arXiv:2304.07651v1 [cs.RO] 15 Apr 2023

从作战需求到机器人驱动：完整的快速集成集群解决方案

Eugene M. TarantaII诺斯罗普·格鲁曼公司 任务系统 佛罗里达州奥兰多 32817
eugene.taranta@ngc.com

Adam Seiwert
诺斯罗普·格鲁曼公司 任务系统
科罗拉多州奥罗拉 80017
adam.seiwert@ngc.com

Anthony Goeckner
诺斯罗普·格鲁曼公司 任务系统 马里兰州安纳波利斯 21401
anthony.goeckner@ngc.com

Khiem Nguyen
诺斯罗普·格鲁曼公司 任务系统
科罗拉多州奥罗拉 80017
khiem.d.nguyen@ngc.com

Erin Cherry诺斯罗普·格鲁曼公司 任务系统 弗吉尼亚州麦克莱恩 22102
erin.cherry@ngc.com

摘要

群体机器人系统有望改变城市环境中的作战方式，但迄今尚未进行大规模实地测试。本文介绍快速集成集群生态系统（RISE），这是一个面向未来多智能体研究与部署的平台。RISE支持快速集成第三方蜂群战术与行为，并通过实体蜂群和仿真蜂群进行了演示。我们的实体测试平台由250多个联网异构智能体组成，并已在五个城市作战训练场的模拟战争场景中进行了广泛测试。RISE具备实况、虚拟与构造仿真能力，能够同时使用虚拟和实体智能体；其“流体保真度”仿真则可根据动态运行时需求，在低保真度与高保真度仿真级别之间自适应调整。虚拟和实体智能体均通过统一的基于手势的界面进行控制，使智能体与操作员之比超过150:1。借助该界面，我们能够高效执行基于集群的任务。RISE通过快速战术集成、可靠的试验平台和高效操作，将任务需求转化为机器人驱动。

1 引言

协调一致的协作机器人集群有望为社会带来变革性影响。例如，集群有望显著提升灾害响应工作（如搜救和损害评估）的成效，还能提供通信网络，甚至运送物资。



图1： 田纳西州坎贝尔堡开展 FX-6野外实验时，大规模发射无人飞行器（UAV）。

群体协同也可能通过自主作业使农业等行业受益，从而降低成本、节省时间并提高精度（Albani、IJsselmuiden、Haken 和 Trianni，2017年）。我们的研究聚焦于通过美国国防高级研究计划局（DARPA）的进攻性蜂群战术（OFFSET）项目，将大规模集群应用于军事领域。部署机器人蜂群可将军事单位的控制幅度从 1:1（士兵：平台）扩大到 1:150+。这样一来，集群可使任务效能提升数个数量级。群体协同被视为一种非线性分散式（NLD）作战。S. J. A. Edwards 在兰德公司的一份报告（2005年）中指出，群体协同是 NLD 作战中最具进攻性的形式；当重型部队无法投入时，它将在未来军事行动中发挥核心作用。

低成本机器人硬件、网络解决方案、感知算法和人工智能方面的进步，正使实用的大规模机器人群逐渐成为现实（Chung等，2016）。这些进步通常分散在各自独立且彼此孤立的问题领域中。然而，机器人群体协同是一门跨学科领域，涉及多个相互关联的领域。自主性、分布式感知、后勤、移动自组网和人机蜂群协同（Chung等，2016；S. J. Edwards，2000）都是这样的领域：一个领域的进步可能会对其他领域产生积极的累积效应。因此，要了解新能力对蜂群任务成果的影响，就必须将相关技术集成到群体生态系统中并进行评估。但这本身就存在难题，因为目前鲜有全面的机器人蜂群系统可用于大规模蜂群的研究、开发和部署，尤其是那些经过实地部署并证明具备足够技术成熟度的系统。我们的技术——快速集成蜂群生态系统（RISE）——正是解决这一问题的系统之一。

更重要的是，我们探讨这样一个问题：能否部署一种旨在实现快速集成和一对多控制的多机器人生态系统，让单个操作员指挥 150+ 自主车辆执行战术机动？大规模集群尚未在仿真之外投入部署，而能够支持这一规模下单操作员指挥的软件和硬件架构仍不明确。Chung 等人（2016）以及 Clark、Usbeck、Diller 和 Schantz（2021）此前曾对此进行探索，但其工作在范围或规模上均有限。我们首次展示了利用 RISE 实现大规模集群

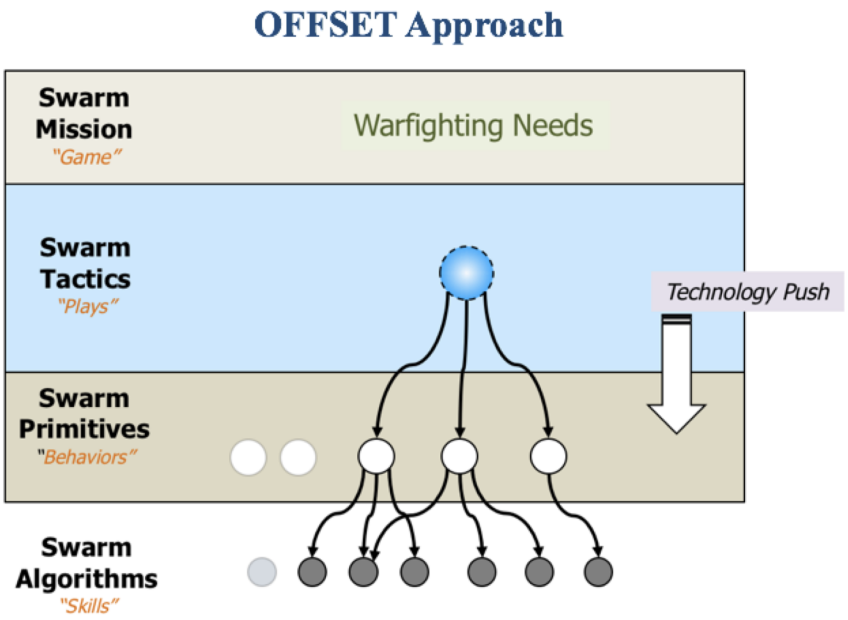


图 2：DARPA OFFSET 对作战需求如何通过分层分解为与机器人硬件交互的算法的看法。人类操作员将任务目标转化为战术集群机动，而战术集群机动又利用算法分解为机器人原语。RISE 中的每一层都通过独特的软件架构得以实现，该架构旨在促进快速开发与集成。图片改编自（DARPA，2017年）。

控制并执行进攻性战术行动是可行的。此外，本文还讨论了我们的架构设计选择，以及指导我们设计的需求，从业人员可以借鉴这些内容来指导自身工作。

通常，群体生态系统需要专用的指挥与控制（C2）界面来控制集群。设计 RISE 时，我们的目标是让 C2 系统能够使单个操作员仅通过指定高级命令，便可控制数百个平台，并使操作员无需编写机器人行为程序即可构建战术。为此，我们从作战手册框架中汲取灵感，并决定采用图 2 所示、为 DARPA OFFSET 计划建立的由战术、原语和算法组成的分层集群框架。该框架与 Giles 和 Giammarco（2017年）的研究相似；两者都受到美国海军研究生院集群实验的启发。

本工作的主要贡献是 RISE[1]，，这是一个端到端的机器人集群生态系统，支持大规模异构机器人集群技术的快速开发、研究、测试和部署。RISE 的贡献还包括：

1. 一种人类–集群团队界面，使单个操作员能够指挥大规模集群，
2. 一个用于集群战术开发与执行的平台，可连接操作员与机器人，并让不熟悉机器人学软件的从业人员也能编写战术，以及
3. 一种轻量级移动自组织网络解决方案，可支持数百个智能体与其他设备之间的大规模可靠通信。

[1]RISE 软件由诺斯罗普·格鲁曼公司和美国政府维护。如需了解有关获取权限程序的更多信息，请联系 Erin Cherry（erin.cherry@ngc.com）。

除了 RISE 软件外，我们还定制、配置并维护了一个由 250 多辆小型无人空中和地面车辆组成的群体测试平台。在过去四年里，我们在五个军事靶场上利用该群体测试平台演示了 RISE。最近一次活动于 2021年11 月 举行，是 OFFSET 群体野外实验的收官之作；我们演示了一名用户控制 174 个平台（84 个空中平台和 90 个地面平台）。

2 相关工作

本节将讨论与 RISE 相关的几个领域中的现有研究，并在可能的情况下介绍这些领域的重要论文。在本节中，我们还会为希望深入了解的读者推荐现有综述。此外，我们指出，群体由许多智能“智能体”组成；每个智能体都是能够自主感知、决策和执行的独立实体（Van der Hoek & Wooldridge, 2008）。我们用“智能体”或“群体智能体”来指代机器人硬件平台及其上运行的软件。此外，我们有时会区分仿真智能体 和实体智能体，前者使用仿真硬件，后者则使用真实硬件。

2.1 多机器人系统

在相关文献中，“群体机器人学”通常指受生物学启发、由同质系统构成，并通过信息素或其他非交换式网络方法进行通信的系统。RISE 是一个“群体生态系统”，但更准确地说，它也是一个“多机器人系统”（MRS），或更广义地说，是一个“多智能体系统”（MAS）。Rizk, Awad, & Tunstel（2019）在对多智能体系统和多机器人系统进行广泛且高质量的综述时，着重指出了这一区别。其他值得关注的综述包括 Dorigo, Theraulaz, and Trianni（2021），该综述聚焦于同质群体机器人学；以及 Schranz, Umlauft, Sende, and Elmenreich（2020），该综述对群体项目和多机器人项目进行了高层次概述。这里我们不再重复这些综述的结果，而只着重介绍特别相关的研究。需要指出的是，尽管本文将 RISE 同时称为“群体机器人学”系统和“多机器人”系统，但我们的讨论仅限于多机器人系统，因为它们与 RISE 最为相似。

具体而言，我们首先关注作为后续研究试验平台的多机器人系统。Jim´enez-Gonz´alez、Martinez-de Dios 和 Ollero（2013年）对多机器人测试平台进行了综述，不过自该综述发表以来已出现多项进展。早期重要的测试平台包括 Mobile Emulab（D. Johnson 等人，2006 年）、HoTDeC（Stubbs 等人，2006 年）和 RAVEN（How, Behihke, Frank, Dale, & Vian，2008 年）。Mobile Emulab 和 HoTDeC 的特点是采用同质化机器人平台，且运行环境受限，只能在实验室中运行。两者的设计目的都是让远程用户能够通过互联网访问系统并控制其运行。在这两个测试平台中，机器人都极为简单，自身的定位或感知能力有限，因此依赖集中式摄像系统。重要的是，这两项研究都没有提出战术或原语等抽象的群体概念，而只是为每个智能体提供基本计算平台，没有配备群体软件框架。

在这些早期的多机器人测试平台中，或许与 RISE 最相似的是 RAVEN。它是一个异构空地集群试验平台，旨在针对各种多机器人场景开展实验。虽然其软件架构与 RISE 相似，但 RAVEN 最多只能容纳十辆车辆，且只能在配备某些外部传感设备的实验室中使用。此外，RAVEN 的指挥与控制范式主要侧重于对单辆车辆进行直接控制，而 RISE 则着重于对整个集群下达命令，以最大化智能体与操作员之比。

在 Pickem、Lee 和 Egerstedt（2015年）的研究中，以及随后 Pickem 等人（2017年）的研究中，作者展示了由其 GRITSBot 平台构建的多机器人测试平台。近期，Ospina、Mojica-Nava、Jaimes 和 Calder´on（2021年）发表了“ARGroHBotS”试验平台，Preiss、Honig、Sukhatme 和 Ayanian（2017年）则发表了“Crazyswarm”。同样，这些试验平台仅限于实验室使用，且不具备或仅具备有限的指挥

与控制功能。

Swarmanoid 项目开发了一种异构集群或多机器人系统，由无人空中车辆（无人机）和无人地面车辆组成（Dorigo et al., 2013年）。与其他项目一样，Swarmanoid 的作者并未明确定义群体战术和整体行为的框架，也未明确说明如何将其他行为整合到系统中。不过，他们提供了一款先进的多机器人模拟器 ARGoS，至今仍用于多智能体研究（Dorigo et al., 2013年）。

COMRADE 多机器人系统是一种先进的多机器人系统，旨在清除冲突地区的地雷和爆炸装置。COMRADE 提供多机器人搜索、数据融合、任务分配和基本的指挥与控制界面，并专为实际应用而设计（Dasgupta, Baca, Guruprasad, Mu˜noz-Melendez, & Jumadinova, 2015年）。然而，作者并未进一步阐述战术概念或用于设计新群体行为的其他抽象；相反，该系统仅用于单一目的。此外，其指挥与控制系统极为简陋，只提供了引导单个智能体所需的最基本控制功能，而且似乎无法通过下达新任务进行分配来手动指挥整个集群。

RUTA 异构群体（Abukhalil, Patil, Patel, & Sobh, 2016）主要是为动态任务分配研究而开发的。然而，它采用了与 RISE 类似的机器人硬件抽象方法，因此能够支持极高的异构性。文中对集群行为软件执行的不同层级作出了一些定义，其中 RUTA 的“动作”与 RISE 智能体层级的“原语”或“算法”相似（Abukhalil et al., 2016）。不过，该集群规模较小，仅有五台机器人，且设计上不支持扩展。此外，该系统主要关注任务分配，不包含任何与指挥与控制相关的概念。

近期，Chamanbaz 等人（2017年）开发了一套软硬件套件，使机器人平台能够轻松集成到多机器人系统中。该套件由计算和网络硬件组成，可与现有平台集成，并配有一个软件库，以便利用这些平台创建新的行为。该平台在两个独立的集群系统中进行了测试。然而，作者没有区分行为栈的不同层级（战术、原语和算法），也没有考虑任何人机交互因素。事实上，作者完全没有提及任何类型的用户界面或控制。

Chung 等人（2016）介绍了一项重要的多机器人系统研究，其中一支由两人组成的团队同时控制 50 架无人机。这项工作可视为 OFFSET 的直接前身，两者的项目经理都是 Timothy Chung 博士，目标也相似：让多机器人系统能够在最少操作员的控制下执行复杂的协同动作。Davis、Chung、Clement 和 Day（2018）在此基础上进一步拓展，让 Chung 等人（2016）创建的两支无人机蜂群争夺制空权。RISE 在这项工作的基础上进一步发展，采用了数量更多的智能体、创新的新型指挥与控制概念、专为集成新战术和行为而设计的软件架构，并引入了异构性等。

近年来，人们开发了许多有助于多机器人系统开发的工具。其中一种工具是 ROS2swarm，这是适用于机器人操作系统 2 框架（ROS 2）的硬件和软件包，可改装到现有平台上。ROS2swarm 提供多机器人编程能力，其集群软件划分方式与我们的战术、原语和算法概念相似（Kaiser 等，2022）。然而，ROS2swarm 不支持战术间的动态切换。换言之，在单次运行中，智能体只能执行一种战术，除非彻底重启机器人软件。与 RISE 不同，ROS2swarm 也不提供任何指挥与控制能力。此外，ROS2swarm 目前采用集中式网络拓扑（基于接入点），而 RISE 采用去中心化拓扑。虽然 ROS2swarm 仅在六个智能体上进行了测试，但我们相当确信，鉴于其使用了 ROS 2 内置的组网机制，ROS2swarm 系统实际上无法支持大量智能体（$150+$）。RISE 以定制的智能体间通信方案取代了 ROS 2 的默认组网机制。有关 RISE 组网方案的更多详情，请参见第4.3节。

Villemure 等人（2022）提供了另一种平台 SwarmUS，其中包含硬件改装板和软件包，与 Kaiser 等人（2022）的方案类似。该平台为多机器人系统提供通信、协调和定位等必要服务。此外，它还提供一个基于 Android 的指挥与控制界面，但该界面似乎较为基础，并非为单个操作员控制大规模集群而设计。事实上，SwarmUS 测试的机器人不超过六台，而 RISE 则测试了数百台（Villemure 等人，2022）。不过，重要的是，SwarmUS 似乎主要侧重于硬件设计，而 RISE 则主要侧重于软件设计。因此，两者难以进一步比较。

RISE 与 DARPA OFFSET 计划中由我们的同行开发的“CCAST”同期出现（Clark 等人，2021年）。由于 CCAST 和 RISE 解决相同的问题，且面向同一计划和实验场景，因此两者的大多数功能都相当相似。不过，RISE 与 CCAST 的不同之处在于，前者采用去中心化网状网络进行通信，而 CCAST 采用接入点方案。此外，RISE 使用创新的通用手势命令，便于在笔记本电脑、触屏设备以及虚拟现实和增强现实平台之间跨平台使用；而 CCAST 使用基于点击操作的虚拟现实界面。CCAST 与 RISE 一样，采用了由战术、原语和算法构成的行为分解方式。

2.2 机器人平台

由于 RISE 是一个复杂的软件系统，可搭配多种硬件平台使用（见图 25），因此我们不详细介绍以往用于群体机器人学或多智能体研究的硬件平台。相关综述可见于文献中，例如 Schranz 等人（2020年）的综述，其中列举了多个平台。

2.3 群体网络

文献中的群体机器人系统大致可分为三类：采用受生物启发的“信息素”系统的系统、采用定向通信的系统（如 Kornienko、Kornienko 和 Levi（2005年）所述），以及采用非定向无线电网络的系统。本文将重点讨论无定向数字无线电网络，因为我们认为前两种方法与我们的方案差异太大，几乎无法比较。

常用的无线电网络架构有三种：中央接入点、本地无线广播和移动自组织网络（MANET）。在传统无线网络中，智能体只能与固定基础设施（称为接入点）通信，无法直接进行智能体间通信。相反，智能体必须先将数据发送到接入点，再由接入点将数据转发给目的智能体。相比之下，MANET 既支持传统的智能体与基础设施间通信，也支持直接的智能体间通信（Hoebeke、Moerman、Dhoedt 和 Demeester，2004年）。

CCAST 系统采用中央接入点方案（Clark 等人，2021年）。该研究的作者发现，智能体与接入点的连接成为一项主要问题，因为智能体无法直接与附近邻居通信，从而增加了碰撞风险。两个相邻智能体要进行通信，各自都需要与接入点保持有效连接，而这在城市环境中并非总能实现。Stubbs 等人（2006年）以及许多其他实现也采用了这种方案。

本地无线广播架构更为常见。虽然这类架构通常不需要中央接入点，但一般不支持数据中继；根据 Hoebeke 等人（2004年）的定义，这一点使其有别于真正的“移动自组网”。Konolige 等人（2006年）采用了智能体间本地无线广播。Rubenstein、Cornejo 和 Nagpal（2014年）也采用了类似系统：数据不会中继给超出发送智能体接收范围的智能体，因此只能与附近邻居进行“本地”通信。Caprari 和 Siegwart（2005年）以及 Dorigo 等人（2013年）也采用了类似方法。

钱奇、雷米、帕格 h 和 Martinoli（2007）实现了真正的移动自组网。他们的方法利用了 e

ZigBee网状协议。Fernandes、Couceiro、Portugal、Machado Santos 和 Rocha（2015），Jevti´c、Gazi、Andina 和 Jamshidi（2010），以及 Zahugi、Shabani 和 Prasad（2012）等人也使用 ZigBee 实现移动自组网。在文献中，ZigBee系统是迄今最受欢迎的移动自组网实现方案。Rajant Corporation 的移动自组网解决方案也成功应用于 Engebr°aten、Nummedal、Gilbreath、Yakimenko 和 Glette（2019）的无人机集群项目，以及 I. D. Miller 等人（2020）的矿山勘探集群项目。

2.4 仿真

与任何机器人开发一样，仿真都是一项关键工具；对于多智能体和集群系统而言尤其如此。涉及多个智能体的仿真在实现方式上差异很大，而在将仿真结果迁移到实际硬件的能力方面，差异则更为显著。大多数集群仿真通常由定制仿真环境构成，用于评估特定算法（Hamer和Ortega-Sanchez，2010年）。这种方式适用于相应的特定评估，但对于RISE这样的快速集成群体协同生态系统而言，此类设置并不足够。鉴于此，我们更注重那些强调与硬件一对一迁移的仿真环境，例如Gazebo（Aguero等人，2015年）。我们借鉴了此类仿真的经验教训和最佳实践，随后着手进行必要调整，以在维持向真实世界迁移能力的同时，达到所需的仿真智能体数量（根据OFFSET计划，$250+$ ）。第9节将更详细地介绍这种方法，并进一步致谢其他相关仿真项目。

2.5 人机群接口

具体而言，人群交互，以及更广泛意义上的人机交互，是一个活跃的研究领域，涵盖界面设计、控制、通信、自主性，以及态势感知和认知负荷等人为因素（Drew，2021年；Chen 和 Barnes，2021年；Kolling、Walker、Chakraborty、Sycara 和 Lewis，2015年；Hocraffer 和 Nam，2017年）。不过，本小节聚焦于能够实现操作员一对多控制进攻型集群的研究，从命令复杂度谈起，最后讨论用户界面定制。

命令复杂度与计算复杂度有相似之处：完成一项任务所需的总投入，取决于操作员必须作出的决策和采取的动作数量（Lewis、Wang 和 Scerri，2006年）。当机器人独立工作时，投入会随着操作员必须命令的机器人数量线性增长 n $\mathcal{O}$（n）。相反，需要机器人之间进行精细协调的任务可能导致超线性复杂度 $\mathcal{O}$（$> n$）。Lewis 举例说明了两辆无人驾驶车辆协同推动箱子两个角的情况：操作员需要在两台机器人之间来回切换，才能移动并摆正箱子。

尽管命令复杂度使从业人员能够比较不同命令策略的有限效率，但扇出是 Olsen 和 Goodrich（2003 年）提出的另一项有用指标，可用于估算操作员一次能够有效操作多少个同质机器人。扇出在数学上定义为 ($NF + IT$)/(IT)，其中，忽略时间 NT是指在不使性能降至最低性能阈值以下的情况下，机器人可被忽略的预期持续时间；交互时间 IT是指与机器人进行交互所需的持续时间（Crandall、Goodrich、Olsen 和 Nielsen，2005年）。直观而言，机器人在无需操作员关注的情况下能够完成的任务越多，操作员设置命令的速度越快，操作员就能在不低于所需性能水平的情况下命令更多机器人。先前研究表明，操作员可以同时命令八至十二个机器人（H. Wang、Lewis、Velagapudi、Scerri 和 Sycara，2009 年），具体取决于多种因素。这意味着，为支持大规模进攻性蜂群作战，我们必须从直接的机器人控制转向更高层次的抽象。

一种解决方案是提高蜂群自主性（Mi & Yang，2013年）。通过使用规划器，可以在一定程度上实现向命令 $\mathcal{O}$(1) 迈进的命令复杂度（Lewis，2013年）。试想，如果没有路径规划器和避障功能，研究发现操作员无法监管超过少数几架

无人机（Cummings、Nehme、Crandall 和 Mitchell，2007年）。将某些任务委派给自主系统，可以进一步提升命令复杂度和扇出。例如，McLurkin 等人（2006年）设计了一种集群界面，其灵感来自流行的即时战略游戏 WarCraft 和 StarCraft。这些游戏允许玩家通过“采矿”“建造”和“攻击”等简单的高级命令，控制单个单位、多个单位组成的组，或整支军队。将单条命令分发给多个机器人，可以显著降低分摊后的交互时间，并提高扇出。当操作员能够依赖底层自动化，使用单条命令指挥多个机器人时，命令复杂度也会呈次线性增长。另一个名为 Playbook 的示例（C. Miller、Pelican 和 Goldman，2000年；Calhoun、Ruff、Behymer 和 Frost，2018年）为操作员提供了一个战术模板库，操作员可以修改其中的模板，而自动化系统会根据情况进行调整。Playbook 类似于运动队的战术手册：领队选定一套战术，由团队执行；多项研究已经表明，使用模板化战术具有诸多益处。

我们采用了与这些委派策略类似的方法。在 RISE 中，操作员通过使用能够调配适当资源并协调多个机器人的动作的任务级战术指令来控制集群。操作员可以组合多种战术以形成行动方案（COA），并在情况变化时实时修改计划。在与人类作战人员共同进行任务规划期间，指挥官通过行动方案图传达战术；行动方案图包含战术控制措施，用于明确责任并限制行动，从而防止各单位相互干扰（美国陆军，2015年）。这一概念可以通过基于草图和手势的界面轻松扩展到集群指挥中，以模拟自然使用纸笔的方式。例如，Hammond 等人（2010）开发了一种徒手行动方案图草图识别系统，支持军事符号，并在 485 个符号上实现了较高的准确率。

一个问题在于，行动方案图采用标准化的军事符号绘制，例如 MIL-STD 2525D 所定义的符号（美国陆军，2020）。然而，RISE 的群体战术和控制措施仍在不断演变，并且在一段时间内仍将处于非标准化状态。因此，开发新战术和控制措施的从业人员需要能够快速集成界面，以支持开发、测试和部署。这可以通过使用快速原型手势识别器进行手势定制来实现（Taranta 等人，2017年）。通过这种方式，我们能够扩展战术库，而无需直接获得界面开发人员的支持，也不会使界面变得杂乱，同时支持一对多的低命令复杂度关系。

3 背景

3.1 将作战需求分解为机器人执行

图2展示了美国国防高级研究计划局 OFFSET计划构想中作战需求与机器人学算法之间的层级关系。这种分解形成了一个多级分类系统，将作战需求映射为操作员、集群、机器人和硬件组件目标。每个层级都对应着独特的视角、世界观和开发方法。尽管我们并未刻意将 RISE 的架构设计成反映这一层级结构，但它仍自然而然地演变为一个由四个组件构成的分布式系统，且这些组件与该层级直接对应：用于将作战需求转化为战术的用户界面、用于战术开发的系统、由原语组成的平台组合，以及用于算法设计与集成的框架。具体而言，自上而下：

- **作战需求**：作战人员根据任务目标制定任务计划，这些计划会随时间演进并驱动集群指挥。因此，必须将任务计划转化为满足任务目标的群体战术。然而，为了根据新信息调整任务计划，必须保持态势感知（SA）。这就要求集群向指挥与控制（C2）团队报告相关信息。第5节将介绍我们用于任务规划、战术执行和态势感知的指挥与控制界面。

- **战术**：Ta　战术涵盖战场上或战场附近兵力的有序部署与机动 -

（LeFavor，2020）。原语以机器人为中心，而本文所述战术以集群为中心，是用于组织和运用智能体以实现任务级目标的软件。“俯视扫描”就是一种战术示例，它将空域划分为多个区域，每个区域都需要进行侦察。随后，俯视扫描可通过下达多个移动至原语，调动多个智能体来完成目标。与原语类似，战术也可以分层构建。“突破”战术可包含持续监视和建筑物进入战术，之后进行探索和安全控制。在 RISE 中，我们通过名为 PyC2 的基于 Python 的软件应用程序（见第 6 节）来支持战术开发。

- **原语**：原语是利用算法实现以机器人为中心的目标的软件组件。一个原语的例子是“移动至”行为，即智能体在避开威胁的同时前往特定战场位置。“移动至”会使用定位、路径规划和避障算法。与此同时，另一个持续运行的原语使智能体能够通过传感器输入分析以及与其他智能体通信来维持态势感知。尽管“移动至”会努力避开威胁，第三个原语也可能识别出迫在眉睫、不可逃避的危险，中断“移动至”并与威胁交战。因此，原语具有组合性和层级性。我们用于原语开发和集成的界面详见第7节。

- **算法**：算法是通过分析传感器输入并驱动执行器输出来与硬件交互的软件组件。算法的示例包括：YOLO 等计算机视觉技术（Redmon、Divvala、Girshick 和 Farhadi，2016），用于识别视频帧数据中的物体；定位技术，用于根据连续的惯性测量单元（IMU）数据进行航位推算（Brossard、Barrau 和 Bonnabel，2020）；以及根据速度指令驱动电机控制系统。单个算法即可实现平台能力；它们也称为技能。我们用于算法开发和集成的界面详见第8节。

3.2 人机群团队

在五次大规模实地实验和若干次中间集成测试过程中，我们将人类团队自行组织成五种不同的角色，分别称为蜂群指挥官、蜂群操作员、群体健康工程师、现场行动官和现场支援人员。每种角色不仅负责成功执行任务的一个方面，也对应着我们发现一个人能够承担而不会不堪重负的责任量。我们设想，随着机器人技术、群体协同和 RISE 持续成熟，这些角色将进一步演变，并可应用于各种组织结构。各角色的具体职责如下：

- **蜂群指挥官**：指挥官通过明确任务目标、将目标转化为任务计划、随着任务推进修改计划，并与指挥与控制团队成员协调来领导指挥与控制团队。指挥官通过与指挥与控制软件交互来保持态势感知，但不会直接指挥集群。由此，蜂群指挥官与指挥与控制软件的交互方式，类似于排或连队指挥官使用该工具的方式。

- **蜂群操作员**：操作员使用指挥与控制软件将任务计划转化为群体战术，从而与集群交互。操作员还负责保持态势感知，向指挥官报告情报信息，并根据不断变化的情况和蜂群能力提出战术建议。操作员还可以请求群体健康工程师提供支持，以便向单个智能体下达命令或调查错误状况。目前，虽然还没有专门针对蜂群操作员的军事职业专业（MOS），但我们预计这一职责将对应多个MOS类别，其中包括15W，即单架无人机操作员。

- **群体健康工程师**：健康工程师负责监控通信网络以及单个智能体的健康状况。工程师是能够诊断并远程解决机器人技术问题的技术人员。在行动环境中，这一角色可由多种工程类军事职业专业人员担任。

- **现场行动官**：现场行动官负责与智能体部署、回收、实体维护、现场人员协调和行动安全相关的后勤工作。行动官与指挥官协调，并可率领一支由现场支援人员组成的小型地面小队。现场行动官这一角色源于实地实验活动的后勤安排、所用集群硬件（例如，电池续航有限），以及大规模系统测试的安全需求。

- **现场支援**：现场支援人员向现场行动官汇报，并提供部署和维护集群所需的后勤保障。鉴于硬件已知的局限性，以及长期持续行动的需求，该团队负责回收车辆并做好重新部署的准备。我们预计，随着技术不断成熟，对这一角色的需求将逐渐减少。

为符合 DARPA OFFSET 计划的目标，我们采用了由低成本地面和空中车辆组成的异构群体。在空中行动中，我们使用了四旋翼平台和垂直起降（VTOL）固定翼飞行器。每种平台类型都具备独特的行动优势，但也各有局限。关于我们采用的平台及其优缺点的更多详情，请参见第11.2.2节。

3.3 实况、虚拟与构造

时间、后勤和成本方面的限制，使得频繁测试单个智能体和大规模集群解决方案难以实现。因此，为了支持快速开发和集成，工程师能够在无法使用实体硬件的情况下迭代软件设计至关重要。仿真有助于实现这一目标，但前提是仿真环境与部署环境保持一致。否则，工程师就会浪费时间来支持多个相关但彼此不兼容的接口。换言之，在仿真中开发的软件应能直接迁移到真实世界，而无需顾虑当前是真实环境还是虚拟环境。因此，我们确保软件与底层现实抽象隔离，使用户界面功能、战术和原语不受底层平台真实属性的影响。我们将在第9节进一步讨论实况、虚拟与构造（LVC）方法。

4 快速集成蜂群生态系统（RISE）

4.1 需求

我们的集群架构设计既基于项目需求，也借鉴了我们作为 OFFSET系统集成商的经验。以下列出一些高级系统需求：

需求 1 能够实现一对多（*150+*）的操作员与智能体比例，以控制群体智能体。

在审查以往系统时，我们发现操作员无法有效命令数量超过少数的单个平台（见第2.5节）。因此，RISE的主要目标之一是让单个操作员能够同时操作数百辆车辆。这就需要采用“群体操作”概念，而非单个智能体行动；在群体操作中，操作员将集群作为一个整体进行指挥，而不是向单个智能体下达命令。

需求 2 能够支持*250*个联网智能体。

作为 DARPA OFFSET 计划的一部分，RISE 必须支持至少 250 个联网智能体同时运行，尽管在任何官方现场实验中都未达到这一数量（参见

第11节，原因是后勤方面（而非技术方面）的限制。该要求促使人们开发先进的网络协议和技术（参见第4.3节）。

需求 3 能够在真实且不确定的环境中运行。

RISE蜂群必须能够在真实的非实验室环境中运行。为了在现场实现智能决策和有效的指挥与控制，需要具备智能体级感知和自主性，以及稳健的组网与通信能力。

需求 4 将战术开发与机器人软件抽象分离。

我们发现，许多参与集群研究的人并非机器人学家，尤其是那些专注于战术开发的人。这些人倾向于将机器人视为能够在环境中自主导航、但只与环境偶尔交互的实体。他们主要关心独立智能体的组织与协调，以实现某项作战目标，例如智能地获取并维持态势感知。换言之，战术工程师通常采用外中心世界观，在高于机器人的抽象层面解决问题。满足战术工程需求的架构将把战术与行为分离，使工程师能够专注于将任务级目标转化为一组机器人指令。因此，RISE必须提供这种抽象，以便这些工程师轻松开发新战术。

需求5 用于支持战术创建的机器人原语与算法开发。

战术开发经过抽象，无需具备机器人学专业知识，但仍需要功能完备的机器人平台。大多数现成平台在出厂时并不具备执行 RISE 中创建的战术所需的能力。这些战术依赖机器人能够在环境中自主导航并与环境进行交互。这些依赖促成了第7节和第8节所述的开发工作。

需求6 用于战术协调的灵活集群指挥接口。

战术的复杂程度和所需监督各不相同。战术工程师或许不关心底层任务管理系统，但系统必须支持分配机器人群组、发起直接和间接指令、同步指令执行、取消未完成任务以及接收指令状态。

Req. 7 用于公开新战术、战术参数和战术控制措施的界面 *s.*

为了促进快速开发，并且由于战术工程师通常不使用最终用户界面软件，工程师必须能够轻松定义新的战术描述、调用方式、相关参数和所需的战术控制措施，并使其立即对操作员可用。

需求 8 支持在真实实验中使用虚拟硬件。

虽然 RISE 软件与硬件无关，但 DARPA OFFSET 等研究项目通常受到资金限制，并使用能力有限的现成平台（见图 25）。因此，并非所有期望的实验场景都能通过实体硬件实现。为此，RISE 必须支持仿真智能体与实体智能体同时运行。

需求 9 用于快速开发的低保真仿真。

虽然高保真仿真有助于机器人行为的开发和测试，但我们发现，战术开发人员通常缺乏运行完整仿真所需的计算资源。因此，RISE 最早提出的需求之一，就是支持可在普通笔记本电脑上运行的低保真仿真。这样一来，第三方合作者便能快速迭代战术开发（见第10节）。

需求 10 用于机器人开发和评估的仿真。

虽然低保真仿真主要用于战术开发，但仍需要一种支持基元和算法开发的仿真。缺少这类仿真，机器人开发就会十分繁琐，而且总是需要进行平台测试。因此，仿真必须同时支持模拟传感器数据流和实际硬件端点。

4.2 RISE架构

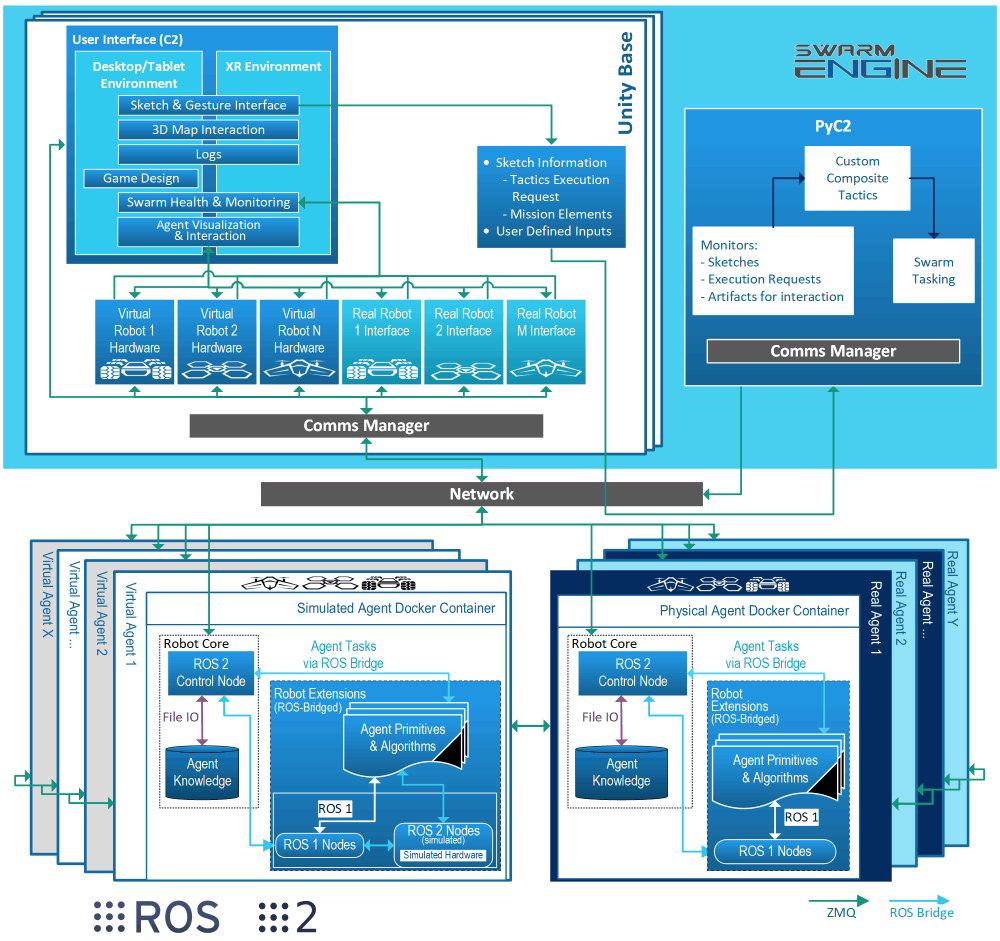


图3： RISE 架构。由四个主要组件组成，分别属于智能体侧计算和指挥与控制计算。

图3所示的RISE架构符合图2中将作战需求分解为机器人执行的方式。其主要组件之一称为Swarm EngineTM，内含指挥与控制解决方案，其中包括C2和PyC2。前者（C2；在图3中称为“Unity Base”）实现了我们的用户界面软件，用于环境可视化、态势感知和基于草图的集群指挥。我们使用Unity 游戏引擎[2]实现C2，因为它具备跨平台兼容性，能够支持高级桌面、虚拟现实和增强现实交互，并内置仿真支持，同时还拥有庞大的活跃社区。PyC2是一款基于Python的软件应用程序，专为快速战术开发而设计。它包含多种工具，旨在帮助开发人员

[2] https://unity.com

编写能够与草图输入交互、查询环境并生成机器人原语命令的战术。PyC2实现了一个任务分配系统，并提供低保真仿真支持，使工程师无需运行机器人软件即可迭代战术设计，符合需求4。我们选择使用Python构建PyC2，因为这门语言易于使用、拥有丰富的库且应用广泛，从而满足需求7。

智能体软件分为“机器人核心”和“机器人扩展”模块，二者均封装在 Docker容器中。这些组件共同满足需求5。机器人核心是一套与平台无关的 ROS/ROS 2节点，负责处理任务竞标、任务执行、运行状况监控、集群通信以及态势推理和报告等关键功能。智能体之间或智能体与集群之间的所有外部通信也都由机器人核心系统处理。这种设计创建了一层抽象，使任何运行机器人核心软件栈的智能体都能成为集群的一员。机器人核心通过智能体各自的机器人扩展代码库与智能体交互。机器人扩展代码库包含所有负责传感器或硬件交互的 ROS节点，以及机载智能体算法（参见第8节）。机器人核心与机器人扩展之间通过标准 ROS 消息传递进行交互，并在适当情况下使用双向 ROS 1至ROS 2桥接器[3] 。

请注意，在架构图（图3）中，列出了两种智能体软件变体——一种代表运行在实体智能体上的软件，另一种代表仿真智能体。二者唯一的区别在于所用的 Docker镜像和 ROS 1传感器数据源。仿真环境与用于指挥真实平台的 Swarm Engine 环境相同，它会生成传感器信息，以取代真实传感器（参见第9.1节）。这种对传感器数据流的独立抽象有助于实现仿真到硬件的迁移，并满足需求10。智能体软件的所有部分均包含在Docker容器中，以便于部署和仿真测试。我们还使用 Ansible[4] ，通过这一基于 Docker 的系统进行集群管理和部署（参见第11.2.2节）。

4.2.1 ROS

ROS [5] 是 RISE架构 的关键组成部分。开发过程中，所有内容都以 ROS 为基础进行设计，以便于集成和开发。智能体软件由 ROS 和 ROS 2 节点混合构成。由于开发时 ROS 2 缺乏硬件驱动支持，机器人核心模块使用 ROS 2，而机器人扩展模块目前使用 ROS；待 ROS 2 中提供所有必需的驱动程序后，机器人扩展模块可能会在之后升级到 ROS 2。所有智能体之间及智能体内部的消息，均通过标准 ROS 网络中间件使用原生 ROS 1/2 消息传输，或通过自定义 ZeroMQ中间件使用 ROS 2消息传输（见第4.3节）。Swarm Engine 的组件与 Unity环境 的不同实例之间的所有通信均使用 ROS 消息传递。仿真环境会广播 ROS传感器消息，以便与智能体软件集成，用于开发和测试。所有算法和原语都通过 ROS 消息传递进行通信，该机制贯穿系统各个层级，从战术任务分配到智能体级执行控制。每个智能体上的传感器与执行器交互均由 ROS 的 MAVLink/MAVROS 软件包处理。在 RISE 的开发过程中，使用过多个版本的 ROS 和 ROS 2，但目前该项目使用 ROS Melodic 和 ROS 2 Eloquent。

4.3 集群组网

本节介绍用于实现智能体之间以及智能体与操作员之间通信的网络。我们首先概述早期原型及其面临的挑战，然后介绍最终的拓扑和协议。

[3]https://index.ros.org/p/ros1 bridge/ [4] https://www.ansible.com/ [5] https://ros.org/

4.3.1 初始原型

在 RISE 开发的早期阶段，群体网络采用了极为简单的现成机制。受组件供应情况影响，RISE 最初使用标准的 802.11 Wi-Fi，所有智能体都连接到集中式接入点。然而，实地实验（见第11节）很快就暴露了这一方案的弊端。任务执行开始前，操作员必须预先部署通信基础设施。这一过程十分困难，需要仔细进行现场勘测，以确保覆盖范围最大化，并在选定的接入点位置建立电源供应和网络回传链路。显然，这种准备工作被认为不适用于现实世界中的战术演练。此外，即使进行了现场勘测并谨慎布置接入点，在实地实验中使用的混凝土建筑密集的城市环境（见第11节）仍会形成大片“盲区”，导致智能体无法与预先安装的接入点通信。

为缓解这些问题，并在获得额外资金后，我们评估了商用现成组网解决方案，并从 Rajant Corporation 购买了用于移动自组织网络（MANET）的设备。有关此方法优势的更多信息，请参阅第4.3.2节。

然而，改用 MANET 架构后，我们又遇到了更多问题。起初，我们一直使用 ROS 2 默认的消息传递机制进行智能体间通信，但发现使用 MANET 时，该机制无法很好地扩展到几十个以上的智能体。ROS 2 消息传递基于数据分发服务（DDS）。由于 MANET 使用共享介质（2.4 GHz 和 5 GHz 无线电频段），大量无线电设备同时尝试发送数据，造成严重的数据丢失和重传，使网络实际上无法使用。我们与 DDS供应商的工程师进行了大量合作，试图解决这一问题，但仍无法让数百个智能体同时运行。于是，我们开始实施自有的智能体间通信协议解决方案。更多信息请参阅第4.3.4节。

4.3.2 移动自组网

部署多智能体机器人系统的一项关键挑战是设计稳健的通信网络。最初我们考虑了多种架构，最终采用移动自组织网络（MANET），通常称为网状网络，以实现智能体之间的所有通信，以及与C2操作员之间的所有非紧急通信。

我们出于多种原因选择了MANET架构，具体如下：

- 城市环境通常包含材料各异、几何结构复杂的建筑物，这些建筑物会阻碍无线电通信，使基于接入点的网络无法与建筑物内部或建筑物之间的智能体有效通信。我们发现，在实地测试所用的环境中，这一点尤为明显（见第11节）。MANET通过允许附近的其他智能体充当网络中继来缓解这一问题，而RISE的战术正是利用了这一能力。例如，请参见SoarTech在第10.3节开发的建筑物清剿能力。

- 通过使用移动自组网（MANET），我们可以不必预先在作战区域内部署网络设备；而对于围绕中心接入点设计的网络，这项任务则是必需的。
- 移动自组网（MANET）架构消除了系统中的任何单点故障，即使一个或多个地面无线电设备发生故障，智能体仍可继续运行。智能体可以直接与网络中的任何其他设备通信，而无需通过任何集中式基础设施路由消息。

- 移动自组网架构便于集群部署；在运行大规模集群系统时，这是一项至关重要的能力。智能体无需针对不同的部署策略重新配置，



图 4：左侧展示了安装在地面车辆上的 Rajant ES1 无线电。空中车辆的底部安装了 Rajant DX2 无线电。右侧展示了小规模实地测试期间，网状无线电如何在 IFO 平台之间进行通信，Rajant 网络管理软件对此进行了呈现。

以及 th 地面基础设施可以随时更改，而无需重新配置网络 n.

基于这些原因，我们选择了 Rajant Corporation 的商用现成 MANET 解决方案，并在系统中采用了其产品线中的多种无线电。空中飞行器配备 Rajant DX2 或 DX4 系列无线电，地面车辆则配备 ES1。ME4 和 Peregrine 系列无线电用于地面／基站通信，将操作员以太网连接到 MANET。这些无线电的安装方式如图 4 所示。所有无线电均在 2.4 GHz 频段运行。除 DX2 外，所有无线电还使用 5 GHz 频段上的第二个信道。Rajant 软件负责管理这些频段的使用，以便在近距离有多台无线电时实现高效传输。

4.3.3 网络概述

虽然上述移动自组网对于智能体通信至关重要，但它无法处理 LVC 传感器仿真所需的海量数据（见第 9 节）。因此，我们增设了一个 1000BASE-T 以太网，并通过交换机将其连接到移动自组网。该以太网为多台 LVC 仿真计算机和各种 C2 计算机提供服务，并支持与第三方地面系统连接（见第 10 节）。整体网络如图 5 所示。

4.3.4 网络与传输协议

最初，我们尝试使用 ROS 2 内置的网络协议栈，该协议栈基于数据分发服务（DDS）。然而，我们很快发现，由于 DDS 的发现阶段开销过大，它不适用于我们的大规模移动自组网（对象管理组，2015年）。因此，我们改用 ZeroMQ（ZeroMQ，2021年）作为套接字库，实现了自定义方案，以支持所有联网系统之间的可靠多播通信。

我们采用互联网协议（IP）（互联网协议，1981）作为集群网络的基础。每个智能体都有一个 IP 地址，并通过互联网组管理协议（IGMP）加入一个或多个 IP 多播组。在多播中，设备只需发送一条消息，该消息发往一个组，组内所有成员均可接收（Deering，1989）。这对我们的架构很有优势，因为它减少了车辆无线电传输消息所需的空中传输时间。对于每个需要接收消息的智能体（通常为 200+）分别发送一条单播消息，所需的空中传输时间远多于发送一条多播消息。在附近可能有数百台无线电设备的集群中，这是一个至关重要的考量。我们利用 Rajant 的“Tactical Multicast”功能，防止多播消息在网络中“回响”。这一功能使多播消息能够传送到所有车辆，同时最大限度地减少在移动自组网中的重传（Acker、Hellhake、Jordan 和 Parks，2016）。

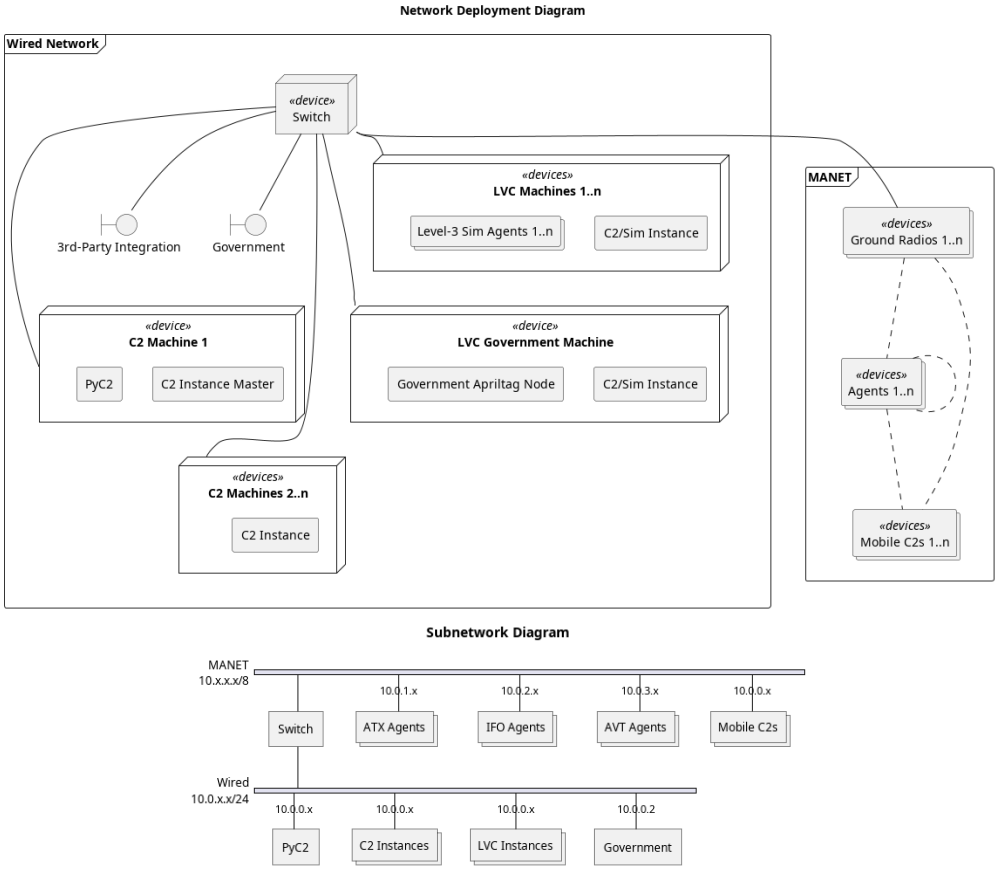


图 5： 上方为整体 RISE网络的部署图。标有“1..n”后缀的项目数量可以任意。下方为 RISE网络的子网图。在这两张图中，“政府”指政府实验基础设施（参见第11节）。

我们将消息分为两种传输类别：可靠和不可靠。两者都使用 IP 网络协议，但采用不同的传输协议。对于不可靠消息，我们使用标准用户数据报协议（UDP）数据报来传输消息。该协议可确保消息正确，但不保证消息送达，也不保证接收顺序（用户数据报协议，1980）。这种协议非常适用于密集发射器环境，因为它能最大限度地减少传输次数，且不会因尝试重传不重要的数据而浪费空中传输时间。

对于需要确保送达的消息，我们使用实用通用多播（PGM）协议。PGM 使用否定确认，使丢失或损坏的消息能够重传。也就是说，每个设备都会维护一个序列号，并定期通过心跳消息（SPM数据包）将其发送给其他设备。这样，接收设备便能检测到消息丢失，并通过否定确认（NAK）请求重传这些消息（Speakman等人，2001年）。我们调整了若干 PGM参数的默认值，以便更高效地利用有限的网络资源。有关这些参数调整的信息，请参见附录B。

4.3.5 应用协议

由于系统中的所有智能体都使用 ROS 2 实现高级逻辑和控制，因此将网络协议栈与 ROS 2 集成至关重要。为此，我们创建了一个“通信处理器”ROS 2 节点和软件包

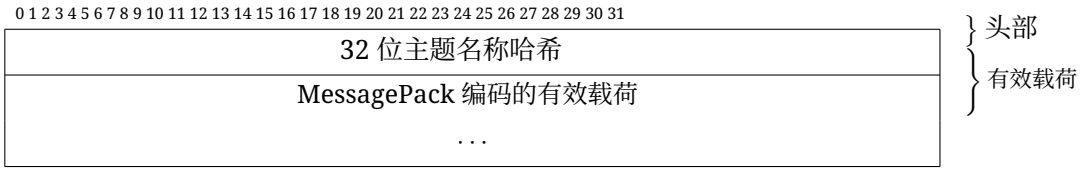


图 6：网络应用协议。

。该节点接受用户指定的 ROS 2 主题列表（参见第 4.2.1 节），用于向网络发送或从网络接收数据。应用协议的设计力求简单明了、易于实现。我们首先使用 DJB2 哈希算法（Bernstein，1991）将主题名称编码为一个 32 位值。然后，使用 MessagePack 库采用 MessagePack 高效的编码方案对消息的每个字段进行编码（Furuhashi，2021）。ROS 2 消息的每个字段都会自动解码，然后编码为 MessagePack 字段，整条消息则被打包到图 6 所示的结构中。随后，使用 ZeroMQ 库通过上述传输机制之一发送该消息。

接收消息时，过程则相反。ZeroMQ 接收消息后，会将主题哈希与已知主题字典进行比对（如果主题未知，则丢弃该消息），随后使用 MessagePack 解码每个字段，并将其重新编码为 ROS 2 消息。然后，通过标准的 ROS 2 发布机制发布该 ROS 2 消息。

支持 ROS 2 发布、订阅、服务和服务客户端，并可在两个方向上自动转换消息。有关集群中使用的主题的更多信息，请参阅第4.4节。ROS 2 项目最初曾研究使用 ZeroMQ，但为了便于开发，最终选择了 DDS，代价是消息传递定制能力有所降低（Woodall，2014）。我们已实现基于 ZeroMQ 的 ROS 2 消息传递，并获得了其中许多性能和定制方面的优势。虽然我们牺牲了 ROS 2/DDS 的部分灵活性（例如自动主题发现），但这些改动使系统能够在我们的 $200+$ 车辆网状网络所施加的严格约束下运行。

4.3.6 EmergencyControl 控制通道

我们还在 2.4 GHz 频段维护一个紧急带外控制通道。蜂群操作员可以通过该通道命令无人机紧急停止，这将立即使所有无人机着陆并解除武装。

4.4 集群 API

如第 4.2.1 节所述，RISE 全程采用 ROS，ROS 是通用 API 接口。与任何基于 ROS 的系统一样，节点之间的接口由 ROS 消息、发布者和订阅者明确定义。集群中的智能体主要通过我们定义的三种 ROS 2 消息进行交互：Heartbeat、Cmd 和 Job 消息。图 7 展示了群体智能体和 Swarm Engine 所使用的这三种主要消息的内容。集群中的单个智能体必须填充 Heartbeat 消息的各个元素，并在 /a2c/heartbeat 主题上重复发送这些信息。当智能体收到任务分配或发生其他情况时，必须更新消息中的相应字段。所有 Swarm Engine 实体都会监听该消息主题，以自动发现群体智能体并将其添加到智能体表中。同样，Swarm Engine 实例会在 /heartbeat 主题上广播相同的 Heartbeat 消息信息，使群体智能体能够获知所有在线的 Swarm Engine 实例，同时其他 Swarm Engine 实例也能彼此发现。

Job 和 Cmd 消息用于构成群体任务分配的竞标过程。图8展示了如何通过战术执行生成任务，并利用这些任务促使群体智能体

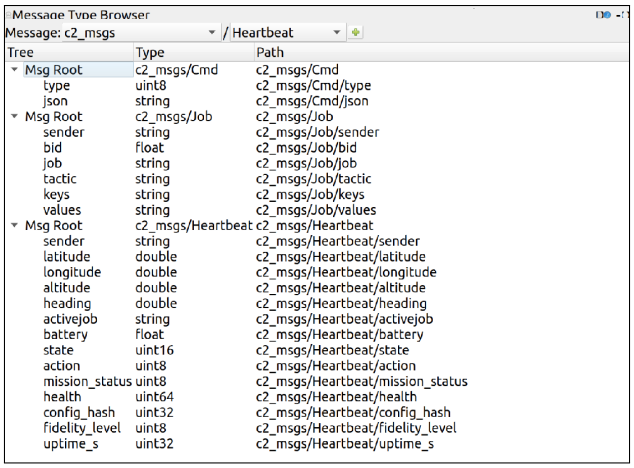


图7：RQT 消息视图，显示一些主要的应用程序接口消息：Heartbeat、Cmd 和 Job

通过 Job 消息和竞标主题进行竞标。根据竞标响应完成任务评估后，会向集群发送一条命令消息，指示智能体完成指定的战术。此交互通过 Cmd 消息实现，智能体订阅 /command主题，以确定自己是否已被分配任务。如果智能体已被分配任务，逻辑流便进入架构中的智能体原语部分。否则，在任务执行期间，随着其他战术陆续执行，智能体会继续响应 /bidding主题上的任务请求。图9展示了集群运行情况的时序图。

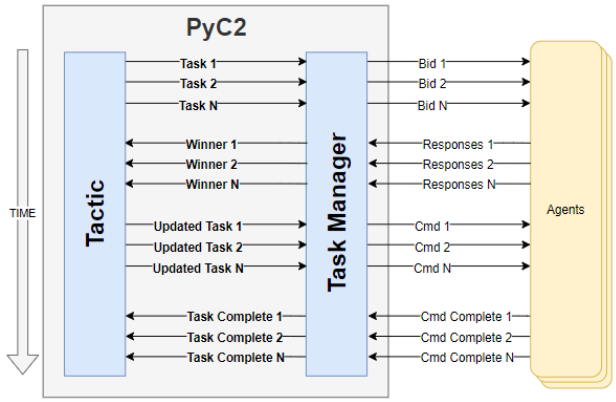


图8：竞标命令流程。

4.4.1 命令应用程序接口

RISE 中智能体的任务分配通过集中式竞标系统来完成。尽管去中心化任务分配方面的研究相当普遍（L. Johnson、Ponda、Choi 和 How，2010），而且 RISE 甚至已与实施了去中心化任务分配的第三方系统集成（见第10.1节），但我们发现，集中式任务分配是满足 RISE 需求的理想方案。首先，集中式任务分配能够对集群实施更高程度的控制。在集中式系统中，智能体何时接受任务分配以及预期由哪些智能体执行，都十分明确。集中式任务分配的另一个主要原因是网络负载。鉴于我们的应用场景中智能体数量众多，我们需要尽可能减少网络流量，而集中式系统正好可以做到这一点。

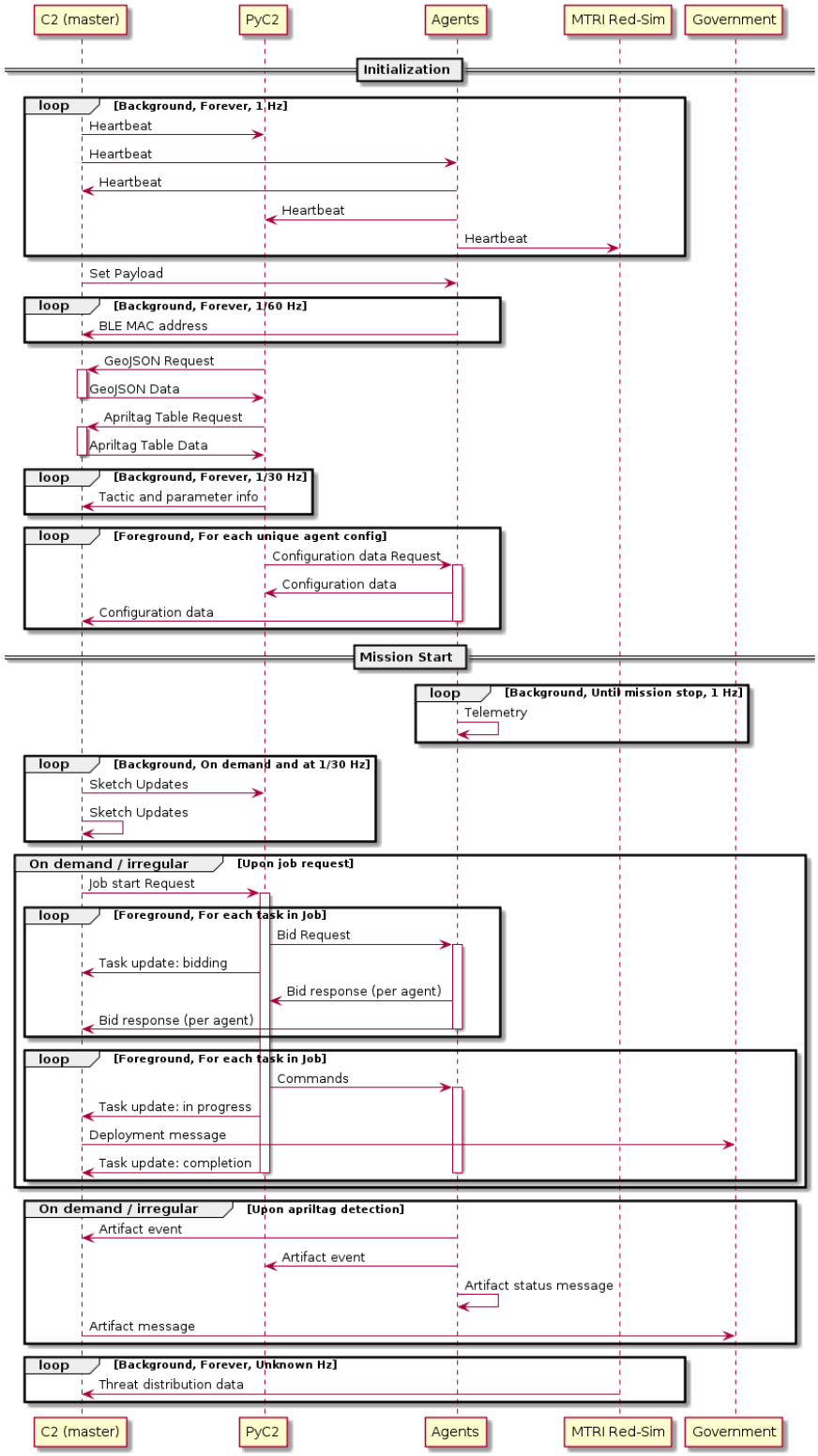


图 9: 展示群体操作期间通过网络发送消息的时序图 .

竞价系统通过使用第4.4节所述的作业消息和命令消息来运行。蜂群操作员提供战术输入后，系统会创建一系列任务或“作业”。随后，这些作业会通过 /bidding主题发送出去，智能体则以基于启发式规则的竞价作出响应。智能体启发式规则会在回复竞价前，考量所请求的作业、作业相对于自身的位置、电池续航、车辆健康状况以及其他一些因素。PyC2 随后会评估收到的所有作业响应，并最终为相应战术中的所有作业分配任务。最终的任务分配通过 /command主题和消息发送出去。图8展示了这一完整流程随时间推移的标准交互过程。多个战术和作业可以同时参与竞价。

4.4.2 情报API

集群中的所有智能体始终都在搜索情报，无论它们正在执行什么任务。这些情报主要通过各平台的 RGB摄像头获取。在 RISE 期间曾使用 YOLO 或其他目标检测变体；而在 DARPA OFFSET 现场实验中，情报主要由 AprilTag标签表示 [6] （见第11.3.1节）。智能体使用 AprilTag检测算法，通过一条通用检测消息向集群报告所见内容。该消息包含诸如哪个智能体发现了物体、该物体在当前上下文中代表什么、物体位于何处等信息。所有智能体和集群引擎实例都会监听彼此报告的所有检测结果。智能体在报告检测结果时，会将这些事件记录到机器人核心的另一个子组件——中央数据存储中。当某个原语执行需要任何已有或动态情报时，特定原语会查询这些存储的信息。第5.5.1节讨论了 Swarm Engine 如何以可视化方式呈现群体智能体报告的情报。

5 RISE：用户界面

我们构建 C2用户界面时，兼顾了集群指挥官、操作员和健康工程师的需求。集群指挥官主要关注态势感知（SA），其中一个方面是“对一定时空范围内各要素的感知”（Endsley，1995年）。相关要素包括蓝方和红方兵力位置、地形布局以及任务计划数据。尽管操作员同样必须保持态势感知，但他们还负责集群指挥——发布能够执行指挥官计划的战术。鉴于机器人集群会与环境互动，战术必然植根于时空背景之中。因此，态势感知和战术调用的性质意味着，指挥官和操作员主要通过环境与集群进行通信。另一方面，集群健康工程师以及研究人员、从业人员和开发人员，都需要访问指挥官和操作员不一定需要的详细信息。智能体任务状态、日志、组件级健康信息和轨迹信息，都是工程师用于诊断任务期间出现的问题的示例数据。尽管健康工程师会利用态势感知并可能协助集群指挥，但能够快速定位并与特定智能体交互，对他们的工作成效至关重要。图10说明了我们如何平衡这些需求。

我们提出的跨平台兼容界面采用迭代式设计方法，历经四年不断完善。我们参与了五个联合兵种集体训练设施（CACTF）的五次大规模实地实验，并定期进行现场集成测试和常规开发，这些经验为设计提供了参考。如图所示，界面分为交互面板（左侧）和场景视图（右侧）。鉴于环境是我们实现态势感知和集群指挥的主要通信渠道，我们为场景视图分配了大量屏幕空间。我们还摒弃了传统的窗口、图标、菜单和指针（WIMP）设计模式，以进一步最大限度地利用带宽。相反，我们采用了一种上下文敏感的基于手势的用户界面，用户通过草图命令与指挥与控制系统交互。这一选择也符合我们对跨平台能力的需求，因为这样可以保留移动设备有限的屏幕空间。在本节余下部分，我们将进一步阐述这些理念，并详细介绍我们的界面。

[6] https://github.com/AprilRobotics/AprilTag



图10： 蜂群操作员在 DARPA OFFSET 第四次现场实验期间，于刘易斯-麦科德联合基地绘制的任务计划草图。

5.1 跨平台兼容性

我们通过实现仅使用二维输入的技术，设计出跨平台兼容的指挥与控制系统；我们认为，二维输入是所有输入设备的最小公分母。鼠标、触控笔和触控输入本质上都是二维的。三维控制器和手部姿态数据则可通过平面投影转换为二维。因此，在一个系统上学会的基础指挥与控制操作技巧可以迁移到所有受支持的系统上。截至目前，我们已将软件移植到 Windows、Linux、Android、HoloLens 2 和 HTC Vive。不过需要指出的是，虽然我们实现了通用的跨平台界面，但仍能利用功能更强大的系统所提供的可供性。有关 RISE 支持的其他人机交互界面的更多信息，请参阅（Williamson、Taranta II、Moolenaar 和 Laviola Jr.，2023年）。

5.2 手势界面

有意传递信息的人体运动称为手势。在本文的上下文中，手势是由输入设备捕获的运动模式，可映射为软件命令——当我们识别出已知模式时，就会调用相应的功能。为识别输入模式，我们采用 Jackknife（Taranta 等人，2017年），这是一种设备无关的自定义手势识别器。

设备无关意味着我们能够使用同一个识别器识别鼠标、触控、触控笔、手部和 3D 控制器等模态的手势输入，从而快速集成新的输入模态。可定制意味着识别器能够从少量示例输入模式中学习。Jackknife 只需加载一个训练样本便能达到较高的准确率（$> 90\%$），并且会随着训练数据的增加而提升[7]。由于我们只需要极少量的训练数据，用户可以根据自己的偏好定制界面，这有望提高界面的易学性和易记性（Nacenta、Kamber、Qiang 和 Kristensson，2013年）。此外，由于 Jackknife 采用最近邻模式匹配策略，我们可以实时在线训练识别器，从而按需定义新的手势类别。当 RISE 实现一项新功能，但其对应的调用手势尚未确定时，我们的界面会提示用户提供训练数据；随后，识别器便会重新训练，新功能也随即可以使用。这

[7]具体而言，在二维手势方面，Jackknife 在加载两个模板的书写者无关识别场景中，准确率可达 95%。

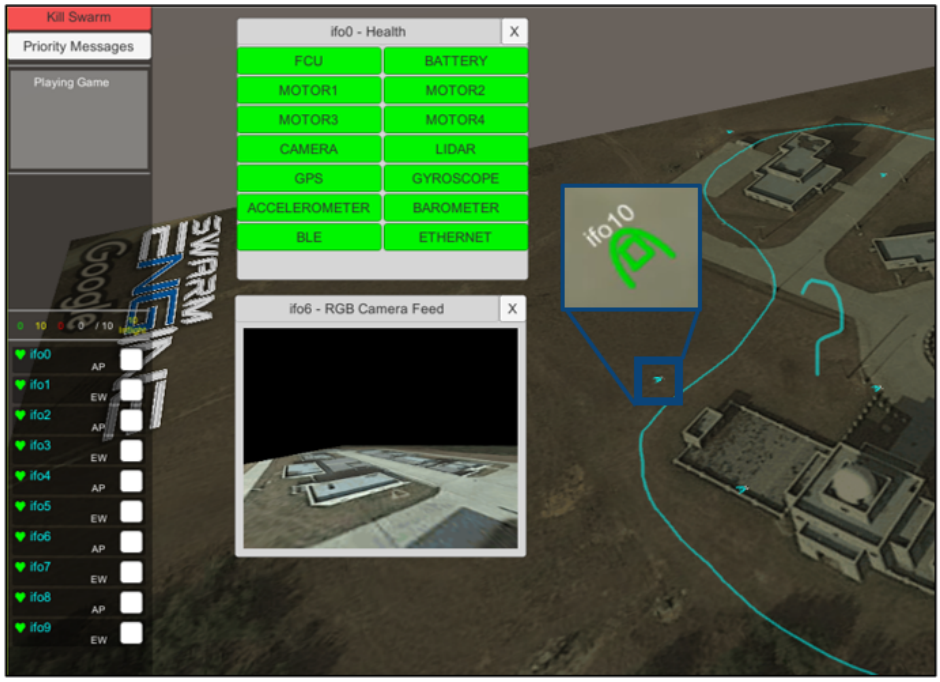


图11： 用户界面元素，包括智能体列表（左侧）、视频流、组件健康面板，以及地图视图仿真中的智能体图标。

使 PyC2 能够在没有直接 C2 支持的情况下采用新的战术和草图参数类型。

所有界面功能均通过手势命令访问，包括用于启动和停止任务的系统命令、用于访问日志和视频馈送的智能体命令，以及战术和战术参数命令。除了节省屏幕空间之外，与窗口、图标、菜单和指针设计模式相比，采用手势的另一项优势是，C2和PyC2开发人员可以添加新功能，而无需重新组织菜单层级或工具栏，从而加快开发进度。更重要的是，手势可以减少模式切换，并允许用户交错执行命令，从而有可能提高命令吞吐量。

为了帮助蜂群操作员学习用户界面，我们开发了一个交互式培训模块，引导用户熟悉基本的智能体、指挥与控制及导航交互。此外，用户可通过在线帮助系统查看所有可用的手势命令；该系统会演示每个符号的绘制方法，并说明其用途。

5.3 智能体界面

与指挥官或操作员相比，集群健康工程师更关注单个智能体。因此，我们通过左侧面板（如图11所示）提供对重要智能体交互的访问。在此面板中，我们会列出所有已知智能体——即至少收到过一条心跳消息的智能体。每个列表条目都会显示智能体名称、有效载荷、状态和战术图标[8]。智能体名称的颜色表示其状态，可能为空闲（绿色）、已击毁（橙色）、手动禁用（红色）、已分配任务（蓝色）或未知（橙色）。后一种状态表示C2已失去与该智能体的通信[9] 。

用户可以通过在智能体按钮上做手势来访问智能体专属数据，最初的触点用于选择智能体。向右滑动会使指挥与控制将摄像机定位到场景视图中的智能体上。h

[8]尽管智能体会执行原语，但我们能够将其工作追溯到生成该原语的战术。[9]如果指挥与控制在七秒内未收到 Heartbeat，通信就会中断

手势会打开智能体的健康信息窗口，其中按颜色列出所有相关硬件组件的状态：正常（绿色）、缺失（灰色）、中间状态（黄色）或严重（红色）。分别通过h和$)$查看智能体的彩色摄像机流和深度相机流。最后，L会显示智能体的日志。虽然我们可以提供其他功能特性，但发现这些功能对于初步了解智能体态势感知和进行诊断最为相关。集群健康工程师还可以通过 SSH 远程终端访问获取更多信息。

我们在场景视图中使用平台专用符号将每个智能体显示为一个图标。图标颜色采用上述相同的方案，反映智能体的状态。在智能体图标中，我们绘制一个战术符号，以表示该智能体正在执行或曾经执行过的工作。在早期的一个用户界面版本中，我们曾改用 3D 模型来表示智能体。然而，除了更具美感之外，这些模型并无明显优势，因此我们选择了资源消耗较低的方案。

5.4 指挥与控制（C2）

RISE 中集群指挥的核心在于指定战术指令、其输入以及执行顺序。正如下文所述，我们通过基于草图的界面将这三个要素整合在一起。

5.4.1 战术调用

要调用特定战术，操作员必须在环境中绘制与之关联的手势。识别出该手势后，C2 会在笔画的质心处插入一个交互式战术图标。随后，操作员可以点击该战术，打开对应的弹出窗口。此窗口包含一个输入参数列表和两个交互按钮：一个会立即将战术发送给 PyC2 进行后续处理，另一个则只关闭窗口，以便进行任务规划。表1展示了五种战术示例，以及各自的调用手势和所需输入参数。“上下文”参数用于指定战术将作用于哪种草图输入参数类型。

战术图标在环境中的位置很重要，因为战术可以选择使用位置信息来推断操作员的意图。例如，我们的建筑扫描战术会推断操作员希望扫描距离图标最近的建筑。战术也会根据草图参数输入做出类似推断。例如，我们的俯瞰扫描战术会处理距离最近的探索区域草图（这是一种下文将讨论的输入参数类型）。因此，操作员可以在环境中拖动战术图标，以精确指定位置。

调用战术后，其状态会通过图标颜色表示，分别为：待处理（黑色）、进行中（蓝色）、失败（红色）或成功完成（绿色）。我们还在左侧面板中添加了一个战术按钮，以便快速列出正在进行的工作，并提供访问其他开发人员数据和集群健康工程师数据的途径。在适用情况下，这包括访问子战术和原语数据，以及智能体航路点列表。随着列表变长，将特定战术按钮与正在进行的现场工作对应起来会变得困难。不过，借助空间记忆，用户也可以随时返回场景视图中的战术图标，通过弹出窗口访问相同的信息。若要取消某个战术及其子战术和原语，只需涂抹擦除该图标即可。

5.4.2 战术链式组合

操作员很少只执行单一战术。相反，他们会按照特定顺序部署多种战术，以推进特定任务目标。因此，我们通过战术链式组合支持多种战术之间的协调，这是一项重要的功能特性。如图12所示，操作员可以通过草图手势将两个战术图标连接起来。多个连接组合后会形成一个有向无环图。操作员发出根节点战术后，指挥与控制会将整个图结构提交给 PyC2 进行进一步处理。

战术名称	手势	参数	Type	描述
检查物体		上下文 半径	POI 浮点数	使用无人机扫描感兴趣的目标。 目标周围球体的半径。
沿路线飞行		上下文 高度 距离 使用链式处理	Path 浮点数 浮点数 bool	请求智能体沿最近的路径移动。 无人机将保持的高度（米）。 点之间的距离。设为零以强制简化。 每次只发出一个点，仅用于测试。
保持位置		上下文 高度 持续时间 智能体数量	Path 浮点数 浮点数 int	将一组智能体移动到周界上的各个点并保持位置。 无人机将保持的高度，单位为米。 保持时长。 沿周界部署的智能体数量。
俯视扫描		上下文 高度 单元格大小 智能体数量	探索 浮点数 浮点数 int	派遣无人机飞越区域，寻找人工制品。 无人机将采用的高度（单位：米）。 航路点之间的最小直线距离，单位为米。 用于扫描区域的智能体数量。
安全着陆		上下文	None	对于给定的飞行器，寻找附近可安全着陆的位置。

表1： 操作员可调用以完成任务目标的部分 PyC2 战术。操作员在环境中绘制战术手势，以调用相应命令。每种战术需要零个或多个参数，其中上下文是该战术执行所依据的草图输入类型。操作员可在提交命令至 PyC2 之前，通过弹出窗口修改其余参数。请注意，此 LaTeX 表格由 PyC2 的自动化文档系统生成。

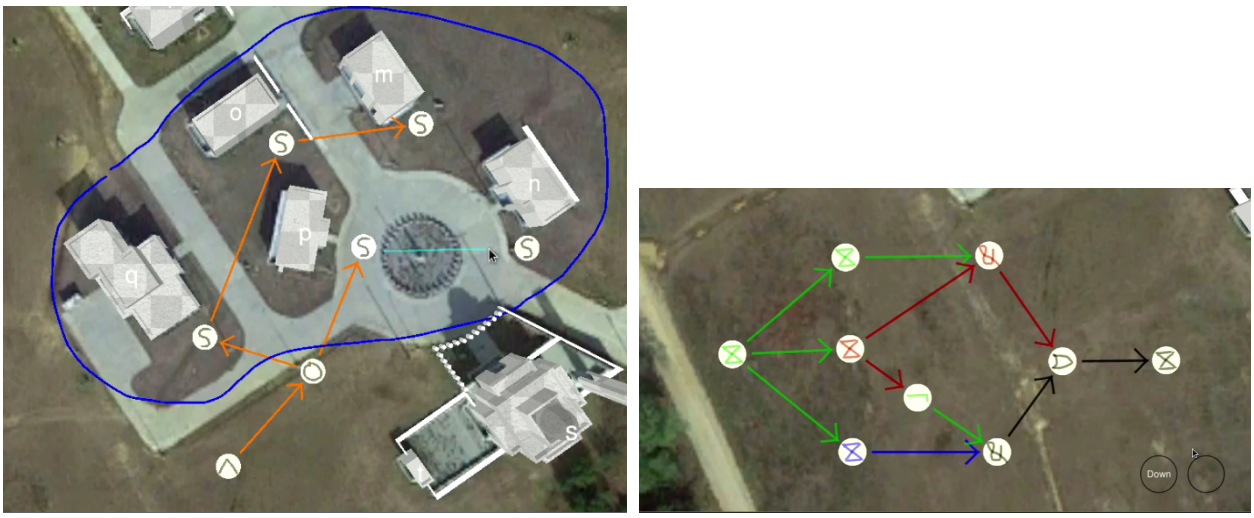
默认情况下，只有在父战术成功完成后，子战术才会执行。换言之，父节点状态会传播到子节点；如果任何父战术失败，相应的子战术也会失败。不过，可以通过使用逻辑门战术节点来修改这种传播行为。

逻辑门战术节点在实现方式上与标准战术相同。区别之一是，它们会覆盖默认的传播行为（参见第 6 节）。PyC2 目前实现了否定、合取和析取。否定门会反转其父战术的状态，这对于规划应急方案很有用。析取门只要任一父战术成功，就会输出成功响应；而合取门则要求所有父战术都成功。逻辑门战术节点使操作员能够以草图输入的形式表达复杂的任务计划。

5.4.3 战术参数

战术需要输入数据来指定调用时的精确行为，例如，持续监视的盘旋高度或建筑物安全距离。大多数参数通常通过经验确定或借助自动化处理，但在开发过程中或特殊情况下，操作员可能需要指定其他值。因此，操作员可以通过弹出窗口界面修改某些输入数据。我们发现，适用于弹出窗口的数据类型包括数值、文本和布尔输入。不过，其他数据类型更适合通过基于草图的交互来描述。在 DARPA OFFSET 计划的整个过程中，我们开发并集成了众多原语和战术，并发现始终只需要三种草图交互类型：选择组、点和折线。

- **选择组**：有时，尤其是对于开发人员和集群健康工程师而言，必须能够选择单个智能体或一组智能体，以便向其分配后续命令。为支持此操作，我们实现了套索选择技术，操作员可以在希望编组的对象周围画出笔画。操作员可以持续使用套索选择对象，直至满意为止。每个新笔画圈选的对象都会与先前



手势图例

合取	创建智能体	析取	否定
俯视扫描	扫描建筑物	计时器	

图12： 战术链式组合。左：操作员在两个节点之间绘制连接，形成战术链中的一条连接。右：包含类似逻辑门节点的战术链示例。战术图标和连接采用颜色编码：蓝色表示战术正在进行，黑色表示等待执行，绿色表示成功完成，红色表示失败。

的组按照析取并集逻辑合并，从而允许取消选择。此后，操作员发出的任何战术都将仅应用于最终选择组中的智能体。如果指定的战术生成子战术，这些子战术也将遵循相同的限制。

- **点**：点是地理空间位置标记，用于指定环境中的某个位置，例如兴趣点、会合点和突破点。点与手势相关联，因此绘制手势时，相应的点会出现在场景视图中该手势的质心处。随后可将点拖动到三维空间中的精确位置；通过自定义设置，还可以锁定坐标轴以减少误差。例如，通常会将点锁定在地平面上。

- **折线**：折线是由一系列按顺序排列的地理空间位置定义的曲线，可以是开放的，也可以闭合形成环路。草图笔画是操作员输入折线数据的方式，因为笔画是一种自然、流畅的通信形式。因此，可以使用折线指定轨迹，包括首选路线和边界（禁入区、探索区域和部署区域）。操作员在场景视图中输入手势，以选择折线类型。随后，操作员绘制笔画来定义初始折线，之后可直接在折线上绘制草图进行修改。修改笔画如同磁性工具，会将折线拉向光标方向。如果折线类型为闭合环路，则会移除自环和交叉点。图10展示了路线和边界示例。

为了帮助感知和理解，战术开发人员可以自定义点和折线类型，并为其设置独特的手势调用和渲染属性。更多信息请参阅第 6 节。

5.5 态势感知

指挥与控制提供了多种工具，帮助感知和理解集群智能。集群指挥官可以利用这些信息估算未来状态并决定下一步行动。

5.5.1 人工制品

我们将人工制品定义为感兴趣的物体。指挥与控制使用如图13所示的军事图标，将集群检测到的相关人工制品进行可视化。我们使用标准化图标来帮助理解，因为这些图标应用广泛，并且能够传达基本信息。系统支持两个DCRI级别（Self、Miller 和 Dixon，2005年）：指挥与控制使用通用类别图标呈现已识别类别的人工制品，并使用具体图标呈现已确定身份的人工制品，例如，人员与非战斗人员、医护人员或敌对人员。为获取更多信息，操作员可以查看某个智能体的视频流，或命令集群监视该人工制品。对于构成威胁的人工制品，我们还会在其周围呈现一个红色球体。威胁解除后，我们会移除该球体。

5.5.2 情报信息

作为战术视图和智能体列表视图的替代方案，指挥官可以选择使用我们的情报消息视图。在此模式下，集群智能消息会列在左侧信息面板中。未读消息按优先级排列，并要求指挥官进行反馈，以确保这些消息已被阅读。引用场景视图数据的消息包含超链接，点击后会将摄像机移动到相关位置。例如，“在10号建筑内发现高价值目标”中嵌入了一个可点击链接，点击后会将摄像机移动到该建筑上方。

5.5.3 建筑可视化

指挥与控制会将集群收集到的建筑状态信息所体现的集群智能可视化，如图14所示。具体而言，我们以棋盘格图案呈现先验已知的建筑几何形状。这表示集群预期此处有一栋建筑，但其状态尚未确认。一旦集群核实其状态，我们便以实心显示该建筑。此外，藏有威胁或情报的建筑会分别以明暗交替的红色或蓝色着色；快速浏览场景视图，即可迅速发现可能需要额外关注的区域。

5.5.4 网格数据可视化

集群可能会跟踪适合用网格叠加层表示的数据。因此，我们提供了一种机制，使集群实体（包括智能体或第三方程序在内的任何联网组件）能够向指挥与控制传输自定义网格数据。图15展示了我们与密歇根理工研究所合作过程中产生的两个示例。左侧的网格可视化呈现威胁概率分布，概率越高（红色越亮），对应的威胁越大。起初，均匀分布会随着集群收集情报并孤立红方力量，逐渐演变为多模态分布。右侧的网格可视化展示了一幅声学地图。如果从操作员光标下方投射欺骗性交火声，便会在该区域听到。由此可见，网格数据可视化在多种上下文中都很有用；例如，集群可以报告网络通信范围等数据，而不是可视化声学区域数据。

5.5.5 时间覆盖

为了可视化随时间变化的覆盖范围，RISE C2 会跟踪智能体在环境中的移动。具体而言，对于每个智能体，在每一帧中，我们都会从每台摄像机向虚拟环境中投射射线。由此得到的命中点集合可估算该帧的覆盖范围。随后，所有命中点都会

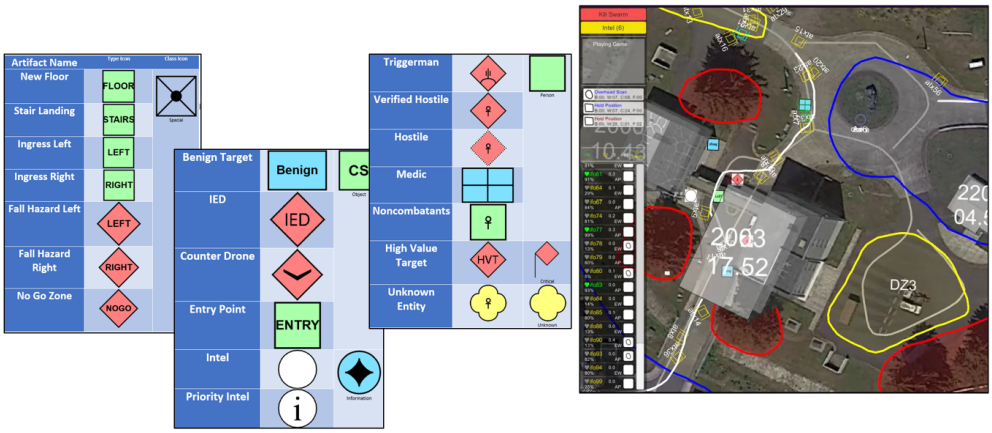


图13：DARPA OFFSET 刘易斯-麦科德联合基地现场实验中用于传达物体信息的图标示例。



图14：建筑物按照群集收集的情报信息进行颜色编码。当建筑物已通过先验知识确知存在时，我们以棋盘格图案显示；当集群发现该建筑物后，则将其显示为纯色。已知内部存在威胁的建筑物以红色阴影表示，已知包含情报信息的建筑物则以蓝色表示。

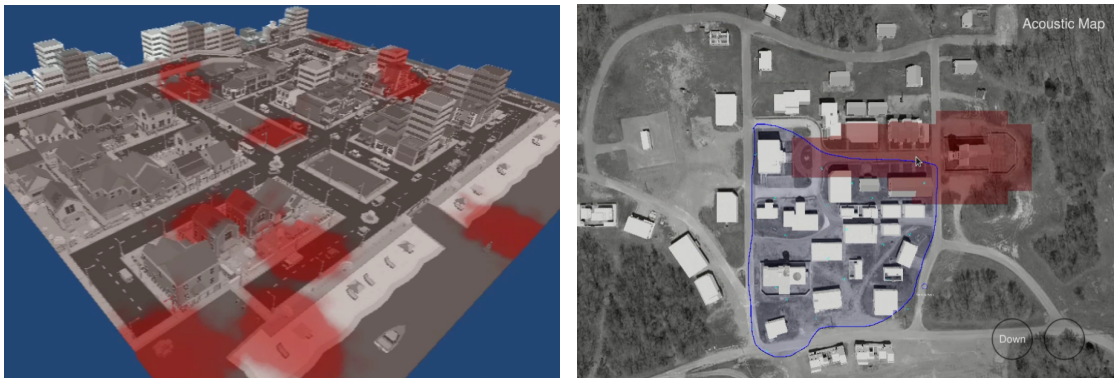


图15：网格数据可视化示例。左：威胁分布图。右：与操作员光标位置相关联的声学区域图。

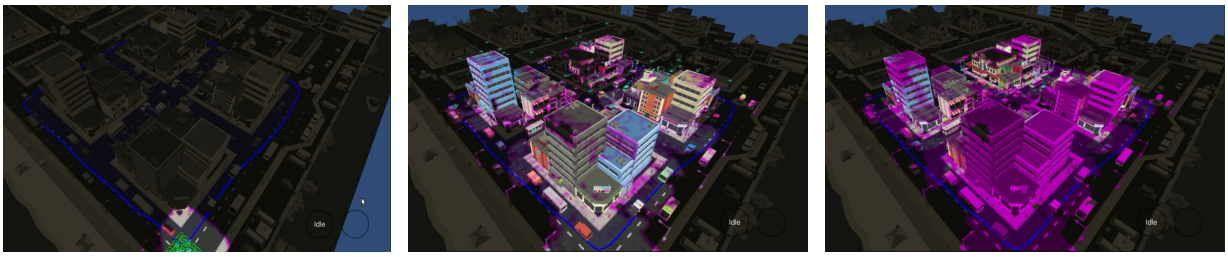


图16： 时间推移时的时间覆盖可视化，时间从左向右推进。最初，未检查区域呈现深褐色。随着智能体在环境中导航，已探索区域的场景几何逐渐呈现完整色彩。最后，随着时间推移，我们对特定区域失去态势感知，这些区域的颜色逐渐褪为品红色。



图17： 在坎贝尔堡举行的美国国防高级研究计划局 OFFSET 第六次野外实验期间，诺斯罗普·格鲁曼与 BBN 联合演练过程中捕获的时间覆盖可视化。深色区域尚未检查。全彩区域表示当前具备态势感知。品红色区域此前已检查过，但其态势数据现已过时。

缓存于基于图形处理器的体素哈希表中，以表示环境；自定义时间覆盖着色器使用该表来渲染环境。每个条目存储一个体素标识符和时间戳。当该着色器渲染一个片段时，会查询哈希表，以确定智能体上次看到该体素的时间。如图16所示，如果哈希表中不存在相应体素，我们就假定该体素从未被观察到，因此将该片段渲染为深褐色。否则，片段的颜色会根据自该体素被观察到以来经过的时间，在全彩与品红色之间进行线性插值[10]。

这种可视化有多种实际用途。例如，从业人员可以利用时间覆盖数据，快速验证侦察和持续监视战术是否按预期发挥作用。蜂群操作员可以保持态势感知，确认任务区域内哪些位置的信息缺失或已过时。图17展示了在美国国防高级研究计划局 OFFSET 第六次野外实验中，诺斯罗普·格鲁曼与 BBN 联合运行场景下使用的时间覆盖可视化。蜂群指挥官能够快速衡量任务目标的完成进度，并据此调整任务计划，以扩大覆盖范围。

[10]二十分钟后，我们会逐渐淡变为纯品红色。

5.6 支持的环境

指挥与控制支持三种复杂程度各异的环境类型。我们支持的最简单环境是基于图像的俯视地图视图，指挥与控制会根据给定的 GPS 坐标，从公共存储库加载该视图。这种环境适用于临时测试，例如旅途中进行的测试，或无法获取 3D 模型时进行的测试。其次，指挥与控制可以加载 GeoJSON 编码的数据，这有助于进行随机测试以及自定义场景测试。然而，如果需要更复杂的环境，开发人员可以导出自定义 Unity 场景，供指挥与控制加载。最后，RISE 中还内置了若干自定义场景，即刘易斯-麦科德联合基地、谢尔比营地和坎贝尔堡的 CACTF 场地。

6 战术开发界面

PyC2 提供了一个简洁明了的界面，可用于快速开发战术。要实现新战术，只需从 PyC2 战术基类派生一个新类，并按清单 2 所示指定战术定义。该定义包含指挥与控制调用战术所需的全部信息，包括命令手势、所需输入参数，以及有关战术及其参数的面向用户的描述。PyC2 通过心跳发现新的指挥与控制设备后，我们会通过网络传输所有战术定义。指挥与控制收到新的战术定义后，会重新训练手势识别器（如有需要，则收集新样本），使操作员能够立即调用这些战术。关联的战术弹出窗口（参见第5.4.1节）会自动填充数值、布尔值和文本参数数据项。这种设计使人们可以专注于核心战术逻辑，而无需操心用户界面集成，例如 GUI 布局设计、排列方式，以及在分层菜单中的嵌入。

当指挥与控制调用某个战术时，PyC2 会收到一条网络消息，其中编码了所有非草图参数数据。PyC2 会解析该消息、验证数据有效性、检查错误并解析上下文。也就是说，由于可以编写适用于不同上下文的战术，PyC2 还会检查所有可用的上下文选项，并选择与手势位置最接近的选项。例如，我们的俯瞰扫描战术可以作用于探索区域或扇区。如果最近的探索区域折线点距离小于最近的扇区点距离，PyC2 就会选择探索区域。若有错误，则会将错误报告给指挥与控制；否则，所有参数数据（包括套索选中的智能体）都会被复制到任务对象中。随后，PyC2 会实例化相应的战术类，并将任务对象作为唯一参数传入。这样，所有样板处理都会自动完成，使人能够再次将主要精力集中在战术逻辑上。

任何战术都可以调用其他战术或机器人原语，并将它们分别作为独立的子任务处理。因此，每个子任务都会编码一个战术或原语名称，以及所需的输入参数值。PyC2 为每个原语实现一个包装任务类，其中包含针对该原语的参数数据选择、操作和转换例程。在大多数情况下，无需专门编写逻辑，因为编码后的原语是键值（参数-值）对的直接副本。然而，在更复杂的情况下，原语编码可能取决于具体情境。每个子任务都会通过战术基类的排队机制进入队列，PyC2 的任务管理系统则会按照第4.4.1节所述处理任务竞标、分配和调用。附录A提供了一个完整的战术示例。本节余下部分将讨论如何定制战术行为，以及 PyC2 提供了哪些工具来支持高效的战术开发。

6.1 Tactic ExecutionCustomization

最简单的战术只负责生成子任务，不做其他事情。我们的战术基类为所有任务处理阶段和错误处理提供了默认行为。然而，在许多情况下，为了扩展战术的功能，必须覆盖默认的战术行为。例如，某个战术可能需要随时间向特定智能体下发多个原语，比如先前往特定位置，再消除威胁。另一个例子是，战术可能会根据竞标获胜的智能体来设置某些参数。或者

```
# SketchInputTypes.PointOfInterest {
  'basic_type' : PARAM_CUSTOM_POINT,
  'type_id' : int(SketchInputTypes.PointOfInterest),
  'gname' : 'lowercase_p', 'parameters' :
      json.dumps(PointProperties(
              display_name = 'Point of Interest',
               sphere_color = [0.0, 1.0, 0.0, 1.0],
               beacon_color = [0.0, 1.0, 0.0, 0.1],
               radius_m = 1.0,
               lock_axes = 2).__dict__) },
```

清单1： PyC2 中兴趣点类型的字典条目。每当 PyC2 收到新的 C2 心跳时，这种自定义点类型的类型、ID、手势命令名称和渲染属性都会自动编码并通过网络传输。

```
class OverheadScan(Tactic):
  # Tactic description data that is sent to C2 during registration.
  TacticDefinition = TacticDefinition('Overhead Scan',
      'Fly agents over an area to find artifacts.', 'circle',
      context = [SketchInputTypes.ExploreArea,
          SketchInputTypes.Sector], parameters = [
          TacticParameter('Altitude',
              'Height in meters that UAV will assume. Use -1 for automatic selection.',
              float, -1), TacticParameter(
              'Cell Size',
              'Minimum linear distance between waypoints in meters.',
              float, 10), TacticParameter(
              'Agent Count', 'Number of agents used to scan area.',
              int, 5)])
```

清单2： 俯视扫描的战术定义：根据输入参数将给定区域细分为若干较小区域，并为每个区域生成一个基元移动至命令。战术定义包含战术名称、简短的易读描述、手势、上下文（本例中可以是探索区域或 扇区），以及参数列表，其中每个条目都指定一个易读名称、描述、数据类型和默认值。每当 PyC2 收到新的 C2 心跳时，每个战术定义都会编码并通过网络广播。

战术可以提供自定义的错误处理逻辑，等等。因此，可以覆盖默认的竞标完成、竞标失败、任务完成、任务取消、任务失败和战术完成回调处理程序。

还可以重写战术的前置条件检查。如第5.4.1节所述，操作员可以使用类似门控的战术节点将多个命令串联起来。默认情况下，只有当子战术所依赖的所有父战术均已成功完成后，我们才会调用该子战术。否则，该子战术将失败。如果这种行为不合适，可以使用自定义前置条件检查，就像我们在合取、析取和否定战术中所做的那样。

6.1.1 草图输入定制

如清单1所示，可以基于 C2 的通用点类和折线类生成新的自定义草图输入类型。定义新类型时，需要指定其底层类型名称、唯一标识符、命令手势和渲染属性。定义完成后，自定义参数类型便可用作上下文，由 PyC2 在调用战术时解析。此方法有两项优势。首先，使用唯一的参数类型，用户可以更轻松地区分它们在 C2 中的存在及用途。其次，当某个手势与多种不同的草图类型相近时，这有助于消除歧义，从而降低操作员出错的可能性。

自定义参数也能快速实现并投入使用。例如，在坎贝尔堡进行最终现场实验期间，我们决定让智能体返回回收点，以便开展野外行动。此前，当智能体的电池电量低于某个阈值时，我们的安全着陆战术会让智能体降落在没有障碍物的位置。因此，我们创建了一种新的输入参数类型，并更新了战术：如果场景中的某处指定了回收点，就改为命令智能体前往回收点附近的位置。这项战术在两小时内完成实现并通过仿真验证，并于当天投入使用，部署了七十多架空中智能体。

6.2 战术开发工具包

PyC2 配备了多种功能和实用工具，可支持智能战术设计。以下是其中最相关的工具：

- **场景几何**：指挥与控制系统在初始化期间会向 PyC2 传输先验场景几何信息。这些信息包括以 GeoJSON 编码的建筑物标签、边界、墙体和高度数据。战术可以利用这些信息，指导侦察、监视以及其他与建筑物交互的机动。战术也可以将建筑数据用作上下文，例如，用于选择操作员打算扫描的建筑物。

- **目标物支持**：PyC2 会跟踪集群可能与之交互的物体，以及基本的状态信息，例如该物体是否已被识别或检测到，以及该物体是否构成威胁或已被中和。与草图输入和建筑物一样，战术可以将物体用作上下文。“防护物体”是一个示例战术，操作员可以在其中手动向集群发出命令，使其消除威胁。在此情况下，该战术会选择距离操作员的防护战术手势最近的物体。

- **路径规划器**：我们采用基于跳点搜索（JPS）（Harabor & Grastien，2011年）的路径规划器，并以多层网格的形式实现。底层网格结构编码了已知障碍物，包括建筑物和禁入区草图。战术可使用 JPS规划器，为空中和地面智能体推荐穿越环境的最优路径。智能体收到作为原始输入参数的路径信息后，可选择使用自身的局部路径规划器，沿指定路径前往指定目的地，同时与其他智能体协调空域机动，以避开动态障碍物。

- **草图输入**：操作员创建草图参数或对其进行修改时，所有参数都会通过网络传输。PyC2 会将每个参数存储在本地数据库中，供战术与之交互。系统提供多种实用工具，可按类型和距离查询草图、分析长度和绕行等几何属性，并通过空间重采样和线段简化转换草图。这些工具使开发人员能够高效分析上下文，并根据自由形式的草图用户输入生成任务。

- **草图生成**：我们还提供了让开发人员生成并发布新草图数据的机制。在 DARPA OFFSET 第六次野外实验之前，海军信息战中心太平洋分部（NIWC）提供了路缘、墙体和电力线占地数据。我们将这些信息编码为自定义草图数据，PyC2 将其视为禁入区。因此，我们的路径规划器能够绕开这些障碍物。

7 基元开发界面

所有战术都由基元构成，就像所有单词都由字母构成一样。基元组合起来，便能表达复杂得多的动作。在 RISE 中，我们将基元定义为特定智能体独有的基本动作。它们组合成涵盖整个集群的战术（参见第 6 节）。单个战术可能涉及在同一智能体上运行多个基元，也可能涉及在不同智能体上同时运行不同基元。

基元将高层的单个智能体逻辑与底层的机器人感知和控制算法相结合，以实现预期行为。

我们利用 FlexBE 行为引擎提供了一个高度模块化的系统，用于集成新基元。FlexBE[11] 是一个 ROS 软件包，提供状态机的创建和执行功能，并支持简便的 ROS 集成和易于使用的可视化编程接口（Schillinger, Kohlbrecher, & von Stryk，2016）。有关使用 FlexBE 状态机编辑器创建基元的示例，请参见图 18。这些 FlexBE 状态机与每个智能体的基本运行紧密交织，提供了许多核心功能。这一点很重要，因为它使基元能够利用现有的智能体行为，包括 FlexBE 的状态机嵌套和并发功能。无需重写代码（例如让智能体移动到某个航点），新基元可以直接嵌套现有的航点移动基元。这样，我们就能快速将新基元集成到智能体代码库中，并实现新的行为。每个智能体独有的基元集合都包含在平台专用FlexBE状态库中。

虽然智能体原语系统旨在便于集成第三方基元，但我们也为 RISE 提供了多种原语，包括基本移动、任务交互、建筑物破拆与进入、建筑物内部探索等。

8 算法开发界面

底层机器人算法构成了智能体能力的基础。我们从 ROS生态系统中实现了其中许多算法，但也为 RISE 自行开发了不少算法。这些算法大多以 ROS节点的形式运行在每个智能体的主计算板上，但也有一些运行在基于 Pixhawk 标准的飞行器飞行控制单元（FCU）上。

[11] http://philserver.bplaced.net/fbe/index.php

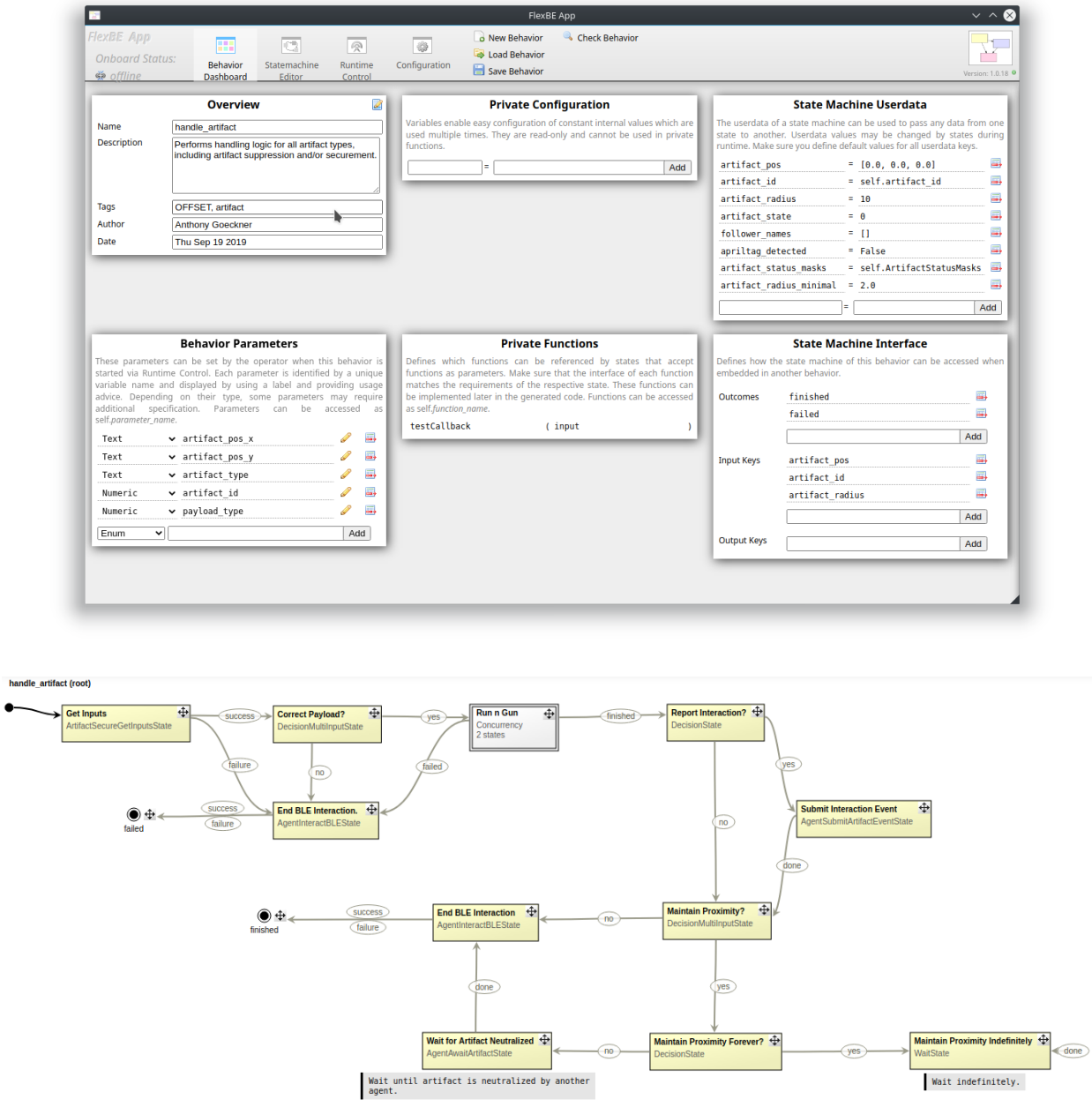


图18：用于创建智能体原语的 FlexBE 应用程序图像。底部的高级任务逻辑基元负责与环境中发现的物体进行交互。

8.1 感知

集群中的每个智能体都会收集其局部环境的数据。根据智能体的类别，可使用的传感器也有所不同。例如，IFO 车辆配备了前向双目摄像头、朝下摄像机、朝下单点激光雷达、全球定位系统（GPS）接收器、磁力计、高度计和加速度计。ATX地面车辆没有朝下的传感器，而是配备了 360 度环形激光雷达。

我们实现了驱动程序，将原始传感器数据转换为可用的 ROS消息，并提供给智能体上运行的其他 ROS节点。根据平台的不同，磁力计、高度计、加速度计和 GPS 数据由车辆的飞控单元处理，使用 PX4 或 Ardupilot 系统。然而，摄像机和激光雷达数据由车辆的主计算板处理，并通过 ROS节点实现。我们使用简单的散斑滤波器，从激光雷达数据中去除草茎等非障碍物。对于双目摄像头，驱动程序会计算视差图像，并将其提供给其他 ROS节点用于感知。

8.2 感知

为了在地面车辆上实现视觉障碍物检测，我们利用双目摄像头执行基于双目视差的检测。我们的实现基于 Oleynikova、Honegger 和 Pollefeys（2015年）的研究，能够高效检测障碍物，同时忽略诸如草茎之类不会对车辆造成碰撞风险的环境伪影。

对于空中车辆，我们实现了 PX4 避障系统，该系统利用 g 前向 深度摄像机检测并避开障碍物（见下文第8.3节）（Baumann，2018年）。

除了本地数据外，我们的智能体还可以使用其他车辆的遥测信息来避免碰撞。所有车辆通过移动自组网共享遥测信息。未来的工作可以扩展此功能，加入预测，从而提高其效用。

8.3 运动规划与控制

我们使用上述感知和感知部分所述的数据，为每个地面智能体构建两种代价地图。二维局部代价地图存储智能体附近障碍物的详细信息。该代价地图具有衰减功能，若一段时间内未检测到障碍物，就会将其从代价地图中移除。这有助于准确跟踪动态障碍物。我们还维护另一张全局代价地图，用于存储整个任务区域的障碍物信息。该代价地图没有衰减功能，这意味着障碍物会一直保留在代价地图中，直到新的本地传感器数据对其进行更新。这张二维全局代价地图用于全局路径规划。

地面车辆运动规划由 ROS 的 move_base 系统执行。其中包括用于远距离目标的全局规划器，以及能够对阻挡全局路径的障碍物作出反应的局部规划器。move_base 系统还负责高层执行器控制，并向飞控单元（FCU）发送速度指令。

-

对于 IFO 车辆，PX4 避障使用其自身的内部代价地图控制所有避障操作。这涉及直接控制车辆速度和航向。除了允许暂时关闭避障功能外，我们并未对公开提供的 PX4 避障库进行重大修改。这样，智能体原语便可控制车辆航向，实现原地旋转；而使用标准 PX4 避障时，这是不可能做到的。

8.4 智能体逻辑

为启用有助于实现任务目标的智能体原语，我们提供了用于高层智能体逻辑的算法。这些算法包括使用 ROS AprilTag 库自动检测和识别 AprilTag标签（J. Wang & Olson, 2016）、利用低功耗蓝牙信标与环境和任务场景交互等。关于如何让智能体与环境交互的更多信息，请参阅第11节。

9 实况、虚拟与构造

我们将智能体视为逻辑实体，其底层组件可以是集中式或分布式的，也可以是物理的或虚拟的。这之所以可行，是因为 ROS 通过基于网络的命名总线实现发布-订阅消息模式，以进行进程间通信（见第4.3节）。因此，使用 ROS 的 ROS 软件包和库具有自包含特性，使机器人模块能够作为独立进程运行。利用这一功能，一个模块可以发布传感器数据，另一个模块则分析这些数据并生成运动指令，供第三个模块执行；尽管所有进程都属于同一逻辑实体，但实际上它们可以分布在彼此不同的系统中。因此，我们将逻辑实体视为实现某个机器人实例功能的所有命名空间主题的集合。

随着 OFFSET 的推进，我们逐渐发现，需要设计一种方法，使仿真环境能够支持不同保真度级别。典型的机器人仿真环境（例如 Gazebo（Aguero等人，2015年））提供了稳健的开发环境，但通常仅适用于单个智能体。尽管我们最初尝试使用 Gazebo 进行群体开发，但很快发现，该系统在平台数量达到约十个时就会受到计算能力的限制。我们在其他各种高保真度仿真环境中也观察到了类似问题。这促使我们提出了一个概念，并在系统中称之为流体保真度（第9.3节）。

我们认为另一项至关重要的实现选择，是让虚拟智能体与实体智能体协同工作的能力。仅从后勤角度来看，将数百个平台运送到各种真实世界环境中进行间歇性测试，往往并不可行。因此，尽管我们希望在实体平台上测试新的群体战术，但我们需要一种方法，让少量实体智能体与虚拟智能体协同工作，以便快速评估新能力，而不必带上整个集群。由于仿真与指挥与控制（第 4.2 节）使用了同一个应用程序，并且我们实现了 ROS 接口（第 4.2.1 节），因此能够集成这一能力。

9.1 高保真仿真

在构建高保真度仿真环境时，我们借鉴了许多行业标准仿真系统，例如 Gazebo 和 AirSim（Shah、Dey、Lovett 和 Kapoor，2017年）。不过，我们的高保真度仿真环境也有一些特点，使其与许多常见仿真系统有所不同。两个主要区别是：高保真度仿真中没有启用物理引擎，且智能体本体由大小简单、相对设置的立方体对象组成。我们没有采用物理环境，而是为每种平台类型配备了基于几何的PID控制器来驱动运动。智能体通过 ROS Twist消息或设定点位置指令发送驱动指令，就像在真实平台上一样。与其他仿真器类似，单个智能体通过 ROS消息接收传感器信息。通常由物理传感器提供的、与传感器数据源相关的所有 ROS消息，均改为由 Unity游戏环境生成。仿真中的 ROS节点包括 TF信息、MAVROS数据、激光雷达、RGB 和深度等。每个智能体的各路传感器数据流都通过 ROS命名空间进行组织。所有数据均以 ROS 2消息的形式生成，但由于智能体代码库期望通过 ROS 消息传递来接收这些信息，我们使用 ROS 2桥接器在两者之间进行转换。这种 ROS 2转换以及集中式 ROS主节点由一款名为 ROS仿真服务器的应用程序负责处理。ROS仿真服务器使我们能够在单个系统上运行多个高保真度智能体，并与仿真系统配对以实现流体保真度（第9.3节）。尽管 Unity

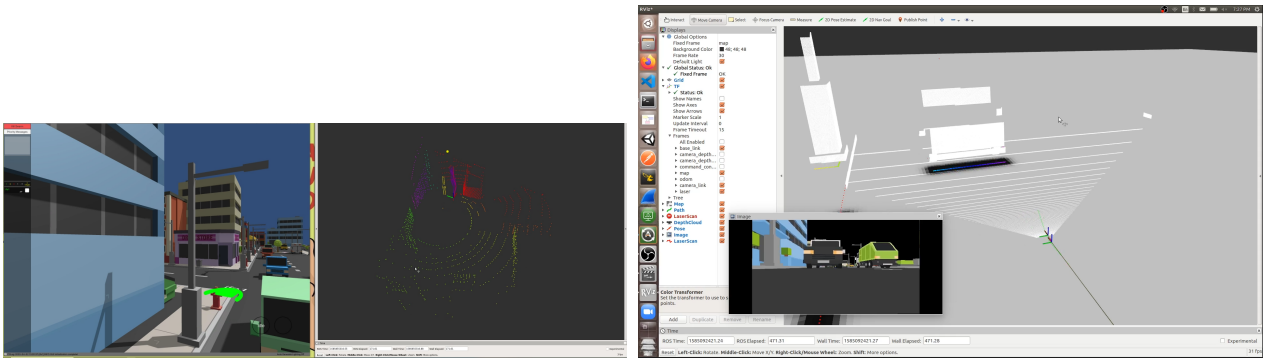


图19： 左：无人机在仿真中绕过路灯导航，以及在 RViz 中可视化的相关传感器数据流。右：无人地面车在仿真环境中导航，以及在 RViz 中显示的相关深度摄像头传感器数据流。

现在已正式支持 ROS 集成[12]，但在该系统问世之前，我们就需要 ROS 支持，因此我们改为利用自己的 ZMQ 消息系统实现了一套方法（参见第4.3节）。

在数据流得到支持后，通过仿真 Docker 容器对实际的智能体原语和算法进行测试。底层代码库与真实群体智能体在使用高保真度仿真时运行的代码完全相同。只有少数 ROS 节点需要进行微调，这些调整通过运行时 ROS 参数确定，用于更改组网配置。直接与硬件传感器交互的 ROS 节点会被禁用，因为 Unity 已发布数据，不再需要这些节点。每个智能体都有对应的 Docker 实例，彼此仅通过容器主机名和挂载的文件目录代码库来区分。图19展示了 IFO 和 ATX 在该系统中进行高保真度仿真的情形，以及它们在导航栈运行时于 RViz 中渲染传感器数据流的情况。

9.2 低保真仿真

我们的低保真仿真能力与标准机器人仿真器截然不同。在这种配置下，没有任何机器人算法运行，整个概念都围绕模仿展开。我们发现，使用高保真仿真器时，单台机器最多只能支持约十个智能体。虽然我们的高保真仿真支持将计算负载分散到多个系统上，但我们发现，要达到预期规模并不可行，尤其是因为我们缺乏云计算资源。战术开发有时也会在现场快速进行，而额外的计算资源或互联网连接并非总能得到保障。这些限制促使我们创建了低保真仿真器。在这种配置中，没有运行仿真智能体 Docker容器，整个配置仅由 Unity Base 和 PyC2 应用程序组成（见图3）。与高保真仿真器类似，Unity 会渲染仿真智能体各自的游戏对象，但除了位置信息外，不会生成任何仿真传感器数据源。Unity环境中的组件不会生成原始传感器信息，而是对智能体损耗或智能体检测等情况进行粗略估算，从而模拟高保真仿真中完整传感器管线的响应。另一方面，PyC2 会模仿智能体原语。由于没有实际运行的智能体原语或算法，PyC2 负责运行智能体实例，并通过模拟算法模拟特定战术执行中的相对智能体响应。这些模拟算法仍像高保真仿真环境一样，通过设定点位置或 Twist API 发送命令。让 PyC2 为每一个智能体原语都维护一份模拟副本将难以为继，因此需要注意，只有最常用的原语才有直接的低保真映射。其余原语则由流体保真度这一概念来处理（见第 9.3 节）。

[12]https: //github.com/Unity-Technologies/ROS-TCP-Conn ector



图20： 在密西西比州谢尔比营的环境中，使用单台机器运行低保真仿真，支持超过100个智能体。

图20展示了在单台机器上运行的 100+ 低保真智能体。值得注意的是，无论保真度级别如何，也无论智能体是真实的还是仿真的，面向最终用户的界面始终相同。这种低保真仿真在进行特定于智能体的开发时用处不大，但在进行战术开发时却非常强大（见第 6 节）。如果没有低保真仿真，就不会有这么多战术。这种较低的保真度还支持快于实时的性能等更多功能，从而能够快速进行战术分析，并为最优行动方案生成等概念打开大门。

9.3 流体保真度

由于低保真仿真智能体无法仿真某些战术所需的原语，因此我们需要高保真仿真智能体能够介入并执行这些原语。如第9.2节所述，低保真智能体仅使用模拟算法来模仿实际智能体原语，但这些算法并不完整。这正是流体保真度概念发挥作用的地方。操作员执行任务时，可能会遇到需要调用低保真智能体无法执行的战术的情况。我们需要一种方法，能够无缝地将低保真智能体转换为高保真智能体，从而无需重启运行时环境即可继续执行该战术。如第4.4节所述，心跳消息包含两个名为“保真度级别”和“配置哈希”的属性。仿真环境利用这两个字段来确定应生成何种智能体及其机载传感器。通过调整低保真仿真和高保真仿真中的这些字段值，我们可以实时更改正在运行的智能体的保真度级别。

9.4 跨仿真支持

在开发过程中，我们发现有些特殊情况下，仿真缺少我们所需的特定功能。这种情况大多发生在开发非常具体的机器人算法时。无论是需要详细的物理模型、更高品质的视觉效果，还是具备准确模型描述的独特平台，这些都是我们的仿真环境所欠缺的功能。所幸，还有许多其他仿真环境专门擅长这些领域。由于智能体集成所需的最低限度应用程序接口（第4.4节），我们能够同时运行 Swarm Engine、仿真 Docker 实例和一个独立的



图 21：Swarm Engine 与 ArduPilot 仿真器并行运行，其中包含两架模拟 AVT

仿真。采用这种配置时，Swarm Engine 会将仿真智能体视为实体平台，而不会生成任何仿真传感器数据源。相反，独立的仿真实例通过 ROS 消息传递生成仿真传感器数据源。仿真 Docker 实例需要接收 ROS 信息，因此只要模拟器配置了正确的话题名称和命名空间，我们就能完成这种切换。不过，Swarm Engine 和独立的仿真环境必须能够在同一环境中定位；而大多数模拟器都支持配置 GPS 信息。我们利用这种配置，快速启动了近期集成的垂直起降平台 AVT。图 21 展示了一个示例：Swarm Engine 为两架仿真 AVT 分配任务，同时使用 ArduPilot 仿真提供传感器信息。这样，我们无需在仿真环境中支持额外的飞行器类型，就能快速开发并验证 AVT 基元和算法，并将其用于实体平台。在与 Hive 集成期间，我们也采用了这种仿真环境配置，并使用了 Gazebo（见第10.4节）。

10 RISE 可扩展性：第三方集成

本节重点介绍作为 DARPA OFFSET 计划一部分集成到 RISE 系统中的各种第三方能力。这些第三方被称为“冲刺团队”，由提出特定重点领域能力的小型企业和大学研究团队组成，这些能力在“冲刺周期”期间开发并集成。DARPA 在该计划实施期间安排了五个冲刺周期，与定期现场实验（简称“FX”）期间展示冲刺团队能力的计划相呼应（参见第11节）。

10.1 Heron 分布式任务分配系统

Heron Systems 专注于通过去中心化的异步竞价方开发自主任务分配框架，以处理分布式任务分配问题。该框架将取代现有的竞价任务分配系统，并提供一种具备更多功能和能力的替代方案。我们与 Heron 密切合作，将其能力集成到 RISE 中，并利用基于 ROS 的设计处理 Heron 核心模块与 RISE 之间传递的数据。Heron 的任务分配框架已完成全面集成，并在佐治亚州本宁堡的现场实验中进行了演示。该功能取代了原有的竞价系统，实现了去中心化解决方案，使智能体即使未与 RISE C2 直接通信，只要与其他智能体通信，也能够承担任务。借助预期群体智能体与 C2 之间的间接通信，该方案实现了任务执行保障和任务持续性。RISE 设计的模块化特性及其基于 ROS 的基础，使整体群体操作中的关键组件能够轻松替换。

10.2 MTRI-SoarTech 合成扫描与下车人员探测

密歇根理工研究所（MTRI）参与了 DARPA OFFSET 计划的多个冲刺阶段，并积累了丰富的 RISE 集成经验。在第 4 轮冲刺中，MTRI 专注于虚拟试验平台方向，利用一种在真实世界中可能不存在、但可以在虚拟环境中进行模拟的合成技术。MTRI 与 SoarTech 合作，旨在开发蜂群结构态势感知（SSAS）功能。该功能通过与 RISE 集成的软件，实现合成的隔墙平面图生成和下车人员探测。我们与 MTRI 和 SoarTech 合作，完成了 SSAS 的全面集成，并在华盛顿州刘易斯-麦科德联合基地的现场实验中，利用实体平台展示了该功能。借助 RISE 蜂群引擎指挥与控制，我们能够分配无人机在建筑物周围执行飞行战术，并实际开展扫描作业。实体扫描利用 SSAS 行为，对真实建筑物的模拟信息进行数据采集，随后生成该建筑物的隔墙平面图，并探测模拟下车士兵。其生成的数据产品包含不确定性信息；如果这项技术在现实中存在，这些信息将与真实世界的数据采集结果非常相似。基于实体飞行和生成的数据产品，MTRI 得以制作该数据产品的可视化结果，展示生成的平面图以及执行的下车人员探测结果。通过与 RISE 集成，借助实际实体平台展示了虚拟合成技术及其应用。RISE 能够支持完整工作流，无需 MTRI 和 SoarTech 开发整套端到端管线，使其能够专注于开发各自的具体能力，同时也能将这些能力应用于更大型的机器人系统中。

10.3 SoarTech 室内建筑物清剿

SoarTech 是一支经验丰富的冲刺团队，参与过众多冲刺阶段，并与我们密切合作，将各种能力集成到 RISE 中。在第 3 轮冲刺期间，SoarTech 提议开发的能力聚焦于室内环境，以及建筑物内部的搜索与清剿。他们提出的方法利用集群的多智能体特性，采用持续且同步的搜索方式，从而提升清剿建筑物楼层内部的效能与效率。SoarTech 开发了这项能力，并将其封装为一个 FlexBE 状态，集成到我们现有的建筑探索战术中。由于他们的工作重点完全在室内，因此利用 RISE 现有的室外导航能力，首先机动前往建筑物。SoarTech 在 RISE 智能体 Docker 容器中以独立 Servlet 的形式运行自己的执行环境，以便在 FlexBE 行为中执行该能力。SoarTech 利用 RISE 的网络，在智能体之间共享生成的建筑地图。他们通过动态指派“中继”智能体，并自动将其部署在关键位置，确保数据能够借助 RISE 的移动自组网（参见第 4.3.2 节）传出建筑物并送达集群中的其他成员，从而在大型建筑物内维持通信。通过广泛的协作开发和迭代测试，我们成功将这项能力全面集成到 RISE 和我们的地面车辆中。我们在密西西比州谢尔比营的 FX-3 演练期间，展示了这些能力的执行效果。

10.4 HIVE

在 Sprint 5 的硬件重点领域，Sentien Robotics 提出了一套自动化无人机地面管理系统，可为比其此前开发的系统规模更大的智能体机队提供存储、充电以及起降后勤支持。该系统名为 Hive，他们开发了两种型号：HiveXL 和 HiveISO。两者本质上相同，只因制造工艺不同而在外观上有所差异。该系统完全装载在一辆拖车内，拖车中设有储存舱，可容纳大量无人机；回收系统则可将集成后的无人机运送至两个屋顶起降平台之一，并在其充电舱与平台之间往返。系统还为每个起降平台配备了机动龙门机械臂，用于调整无人机的位置，使其处于起飞和着陆回收所需的状态。RISE 与 Sentien 团队紧密合作，全面集成了无人机作业流程：RISE C2 可以向 Hive 中的智能体下达任务，而 Hive 会回收这些智能体



图 22：RISE 无人机智能体在第六次现场实验中降落到 HiveISO 上

将它们从充电舱取出，并送至屋顶发射台起飞。飞行作业完成后，Hive无人机会返回 Hive，并精确着陆在两个充电舱中的一个上（见图 22）。着陆后，Hive 会调整无人机的朝向，然后通过舱门将其放到充电托盘上。朝向十分重要，因为充电舱内的充电机构需要以特定朝向与无人机连接，才能正确接入托盘进行充电。无人机放上托盘后，内部回收系统会将其送回充电舱，从而完成一个完整的飞行作业周期。RISE 和 Sentien 在田纳西州坎贝尔堡举行的最终现场实验中，成功演示了这一完整的飞行作业周期。Hive 为群体智能体的运输、充电、部署和回收提供了重要的后勤解决方案，但在支持多个平台同时发射方面存在一些局限。它更适合数量较少的无人机持续作业，或连续多次发射；若同时使用过多平台，起降平台便会出现拥堵。Hive 与 RISE 的集成表明，RISE 能够纳入机器人平台以外、支持集群作业的其他技术；在本例中，这种技术属于后勤技术。

10.5 集成总结

Heron、MTRI、SoarTech 和 Sentien Robotics 的能力集成案例，展现了 DARPA OFFSET 计划中冲刺团队开展集成工作的若干不同实例。各个冲刺阶段的重点各不相同，吸引了广泛的能力。我们与不同的冲刺团队合作，为其能力建立了集成到 RISE 的路径，并提供集群机器人框架，以实现这些团队的能力，涵盖蜂群自主性、群体战术、机器人硬件、人机界面、人工智能等。

实验	Date	位置	RISE目标
FX-0	2018年3月	纽约市消防局，纽约市	机器人代码库和架构测试。 此外，还进行了野外机器人学演练。
FX-1	2018年10月	罗伯茨营地，CA	多智能体框架的实现 以及指挥与控制界面
FX-2	2019年6月	佐治亚州本宁堡	面向原始环境的群体协同作战， 情报、监视与 侦察（ISR），采用下一代指挥与控制迭代， 通过平板电脑指挥与控制模态实现移动指挥， 以及虚拟现实（VR）界面
FX-3	2019年12月	密西西比州谢尔比营	大规模蜂群组网与作战 采用下一代手势 的 C2 迭代版本
FX-4	2020年6月	刘易斯-麦科德联合基地，WA	增强群体战术执行并改 进借助增强现实（AR）实现的指挥与控制 模态
FX-5	已取消	N/A	N/A
FX-6	2021年11月	田纳西州坎贝尔堡	使用最新版本进行全规模集群测试， 其中指挥与控制包括实况，虚拟和建 构式功能特性和改进的增强现实界面 包括指挥功能

表2： 过去四年中，RISE 已在一次室内和五次室外实地实验中接受测试。请注意，FX-5 因新冠肺炎而取消。

11 实地实验

作为 DARPA OFFSET 计划的一部分，项目开展了实地实验，以测试和展示技术进展。这些实验大约每六个月进行一次，由海军信息战中心太平洋分部（NIWC）的实验团队负责组织。总计四年间，RISE 在五个城市训练场反复接受测试。表2总结了 OFFSET 的所有现场实验。FX-0 是一场仅在室内进行的活动，其余现场实验则都在室外进行。FX-1—FX-3 为我们提供了在典型环境中测试系统的机会，而这些实验中总结的经验为架构和用户界面的设计提供了依据。FX-3 让我们进一步认识到原有网络解决方案（见第4.3.1节）所带来的可扩展性问题，从而使我们能够在后续实验中实现更大的规模和更强的作战能力。到 FX-4 和 FX-6 阶段时，我们已积累了相关经验和信息，能够始终如一地根据任务级目标部署大规模集群。

在本节余下部分，我们将讨论倒数第二次实地实验 FX-6。该实验使我们能够回答以下问题：为快速集成和一对多控制而设计的多机器人生态系统，能否投入实地使用，让单个操作员指挥 150+ 自主车辆执行战术机动?



图23：与表2对应的六个现场实验地点。上方：FX-0（左）—FX-3。下方：FX-4 和 FX-6。



图24：田纳西州坎贝尔堡 Cassidy CACTF 的屋顶视图。

11.1 参与者

诺斯罗普·格鲁曼 RISE 团队承担了所有主要角色，包括一名蜂群指挥官、一名蜂群操作员、一名健康工程师和一名现场行动官。这些人员都是 RISE 专家，深入了解实验中使用的指挥与控制界面和平台。此外，除健康工程师外，其他人员都参与过以往的实地实验。核心团队由 8–9 名现场支援人员提供支援，协助处理后勤、部署和安全事宜。最后，一名作为辅助团队成员参与其中的 Rajant 现场工程师协助处理网络基础设施。

11.2 设施与设备

11.2.1 环境

FX-6 在田纳西州坎贝尔堡的 Cassidy CACTF 举行，见图24。Cassidy 直径约为305米，由一条城市道路网络连接约45栋建筑物组成，其中包括教堂、大使馆、联排住宅和清真寺等。除建筑物密集外，电线、路缘、树木和其他障碍物遍布整个 CACTF。

NIWC 对 CACTF 进行了增强，通过在整个环境中布置基于场景的标记物，以支持实验。这些标记物是 AprilTag（J. Wang & Olson，2016）（一种基准标记），用于表示复杂物体。例如，AprilTag 可用于表示高价值目标、敌对目标、简易爆炸装置（IED）、医护人员、无害物体、建筑物标签和优先情报信息等。某些 AprilTag 与现场节点相连，节点内搭载树莓派和低功耗蓝牙（BLE）设备，使机器人能够与物体交互。例如，当机器人进入简易爆炸装置的作用范围时，该装置可通过蓝牙交互使机器人失效。本次实验使用的特定 AprilTag 标记族为 48h12。采用这一标记族是因为它具备嵌套标签能力。智能体能够从更远的距离检测到外层标签，且该标签通常只包含人员或物体等概括性信息。只有内层标签才能揭示特定基准标记的真实属性，说明它是无害、敌对还是友方。在 RISE 中，大多数内层标签与外层标签之间的交互都由机载智能体逻辑自主处理（见第7节）。为构建测试场景，总共使用了 8000 多米以太网电缆、300 米光纤、1901 个不同的 AprilTag（其中 26 个为动态标签）、136 个树莓派节点和 46 台网络交换机。

11.2.2 RISE蜂群

所选任务场景和场地旨在使蜂群规模从一次现场实验到下一次实验逐步扩大。因此，到 FX-6 时，我们不仅需要一个能够持续执行 3.5 小时任务的大型蜂群，还需要一个具备多样化能力、能够满足不同任务目标的蜂群。基于这一考虑，我们组建了一个由 274 个平台构成的大型异构群体，如图 25 所示。这一组成使我们能够支持三个作战层级：由 ATX平台承担地面层，由 IFO 平台承担低空至中空空域，以及由 AVT平台承担高空。以下是关于各平台类型的更多信息：

- **地面平台**：地面车辆续航时间长，能效高于空中平台。与空中平台相比，地面车辆可以搭载更多有效载荷和额外硬件，而不会对整体性能造成重大影响。它们具备地面作业优势，作业期间受损风险较低。与空中平台相比，它们更能承受碰撞和／或更严苛的作业环境。这些平台还配备了额外的传感器，即全向测距扫描激光雷达（RPLidar A2），用于建图和定位。激光雷达与车轮编码器配合，使这些平台也能在 GPS 信号拒止或受限环境中运行，而其他平台则无法做到。然而，它们的通行能力有限，必须避开比空中车辆更多的障碍物，因此抵达目的地所需时间更长



图 25：FX-6采用的274个机器人平台，从左至右分别为：158个平台为 Uvify IFO-S（IFO）、97个平台为 Aion R1（ATX），以及19个平台为 Aion（AVT）垂直起降（VTOL）平台。

我们最初使用 Kobuki Turtlebots 和定制的滑移转向漫游车。然而，为了支持规模更大的行动，我们将 AION R1 漫游车集成到系统中，使其成为主要的地面资产。R1 漫游车使用 Jetson TX2 伴随计算板，简称“ATX”（见图 25）。

- **四旋翼飞行器**：四旋翼飞行器能够更精准地控制空中车辆，且无需横向移动即可维持飞行。它们机动性极强，能够按需悬停，因此可以在障碍物更多的低空运行。四旋翼飞行器能够在三维空间中穿越环境，既能飞越障碍物，也能从其下方穿过，并且比地面车辆更快地导航至目标位置。不过，这类平台也有一些局限性，例如其功重比对于获得并维持升力至关重要。飞行时间和效率会受到车辆重量及有效载荷的影响。此外，在密集环境中飞行时，受损风险也会增加。我们此前集成过定制平台以及 Intel Aero。为满足 OFFSET计划的需求及其范围，我们集成并大量使用了 Uvify IFO-S 平台，称为“IFO”（见图 25）。

- **固定翼**：此外，我们还集成了图 25 所示的 AION垂直起降尾座式飞行器（AVT）。虽然该飞行器比四旋翼飞行器更加节能，因此可能能够在空中停留更长时间，但它也必须保持最低空速。垂直起降类型的固定翼飞机无需发射装置和较长的起飞跑道，同时具备四旋翼飞行器的悬停能力，但更易受风影响。将非垂直起降固定翼平台用于群体协同，会限制可同时发射的平台数量（Chung等，2016）。然而，尽管垂直起降平台的转向灵活性有限，需要较大的开放区域进行转向，但与四旋翼飞行器相比，其更长的飞行时间可以弥补这一不足。因此，垂直起降飞行器非常适合高空作业，例如空中监视。

每个平台还携带了一个用于模拟有效载荷的蓝牙发射器。潜在的有效载荷包括电子战（EW）、杀伤人员（AP）、声学欺骗（AS）和掩护火力（CF）。这些有效载荷类型会影响智能体能力、它们能够交互的野外节点工件，以及它们在任务中的部署方式。

鉴于任务持续时间较长，维护和更新大型集群是一项繁重的工作。除了维护硬件以及确保智能体性能所需的后勤工作外，还必须能够在智能体执行任务前，将其全部更新至最新软件版本。RISE 的架构设计支持利用 Ansible 高效部署软件更新至集群。针对不同的更新需求，开发了多种用于部署的剧本。这些更新可以细化到部署更新后的 Docker 容器、机器人扩展代码库，或平台代码库中的其他组件。

11.2.3 命令与控制

蜂群操作员在一台 Alienware 笔记本电脑上运行 RISE C2 软件，该电脑配备 2.20 GHz Intel© CoreTM i7-8750H CPU、32 GB 2667 MHz DDR4 内存和 NVIDIA® GeFore® RTX 2080 移动版图形处理器。所有集群指挥与控制操作均使用鼠标完成，键盘方向键则用于切换可视化模式。群体健康工程师和蜂群指挥官使用了类似的系统，但指挥官仅使用 C2 进行态势感知。指挥官、操作员和工程师在 3b号建筑的屋顶上操作，如图 26 南侧（底部）所示。

11.3 协议

11.3.1 场景

在整个 OFFSET 计划期间，NIWC 开发的任务场景聚焦于城市环境中的集群作业，并自首次现场实验以来不断迭代演进。任务故事通常围绕敌对部队（OPFOR）占据城市环境展开，其中主要和次要高价值目标（HVT）分布各处，并由各种防御措施保护。蓝方集群（BLUFOR）的任务是进入该环境，突破防御措施并夺取高价值目标。对于 FX-6，NIWC 发布的场景与实验指南中给出的任务陈述如下：

> DARPA 总部收到情报，指出一种武器化传染病原体正在坎贝尔堡卡西迪 CACTF 设施区内的建筑物中。蓝方部队已在 CACTF 南端（第1区）建立立足点。主要突击将从南端（第1区）开始。

此外，任务分为以下四个阶段：

- 阶段 0：集群部署与情报、监视与侦察（ISR）
- 阶段 1：战斗行动与隔离主要目标
- 阶段 2：实施城市突袭
- 阶段 3：夺取关键城市地形

阶段 0 的目标是部署集群、收集有关环境的情报信息、在作战区域内持续监视，并保持态势感知。定位问题和传感器误差使我们的地面车辆无法自主移动并以必要的精度进入建筑物，因此我们主要只参与了阶段 0，因为后续阶段需要具备破障能力。

11.3.2 任务准备

蓝方团队每次任务运行前约有一小时的准备时间。在此期间，现场作战人员根据任务需求，将集群布置在初始部署区域。按照既定场景，突击行动均从一区发起，因此 IFO 和 ATX 平台沿南侧道路分布，充分利用所有可用空间。出于安全考虑，AVT 固定翼平台被部署在 3b号建筑以东的一处空旷区域，远离大多数人员。在此期间，蜂群指挥官与蜂群操作员根据任务需求和平台限制，共同制定行动方案。计划敲定且平台就位后，系统通电，以进行通信

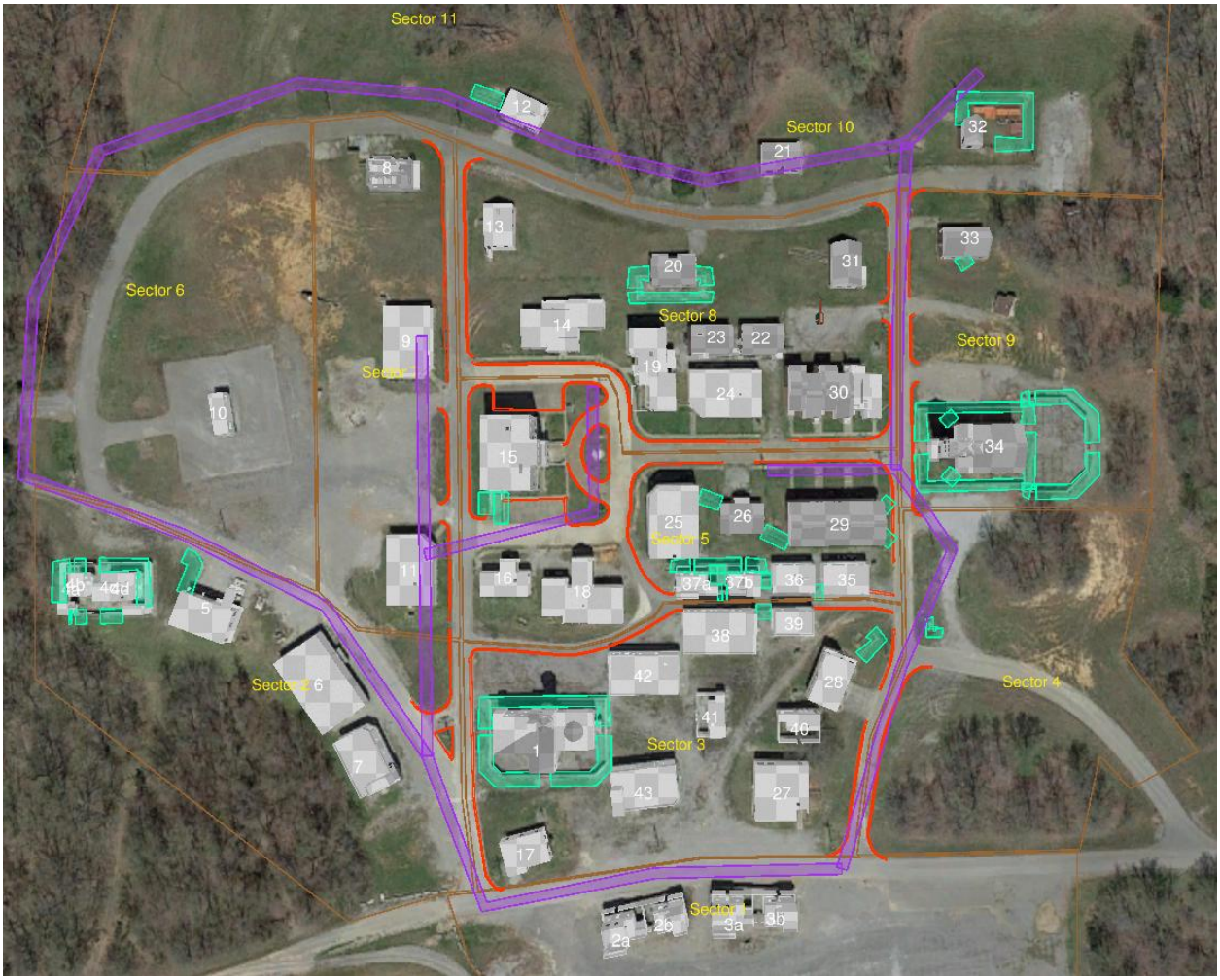


图 26：由 NIWC 提供的先验现场情报增强后的坎贝尔堡卡西迪 CACTF，其中包括路缘（红色）、墙（绿色）和电力线（紫色）障碍物数据，以及扇区边界信息。建筑物在 FX-6 之前根据平面图数据和现场勘察信息建模，而障碍物数据则在每次任务运行开始时加载。地图方向为上北。

检查。发生网络问题时，均已解决，并关闭了集群电源。随后，蜂群指挥官向实验团队、活动组织者和贵宾介绍了任务计划及预期结果；与此同时，蜂群操作员单独输入最终的初始行动方案，为任务开始做准备。

11.3.3 任务执行

任务简报结束后，现场作战人员返回部署区域，或在 CACTF 各处就位担任安全观察员。需要执行初始行动方案的机器人已启动；当集群中足够比例的机器人投入运行后，蜂群指挥官便开始突击。在整个任务运行期间，蜂群指挥官会监控局势、响应情报信息，并决定后续行动方案，再由蜂群操作员通过指挥与控制加以执行。任务将持续进行，直至蜂群指挥官宣布任务结束，或规定的 3.5小时 用尽。

11.4 结果

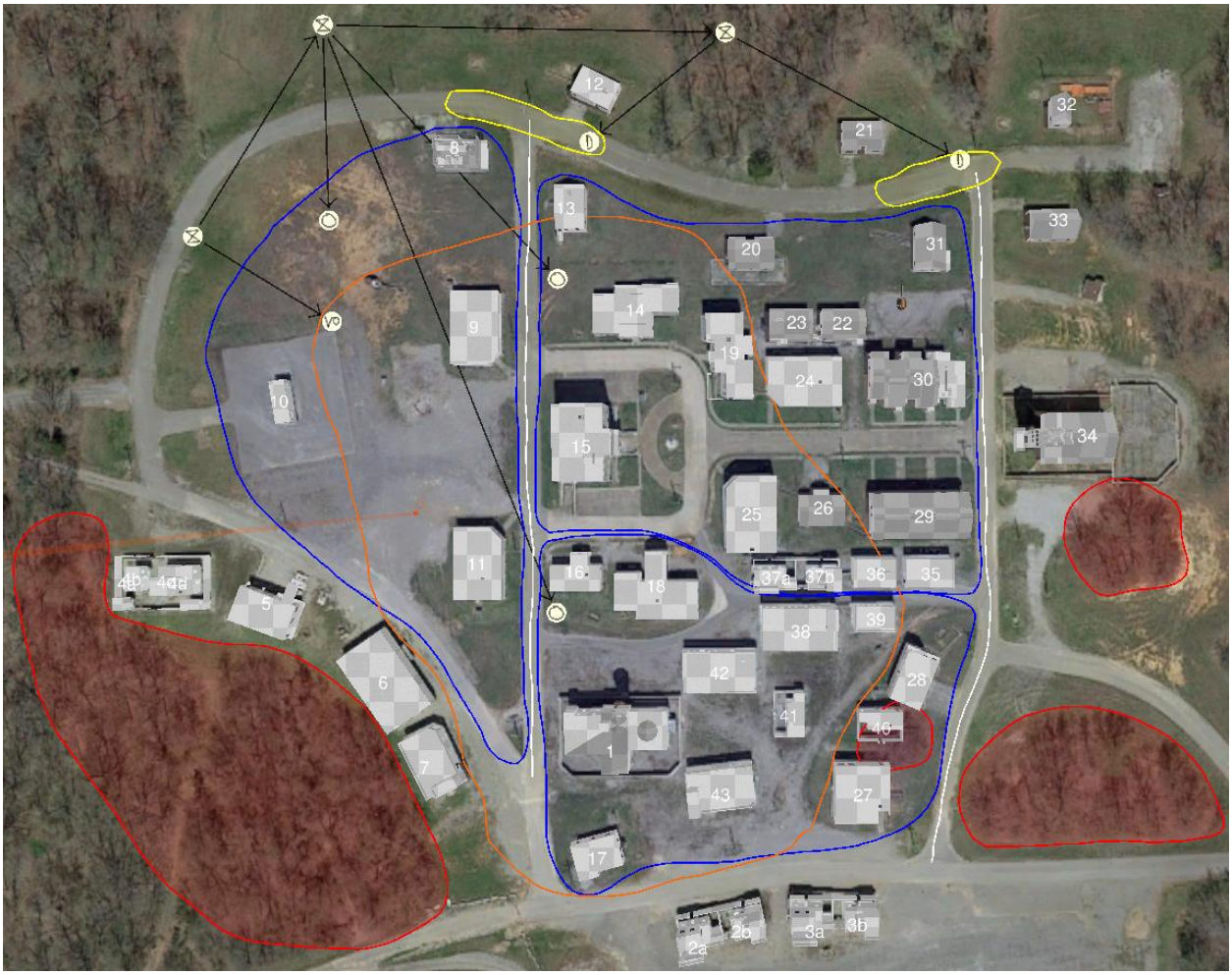


图27： FX-6 使用的初始行动方案示例；为清晰起见，图中移除了障碍物（路缘、墙和电力线）。详情见第11节 11.4。

借助已建成的现场实验场景基础设施，RISE 得以部署其能力与技术创新，为机器人集群作业提供了有益的洞见。

11.4.1 命令与控制

图27展示了在 FX-6 使用的一种行动方案示例，该方案用于启动一次任务运行。由于 Cassidy CACTF 中有一些树木未包含在我们的虚拟环境表示中，蜂群操作员手动设置了禁入区，以确保 PyC2 的路径规划器绕开这些障碍物。蜂群操作员还使用战术链对三个事件进行排序。首先，零延迟计时器（最左侧的沙漏符号）启动垂直起降扫描，使用三架 AVT 扫描橙色草图区域。接着，延迟一分钟后，第二个计时器启动三次空中扫描，分别覆盖每个蓝色探索区域，并使用 IFO 四旋翼飞行器资产。最后，再过一分钟，第三个计时器启动两项部署战术，旨在将 ATX 地面智能体调入北侧的黄色部署区。由于每个部署区都与一条路线相连（白色的南北向草图线），智能体将沿指定路径移动，而不会选择可能更短、但要穿过 CACTF 的路径。此外，还设置了一个回收点（建筑物 11 附近带有信标的橙色点），用于标明 IFO 智能体在电量不足时应飞往并降落的位置。通过这种方式，操作员仅使用 14 个战术草图参数（战术控制措施）和 9 项战术，就部署了所有可用资产并实现了较高的覆盖范围。

随着任务推进，指挥官通常会指示蜂群操作员继续执行扫描行动，包括根据情况进行俯视扫描和建筑扫描。期间曾尝试让地面智能体与敌对目标交战并突破建筑物，但定位和传感器误差往往阻碍了这些行动。

整个活动期间共使用了13种不同的战术，其中两种采用了虚拟硬件传感器。将这些战术转化为实际行动时采用基于草图的战术参数，这为我们提供了足够的灵活性，能够满足需求，同时适应平台自主性限制。如果平台在避障和导航方面具备更高的自主性，我们就需要更多战术，因为届时会让集群承担其他任务，尤其是协调突击行动和确保建筑物安全。

11.4.2 作战持续保障

在FX-6，RISE能够在每次试验运行规定的3.5小时内持续开展作战，并且还可以进一步延长。鉴于在高密度城市环境中开展低成本机器人实地实验的性质，出现损耗在所难免，例如传感器与系统故障、飞行条件突然恶化、平台生命周期延长以及其他物理限制因素。然而，得益于作战审慎性和我们的战术设计方法，带到FX-6的274个资产经过了适当配置，足以应对为期两周的活动。尽管偶尔会发生载具与环境碰撞，现场维修和资产恢复不仅维持了集群，还使其规模得以扩大。随着每次演练轮班，RISE都会为任务运行调配更多比例的资产；到活动结束时，RISE已能够同时使用174个资产，并让其他资产处于待命状态，以确保任务持续执行。该项目的一个关键问题是，单个操作员能否控制 $150+$ 架无人机；最终结果证明这是可行的。

11.4.3 任务覆盖

图28展示了使用100个智能体进行的一次运行中，约前一个半小时的检测结果。在最初约20分钟内，集群就检测到600件人工制品；随着任务推进，检测到的数量还在增加。图29展示了一次运行过程中任务区域的覆盖情况：上图显示了十个群体智能体中部分智能体的飞行路径，下图则呈现了整个环境中的蜂群活动。这些结果不仅凸显了集群实现的全面区域覆盖，也表明蜂群活动与CACTF的建筑密度相吻合。尽管这些图表和图像只是RISE蜂群在FX-6期间单次运行的快照，但它们体现了RISE一贯具备的能力：部署蜂群以覆盖整个作业区域并执行任务。

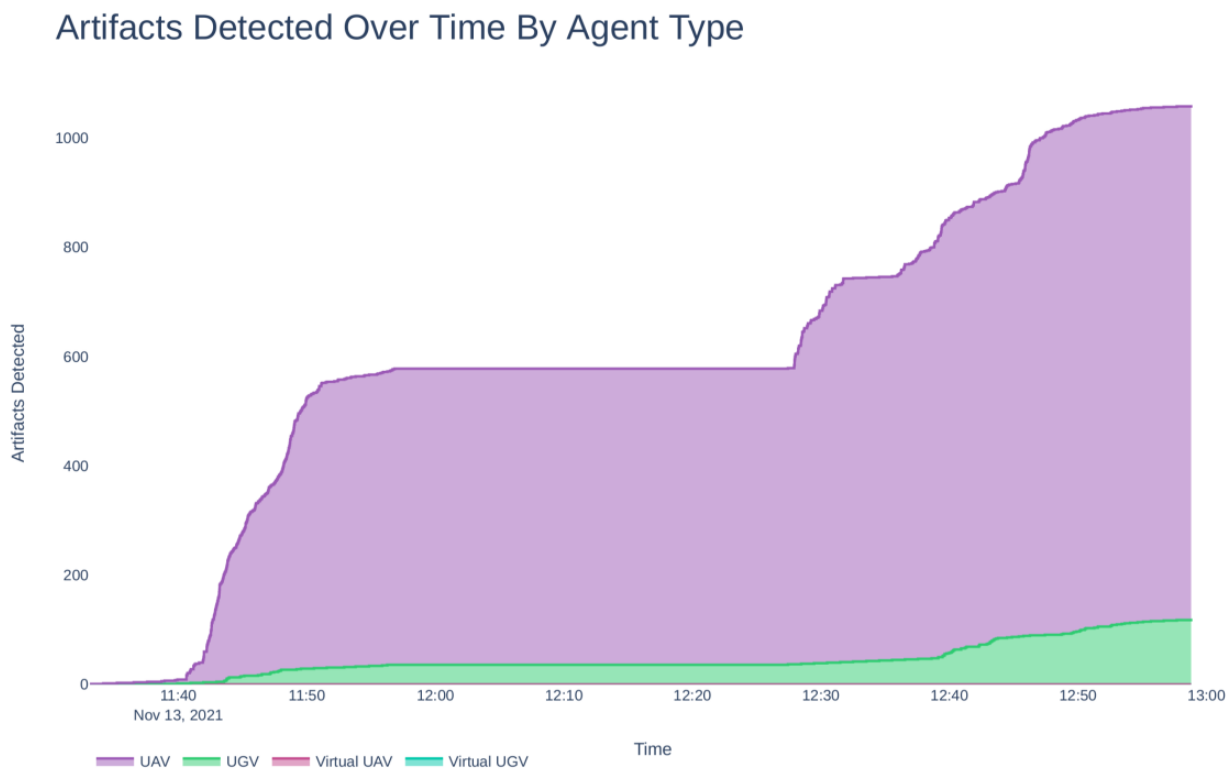


图28： NIWC 的编排软件在 FX-6 的 100 个智能体运行初期记录的独特物体检测事件。

12 讨论、经验教训与未来工作

借助RISE，我们旨在证明，专为快速集成新技术和新构想而设计的系统，可由单名操作员用于控制由自主智能体组成的大型异构群体。为此，我们结合自身作为集成商与第三方合作的经验，设计了一个生态系统；这些第三方专注于关键功能，而非整体系统。我们将这些经验提炼为第4.1节所定义的一系列需求，并结合OFFSET计划的项目需求，开发出一种将不同关注事项划分为四个核心模块的解决方案，分别涉及用户界面、战术、行为和算法。尤其重要的是，将战术单独划分为一个模块，使其能够在高于智能体本身的抽象层级上协调智能体；此外，还采用了基于草图的界面，通过草图语言描述操作。这两点是实现快速集成和一对多蜂群控制的关键。借助PyC2，战术开发时间从数周缩短至数小时。PyC2还使战术开发人员能够专注于操作员如何通过用户界面将战术付诸实施（而无需成为用户界面开发人员），并进一步关联到如何协调各个机器人以实现特定的战术目标。所有模块就绪后，我们得以快速编写战术，使操作员只需输入一条命令即可调用众多智能体；由于行为层面的自主性等因素，操作员无需操心单个机器人的操作。因此，RISE成功使单名操作员能够在真实世界中同时控制174个平台。未来，这或许能将士兵的控制幅度提高到1名士兵对应 150+ 个平台，有别于当今1:1的模式。我们的指挥与控制界面和战术开发工具为操作员提供了蜂群控制能力。

与大多数以往的群体协同研究不同，RISE 从一开始就在真实环境中使用硬件运行。我们已在 5 个军事靶场开展部署和测试，并计划在以下地点进行更多现场工作

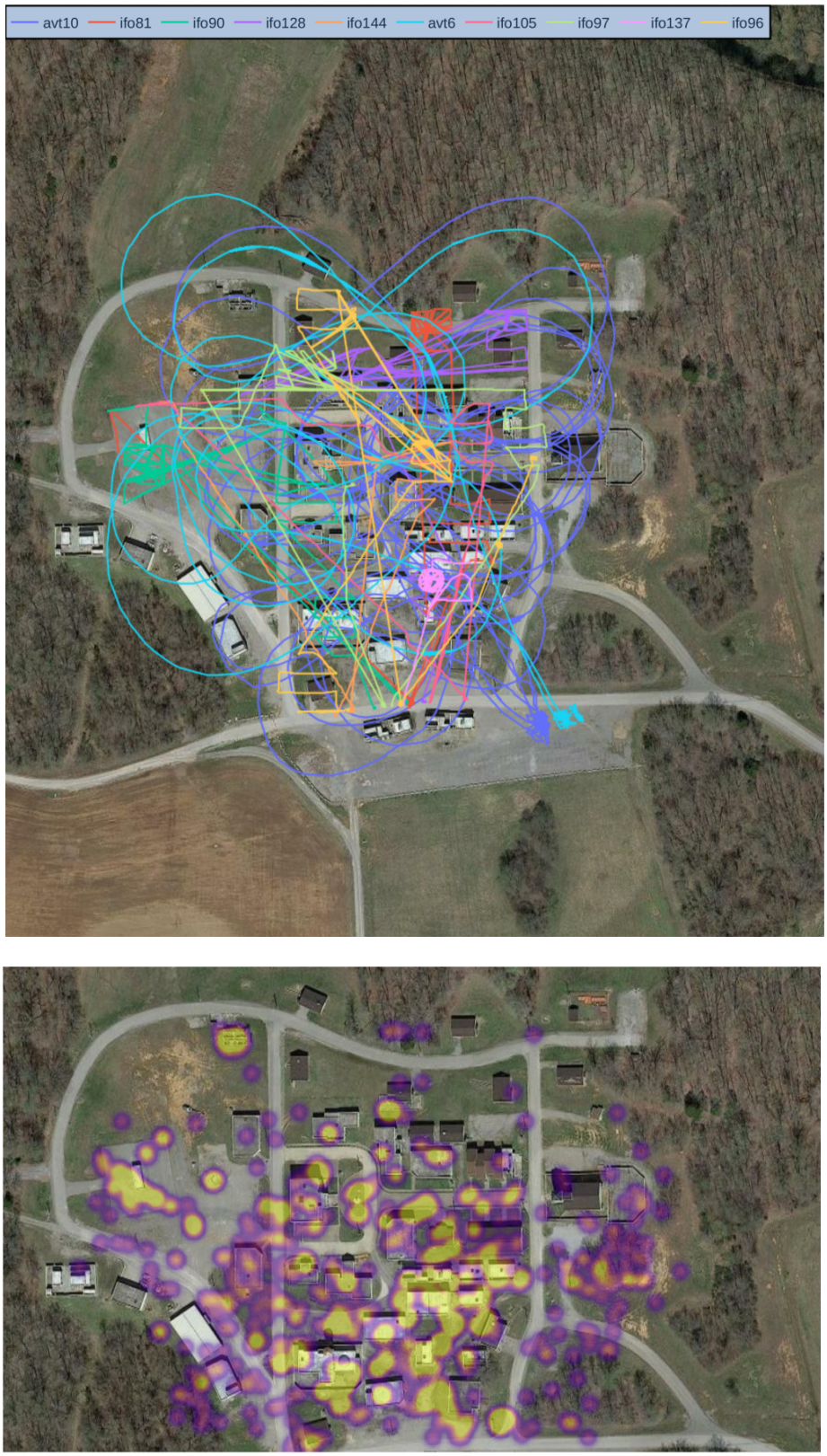


图29：在 FX-6 期间，NIWC 的编排软件记录的坎贝尔堡一次运行过程中的智能体轨迹（上）和覆盖范围（下）。

未来。大规模群体协同的野外机器人学面临诸多挑战，但群体后勤也是一个亟待开展新研究的领域。

在本次讨论中，我们重点介绍 Swarm Engine C2、战术开发和野外机器人学，并分别总结各主题的经验教训和未来工作。

12.1 集群指挥与控制

我们设计 Swarm Engine C2，旨在让单个操作员能够指挥大规模集群。通过实地实验，我们证明操作员能够在较长时间内持续执行任务，同时保持充分的态势感知。测试中出现的问题主要源于硬件错误和自动化程度不足。在这些情况下，操作员无法生成新的群体级任务，智能体因而处于空闲状态。这些情况是由程序和实验的性质所致，并不反映实际情况下的正常运行。改进机器人群体硬件（而非我们的 RISE 平台），再加上正在开发的新战术，将能够克服这些问题，使操作员得以按预期开展工作。待这些战术可用后，有必要开展完整的基于仿真的用户研究。

我们的用户界面设计为可自定义，因此操作员可以根据个人经历和联想选择容易记忆的手势。然而，我们发现，团队使用的是由实现相关功能特性的工程师定义的手势。众所周知，个人定义的开发者手势不如大型群体提出的手势受欢迎（Morris、Wobbrock和Wilson，2010年），因此有时会出现难以记忆的问题。也就是说，正如预期的那样，个人能够学会我们的界面命令，但这套手势并非最优。尽管自定义功能能在短期内实现快速原型开发，但从长远来看，随着群体战术日趋成熟并逐渐标准化，我们预计需要定期开展引出研究和一致性分析（Vatavu和Wobbrock，2015年）。因此，群体草图语法将像数学符号、电路图和军事图标一样实现规范化。

12.2 通过PyC2进行战术开发

PyC2 是 OFFSET 计划第二阶段期间后期开发出来的。在 PyC2 出现之前，战术一部分用指挥与控制代码编写，另一部分则用机器人技术代码编写，这种做法既繁琐、耗时，又有诸多限制。因此，PyC2 应运而生，旨在加快战术集成，并使非机器人学家也能够开发新战术（参见需求4）。PyC2 部署后，我们的团队迅速开发并迭代了众多战术，其中包括促成第三次和第四次实地实验之间性能大幅提升的战术。无需修改指挥与控制代码，借助草图和上下文即可编写新战术，这被证明是一项很有价值的工具。战术开发之所以放缓，完全是因为平台限制问题，以及团队需要转而处理其他优先事项。借助 PyC2，冲刺型机器人的集成也有所增加，详见第10节。

我们计划近期解决的一个关键问题是，PyC2 是一个断裂关键组件。这意味着，如果 PyC2 出现故障或无法与智能体通信，我们就无法指挥集群。我们的解决方案是让 PyC2 能够直接在机器人平台上运行，并改进竞标协议，以更好地支持协同任务分配。这样一来，PyC2 的战术、集成和开发不受影响，同时 PyC2 也将成为一个分布式系统。

12.3 野外机器人学

RISE 采用自底向上的开发方法，因此整体战术性能取决于智能体能否可靠地完成任务。当智能体无法可靠地执行任务时，就会引发连锁问题：其他智能体必须接手失败的任务，或者迫使蜂群指挥官重新考虑任务方案。大多数任务失败都是由于单个智能体无法准确定位并在环境中导航所致。导航与定位是机器人学领域持续研究的方向，尤其是在城市环境中。推进这些领域的发展，将进一步提升集群的效能，并在未来实现更加复杂的智能体行为。

随着 RISE蜂群规模不断扩大，后勤以及平台的整体维护（包括硬件和软件）成了一个重大难题。例如，随着时间推移，智能体的传感器或电机会发生损坏，需要排查故障和／或维修。从软件角度来看，每个平台运行的代码库版本各不相同，飞行控制器所用的固件也各有差异。所有在用平台上的这两类系统都需要保持最新，以确保各平台之间的集群作业一致。此外，平台的飞控单元（例如 Pixhawk）需要校准；尽管校准本身并非特别费时，但要为蜂群中的众多平台逐一校准，工作量就会非常大。另外，尽管从纸面上看，所有平台子组都完全相同，但我们发现各个平台之间缺乏一致性，这给平台开发增添了复杂性。通过使用 Ansible 和 Docker，我们为智能体代码库创建了易于管理的平台更新流程，并能根据所需更新的内容，在 5 至 15 分钟内将更新部署到 50 个资产上。在整个项目期间，固件更新和校准仍然是持续存在的痛点。未来的工作可能包括更直接地参与固件代码库，摆脱对这些子组件的依赖，或与制造商更紧密合作，帮助管理这些方面。

借助 FlexBE，我们为系统自下而上的战术开发方法奠定了坚实基础。状态开发的可复用性，以及基于底层算法或状态、通过简单拖放即可创建原语的方式，使我们能够快速创建新的智能体行为。然而，最终看来，发展方向已从层次状态机大幅转向行为树。理论上，FlexBE 可以支持这一功能，但系统目前尚不具备此功能。随着我们的系统和架构在过去 4 年中不断演进，我们发现 FlexBE 和 PyC2 将来可能会出现功能重复。例如，某些逻辑流究竟应该以原语的形式存在于智能体代码库中，还是应该置于 PyC2 的战术层级，目前仍未有定论。

设计 RISE 时，使用 ROS 是绝对必要的；如果再做一次选择，我们团队仍会选择 ROS。不过，我们希望 ROS 2 的迁移能更早进行。我们采用每一项新的 ROS 2 功能时，往往都稍微早了一些。我们认为，随着 ROS 2 日趋成熟，未来无需 ROS 主节点的去中心化概念将成为集群算法开发乃至集群开发领域的有力工具。

13 结论

我们的快速集成蜂群生态系统（RISE）为未来的多智能体研究和部署提供了一个平台。通过实体和仿真蜂群，我们展示了 RISE 快速集成第三方蜂群战术与行为的能力。我们的实体测试平台由 250 多个联网异构智能体组成，已在五个 CACTF 场地的模拟战争场景中接受了广泛测试。凭借实况、虚拟与构造（LVC）仿真能力，RISE 可同时使用虚拟和实体智能体。其他仿真方面的进展，例如“流体保真度”理念和超实时仿真，使蜂群战术原型设计得以快速实现。我们的基于手势的界面可实现超过 1:150 的操作员比例，使 RISE 成为强大的力量倍增器。这些功能相结合，使 RISE 能够以前所未有的便捷性和灵活性，将任务需求转化为机器人驱动和群体操作。

致谢

本项目部分由美国国防高级研究计划局（DARPA）通过合同 N66001-17-C-4066 资助。我们感谢 OFFSET计划项目经理 Timothy Chung 博士在该计划四年期间提供的专业指导和真知灼见的反馈，也感谢我们的合作伙伴诺斯罗普·格鲁曼、中佛罗里达大学和 Intelligent Automation, Inc.。我们还要感谢全球 ROS 社区以及所有开源机器人开发者。

参考文献

Abukhalil, T., Patil, M., Patel, S., & Sobh, T. (2016年4月). 使用基于机器人效用的任务分配（RUTA）协调异构机器人群. 收录于*2016*年 *IEEE* 第*14*届高级运动控制国际研讨会（*AMC*）（第57–62页）。doi: 10.1109/AMC.2016.7496328 Acker, D., Hellhake, P. R., Jordan, W., & Parks, J. E. (2016). 在高度机动的网状网络上进行多播的系统与方法。（美国专利 9319922B2）Aguero, C., Koenig, N., Chen, I., Boyer, H., Peters, S., Hsu, J., . . . Pratt, G. (2015年4月). 虚拟机器人挑战赛内幕：仿真实时机器人灾害响应. 《自动化科学与工程 *IEEE* 汇刊》，*12*(2), 494-506. doi: 10.1109/TASE.2014.2368997 Albani, D., IJ sselmuiden, J., Haken, R., & Trianni, V. (2017). 面向农业应用的机器人群监测与建图. 收录于*2017*年第*14*届 *IEEE* 高级视频与基于信号的监视国际会议（*AVSS*）（第1–6页）。Baumann, T. (2018). 使用 *3DVFH** 算法实现无人机避障（未发表的硕士论文）。苏黎世联邦理工学院。Bernstein, D. J. (1991). *djb2* 哈希函数。http://www.cse.yorku.ca/~oz/hash.html。（最后访问于2022年1月26日）Brossard, M., Barrau, A., & Bonnabel, S. (2020). AI-IMU 航位推算. 《*IEEE* 智能车辆汇刊》，*5*(4), 585–595. Calhoun, G., Ruff, H., Behymer, K., & Frost, E. (2018). 多无人载具控制的人类-自主系统协同界面设计考量. 《人体工程学科学理论问题》，*19*(3), 321–352. Caprari, G., & Siegwart, R. (2005). 即用型移动微型机器人：Alice. 收录于*2005*年 *IEEE/RSJ* 智能机器人与系统国际会议（第3295-3300页）。doi: 10.1109/IROS.2005.1545568 Chamanbaz, M., Mateo, D., Zoss, B. M., Toki´c, G., Wilhelm, E., Bouffanais, R., & Yue, D. K. P. (2017). 多机器人系统的群体赋能技术. 《机器人与人工智能前沿》，*4*。Chen, J. Y., & Barnes, M. J. (2021). 人机交互. 《人为因素与工效学手册》，1121–1142. Chung, T. H., Clement, M. R., Day, M. A., Jones, K. D., Davis, D., & Jones, M. (2016年5月). 面向大量固定翼无人机的实飞、大规模现场实验. 收录于*2016*年 *IEEE* 机器人与自动化国际会议（*ICRA*）（第1255–1262页）。doi: 10.1109/ICRA.2016.7487257 Cianci, C. M., Raemy, X., Pugh, J., & Martinoli, A. (2007). 微型机器人群中的通信：作为群体机器人学教育工具的 e-puck. 收录于 E. S¸ahin, W. M. Spears, & A. F. T. Winfield (编), 群体机器人学（第103–115页）。柏林、海德堡：Springer Berlin Heidelberg。Clark, S., Usbeck, K., Diller, D., & Schantz, R. E. (2021年3月). CCAST：异构无人系统群的框架与实际部署. 《*GetMobile*：移动计算与通信》，*24*(4), 17–26. doi: 10.1145/3457356.3457362 Crandall, J. W., Goodrich, M. A., Olsen, D. R., & Nielsen, C. W. (2005). 在多任务环境中验证人机交互方案. 《*IEEE* 系统、人类与控制论汇刊 *A*辑：系统与人类》，*35*(4), 438–449. Cummings, M. L., Nehme, C. E., Crandall, J., & Mitchell, P. (2007). 预测多无人机监督控制的操作员能力. 收录于智能机器创新-*1*（第11–37页）。施普林格。美国国防高级研究计划局（DARPA）. (2017年2月15日). 广泛机构公告：进攻性群体战术（*OFFSET*）（编号 HR001117S0011）。

Dasgupta, P., Baca, J., Guruprasad, K. R., Mu˜noz-Melendez, A., & Jumadinova, J. (2015). C OMRADE 系统：用于冲突后地区多机器人自主地雷探测.《机器人学杂志》，2015。doi: 10.1155/2015/921370 Davis, D. T., Chung, T. H., Clement, M. R., & Day, M. A. (2018). 用于蜂群对抗蜂群实验的多蜂群基础设施. 收录于 R. Groß 等（编），分布式自主机器人系统（第6卷，第649–663页）。Cham：施普林格国际出版。doi: 10.1007/978-3-319-73008-0 45 Deering, D. S. E. (1989年8月). IP 多播主机扩展（编号 1112）。RFC 1112。RFC 编辑部。取自 https://www.rfc-editor.org/info/rfc1112 doi: 10.17487/RFC1112 Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., . . . Vaussard, F. (2013年12月). Swarmanoid：研究异构机器人蜂群的新概念.《IEEE 机器人与自动化杂志》，20 (4), 60–71. doi: 10.1109/MRA.2013.2252996 Dorigo, M., Theraulaz, G., & Trianni, V. (2021年7月). 群体机器人学：过去、现在与未来 [观点].《IEEE 会议录》，109 (7), 1152–1165. doi: 10.1109/JPROC.2021.3072740 Drew, D. S. (2021). 搜救应用中的多智能体系统.《当前机器人学报告》，2 (2), 189–200. Edwards, S. J. (2000). 战场蜂群作战：过去、现在与未来. 兰德公司国防研究所，加利福尼亚州圣莫尼卡。Edwards, S. J. A. (2005). 蜂群作战与战争的未来（博士学位论文）。帕迪兰德研究生院，加利福尼亚州圣莫尼卡。Endsley, M. R. (1995). 面向动态系统态势感知理论的构建.《人为因素》，37 (1), 32–64. Engebr ̊aten, S., Nummedal, O. R., Gilbreath, D., Yakimenko, O., & Glette, K. (2019). 配备网状无线电的无人机蜂群：开发进展（技术报告）。EasyChair。取自 https://easychair.org/ publications/preprint/XxRx Fernandes, A., Couceiro, M. S., Portugal, D., Machado Santos, J., & Rocha, R. P. (2015). 使用 Zigbee 技术实现移动机器人团队的自组织通信.《工程教育中的计算机应用》，23 (5), 733-745. 取自 https://onlinelibrary.wiley.com/doi/abs/10.1002/ cae.21646 doi: https://doi.org/10.1002/cae.21646 Furuhashi, S. (2021). MessagePack。https://msgpack.org/。（最后访问于2022年1月26日）Giles, K., & Giammarco, K. (2017). 面向蜂群可组合性的基于任务的架构（ MASC）.《计算机科学 Procedia》，114，57–64. Hamer, M., & Ortega-Sanchez, C. (2010). 用于评估机器人蜂群算法的仿真平台. 收录于 TENCON 2010—2010年 IEEE 第10区会议（第1583–1588页）。Hammond, T. A., Logsdon, D., Paulson, B., Johnston, J., Peschel, J., Wolin, A., & Taele, P. (2010). 用于识别手绘行动方案图的草图识别系统. 收录于第二十二届 IAAI 会议。Harabor, D., & Grastien, A. (2011). 在网格地图上寻路的在线图剪枝. 收录于 AAAI 人工智能会议论文集（第25卷）。Hocraffer, A., & Nam, C. S. (2017). 无人机蜂群管理中人机系统界面的荟萃分析.《应用人体工程学》，58，66–80. Hoebeke, J., Moerman, I., Dhoedt, B., & Demeester, P. (2004). 移动自组织网络概述：应用与挑战.《通信网络杂志》，3 (3), 60–66. How, J. P., Behihke, B., Frank, A., Dale, D., & Vian, J. (2008年4月). 实时室内自主车辆测试环境.《IEEE 控制系统杂志》，28 (2), 51–64. doi: 10.1109/MCS.2007.914691 互联网协议（编号 791）。（1981年9月）。RFC 791。RFC 编辑部。取自 https://www.rfc -editor.org/info/rfc791 doi: 10.17487/RFC0791 Jevti´c, A., Gazi, P., Andina, D., & Jamshidi, M. (2010). 构建一群机器人蜜蜂. 收录于 2010年世界自动化大会（第1-6页）。Jim´enez-Gonz´alez, A., Martinez-de Dios, J. R., & Ollero, A. (2013年12月). 泛在机器人学测试平台：综述.《机器人与自主系统》，61 (12), 1487–1501. doi: 10.1016/j.robot.2013 .07.006 Johnson, D., Stack, T., Fish, R., Flickinger, D. M., Stoller, L., Ricci, R., & Lepreau, J. (2006年4月). Mobile Emulab：机器人无线与传感器网络测试平台. 收录于 IEEE INFOCOM 会议录

*2006*年。第*25*届电气电子工程师学会国际计算机通信会议（第1–12页）。doi: 10.1109/INFOCOM.2006.182 Johnson, L., Ponda, S., Choi, H.-L., & How, J. (2010). 通过异步通信提高无人机团队分布式任务分配算法的效率。载于*AIAA*制导、导航与控制会议（第8421页）。Kaiser, T. K., Begemann, M. J., Plattenteich, T., Schilling, L., Schildbach, G., & Hamann, H. (2022年5月). ROS2SWARM——用于群体机器人行为的 ROS 2 软件包。载于*2022*年国际机器人与自动化会议（*ICRA*）（第6875–6881页）。doi: 10.1109/ICRA46639.2022.9812417 Kolling, A., Walker, P., Chakraborty, N., Sycara, K., & Lewis, M. (2015). 人与机器人群的交互：综述。*IEEE*人机系统汇刊，*46*（1），9–26。Konolige, K., Fox, D., Ortiz, C., Agno, A., Eriksen, M., Limketkai, B., . . . Vincent, R. (2006). Centibots：超大规模分布式机器人团队。载于 M. H. Ang & O. Khatib（编），实验机器人学 *IX*（第131–140页）。柏林，海德堡：Springer Berlin Heidelberg。Kornienko, S., Kornienko, O., & Levi, P. (2005). 微型机器人群体通信与感知的极简方法。载于*2005*年电气电子工程师学会/日本机器人学会智能机器人与系统国际会议（第2228-2234页）。doi: 10.1109/IROS.2005.1545594 LeFavor, P. D. (2020). 美国陆军小部队战术手册。载于（第Tactics章）。Blacksmith LLC。Lewis, M. (2013). 人与多台远程机器人的交互。《人为因素与工效学评论》，*9*（1），131–174。Lewis, M., Wang, J., & Scerri, P. (2006). 面向真实复杂多机器人系统的团队协作协调。载于北约无人驾驶军用车辆作为力量倍增器的人因研讨会（第1–12页）。McLurkin, J., Smith, J., Frankel, J., Sotkowitz, D., Blau, D., & Schmidt, B. (2006). 群体机器人交流语言：面向大型自主移动机器人群的人机交互界面设计。载于*AAAI*春季研讨会：勇闯无人类—机器人团队涉足之境（第72–75页）。Mi, Z.-Q., & Yang, Y. (2013). 无人载具系统集群中的人机交互：综述。《国际计算机科学问题期刊》（*IJCSI*），*10*（第2期第1部分），273。Miller, C., Pelican, M., & Goldman, R. (2000). 用于灵活交互的“任务分配”界面：让操作员保持控制权。载于复杂系统人机交互会议论文集（第123–128页）。Miller, I. D., Cladera, F., Cowley, A., Shivakumar, S. S., Lee, E. S., Jarin-Lipschitz, L., . . . Kumar, V. (2020). 使用多台四足机器人探索矿井隧道。*IEEE*机器人与自动化快报，*5*（2），2840-2847。doi: 10.1109/LRA.2020.2972872 Morris, M. R., Wobbrock, J. O., & Wilson, A. D. (2010). 了解用户对表面手势的偏好。载于*2010*年图形界面会议论文集（第261–268页）。Nacenta, M. A., Kamber, Y., Qiang, Y., & Kristensson, P. O. (2013). 预设手势集与用户定义手势集的易记性。载于*SIGCHI*计算系统人因会议论文集（第1099–1108页）。Object Management Group, I. (2015). 数据分发服务。取自 https://www.omg.org/ spec/DDS/（最后访问日期：2022年1月26日）Oleynikova, H., Honegger, D., & Pollefeys, M. (2015). 使用嵌入式立体视觉实现微型飞行器的反应式避障。载于*2015*年电气电子工程师学会国际机器人与自动化会议（*ICRA*）（第50-56页）。doi: 10.1109/ICRA.2015.7138979 Olsen, D. R., & Goodrich, M. A. (2003). 评估人机交互的指标。载于*PERMIS*论文集（第2003卷，第4页）。Ospina, N. I., Mojica-Nava, E., Jaimes, L. G., & Calder´on, J. M. (2021年1月). ARGroHBotS：一种经济且可复现的地面同质机器人群体测试平台。*IFAC-PapersOnLine*，*54*（13），256–261。doi: 10.1016/j.ifacol.2021.10.455 Pickem, D., Glotfelter, P., Wang, L., Mote, M., Ames, A., Feron, E., & Egerstedt, M. (2017年5月). The Robotarium：可远程访问的群体机器人研究测试平台。载于*2017*年*IEEE*机器人与自动化国际会议（*ICRA*）（第1699–1706页）。doi: 10.1109/ICRA.2017.7989200 Pickem, D., Lee, M., & Egerstedt, M. (2015年5月). GRITSBot的自然栖息地——多机器人测试平台。载于*2015*年*IEEE*机器人与自动化国际会议（*ICRA*）（第4062–4067页）。doi: 10.1109/ICRA.2015.7139767

Preiss, J. A., Honig, W., Sukhatme, G. S., & Ayanian, N. (2017年5月). Crazyswarm：大型纳米四旋翼飞行器集群。载于*2017*年*IEEE*机器人与自动化国际会议（*ICRA*）（第3299–3304页）。doi: 10.1109/ICRA.2017.7989376 Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). 只看一次：统一的实时目标检测。载于*IEEE*计算机视觉与模式识别会议论文集（第779–788页）。Rizk, Y., Awad, M., & Tunstel, E. W. (2019年4月). 协作异构多机器人系统：综述。*ACM*计算调查，*52*（2），29:1–29:31。doi: 10.1145/3303848 Rubenstein, M., Cornejo, A., & Nagpal, R. (2014). 千机器人群中的可编程自组装。《科学》，*345*（6198），795-799。取自 https://www.science.org/doi/abs/10.1126/ science.1254295 doi: 10.1126/science.1254295 Schillinger, P., Kohlbrecher, S., & von Stryk, O. (2016年5月). 人机协作高级控制及其在救援机器人学中的应用。载于电气电子工程师学会国际机器人与自动化会议。瑞典斯德哥尔摩。Schranz, M., Umlauft, M., Sende, M., & Elmenreich, W. (2020). 群体机器人行为及当前应用。《机器人与人工智能前沿》，*7*。Self, M., Miller, B., & Dixon, D. (2005). 城市行动与全球反恐战争中针对人类目标的采集级别定义及可观测指标（技术报告）。美国陆军通信电子司令部贝尔沃堡夜视与电子传感器局。取自 https://apps.dtic.mil/sti/citations/ADA442798 Shah, S., Dey, D., Lovett, C., & Kapoor, A. (2017). *AirSim*：面向自主车辆的高保真视觉与物理仿真。Speakman, T., Crowcroft, J., Gemmell, J., Farinacci, D., Lin, S., Leshchiner, D., . . . Vicisano, L. (2001年12月). *PGM*可靠传输协议规范（第3208号）。RFC 3208。RFC 编辑部。取自 https://www.rfc-editor.org/info/rfc3208 doi: 10.17487/RFC3208 Stubbs, A., Vladimerou, V., Fulford, A., King, D., Strick, J., & Dullerud, G. (2006年6月). 基于网络的多车辆系统控制：用于联网与去中心化控制的气垫船测试平台。《*IEEE* 控制系统杂志》，*26*（3），56–69。doi: 10.1109/MCS.2006.1636310 Taranta, E. M., II, Samiei, A., Maghoumi, M., Khaloo, P., Pittman, C. R., & LaViola, J. J., Jr. (2017). Jackknife：一种样本少、模态多且可靠的识别器。载于*2017*年*CHI*计算系统人因会议论文集（第5850–5861页）。美国陆军。（2015）。野战手册 *FM 3-90-1* 进攻与防御，第*1*卷，修订*2*，*2015*年*4*月。CreateSpace Independent Publishing Platform。取自 https://books.google.com/books ?id=wJJhrgEACAAJ 美国陆军。（2020）。野战手册 *FM 1-02.2* 军事符号，*2020*年*11*月。独立出版。取自 https://books.google.com/books?id=I4gKzgEACAAJ用户数据报协议（第768号）。（1980年8月）。RFC 768。RFC 编辑部。取自 https:// www.rfc-editor.org/info/rfc768 doi: 10.17487/RFC0768 Van der Hoek, W., & Wooldridge, M. (2008). 多智能体系统。载于人工智能基础（第3卷，第887–928页）。Elsevier。doi: 10.1016/S1574-6526(07)03024-6 Vatavu, R.-D., & Wobbrock, J. O. (2015). 将引出研究中的一致性分析形式化：新度量、显著性检验与工具包。载于第*33*届*ACM*计算系统人因年会论文集（第1325–1334页）。Villemure, ´E., Arsenault, P., Lessard, G., Constantin, T., Dub´e, H., Gaulin, L.-D., . . . Ferland, F. (2022年3月). *SwarmUS*：用于群体机器人学开发的开放式硬件与机载软件平台（第 arXiv:2203.02643 号）。arXiv。doi: 10.48550/arXiv.2203.02643 Wang, H., Lewis, M., Velagapudi, P., Scerri, P., & Sycara, K. (2009). 搜索及其子任务如何随机器人数量扩展。载于第四届*ACM/IEEE*人机交互国际会议论文集（第141–148页）。Wang, J., & Olson, E. (2016年10月). AprilTag 2：高效且稳健的基准标记检测。载于*IEEE/RSJ*智能机器人与系统国际会议（*IROS*）论文集。Williamson, B. M., Taranta II, E. M., Moolenaar, Y. M., & Laviola Jr., J. J. (2023). 使用自然人机界面实现大规模集群的指挥与控制。野外机器人学，*2*，1—22。

Woodall, W. (2014). 基于 *ZeroMQ* 的 *ROS* 及相关项目。 https://design.ros2.org/articles/ros with zeromq .html.（最后访问于 2022年1月26日）Zahugi, E. M. H., Shabani, A. M., & Prasad, T. V. (2012). Libot：面向户外群体机器人学的低成本移动机器人设计。载于*2012*年 *IEEE*自动化、控制与智能系统网络技术国际会议（网络技术）（第 342-347 页）。doi: 10.1109/CYBER.2012.6392577 ZeroMQ。（2021）。*ZeroMQ*：一种开源通用消息传递库。 取自 https://zeromq .org/（最后访问于 2022年1月26日）

附录 A 完整战术示例

清单 3 展示了完整的保持位置策略。操作员使用此策略，将多个智能体移动到围绕感兴趣物体或结构绘制的路径上，并使所有智能体朝向内侧。文中提供了详细注释，便于理解。

附录B 网络参数

我们将多个 PGM参数从默认值进行了调整，以便更高效地利用有限的网络资源。首先，我们将环境 SPM速率降低至 $_{1\ 30}$ Hz，并且仅在最后一次数据传输结束 30 秒后发送一个结束传输的“Heartbeat”SPM。最重要的是，我们将 NAK退避间隔增加至两秒，将 NAK重复间隔增加至 10 秒，并将允许的 NAK重试次数减少至 3。我们还将 NAK修复数据间隔增加至 800 毫秒，并将修复数据速率上限设为 1 KB/s。这些调整共同避免了在严重通信干扰情况下发生洪泛和网络崩溃，同时仍能保持可接受的可靠性。

```
类HoldPosition（战术）：
    2 3 # 注册期间发送至指挥与控制的战术描述数据。4 战术定义 ＝战术定义（5 ’保持位置’，6 ’将一组智能体移动到周界上的各点并保持位置。’，7 ’正方形’，8 上下文 ＝SketchInputTypes . 路径，9 参数 ＝ [10 战术参数（11 ’高度’，12 ’智能体将保持的高度，单位为米。’，13 浮点数，14 6），15 战术参数（16 ’持续时间’，17 ’保持位置的时间，以秒为单位。’，18 浮点数，19 10000），20 战术参数（21 ’智能体数量’，22 ’沿周界布置的智能体数量。’，23 整数，24 8）] ,25 需要 ＝[MoveTo ]）26 27 定义（自身，任务）： 28 """ """ 29 super（）。初始化（任务）30 31 # 投影到 XZ 地面平面
32                                  ＝                                  [       0 , 2   ] 折线 任务.参数.上下文.
33                                  ＝
点: ， （）数量 任务.参数.智能体数量 34 位置 ＝重采样（折线，整数（数量 ＋ 1））35 36 # 去掉最后一个点，因为它与第一个点重复 37 位置 ＝位置 [: −1 ]38 39 # 根据路径的绕行方向，获取每个点朝内的航向 40 # 41 向量 ＝获取朝内向量（位置）42 航向度数 ＝向量转航向度数（向量）43 44 # 为每个保持位置创建一个任务 45 对于点，航向度数，遍历 zip（位置，航向度数）： 46 点 ＝ [点 [0] , 任务.参数.高度，点 [1] ]47 48 # 使用机器人 MoveTo 基元向智能体发送命令。49 子任务 ＝MoveTo（50 父级 ＝自身，51 位姿_xyz ＝ [点 ] ,52                                  ＝ [                    ] , 航向度数 航向度数 持续时间_秒
53
                                  ＝ [                                    ] , 任务.参数.持续时间
54                                  ＝
已选智能体 任务.已选智能体）55 56 自身.任务 +＝ [子任务 ]57 58 # 将所有子任务发送给任务管理器 59 自身.执行（）
```

清单 3： 在 PyC2 中实现的完整战术。