过程通过对系统的回报值的相关分析和计算，能够得到一个使得总回报值最大的动作序列。但是这个过程中，智能体并不能精确的知道系统的所在的状态，而是只能得到状态的观测。不同的系统可能得到的观测是相同的。智能体会用一个状态的概率分布来表示其对于当前所在状态的信念，并会根据采取的不同动作以及得到的观测来改变信念。

2.1 POMDP 具体定义

一个部分可观测的马尔科夫决策过程是一个七元组 $(S, A, O, T, Z, R, \gamma)$，其中 S, A, O 分别表示状态、动作以及系统的观测空间。$T\colon S\times A\times S \to [0,1]$ 和 $Z\colon S\times A\times O \to [0,1]$ 分别是迁移函数以及观测函数。对于每个状态 $s \in S$ 和动作 $a \in A$, 以及迁移后的状态 $s' \in S$ 和观测 $o \in O$, 对迁移函数以及观测函数我们有 $T(s,a,s') = p_S(s'|s,a)$ 以及 $Z(s,a,o) = p_O(o|s,a)$。其中概率分布 p_S 和 p_O 分别用来对系统动力学和观测建模；$R\colon S \times A \to \mathbb{R}$ 是回报函数；$\gamma \in [0,1)$ 是用来平衡即期回报和远期回报的折扣因子。在这篇文章中，我们假设动作集合 A 和观测集合 O 都是有穷的。

POMDP 在每个状态的每一步都按照下面的方式进行执行：当系统在状态 $s \in S$ 执行动作 $a \in A$ 时，系统会进行以下行为：1) 获得在状态 s 执行动作 a 的回报值 $r = R(s,a)$；2) 系统迁移到以概率 $T(s,a,s')$ 迁移到状态 s'；3) 以 $Z(s',a,o)$ 的概率在 s' 产生一个观测 $o \in O$。这里需要注意观测依赖于迁移之后的具体状态。

由于智能体仅对当前系统处在的状态保存一个信念，因此在执行动作 a 并获得观测 o 来对当前的信念做改变。我们令 $\mathcal{B}$ 表示信念的集合，也就是状态上的概率分布的集合。给定一个信念 $b \in \mathcal{B}$, 在执行动作 a 并得到观测 o 之后, 智能体通过以下公式来更新对于迁移到的 s' 的信念：

$$b_a^o(s') = \eta Z(s',a,o) \int_{s\in S} T(s,a,s') b(s)\,\mathrm{d}s, \tag{1}$$

其中 η 是一个使得积分 $\int_{s\in S} b_a^o(s) = 1$ 的正规化因子。为了简化标记，当 a 和 o 都清晰的时候，我们 b' 来表示 b_a^o。更新信念的过程在是贝叶斯规则的直接应用，也叫状态估计或者过滤 [**?**].

2.2 POMDP 模型与求解

POMDP 和 MDP 类似, 都有不确定性以及概率的行为。不确定性体现在，对于一个状态可以执行的不同的动作，我们是通过智能体的策略 π 来表示在每个状态智能体的行为选择, 也就是说策略是一个根据当前对于状态的信念到执行动作的一个映射，可以表示为 $\pi\colon \mathcal{B} \to A$.

智能体的执行过程可以看作一个动作选择和信念更新两个操作交织的一个序列。给定一个策略 π 以及当前智能体的信念 b，智能体的控制系统通过策略计算出当前的动作 $a = \pi(b)$. 然后在执行完动作后，又会根据公式 **??**中更新信念的公式来进行更新。

求解 POMDP 的最优策略 π^* 的目标就是最大化智能体的平均总体回报，可以用 $\mathbb{E}(\sum_{n=0}^{\infty}\gamma^n R(s_n,a_n))$ 来表示，其中 s_n 和 a_n 表示系统在第 n 步的状态以及选取的动作。给定一个策略 π 我们可以给出该策略的值函数 $V_\pi\colon \mathcal{B}\to\mathbb{R}$ 来表示期望全体回报。策略 π 的全体回报的值可以根据以下公式给出：

$$V_\pi(b)=\mathbb{E}\big(\sum_{n=0}^{\infty}\gamma^n R(s_n,a_n)|\pi,b\big). \tag{2}$$

在实现层面上，即使是对于例如 [**?**] 中的小规模的问题，求 POMDP 的最优策略这一问题的计算复杂性很高。精确求解最有策略的相关算法可以在一些文献中找到，比如 [**?**] 中的 witness 算法。但是这些精确的求解方法对于大规模的例子在实际计算上并不可行。为了使得在大规模的 POMDP 模型上，其最优策略也能够被求解，常见的方式是考虑求最有策略的近似解，例如在 [**?**] 中所提到的近似求解算法。其中比较成功的算法是基于点的 POMDP 算法，既能支持计算最优策略的近似解，又能应用到大规模的离散系统上 [**?**, **?**]。由于最优值函数 V^* 满足以下 Bellman 方程 [**?**]

$$\begin{aligned} V^*(b) &= H(V^*(b)) \\ &= \max_{a\in A}\Big\{\int_{s\in S} R(s,a)b(s)\,\mathrm{d}s+\gamma\sum_{o\in O}p(o|b,a)V^*(b_a^o)\Big\} \end{aligned} \tag{3}$$

其中 H 称为 *backup* 算子，大多数计算方法都用它来计算最优值 V^*. 值迭代算法是基于 Bellman 方程解 MDP 最优策略的常见的动态规划算法 [**?**]，在 POMDP 的算法中，我们也用到值迭代算法的相关结构 [**?**]。

在值迭代的过程中，首先通过已知初始值的相关信息给出的值函数 V。如果一开始我们不知道任何信息，那么对于所有的信念 V 会被设为 0。然后算法会在 V 上不断的迭代用 H 计算 Bellman 方程直到到一个不动点。当迭代过程终止时，计算出来的策略就是对于每个信念 b 都满足 Bellman 方程的动作 a.

在实现这个算法时，如果直接实现，值迭代算法需要对整个信念空间 $\mathcal{B}$ 进行搜索，在现实问题下就会遇到为多以及指数复杂度的问题。[**?**]. 两个基于点的算法在提高计算效率上起了至关重要的作用。 [**?**]. 首先是用从信念空间 $\mathcal{B}$ 上随机采样得到的点作为信念空间的 $\mathcal{B}$ 的近似表示. 算法的 backup 操作是在信念空间的采样点上去计算而不是整个信念空间，因此计算出来的最有策略是一个近似最优策略。其次，α-向量 [**?**] 被用来作为值函数的近似 $V_\pi\colon \mathcal{B}\to\mathbb{R}$ 来表示从信念 b 开始的执行策略 π 的期望总体回报值。最优

值函数可以用形如 $V^*(b) = \max_{\alpha\in\Gamma} \alpha \cdot b$ 分段函数和图函数 [?] 来表示，其中 $\Gamma = \{\alpha_0, \alpha_1, \ldots, \alpha_m\}$ 是一个含有 m 个向量，我们把它称作 α-向量. 直观上来说, 每个 α-向量都能和一个动作相联系。对于当前的信念 b 来说，求解的策略会选在使得 α-向量最优的动作。因此策略同样也可以用含有 α-向量的 Γ 来表示 [?]. 通过 α-向量的方式, 在一定合适的条件下，我们能用目前计算出来的策略为信念空间中剩下的部分计算最优策略，从而显著地提高计算的效率 [?].

基于点的算法和一般算法的不同在于信念空间上的采样以及执行 backup 操作的不同。PBVI [?] 是第一个在大规模的 POMDP 上成功应用基于点的算法的样例。HSVI2 [?] 使用启发式的方法来指导在信念空间上的采样, 这么做可以减少值函数的上界和下届之间的差距, 从而达到加速的目的。SARSOP [?] 和 HSVI2 的方法相关, 但在这篇文章中，作者尝试用增强学习和约束的技术对从初始信念 b_0 到最优的可达空间 $\mathcal{R}^*(b_0)$ 进行采样。

尽管基于点的算法在求解离散的 POMDP 上有着很好的结果, 在连续状态空间的 POMDP 基于点的算法目前没有太多研究的进展. 连续的 POMDP 模型对于真实世界的行为有着更强的建模和表示能力，然而唯独问题成为连续 POMDP 模型的一大障碍。通常对于连续状态空间 POMDP 的做法是用各自分割出离散的状态集合 S, 然后用已有的对于离散 POMDP 的算法来求解。这个方法的难点是在分割离散状态时，分割出来的状态空间可能还是太大不符合我们的需求。解决这个问题的一个办法是用一个特殊的参数来表示状态的置信度以及值函数。比如高斯分布的线性组合 [?]。另外一个想法是用粒子滤波来表示置信度，比如 MC-POMDP [?]. MCVI (Monte Carlo Value Iteration) [?] 结合了用粒子滤波表示置信度以及离散的基于点的 POMDP 的 α-向量的算法。MCVI 用策略图 G 来表示解得的策略 [?]，其中策略图是一个用编码了最优策略图，在图的每个节点上用动作来做标记，在每条边上用一个观测做标记。置信度并不会显式地存放在这个途中。给定一个置信度 b，MCVI 算法从策略图 G 中，以合适的节点 v 为起始点合成一个策略。策略选择的动作，就是 v 上对应标记的动作，然后智能体根据所选的边 (v, v') 走到下一个状态 v' 并且得到边对应的相应的观测 o。这个过程会一直重复，直到得到最后的策略。

对于每个策略图 G 上的节点 v,我们可以定义 α-函数 α_v,其中 $alpha_v(s)$ 是从状态 s 开始执行策略 $\pi_{G,v}$ 的期望总体回报：

$$\alpha_v(s) = \mathbb{E}\big(\sum_{n=0}^{\infty} \gamma^n R(s_n, a_n) | \pi_{G,v}, s\big). \tag{4}$$

策略 $\pi_{G,v}$ 下的置信度 b 的值是根据 G 的 α-函数如下定义：

$$V_G(b) = \max_{v \in G} \int_{s \in S} \alpha_v(s) b(s) \, \mathrm{d}s. \tag{5}$$

策略图是通过近似动态规划技术来构建的，MCVI 执行蒙特卡罗模拟来对状态空间和置信度空间进行采样。POMDP 的最优值函数 V^* 是利用方程(**??**)中的 backup 操作在可达的置信度集合中的置信度 b 上多次迭代计算得来的。每次 backup 操作可以选择一个使得期望即时回报和期望总体回报相加之和在下一个置信度上最大的动作。找到这个动作之后，该动作将被加到策略图中，并且让动作和最优决策距离靠近。可达置信度集合是从整个置信空间中采样得到的对于之心空间的近似表示。MCVI 方法在处理连续状态上的积分效果较好，因此在使用 MCVI 方法时应该避免离散化状态空间从而提高效率。在 [**?**] 中有着更详细的描述。

3 Collision Avoidance Models

To the best of our knowledge, the algorithms and models used in the latest versions of ACAS X and ACAS Xu are not publicly accessible. Therefore, we define our model based on the early published version of ACAS X [**?**] and ACAS Xu [**?**]. In our implementation, the aircraft model is parametric and can be easily modified to fit different types of aircraft. In this work, we use the parameter values from Pixhawk[1] and build the model with PX4[2].

In our model, we consider a simple encounter situation between two aircraft, our own aircraft and an intruder aircraft. Our own aircraft has no prior knowledge about the flight path of the intruder aircraft, when an encounter happens. During this period, our own aircraft can only get information about the intruder aircraft's elevation and bearing by sensors onboard. The collision avoidance system is responsible to control our own aircraft so to keep a safe distance from the intruder by choosing appropriate actions to avoid collision.

[1] `https://pixhawk.org/`

[2] `https://dev.px4.io/master/en/index.html`, PX4 Development Guide.

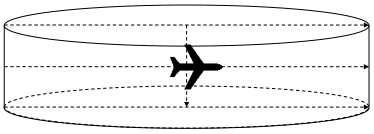

图 1: Threshold range for starting the system in 3D

We will describe the state-transition, observation and reward modeling in our model in detail, similarly to the modeling proposed in [**?**].

3.1 State-Transition Modeling

The position of a flying aircraft can be represented as a point in the 3-dimensional space. The collision avoidance system needs to intervene when the horizontal and vertical distances of two aircraft are less than the given thresholds, which depend on the type of the aircraft and other parameters; this can be represented as the cylinder shown in Fig. **??**.

Similarly to other works on collision avoidance [**?**,**?**], in order to simplify the model's state space and make verification more practical, we omit the altitude and consider only the latitude/longitude of the aircraft. So when an intruder is detected, the system tries to avoid the collision by banking left or right, i.e., by changing our aircraft's direction.

3.1.1 State space.

Since we ignore the aircraft altitude in our model, we let (x, y) be the position of the aircraft with respect to the Earth coordinate system, where the positive x-direction points East and the positive y-direction points North. Let θ be the heading angle of the aircraft with respect to East and u the aircraft's horizontal forward speed. The state information of each aircraft is then (x, y, θ, u). By combining the information about our aircraft and the intruder, a state in the POMDP representing our model is a 7-tuple $(x_1, y_1, \theta_1, x_2, y_2, \theta_2, u)$, where (x_1, y_1, θ_1, u) stands for the state of our own aircraft while (x_2, y_2, θ_2, u) is the intruder's state, by assuming that both aircraft have the same speed.

The state space of the POMDP is continuous, given the nature of the

aircraft information. In order to compute the next action for our own aircraft, traditional methods have to first discretize the state space by means of a grid, where each grid cell becomes a state of the discrete POMDP [**?**]. In our model, we will directly work in the continuous state space instead. In this way, we can apply Monte Carlo methods to improve efficiency of the computation of the policy.

3.1.2 Action space.

Now we introduce our modeling of actions. In order to avoid a collision in an encounter situation, we can change our aircraft direction by changing its forward angle, that is, we bank the aircraft. Thus we can represent the angular velocity $\omega \in \{-\omega_m, 0, \omega_m\}$ as an action in our model, where ω_m is the maximum input value for the banking rate. Consequently, if at step t the heading angle is θ_t and we take the action ω, then the heading angle will become $\theta_{t+1} = \theta_t + \omega\Delta t$ at step $t + 1 = t + \Delta t$ after a small duration of time Δt.

3.1.3 Transition dynamics.

Now we are ready to give the state-transition dynamics of an aircraft in an encounter situation. Recall that in our model, a state consists of the information about both aircraft. Let s_t be the state at step t and s_{t+1} the new state at step $t+1$ after a small duration of time Δt, where Δt is the sum of two small durations Δt_h and Δt_f, i.e., $\Delta t = \Delta t_h + \Delta t_f$. In order to avoid a collision, our own aircraft first changes the heading angle by banking ω degrees for a small amount of time Δt_h and then goes forward at the constant speed u. After a small amount of time Δt_u, our own aircraft will reach the new position (x_{t+1}, y_{t+1}) from (x_t, y_t). This transition dynamics is formalized as follows.

$$\begin{aligned} \theta_{t+1} &= \theta_t + \omega\Delta t_h, \\ x_{t+1} &= x_t + u\Delta t_f \cos\theta_{t+1}, \\ y_{t+1} &= y_t + u\Delta t_f \sin\theta_{t+1}. \end{aligned} \tag{6}$$

Note that at the same time, the intruder will also change its flight

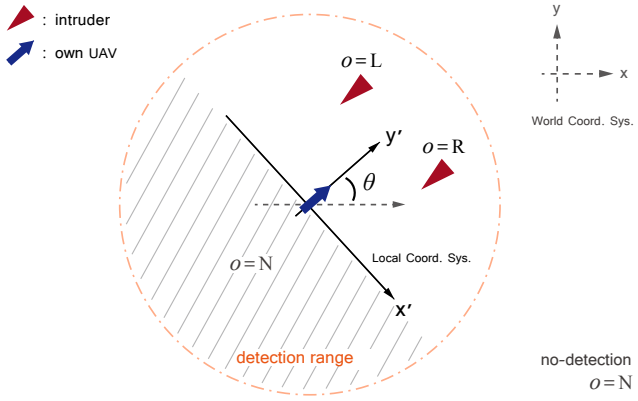

图 2: Observation Model

state (x_2, y_2, θ_2, u) in a similar way by following the action $\omega_{intruder}$. Since our own aircraft does not know the action of the intruder, we can model it in different ways: for instance, the action $\omega_{intruder}$ can be chosen between bank-left, go-forward, and bank-right with equal probability of $\frac{1}{3}$. Other policies can be used to represent different flight plans for the intruder, such as always banking left or right, to represent an intruder flying in circle; we can also consider policies where the intruder tries to intercept our aircraft. We leave the analysis of such policies to the extended version of this paper.

3.2 Sensor Modeling

The threat resolution logic behind a collision avoidance system relies on the aircraft on-board sensors to detect intruders. Electro-optical/infrared (EO/IR) sensors and passive radars are two types of sensors commonly used on large aircraft. On smaller aircraft, the on-board sensors usually have limited accuracy; to model this scenario, we use as possible observations $\{N, L, R\}$, which stand for the fact that no intruder is detected, it is detected on the left of the aircraft, or on the right, respectively. Fig. **??** shows the observation values corresponding to the possible positions of the intruder with respect to our own aircraft.

By following the state modeling introduced in Section **??**, we can let (x_1, y_1) be the coordinates of our aircraft in the Earth coordinate system

and θ its heading angle with respect to East. Let (x_2, y_2) be the coordinates of the intruder in the Earth coordinate system and (x_2', y_2') be the corresponding coordinates in the local coordinate system of our own aircraft, which is represented in Fig. **??** by the $x'y'$-coordinate system whose origin coincides with our aircraft. In order to get the position of the intruder with respect to our own aircraft, we convert the coordinates of the intruder into our local coordinate system as follows:

$$\begin{aligned} x_2' &= (x_2 - x_1)\sin\theta - (y_2 - y_1)\cos\theta, \\ y_2' &= (x_2 - x_1)\cos\theta + (y_2 - y_1)\sin\theta. \end{aligned} \tag{7}$$

We let d denote the horizontal relative distance of two aircraft and o denote the observation value of the intruder; the value d can be computed as

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} = \sqrt{{x_2'}^2 + {y_2'}^2}, \tag{8}$$

while the value for o, for a given detection range r, is

$$o = \begin{cases} N & \text{if } y_2' < 0 \vee d > r, \\ L & \text{if } y_2' \geq 0 \wedge d \leq r \wedge x_2' < 0, \text{ and} \\ R & \text{if } y_2' \geq 0 \wedge d \leq r \wedge x_2' \geq 0. \end{cases} \tag{9}$$

3.3 Reward Modeling

As we have seen in Section **??**, a solution for a POMDP is a policy that maximizes the expected total reward. In order to synthesize a policy piloting our own aircraft to its destination while avoiding collisions, we use rewards to compute a policy that fulfills our requirements.

When at state s and the policy chooses to take action a, a penalty value (i.e., a negative value) $r = R(s, a)$ is computed: it depends on the distance between our aircraft and the intruder, as well as the chosen action.

Since we only consider the 2-D space, we use NMAC to represent the minimum horizontal distance between the two aircraft so that they are considered to be near collision. We give the large penalty of -100 to each action performed when the two aircraft are closer than the NMAC distance, as shown in Table **??**. When the aircraft are enough far away, we give a

表 1: Aircraft action rewards

condition on distance d and banking angle ω	reward value
$d \leq$ NMAC	-100
$d >$ NMAC, $\omega \neq 0$	-1
$d >$ NMAC, $\omega = 0$	0
$d >$ NMAC, at destination	100

penalty of -1 when the action ω of our own aircraft is not zero, i.e., the aircraft changes direction, and of 0 if it goes straight. This allows us to reduce non-necessary actions taken by our own aircraft. Finally, when our aircraft reaches its destination, a reward of 100 is given to each action, as long as the intruder is not closer than NMAC.

Regarding the discount factor for the expected reward (cf. Eqns. (**??**) and (**??**)), we use $\gamma = 0.95$ as discount factor. This values ensures that it is better to immediately avoid the intruder than eventually reach the destination; however, reaching the destination is still taken into account rather strongly (cf. [**?**]).

4 实验结果

我们在 MCVI [**?**] 上面开发,并基于该平台的开源代码`https://github.com/bhy/mcvi`进行了相关实现和实验: 程序的输入是一个 POMDP 模型 (C++ 对象), 并输出一个策略图到文件中。通过扩展 MCVI 中的相关定义, 我们创建了自己的对于放碰撞问题的 POMDP 模型, 并根据这个模型进行策略图的生成。之后我们又扩展了 PX4 平台, 并根据 MCVI 生成的策略, 对 PX4 平台上的无人机加上了放碰撞的模块。我们从一下两个方面评估了放碰撞模块的表现。

PX4 is an open source flight control platform that is supported by the Pixhawk flight control board, an independent open-hardware project for drones and other unmanned vehicles developed by Computer Vision and Geometry Laboratory at ETH Zurich. Due to the clear structure of PX4 flight stack, shown in Fig. **??**. it is easy for developers to add their own flight mode on Pixhawk hardware board with PX4 software, or in the

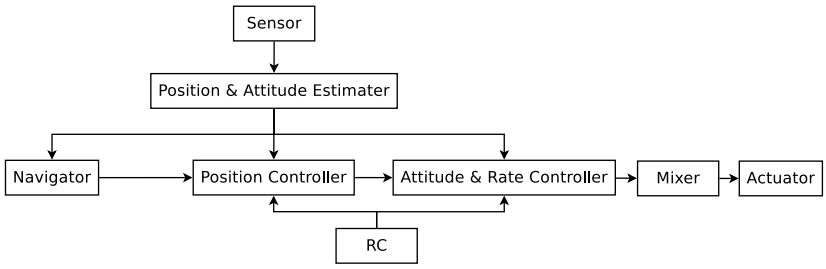

图 3: Overview of the PX4 flight stack

jMAVSim tool that is a simulator for PX4-controlled unmanned aircraft. We integrated the collision avoidance module we developed into the PX4 flight stack by adding it in the position controller, the PX4 component responsible for the flight trajectory.

4.1 Experiments Configuration

In the experiments, since our Pixhawk drone has no sensors for detecting an intruder, we have created synthetic encounter situations where our own aircraft detects an intruder while performing a flight mission, so there may be a collision. The flight mission of our own aircraft is to reach the destination placed at 30 meters North of the initial position. The intruder aircraft starts from our destination and its goal is to reach our initial position. When an encounter occurs, the intruder aircraft is designed to fly according to two predefined policies, introduced later in Sections **??** and **??**. By following the sensor modeling given in Section **??**, our own aircraft makes a detection every second, and the detection radius is set to 10 meters. That is, our own aircraft gets one observation per second about the presence and position of an intruder aircraft; it can be sensed when the distance between the two aircraft is below 10 meters. After getting the observation about the intruder aircraft, our own aircraft will react according to the two different collision avoidance methods introduced below.

Simple rule-based collision avoidance scenario. In the simple rule-based collision avoidance scenario, our own aircraft reacts to observations

as follows: if L or R is observed (i.e., the intruder aircraft is in the detection range on the left or right of our aircraft), then bank to the opposite direction by $\frac{\pi}{6}$ and fly forward by 1 meter; otherwise N is observed, so fly towards to destination.

MCVI-based collision avoidance scenario. In the collision avoidance scenario using the MCVI synthesized policy, our own aircraft reacts to observations as follows: as soon as an intruder aircraft is detected (observations L and R), bank according to the policy as long as the intruder is detected or the graph terminal node is reached. If the intruder aircraft is not detected or the policy graph terminal node is reached, fly towards the destination. The banking angle and step length are the same as those of the simple rule-based method. Note that after the controller stops to follow the policy, if the intruder aircraft is detected again, the controller restarts to follow the policy.

Experiments setting. Finally, we implemented the above two collision avoidance methods in the PX4 platform. Supported by the Software In the Loop (SITL) simulation of the PX4 platform, we used jMAVSimto simulate our experiments. jMAVSim supports the multirotor simulation and allows us to fly drones running PX4 in a synthetic world. To compare the effectiveness of two collision avoidance methods, we recorded the following measure data (with the meter as the length unit) to indicate the performance: the number *actNum* of banking performed to reach the destination, the minimum distance *minDis* between our own aircraft and the intruder aircraft, the maximum deviation distance *maxDvtDis* from the designed mission flight path, and the increased flight distance *incFltDis* from the original mission flight distance.

4.2 Simulated Intruder's Opposite Flight Experiment

In the first simulated experiment, we assume that the intruder aircraft starts from the destination of our own aircraft (viz. 30 meters North of the starting place of own aircraft) and flies straight towards the starting place

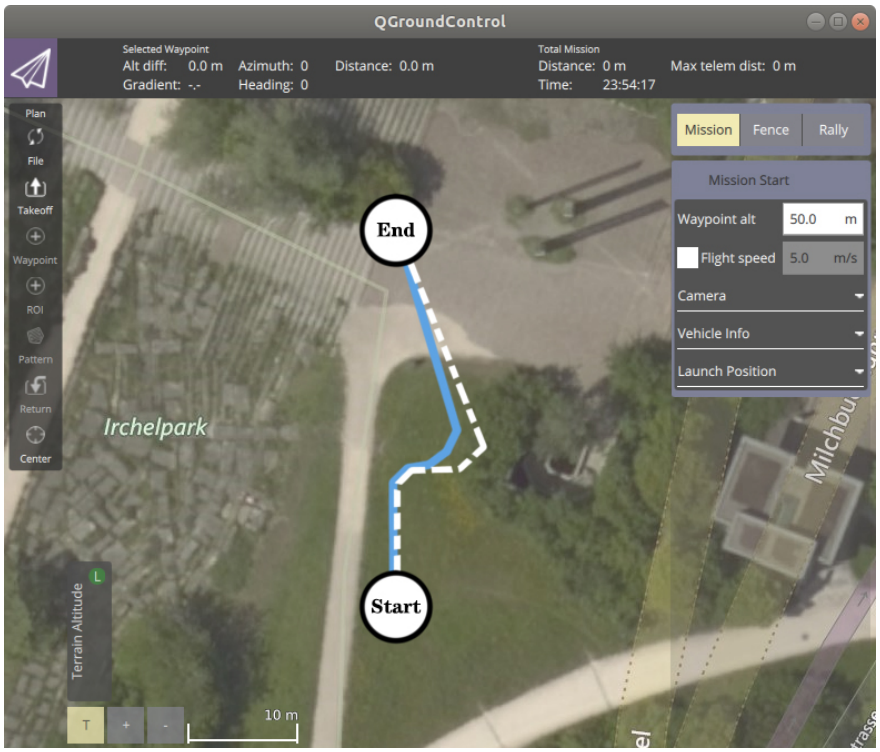

图 4: One flight simulation for the opposite flight experiment. Blue line: MCVI flight path; white line: simple flight path

of our own aircraft. The flight paths of the mission while avoiding a collision are shown in Fig. **??** and the average experimental results are summarized in Table **??**.

Based on the jMAVSim simulation, depicted in Fig. **??**, we can see the flight path of own aircraft in the simulated map, where the blue solid line represents the flight path induced by the MCVI method while the white dashed line tracks the flight path given by the simple rule-based method. It can be intuitively seen from the figure that the MCVI method can reduce the useless avoidance distance of our own aircraft, and avoid the collision risk more effectively.

Table **??** gives more details about the flight paths: we note that although the logic generated by MCVI based on the POMDP model takes

表 2: Opposite flight results; in bold, the best results

	rule-based	MCVI-based
actNum	**4**	5
minDis	2.51	**2.95**
maxDvtDis	9.69	**5.88**
incFltDis	14.83	**9.67**

5 actions instead of 4 to reach the destination while avoiding the intruder aircraft, it does not results in an increase of the flight distance. On the contrary, as shown in Fig. **??** by the MCVI method's blue solid line being closer to the straight line between the terminal points than the simple method's white dashed line, the flight distance is increased by just 9.67 meters, about 5 meters less than the 14.83 meters of the simple method. Furthermore, the MCVI-based approach increases the minimum distance (2.95 vs. 2.51 meters) from the intruder aircraft while deviating less (5.88 vs. 9.69 meters) from the optimal path without intruders than the simple method; this guarantees a higher safety level while decreasing the maximum deviation distance from the new flight path to the original mission flight path.

4.3 Simulated Intruder's Random Interfering Flight Experiment

In the second simulated experiment, we assume that the intruder aircraft takes off from the midway between the start place and destination of our own aircraft (viz. 15 meters North of the starting place of our own aircraft). This time the intruder aircraft chooses randomly an action between bank-left, go-forward and bank-right, i.e., each one with probability $\frac{1}{3}$. Since the flight behavior of the intruder aircraft in this experiment is randomized, we repeated the simulation 50 times and then considered the average outcomes of the two scenarios against the same intruder's random choices, which can be reproduced by fixing the same random seed for the two scenarios.

The statistics for this experiment are presented in Table **??**; as we can

表 3: Random interfering flight results; in bold, the best results

	average value		median value	
	rule-based	MCVI-based	rule-based	MCVI-based
actNum	**3.82**	4.96	**2.5**	4
minDis	5.54	**6.06**	5.88	**6.04**
maxDvtDis	8.85	**5.40**	7.50	**5.60**
incFltDis	9.14	**7.10**	**6.67**	6.87

see, we have a behavior similar to the opposite flight experiment: although the logic generated by MCVI on the POMDP model banks on average one time more to avoid the intruder than the simple rule-based method, it is able to reduce the increased flight distance. Furthermore, it increases the minimum distance between the two aircraft, which implies that the MCVI method gives a higher safety level when dealing with the random behavior of the intruder aircraft. It is particularly noteworthy that when compared with the simple rule-based method, MCVI-based collision avoidance is able to save about 40% of the maximum deviation distance between the actual flight path and the original mission flight path, which is a straight line between the designed takeoff and landing coordinates.

We now give some more details about the safety level achieved by the two collision avoidance scenarios introduced in Section **??**. Out of the 50 simulations for each of the two scenarios, both scenarios ensure that there is no collision since in none of the simulations the minimum distance was below 2 meters, the value we set for NMAC; the minimum distance has been at least 2.5 meters. There have been few simulations where the distance was about 3 meters and the majority above 4 meters, with the MCVI-based scenario usually keeping a larger distance. This is reflected also by the average *minDis* values reported in Table **??**, where the rule-based *minDis* value is lower than the one for MCVI, which means that the former control policy got closer to the intruder than the latter one.

From these experiments, we obtain that the MCVI-based collision avoidance controller provides a better safety level, since it keeps a larger distance between our aircraft and the intruder than the rule-based controller. This

comes at the cost of banking the aircraft few more times, which still allows for reducing the increased flown distance. The flown distance is also an aspect to be considered for drones and other unmanned aircraft, since the increased flight distance reduces the actual operative range of the aircraft, given the limited amount of energy stored in the batteries that can be used to power the aircraft.

While the experiments give a good indication about the safety level provided by the MCVI-based collision avoidance controller, there is no guarantee that it always avoids collisions. We leave the formal analysis of its safety as future work.

4.4 Actual Pixhawk Drone Flight Experiment

In the third experiment, we have taken the PX4 MCVI-based collision avoidance module we developed and uploaded it on a real Pixhawk drone we assembled ourselves. The flight mission is the same as for the previous experiments. The Pixhawk drone communicates to its ground station several data about the position and flight attitude. Since the Pixhawk drone has no sensors for detecting intruders, we used the ground station to inject in the drone's PX4 control system the synthetic intruder from the opposite flight experiment presented in Section **??**. We show in Fig. **??** the actual flight path flown by our drone as registered by the ground control station and superimposed on the map.

As we can see, the MCVI-based collision avoidance module works well not only in a synthetic environment, but also in the real world. This suggests that the policy synthesis for specifically crafted POMDPs can improve the safety of unmanned aircraft equipped with a collision avoidance module basing its decisions on the synthesized policy.

5 Conclusion

We have constructed a collision avoidance system working with limited information, which also tried to save flying resources by means of reducing the change of the original flight path. Our approach used a POMDP to

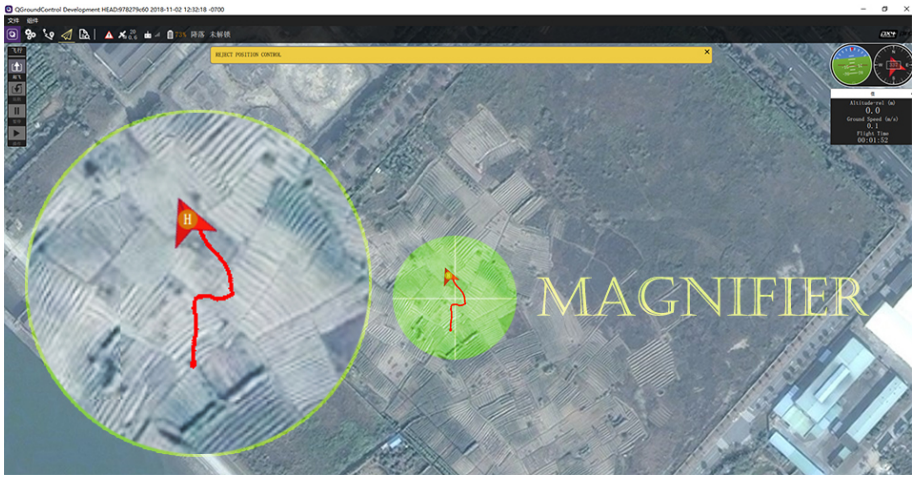

图 5: The actual Pixhawk drone flight path registered by the ground control station

model the collision avoidance system with only the destination information of our own aircraft and rough direction information of the intruder and then generated the collision resolution logic that maximizes the expected sum of rewards of selected actions. We have implemented the collision avoidance module and embedded it into the PX4 flight control platform that can control real unmanned aircraft systems. The effectiveness of our system is witnessed by the experimental evaluation in both a simulated environment and a real world scenario using a Pixhawk drone.

As future work, we consider to study the scalability of the MCVI approach to more detailed flight scenarios, such as the ones with the 3D environment, different aircraft speeds, more accurate observations given by better sensors, and so on. Also, we plan to apply machine learning algorithms to improve the efficiency of the algorithm for searching an optimal policy. Moreover, we also consider the use of rigorous methods, such as model checking, to ensure the correctness of the collision avoidance system.