基于蚁群算法的未知环境主动探索与建图

Active Exploration and Mapping in Unknown Environments via Ant Colony Optimization

2300019005 李昀濠　　2200012181 陆尧
北京大学
北京, 中国

摘要—自主探索未知环境并构建地图是移动机器人的核心能力之一，在灾后救援、星球探测等领域具有重要应用价值。本次大作业，我们旨在研究并实现一套基于蚁群优化（Ant Colony Optimization, ACO）的主动探索与建图方法。我们首先将经典 ACO 算法应用于单机器人的主动构图问题（ACO-Mapping），通过引入信息增益作为启发式信息来引导探索。随后，我们将此框架扩展至多智能体协同探索场景（Multi-ACO），通过设计一个共享的全局信息素地图和目标协调机制，实现了高效的分布式任务分配。此外，我们还设计了利用神经网络学习启发式函数的 DeepACO 和适应真实定位不确定性的 ACO-SLAM。实验结果表明，我们提出的基础和多智能体 ACO 方法在多种模拟环境中均表现出优越的探索效率和鲁棒性，为基于群体智能的机器人主动探索问题提供了一个系统性的研究框架和富有潜力的解决方案。

***Index Terms*—主动 SLAM，机器人探索，蚁群优化，多机器人系统，协同建图**

I. 引言

移动机器人在无先验知识的未知环境中执行任务的能力，是机器人学领域一个长期存在且充满挑战的研究方向。其中，核心任务之一是“探索与建图”，即机器人需要自主地规划路径，以最有效的方式遍历环境，同时利用传感器数据构建一张精确的环境地图。这项技术在诸多领域至关重要，例如在 GPS 信号无法覆盖的灾后废墟中进行搜救 [1]、对外星球表面进行科学考察、以及对矿井或洞穴等复杂室内空间进行测绘。

传统的主动探索方法大多基于“前沿点（Frontier-based）”理论 [2]，机器人不断地移动到已知空间与未知空间的边界。这类方法的决策通常依赖于一个效用函数，该函数权衡了到达前沿点的代价和可能获取的信息增益，如 IGE (Information Gain-based Exploration) [3] 和 URE (Utility-based Robotic Exploration) [5]。然而，这些基于局部最优决策的贪心策略有时会使机器人陷入局部困境或在多智能体场景下产生冲突，导致探索效率低下。

为了克服这些限制，我们转向了受生物启发的元启发式算法。蚁群优化（ACO）是一种模拟蚂蚁觅食行为的概率性技术，因其在解决复杂组合优化问题中表现出的鲁棒性和分布式特性而闻名。我们认为，ACO 的内在机制——通过信息素进行正反馈和协同决策——非常适合解决主动探索问题。

本文的主要贡献如下：

- 我们设计并实现了一种将 ACO 用于单机器人主动构图的新框架（ACO-Mapping），将信息增益整合到 ACO 的决策过程中。
- 我们将其成功扩展为一个高效的多智能体协同探索框架（Multi-ACO），通过共享信息素地图实现了隐式的任务分配和冲突避免。
- 我们系统地探讨了该框架的两种高级扩展：(1) DeepACO，利用神经网络学习启发式函数；(2) ACO-SLAM，将方法推广到存在定位不确定性的完整 SLAM 问题。
- 我们通过在一系列模拟地图上进行仿真实验，验证了所提方法的有效性，并与多种主流方法进行了性能对比。

II. 相关工作

A. 基于前沿点的主动探索

Yamauchi 在 1997 年首次提出了基于前沿点（frontier-based）的探索方法 [2]。其核心思想是，地图中已知自由空间与未知空间的边界（即前沿）是获取新信息最有效的位置。机器人不断地检测所有前沿点，

并选择一个前往。后续研究主要集中在如何 “选择最优前沿点”。

IGE (Information Gain-based Exploration) 是一种典型的方法，它通过一个效用函数来评估每个前沿点 f_i。该方法旨在以最小的代价获取最大的信息量。其信息增益 (IG) 定义为机器人移动到 f_i 后，其传感器范围内新发现的未知区域面积。路径成本 C 则为机器人从当前位置 p_{rob} 到 f_i 的最短路径长度（通常用 A* 算法计算）。效用函数 $U(f_i)$ 定义为：

$$U(f_i) = \frac{IG(f_i)}{C(p_{\text{rob}}, f_i)} \tag{1}$$

机器人将选择效用函数值最大的前沿点 f^* 作为下一个目标。

URE (Utility-based Robotic Exploration) 是对 IGE 的改进。它不仅考虑了信息增益，还考虑了地图构建的完整性和准确性。为了避免机器人反复访问某些信息量大的区域而忽略其他区域，URE 引入了一个惩罚项，该惩罚项与一个前沿点被观察到的次数相关。在我们的实现中（见 ure_planner.py），当所有探索目标的效用低于一个阈值时，算法会进入 “巩固模式”，主动选择前往那些已知但观察次数较少的区域，以提高地图的整体覆盖质量。

B. 蚁群优化算法

蚁群优化（ACO）由 Dorigo 等人在上世纪 90 年代提出 [4]，最初用于解决旅行商问题（TSP）。ACO 的核心是利用信息素（pheromone）这一间接通信机制进行协同搜索。蚂蚁在路径上留下信息素，路径越短，蚂蚁往返越快，信息素积累也越快，从而形成正反馈，吸引更多的蚂蚁选择这条路径。同时，信息素会随时间挥发，避免算法过早收敛到局部最优解。由于其强大的全局搜索能力和鲁棒性，ACO 已被广泛应用于路径规划、调度问题和网络路由等领域。将其应用于机器人探索，是一个充满潜力的方向。

III. 问题建模

我们将机器人探索的未知环境建模为一个二维离散栅格地图 M。地图中的每一个栅格单元 $c \in M$ 具有一个状态 $S(c)$，其取值范围为：S(c) ∈ {KNOWN_FREE, KNOWN_OBSTACLE, UNKNOWN}

机器人的状态由其在地图上的位置 $p_t = (x_t, y_t)$ 在时刻 t 定义。机器人配备一个虚拟传感器，该传感器具有有限的探测半径 R_{sensor} 和 360 度的视场。在我们的robot.py实现中，我们通过从机器人位置向其感知边界发射光线（使用 Bresenham 直线算法），并中止于遇到的第一个障碍物，来精确模拟带遮挡效应的视线 (Line-of-Sight, LoS)。在任意位置 p，其感知范围 $V(p)$ 内的所有可视栅格 c 的状态会被更新。

主动探索的目标可以形式化地描述为一个优化问题：寻找一系列的机器人动作序列 $A = \{a_0, a_1, ..., a_T\}$，使得在总步数或总时间 T 的限制下，最大化探索到的区域面积，即最大化状态为 KNOWN_FREE 或 KNOWN_OBSTACLE 的栅格数量。

$$\max_{A} \sum_{c \in M} \mathbb{I}(S_T(c) \neq \text{UNKNOWN})$$

其中 $\mathbb{I}(\cdot)$ 是指示函数，$S_T(c)$ 是在执行完动作序列 A 后栅格 c 的状态。

IV. 经典蚁群算法回顾

在经典 ACO 中，一群人工蚂蚁在图的节点间移动来构建解。在路径规划问题中，图的节点就是地图上的栅格。第 k 只蚂蚁从节点 i 选择移动到邻居节点 j 的概率 $P_{ij}^k(t)$ 由以下公式决定：

$$P_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta}, \quad \text{if } j \in N_i^k \tag{2}$$

其中：N_i^k 是蚂蚁 k 在节点 i 处的可选邻居集合，$\tau_{ij}(t)$ 是边 (i, j) 上的信息素浓度，η_{ij} 是启发式信息（通常为距离的倒数），α 和 β 是控制两者相对重要性的参数。信息素会随时间挥发并根据蚂蚁构建的解的质量进行增强。

V. 基于 ACO 的主动构图 (ACO-Mapping)

为了将 ACO 应用于没有固定 “终点” 的主动探索任务，我们设计了一个创新的混合框架，该框架并非让蚂蚁在栅格地图上进行底层路径规划，而是将 ACO 作为一个高层决策模块，用于在所有可行的探索目标（前沿点）中进行选择。这充分利用了 ACO 的全局优化能力来克服传统前沿点方法的贪心局限性。算法的具体实现（见aco_planner.py）遵循以下步骤：

1. 探索决策空间定义：前沿点识别

在每次决策循环开始时，算法首先调用find_frontiers方法，扫描当前已知的地图known_map，找出所有状态为FREE且至少与一个UNKNOWN栅格相邻的单元，这

些单元构成了前沿点集合 $F = \{f_1, f_2, ..., f_n\}$。这个集合就是 ACO 算法本次决策的“城市”列表，即所有潜在的探索目标。

2. 启发式信息定义：目标效用计算 (η)

与传统 ACO 使用距离倒数作为启发式信息不同，我们为每个前沿点 $f_j \in F$ 定义了一个更符合探索任务目标的启发式值 η_j。这个值综合了信息增益和路径成本，其计算方式（见_get_heuristic_value_and_path方法）如下：

$$\eta_j = \frac{w_{ig} \cdot IG(f_j) + \epsilon}{C(p_{\text{rob}}, f_j) + \epsilon} \tag{3}$$

其中：

- $IG(f_j)$ 是预期的**信息增益**，计算为如果机器人移动到前沿点 f_j，其传感器范围内新发现的UNKNOWN栅格数量。
- $C(p_{\text{rob}}, f_j)$ 是从机器人当前位置 p_{rob} 到 f_j 的**路径成本**。我们使用 A* 算法（a_star_search）在当前known_map上计算最短路径，成本即为路径长度。如果一个前沿点不可达，其路径成本为无穷大，启发式值为 0。
- w_{ig} 是一个权重参数（ig_weight_heuristic），用于调节信息增益的重要性。
- ϵ 是一个小的正常数，以避免分母为零。

这个启发式值 η_j 定量地描述了前往前沿点 f_j 的“性价比”，成为引导蚂蚁选择的直接依据。

3. ACO 迭代搜索与选择

算法启动一个包含 N_{ants} 只虚拟蚂蚁的种群，进行 $N_{\text{iterations}}$ 次迭代。在每次迭代中，每只蚂蚁根据概率选择一个前沿点作为其本次的目标。第 k 只蚂蚁选择前沿点 f_j 的概率由以下状态转移规则决定：

$$P(f_j|k) = \frac{[\tau_j(t)]^\alpha [\eta_j]^\beta}{\sum_{l \in F_{\text{reachable}}} [\tau_l(t)]^\alpha [\eta_l]^\beta} \tag{4}$$

其中：

- $F_{\text{reachable}}$ 是所有可达的前沿点集合。
- $\tau_j(t)$ 是在第 t 次迭代时，与前沿点 f_j 关联的**信息素浓度**。
- η_j 是根据公式(3)计算出的启发式值。
- α 和 β 是控制信息素和启发式信息相对重要性的参数。

为了平衡探索与利用，我们的实现还引入了 q_0 参数。蚂蚁有 q_0 的概率选择当前效用最高（即 $\tau^\alpha \eta^\beta$ 最大）的前沿点（利用），有 $1 - q_0$ 的概率根据公式(4)进行轮盘赌选择（探索）。

4. 信息素更新机制

在所有蚂蚁完成选择后，信息素地图进行更新，这是算法学习和优化的核心。

1) **全局挥发**：所有前沿点上的信息素都以一个固定的挥发率 ρ 进行衰减：

$$\tau_j(t+1) \leftarrow (1-\rho) \cdot \tau_j(t), \quad \forall f_j \in F_{\text{reachable}} \tag{5}$$

2) **信息素增强**：每只选择了前沿点 f_j 的蚂蚁，都会在该前沿点上增加信息素。关键在于，增加的信息素量 $\Delta\tau_j^k$ 正比于该前沿点的启发式值（即效用）：

$$\Delta\tau_j^k = \eta_j \tag{6}$$

因此，完整的更新公式为：

$$\tau_j(t+1) \leftarrow (1-\rho) \cdot \tau_j(t) + \sum_{k \text{ chose } j} \eta_j \tag{7}$$

这个机制形成了一个正反馈循环：效用高（信息增益大、成本低）的前沿点会吸引更多蚂蚁，从而积累更多信息素，进而在后续迭代中以更大概率被选中。

5. 最终目标决策

经过 $N_{\text{iterations}}$ 次迭代后，信息素分布趋于稳定，反映了蚁群对各个前沿点探索价值的集体判断。机器人最终以确定性的方式选择目标，即选择能使 $\tau^\alpha \eta^\beta$ 值最大的前沿点 f^*：

$$f^* = \arg \max_{f_j \in F_{\text{reachable}}} ([\tau_j(T_{\text{final}})]^\alpha [\eta_j]^\beta) \tag{8}$$

选定 f^* 后，机器人便沿着 A* 算法预先计算好的路径向该目标移动。

A. 实验结果

我们在一个基于 Python 的 2D 栅格化仿真平台上进行了实验，以评估我们提出的 ACO-Mapping 算法的性能。

1) 实验设置: **地图环境：**我们使用了多种地图进行测试，包括：

- **自制地图：**通过交互式绘制工具创建，包含长廊、开放空间和复杂迷宫结构。
- **随机地图：**通过在environment.py中随机生成障碍物来创建。我们测试了不同尺寸（如 50x50, 100x100）和不同障碍物密度（如 0.15, 0.30）的地图。

对比算法：我们将我们的 ACO-Mapping 算法与以下三种主流的探索算法进行了比较，所有算法均在planners目录下实现：

- **FBE (Frontier-Based Exploration):** 基础的前沿点探索，总是选择最近的前沿点。
- **IGE (Information Gain-based Exploration):** 考虑信息增益和成本的探索。
- **URE (Utility-based Robotic Exploration):** IGE 的改进版，能主动进行地图巩固。

机器人与传感器参数：

- 传感器半径 R_{sensor}: 5 个栅格单位。
- 传感器模型：如robot.py中实现，考虑了障碍物遮挡的 Line-of-Sight。

评价指标：主要评价指标是**已探索区域百分比**随**机器人实际移动步数**变化的曲线。曲线越快达到饱和，说明探索效率越高。所有实验由main_simulation.py脚本驱动，并自动生成对比图表。

*2) 结果与分析：*本节展示的实验结果，其中的‘ACO‘算法均指代我们在上一节中详细描述的 ‘ACO-Mapping‘实现。

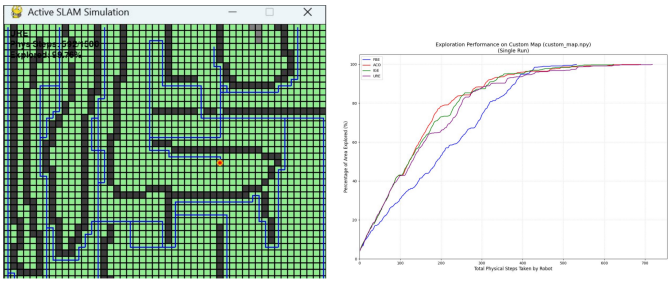

图 1: 在自制地图上的单次运行表现。左侧为 ACO 算法的探索轨迹，右侧为四种算法的探索效率对比曲线。

图 1展示了在自制复杂地图上的一次典型运行结果。从左侧的探索轨迹可以看出，ACO 算法生成的路径（蓝色）能够有效地覆盖地图的各个角落。右侧的性能曲线图显示，ACO（红色曲线）的探索效率显著优于基础的 FBE，并且与和 IGE 和 URE 性能相当，甚至在探索后期表现更优。这表明 ACO 的全局搜索能力帮助机器人避免了陷入局部最优（如在迷宫的某个分支中来回打转）。

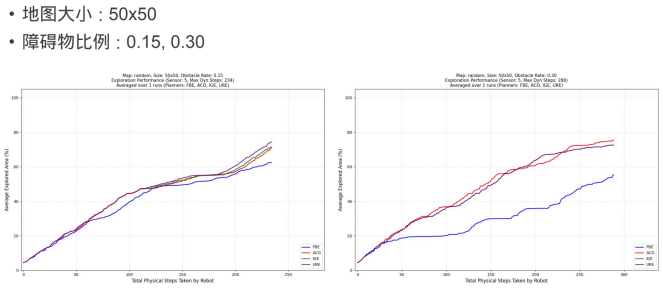

图 2: 在 50x50 随机地图上的性能对比。左：障碍物密度 0.15。右：障碍物密度 0.30。

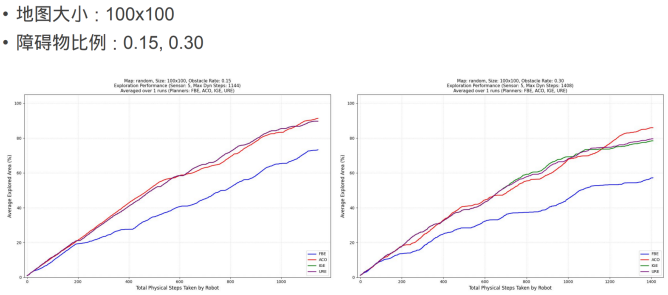

图 3: 在 100x100 随机地图上的性能对比。左：障碍物密度 0.15。右：障碍物密度 0.30。

图 2和图 3展示了在不同尺寸和障碍物密度的随机地图上的平均性能。实验结果具有一致性：

- 在所有测试场景中，ACO 和 URE 的性能都稳定地处于第一梯队，远超 FBE。
- 随着地图尺寸和复杂度的增加（从 50x50 到 100x100，从 0.15 密度到 0.30 密度），ACO 相对于 IGE 和 URE 的优势愈发明显。这进一步印证了 ACO 在处理复杂问题时的全局优化能力。
- 在某些情况下，特别是在障碍物较多的复杂环境中（如 100x100，0.30 密度），ACO 的最终覆盖率和速度显著优于 URE，这可能是因为 URE 的巩固机制有时会过于保守，而 ACO 的概率性选择使其能更大胆地探索潜在的高价值区域。

总的来说，实验结果有力地证明了我们提出的 ACO-Mapping 方法在主动探索任务中的有效性和高效性。

VI. 基于深度学习的启发式函数优化（DeepACO）

在公式(4)中，启发式信息 η_{ij} 对 ACO 的性能至关重要。传统上，η_{ij} 是一个基于简单规则（如距离）的人工设计函数。然而，最优的启发式信息应该能更精准

地预测一个动作的长期价值。DeepACO 的核心思想是用一个深度神经网络来学习这个启发式函数 η_θ。

设想实现：我们可以设计一个深度神经网络作为启发式模型。该网络的输入是当前的局部地图状态（一个以边界点为中心的子图），经过一个 CNN 卷积神经网络压缩得到具有边界点周围空间特征的向量，再与全局信息或局部标量信息进行融合，经过 MLP 输出对于每个可行动作（即移动到邻居节点 j）的启发值 $\eta_\theta(j|M_{\text{local}})$。

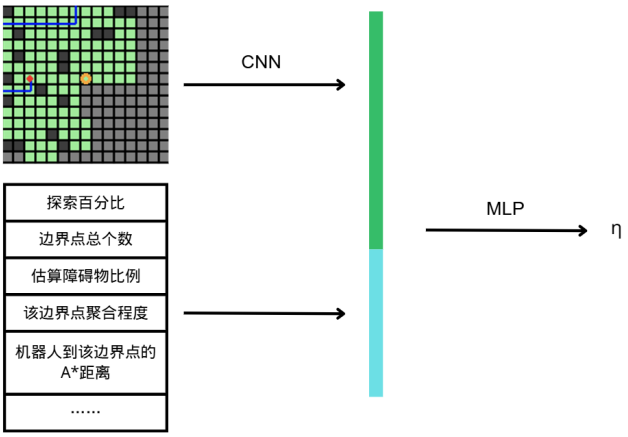

图 4: DeepACO 设想架构。局部地图作为输入 CNN，结合全局信息和局部标量信息，输出每个边界点作为下一步的启发值。

该网络可以通过强化学习进行端到端的训练，奖励由公式(9)给出（这里想要重点惩罚走重复路径，因为观察到机器人容易产生原路返回的路径），所以相当于一次决策就有一次奖励。机器人的动作由修改后的 ACO 概率公式（其中 η 由 η_θ 替代）决定，而环境给予的奖励可以是每一步新探索的栅格数量。通过 PPO 强化学习方法，网络参数 θ 被优化，使得学习到的启发式信息 η_θ 能引导 ACO 做出更有远见的决策。

$$\text{Reward} = \frac{\text{新发现的点个数} - 3 \cdot \text{走的重复路径}}{\text{导航到边界点的步数}} \quad (9)$$

A. 实验结果

1) 实验设置： **地图环境：**我们主要使用 30x30 障碍物密度 0.15 的随机地图上进行训练（我们的设想是只要在超过机器人探测范围的情况下进行训练，就能学到一定的选择原则）。

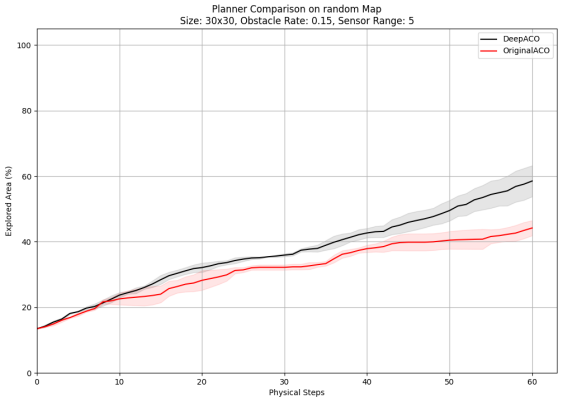

图 5: DeepACO 和传统 ACO 在 30x30 障碍物密度 0.15 随机地图上的探索效率对比（60 步以内）。

蚁群算法超参数：我们调低了原本蚁群算法中的每轮蚂蚁数和迭代数，以期能够更凸显 η_θ 发挥的作用。为了 PPO 训练的稳定性，我们放弃了平衡探索与利用的 q_0 参数，直接根据 $softmax([\tau_j(T_{\text{final}})]^\alpha[\eta_j]^\beta)$ 概率进行选择。

机器人与传感器参数：

- 传感器半径 R_{sensor}: 5 个栅格单位。
- 传感器模型：如robot.py中实现，考虑了障碍物遮挡的 Line-of-Sight。

评价指标：主要评价指标是在一定步数内（60 步）探索的区域百分比。

2) 结果与分析： 我们观察到，大约 20000 次迭代后，DeepACO 算法在 30x30 障碍物密度 0.15 的随机地图上达到了收敛。与传统的 ACO 相比，DeepACO 在探索效率上有了大约 10-15% 的提升（如图 5）。

当然由于 DeepACO 比较依赖于网络的设计与强化学习框架，可能在不同的环境下表现不一。这里我们用在 30x30 随机地图上训练得到的模型应用在 50x50 随机地图上进行测试，发现其探索效率仍比传统 ACO 大约有 10-15% 的提升（如图 6）。

我们相信只要辅以合理的网络设计，DeepACO 可以在更复杂的环境中发挥更大的优势。未来的工作可以探索如何将 DeepACO 与其他启发式方法（如 URE）结合，进一步提升探索效率。

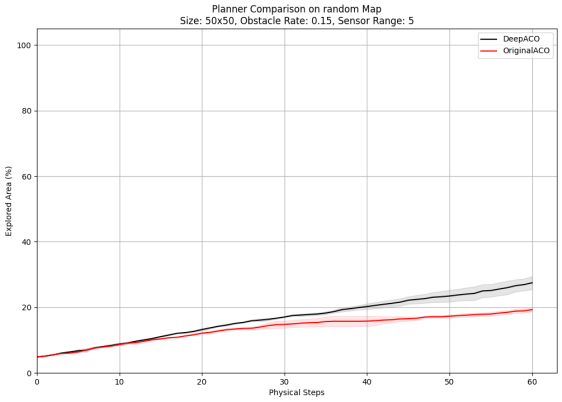

图 6: DeepACO 和传统 ACO 在 50x50 障碍物密度 0.15 随机地图上的探索效率对比（60 步以内）。

VII. 面向 SLAM 的扩展 (ACO-SLAM)

以上讨论都基于一个理想假设：机器人定位是完美的。在真实世界中，机器人需要同时进行定位和建图，即 SLAM。将我们的方法扩展到 ACO-SLAM，需要解决由定位不确定性带来的新挑战。

在 SLAM 框架下（如基于图优化的 SLAM），机器人的位姿 p_t 和地图 M 都是带有不确定性的估计值。这意味着：

1) **路径成本的不确定性：** 路径长度 L_k 不再是确定值，它依赖于未来的定位精度。
2) **信息增益的价值变化：** 除了探索未知区域，前往那些能够减少定位不确定性的区域（如含有丰富特征、能形成回环闭合的区域）也具有很高的价值。

设想实现： 我们可以将定位不确定性整合到效用函数中。修改公式(3)为：

$$\eta_j = w_1 \frac{IG(f_j)}{C(p_{\text{rob}}, f_j)} - w_2 \cdot U_{\text{loc}}(path_j) \qquad (10)$$

其中，$U_{\text{loc}}(path_j)$ 是执行路径 $path_j$ 后机器人位姿的预期协方差（不确定性），w_1, w_2 是权重系数。这个公式鼓励机器人不仅要探索新区域，还要主动进行“主动定位”，以保证地图的一致性。

A. 实验结果

1) 实验设置： **地图环境：** 我们主要使用 30x30 的随机地图上进行训练（这里的随机地图仅有 8 个障碍物，这是由于障碍物密度太高容易使得机器人无法正确规划，所以我们先在较为简单的场景下进行测试）。

机器人与传感器参数：

- 传感器半径 R_{sensor}: 5 个栅格单位。
- 传感器模型：如robot.py中实现，考虑了障碍物遮挡的 Line-of-Sight。

评价指标： 主要评价指标是观察是否能够正确估计障碍物的范围，并能够在回环中降低定位不确定性。

2) 结果与分析： 我们观察到，在 30x30 的随机地图上，ACO-SLAM 能够有效地探索未知区域，并在回环闭合时显著降低定位不确定性。图 9展示了一个基于 ACO 探索策略的简化 SLAM 过程的可视化结果，回环前后降低了地标的不确定度。

VIII. 多智能体协同构图 (Multi-ACO)

在多智能体系统 (Multi-Agent Systems, MAS) 中，主动探索的挑战变得更加复杂，核心在于如何实现有效的协作，避免重复探索和任务冲突，从而发挥群体智能的优势。我们提出的多智能体蚁群协同主动建图算法 (Multi-Agent ACO)，其核心思想是将单体 ACO 中的决策机制扩展为一个全局协作框架。

该框架的设计遵循一个关键原则：**将计算密集型的全局规划与轻量级的局部决策分离**。系统周期性地运行一个“重量级”的全局信息素更新过程，模拟大量虚拟蚂蚁探索，形成一个指导性的全局“热力图”。而每个机器人则基于这张现成的热力图进行“轻量级”的、快速的导航决策。这种分离确保了系统的高响应性和可扩展性。

1) 共享信息素地图与更新机制： 所有智能体共享并共同维护一个与环境等大的全局信息素地图 τ。该地图的更新是整个协同框架的核心，分为信息素沉积和蒸发两个过程。

1. 虚拟蚂蚁模拟与信息素沉积: 与单体 ACO 选择前沿点不同，多智能体框架中的虚拟蚂蚁直接在栅格地图上游走。系统定期（例如每隔 PHEROMONE_UPDATE_INTERVAL 步）从每个机器人 R_k 的当前位置 P_k 释放 N_{ants} 只虚拟蚂蚁。蚂蚁 m 在选择下一步时，会综合考虑信息素浓度和启发式信息。从单元格 i 移动到邻近单元格 j 的概率 p_{ij}^m 定义为：

$$p_{ij}^m = \frac{[\tau_j]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i} [\tau_l]^\alpha \cdot [\eta_{il}]^\beta} \qquad (11)$$

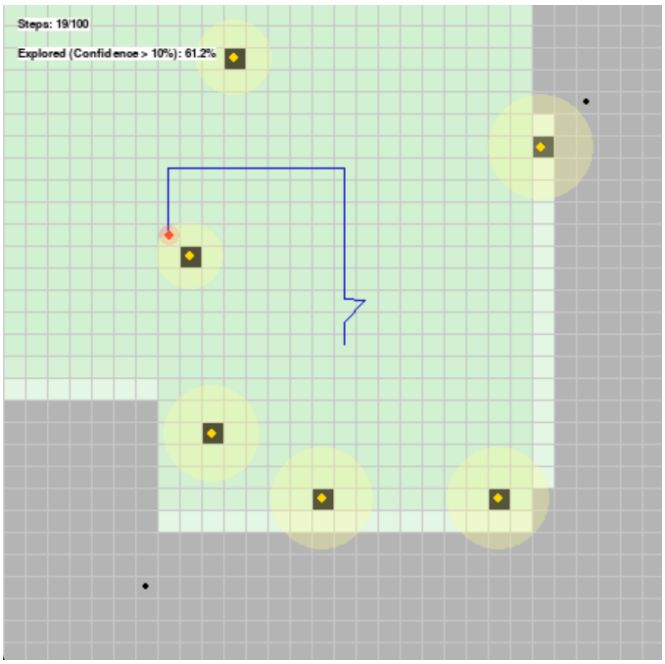

图 7: 回环前 ACO-SLAM 的可视化结果示例。

图 8: 回环后 ACO-SLAM 的可视化结果示例。

图 9: 一个基于 ACO 探索策略的简化 SLAM 过程的可视化结果。

图中各个元素的含义如下：

- **背景网格**: 代表机器人对环境的概率性占据栅格地图。颜色从亮绿色（确信为自由空间）过渡到中灰色（未知），再到深灰色（确信为障碍物）。
- **黑色小点**: 地标在环境中的真实位置（地面真值），仅用于评估和对比。
- **红色圆点**: 机器人对其自身位姿的最佳估计（均值）。
- **半透明红色圆圈**: 机器人位姿估计的协方差（不确定性）。圆圈越大，表示定位越不确定。
- **金色圆点**: 机器人对已观测到的地标位置的最佳估

其中，N_i 是单元格 i 的合法邻居集合，而启发式信息 η_{ij}（见_get_local_step_heuristic）鼓励蚂蚁走向未知区域和全局前沿点质心：

$$\eta_{ij} = w_u \cdot \mathbb{I}(S(j) = \text{UNKNOWN}) + w_c \cdot \mathbb{I}(d(j, C) < d(i, C)) + 1 \tag{12}$$

其中 $\mathbb{I}(\cdot)$ 是指示函数，C 是所有前沿点的质心，$d(a, b)$ 是曼哈顿距离，w_u 和 w_c 是对应的权重。

当蚂蚁完成路径 L^m 构建后，根据路径质量（如路径中独立单元格的数量）在其经过的每个单元格 i 上沉积信息素。

2. 信息素蒸发与整合: 整个信息素地图会定期以一个固定的速率 ρ 进行蒸发。完整的更新规则为：

$$\tau_{ij}(t+1) \leftarrow (1-\rho) \cdot \tau_{ij}(t) + \sum_{m=1}^{M} \Delta\tau_{ij}^{m}(t) \tag{13}$$

其中 M 是所有机器人释放的蚂蚁总数。此过程由update_shared_pheromones函数（见代码清单 1）集中执行，是计算开销较大的“重量级”操作。

2) 目标保留与轻量级决策: 为了避免多个机器人选择同一目标导致聚集，我们引入了目标保留机制。在一个规划周期内，当机器人 R_k 选定目标 T_k 后，会立即将其加入全局“保留列表” reserved_targets，其他机器人决策时会避开这些目标。

机器人的决策过程（见代码清单 2）是一个轻量级的、不涉及蚂蚁模拟的快速操作：

1) **信息素导航**: 从机器人当前位置 P_k 开始，贪婪地沿着信息素浓度最高的路径前进。
2) **目标选择**: 当导航路径到达一个未被保留的边界点时，该点被选为最终目标。
3) **回退策略**: 如果信息素导航失败，则启用回退策略，寻找最近的、可达且未被保留的边界点。

3) 分阶段探索与全覆盖策略: 为确保地图的完全探索，_final_fallback_plan函数（见代码清单 3）实现了一个鲁棒的分阶段策略：

1) **常规探索**: 优先寻找与大片未知区域接壤的“外部边界点”（由find_frontiers实现）。
2) **清扫阶段**: 当外部边界点消失后，系统自动寻找被已探索自由空间包围的未知“洞穴”的入口，即“内部边界点”（由find_internal_frontiers实现），确保地图内部的完整性。

此外，系统通过idle_steps_counter监测停滞状态。若所有机器人长时间空闲，会强制进行一次全局信息素更新，以期打破僵局，直至所有类型的可达边界点都被探索完毕。

A. 算法伪代码

多智能体协同探索的主循环逻辑如算法 1所示。

Algorithm 1 Multi-Agent ACO Exploration 主循环

初始化: 机器人 $R_{1..N}$, 共享信息素图 τ, 共享地图 M_{known}
for step = 1 to MAX_STEPS **do**
  if step mod UPDATE_INTERVAL == 0 **or** isStalled() **then**
    UpdateShared-Pheromones($R_{1..N}, M_{\text{known}}, \tau$)
  end if
  $ReservedTargets \leftarrow \emptyset$
  $isAnyRobotActive \leftarrow$ false
  for 机器人 R_k in $R_{1..N}$ **do**
    if R_k is idle **then**
      $T_k, P_k \leftarrow$ PlanNextAction($R_k, \ldots, ReservedTargets$)
      if T_k is not None **then**
        R_k.setPath(P_k)
        $ReservedTargets$.add(T_k, R_k.id)
        $isAnyRobotActive \leftarrow$ true
      end if
    else
      $isAnyRobotActive \leftarrow$ true
    end if
  end for
  for 机器人 R_k in $R_{1..N}$ **do**
    R_k.moveOneStep()
    R_k.senseAndUpdateMap(M_{known})
  end for
  if not $isAnyRobotActive$ **and** noReachable-FrontiersExist() **then**
    break ▷ 探索完成
  end if
end for

B. 核心代码实现

以下是算法中关键部分的 Python 代码实现，以展示其在项目中的具体应用。

```python
# planners/aco_multi_agent_planner.py
def update_shared_pheromones(self, all_robot_positions, known_map,
                             shared_pheromone_map):
    frontiers = self.find_frontiers(known_map)
    if not frontiers: return

    # 1. 全局蒸发
    shared_pheromone_map *= (1.0 - self.config['evaporation_rate_map'])

    # 2. 蚂蚁从所有机器人位置出发，进行模拟
    for robot_pos in all_robot_positions:
        for _ in range(self.config['n_ants_update']):
            # ... 蚂蚁根据公式 (5) 和 (6) 构建路径 ...
            path = self._build_ant_path(...)
            # 评估路径质量并准备沉积
            # ...
    # 3. 统一进行信息素沉积
    # ...
```

Listing 1: 信息素地图更新 (重量级操作)

```python
# planners/aco_multi_agent_planner.py
def plan_next_action(self, robot_id, robot_pos, known_map,
                     shared_pheromone_map, reserved_targets):
    # 1. 尝试根据信息素导航，避开已保留的目标
    target = self._navigate_by_pheromones(robot_pos, known_map,
                                          shared_pheromone_map,
                                          reserved_targets)
    if target:
        path = self._is_reachable_and_get_path(robot_pos, target
            , known_map)
        if path:
            reserved_targets[robot_id] = target
            return target, path

    # 2. 如果导航失败，使用回退计划
    fallback_target, fallback_path = self._final_fallback_plan(
        robot_pos,
                                       known_map,
                                       reserved_targets)
    if fallback_target:
        reserved_targets[robot_id] = fallback_target
    return fallback_target, fallback_path
```

Listing 2: 机器人规划 (轻量级操作)

```python
# planners/base_planner.py
def _final_fallback_plan(self, robot_pos, known_map,
                         reserved_targets={}):
    # 1. 优先寻找标准边界点
    frontiers = self.find_frontiers(known_map)
    available_frontiers = [fr for fr in frontiers
                           if fr not in reserved_targets.values
                               ()]
    if available_frontiers:
        # ... 寻找最近的可达标准边界点 ...
        best_target, best_path = self._find_closest_reachable
            (...)
```

```
        if best_target: return best_target, best_path

    # 2. 如果没有，则寻找内部边界点（清理未知“洞穴”）
    internal_frontiers = self.find_internal_frontiers(known_map)
    available_internal = [fr for fr in internal_frontiers
                          if fr not in reserved_targets.values()
                          ]
    if available_internal:
        # ... 寻找最近的可达内部边界点 ...
        best_target, best_path = self._find_closest_reachable
            (...)
        if best_target: return best_target, best_path

    return None, None
```

Listing 3: 强化的回退策略，确保全覆盖

C. 实验结果

1) 实验设置: 为了验证多智能体协同探索算法 (Multi-ACO) 的有效性，我们进行了一系列仿真实验。实验在一个包含三个智能体的随机地图上进行，其核心参数配置如表 I所示。

表 I: Multi-ACO 仿真实验核心参数

参数类别	参数及设定值
环境参数	
地图尺寸	50×50
障碍物比例	0.25
机器人参数	
机器人数量	3
传感器半径	7 个单元
ACO 核心参数	
信息素更新间隔	15 步
虚拟蚂蚁数量 (每次更新)	8 只/机器人
蒸发率 (ρ)	0.05
启发式权重 (α, β)	1.0, 2.5

2) 结果与分析: 图 10 展示了 Multi-ACO 算法在探索过程中的一个典型中间状态。

从图中可以清晰地观察到算法的核心机制正在有效运作：

- **信息素的全局引导作用：** 共享信息素地图（蓝色热力图）在连接上下两个区域的关键通道中形成了明显的“热点”。这表明虚拟蚂蚁的探索已成功识别出高价值的探索路径，从而引导所有机器人优先通过这些关键区域，避免在局部区域徘徊。
- **高效的隐式协同：** 三个机器人（红、绿、蓝）被有效地分配到了地图的不同探索前沿。红色机器人在下方区域探索，蓝色机器人在上方，而绿色机器人正穿越中间的已探索通道。这得益于共享信息素和目标保留机制，成功避免了任务冲突和冗余探索。
- **动态的任务分配：** 当一个机器人探索完一个区域后，该区域的信息素会因蒸发而逐渐减弱，而新的前沿点附近会因虚拟蚂蚁的持续探索而形成新的信息素高地，从而动态地将机器人引导至新的、最有潜力的探索区域。

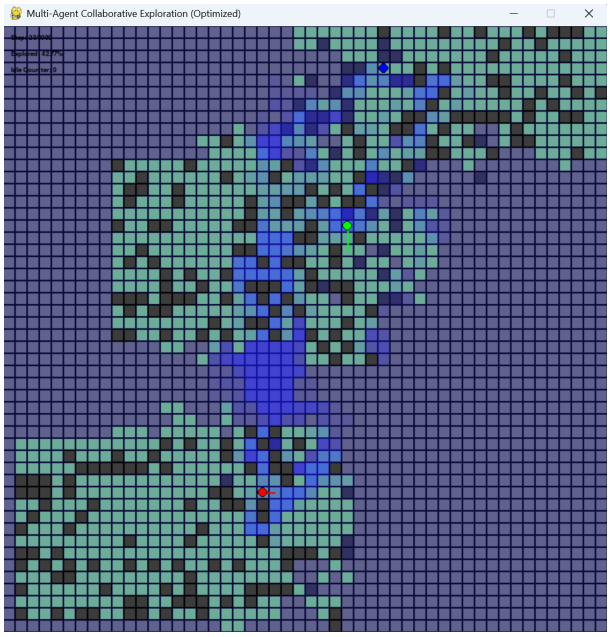

图 10: Multi-ACO 协同探索过程。图中红色、绿色和蓝色的圆点代表三个机器人。浅绿色为已探索自由空间，深灰色为障碍物，深蓝色为未知区域。背景中的蓝色“热力图”代表共享信息素地图的浓度。

综上所述，仿真结果直观地验证了我们提出的 Multi-ACO 框架能够实现高效、鲁棒的多智能体协同探索。

IX. 结论

本次大作业，我们系统地研究了如何将蚁群优化算法应用于移动机器人的主动探索与建图任务。对课程的学习有了更为深刻的认识。

对于单机器人场景，我们提出的 ACO-Mapping 算法将信息增益作为核心奖励机制，实验证明其性能优于传统方法。对于多机器人场景，我们设计的 Multi-ACO 框架通过共享全局信息素地图、目标保留和分阶段探索策略，实现了高效的隐式协同。将计算密集的全局规划与轻量级的个体决策分离的设计，确保了系统的可扩展性和高效率。

除此以外，我们结合深度学习，对 DeepACO 模型进行完整实现与评估；考虑真实世界噪音，实现了 ACO-SLAM，以处理更复杂的现实因素。

我们相信，基于 ACO 的探索方法为解决复杂环境下的单体及多机器人自主导航问题提供了一条富有前景的技术路径。

X. Demo

为了更直观地展示本文提出的蚁群优化算法，我们开发了一个基于 Web 的全栈交互式可视化工具。该项目旨在以交互的方式演示两种核心的蚁群优化（Ant Colony Optimization, ACO）应用：**路径规划**和**自主探索构图**。用户可以通过一个直观的 Web 界面调整参数、生成或手绘地图，并实时观察算法的运行过程和结果。

A. 功能特性

该可视化工具具备以下主要功能：

- **双模式演示**:
  - **路径规划 (Path Planning)**: 在给定的、完全已知的地图中，利用蚁群算法寻找从起点到终点的最优路径。
  - **自主构图 (Exploration)**: 模拟一个机器人在未知环境中，仅依靠有限范围的传感器，逐步探索并构建出整个环境的地图。
- **高度交互性**:
  - **实时参数调整**: 用户可以在网页侧边栏实时调整 ACO 的核心参数，如 Alpha（信息素影响）、Beta（启发式影响）、信息素蒸发率等，并立即在新的模拟中看到效果。
  - **多种地图生成方式**: 支持程序化生成（随机地图、欺骗性走廊）和用户手绘地图。
- **实时可视化**: 实时展示蚂蚁路径、信息素浓度分布、机器人轨迹、已知地图构建过程和边界点等关键信息。
- **双栏视图**: 左侧为参数控制面板，右侧为可视化画布和算法解释，方便用户对照学习。

B. 技术栈

项目的技术选型旨在实现轻量级、高响应和易于部署：

- **后端**:
  - **Python 3**: 主要编程语言。
  - **FastAPI**: 用于构建高性能的 API 和 WebSocket 服务。
  - **Uvicorn**: ASGI 服务器。
  - **NumPy**: 用于高效的地图（网格）数据处理。
- **前端**:
  - **原生 HTML, CSS, JavaScript (ES6)**: 无任何前端框架，确保轻量化。
  - **Canvas API**: 用于绘制所有可视化元素。
  - **WebSocket API**: 用于与后端进行实时、双向的数据通信。

C. 运行指南

1) 安装依赖: 项目所需的 Python 库已在requirements.txt文件中列出。在项目根目录下打开命令行或终端，运行以下命令：

```
pip install -r requirements.txt
```

2) 启动程序:

1) **对于 Windows 用户**: 直接双击运行项目根目录下的run_windows.bat文件。
2) **对于 macOS/Linux 用户 (或手动启动)**:
   a) 打开终端，导航到项目的backend目录。
   b) 运行以下命令启动服务器：

```
uvicorn main:app --host 127.0.0.1 --port 8000
```

服务器启动后，在浏览器中访问 **http://127.0.0.1:8000** 即可与程序进行交互。

D. 核心算法简述

1) 路径规划中的 ACO:

1) **初始化**: 在地图上均匀分布少量信息素。
2) **蚂蚁出发**: 在每次迭代中，从起点释放一群蚂蚁。
3) **路径选择**: 每个蚂蚁在每个决策点，根据脚下路径的**信息素浓度**（历史经验）和**启发式信息**（距离终点的远近）的加权组合，概率性地选择下一步。
4) **信息素更新**: 完成一次迭代后，成功到达终点的蚂蚁根据其路径优劣（通常是长度的倒数）在走过的路径上留下信息素。同时，地图上所有信息素都会有一定比例的**蒸发**。
5) **正反馈**: 较短的路径会获得更多信息素积累，吸引更多蚂蚁，最终收敛到最优或次优路径。

2) 自主构图中的探索策略:

1) **感知 (Sense)**: 机器人使用模拟传感器感知周围环境，更新其内部的“已知地图”，视线会被障碍物遮挡。
2) **识别边界 (Find Frontiers)**: 在已知地图中寻找所有“已知区域”与“未知区域”的交界处，即“边界点”。
3) **决策 (Choose Target)**: 使用一种探索策略评估前往各个边界点的“价值”，选择一个作为下一个目标。
4) **移动 (Move)**: 使用 A* 算法在**当前已知地图**的基础上规划并执行前往目标的路径。
5) **循环**: 重复以上步骤，直至探索完成。

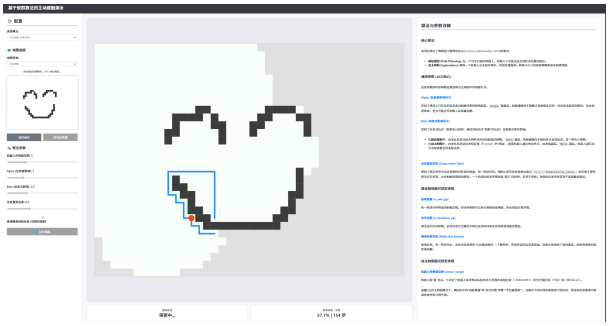

图 11: Demo

参考文献

[1] R. Murphy, S. Tadokoro, D. Nardi, E. Sahin, and T. L. Choy, “Search and rescue robotics,” in Springer Handbook of Robotics, B. Siciliano and O. Khatib, Eds. Springer, 2008, pp. 1151–1173.

[2] B. Yamauchi, “A frontier-based approach for autonomous exploration,” in Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, 1997, pp. 146-151.

[3] C. Stachniss, G. Grisetti, and W. Burgard, “Information gain-based exploration using Rao-Blackwellized particle filters,” in Robotics: Science and Systems, 2005.

[4] M. Dorigo, V. Maniezzo, and A. Colorni, “Ant system: optimization by a colony of cooperating agents,” IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 26, no. 1, pp. 29-41, 1996.

[5] F. Amigoni and V. Caglioti, “A utility-based approach for robotic exploration of unknown environments,” in Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 5496-5502.

[6] C. Campos, R. Elvira, J. J. Gómez-Rodríguez, J. M. M. Montiel, and J. D. Tardós, “ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM,” IEEE Transactions on Robotics, vol. 37, no. 6, pp. 1874-1890, 2021.