基于 OPEN-RMF 的多机器人协同规划在药房自动化中的应用研究

严以鑫

合肥工业大学宣城校区，安徽宣城　242000

【摘　要】随着药房自动化需求的日益增长，多机器人协同技术成为提升效率与精度的关键。本文基于 OPEN-RMF（Open Robotics Middleware Framework）框架，研究了多机器人协同规划在药房自动化中的实际应用。通过构建虚拟药房环境并结合 ROS2 与 Gazebo 仿真平台，实现了多机器人任务分配、路径规划及药品配送功能。实验结果表明，该方法能够显著提高药品分发的效率，减少人工干预，同时适应动态环境变化。研究为药房自动化的智能化发展提供了创新思路与技术参考，具有重要的实践价值。

【关键词】多机器人协同；路径规划；路径规划；ROS2

一、引言

（一）研究背景与意义

1. 药房自动化的发展趋势

随着中国医疗卫生事业的蓬勃发展，公众对医疗服务品质和效率的要求日益提升。药房，作为医疗服务体系中不可或缺的一环，其运营效能与服务水平直接关系到患者的就医满意度。但当前，药房运营中普遍存在的取药等待时间过长、服务效率不高等问题，已成为制约医疗服务质量提升的瓶颈，不仅增加了患者的等待成本，更影响了医疗机构的整体运营效果。当前，为解决这一瓶颈问题，同时紧跟医疗科技创新的步伐，本文立项研发了“药房智能化 ROS 机器人群控制系统”，该系统以机器人自动取药技术为核心，旨在通过智能化手段，实现药品的迅速、精准配送，进而显著提升药房服务效率，降低人工成本，并大幅增强医疗安全性能。

2. 多机器人协同技术的应用潜力

工业生产等领域的多机器人协同技术和融合以及智能化应用才刚刚起步，因为智能工厂目前仍然存在许多不可控变量，多机器人的切入往往还需要在解决复杂环境中对工程应用的不确定性进行评估，但在未来随着 AI 的加入，更智能的分配调度系统下，多机器人在工业上的应用将逐渐增多。

集群机器人具备 5 个典型特点，即资源分布式、信息分布式、时间分布式、功能分布式、空间分布式，这些特点使其展现出更多发展潜力。

多机器人相对于单机器人而言，优势也相当明显，多机器人能通过资源的互补对单个机器人的能力进行提升，将其有限扩大到多个任务，分布到不同的机器人当中。同时多机器人也可以增强机器人的灵活性，特别是在资源的分配调度优化方面，能起到更加广泛的作用。未来成熟的智能工厂，需要多机器人以适应复杂的人工智能调度。

（二）国内外研究现状

1. 国内研究现状

中国药房智能化发展可以追溯到20世纪80年代，当时，中国药房还处于传统的人工服务模式，药品配送、库存管理以及购买记录等都依赖人工操作，然而随着信息技术的快速发展，中国在90年代初期开始引入了计算机管理系统。21世纪10年代，中国药房才开始积极探索智能化发展道路，借助物联网、大数据和人工智能等先进技术实现智能配送、智能终端等项目。当前，国内学者对于药房智能化的发展有了多层次、多角度、深领域的研究。

进入21世纪后，各学者积极探讨各型各类的智能药房的发展现状和使用效果。林艳等（2020）筛选196篇文献，对关于自动发药系统的使用效果进行评价，该系统明显缩短患者取药时间，优化门诊工作流程，效率提高30.0%～50.0%，相对传统发药模式有较为明显的优势，但机器故障，运行稳定性也还需持续改进；陈井泉等（2022）基于其所在医院智慧门诊药房的使用，发现智能化药房提高了调配药发药的准确率和效率，误发率＜0.001%，减少了药师的重复工作，智能软件辅助核对使药品核对准确率达到了100%；周良等（2020）探索智能化门诊药房药品有效期闭环管理的模式，建立了智能化药品调配系统、物流和信息流系统、温湿度监控系统，并制定了近效期药品管理、内部质控和绩效考核制度，优化了药品有效期管理工作流程，改善了药房工作环境，提高了药学服务质量和药房管理水平，为患者提供更加安全有效的药学服务；朱磊等（2018）分析实践综合性医院门诊自动化的应用和效果，认为而药品的合理、分类和规律摆放使得药房的环境得到改善，保证药品存放的整洁性、密闭性，进而从一定程度上减少污染。对医院中药饮片及中药材进行智能化管理，能够明显提升其保管质量，保证药品使用效果。万盟等（2021）观察新型冠状病毒肺炎防控常态化后，智能化设备在门诊药房的应用效果，智能化设备的合理使用可提高门诊药房人员工作效率，减少患者等候时间，增加发热门诊药品储存量，降低交叉感染机会，药房的药品管理更加精准。

2. 国外研究现状

全球COVID-19大流行加速了向远程护理的过渡，42%的医疗保健领导者打算维持这一趋势。此外，以AlphaFold和RoseTTAFold等突破为代表的人工智能的重大进步有望通过潜在地增加发现新药物的机会来彻底改变药物发现。他们还提出在医疗保健领域实施人工智能和机器人技术的伦理考虑和挑战，解决这些问题至关重要，以确保这些技术在医疗环境中负责任且公平地部署。在药房系统中实施人工智能可以通过多种方式帮助世界变得更美好。首先，它可以通过为患者提供个性化药物管理和24小时支持来改善医疗保健服务的可及性。这可以帮助患者更有效地管理他们的药物，并减少频繁拜访医疗保健提供者的需要，这在医疗保健服务有限或无法获得的地区尤其有益。其次，人工智能可以通过最大限度地减少用药错误和防止不良药物相互作用来降低医疗成本，从而减少住院次数并降低医疗费用。最后，人工智能可以减少医疗保健提供者的工作量，使他们能够专注于更复杂的任务并提高他们可以提供的护理质量。总体而言，在药房系统中实施人工智能有可能改善医疗保健结果、降低医疗保健成本并增加获得医疗保健服务的机会，使世界变得更美好。

3. OPEN-RMF及相关技术在机器人领域的应用

虽然OPEN-RMF最初为医疗场景（如新加坡樟宜综合医院）开发，但其应用已扩展至多个领域。

在丹麦召开的ROSCON2024中，发布会上具体描述了RMF技术是如何从仿真到实验室到生产应用

的。由此可以了解到RMF的具体部署过程，同时OPEN-RMF已成功应用于实际场景，例如新加坡的医院，物流系统、物流与仓储、矿山显著提升了机器人舰队的运营效率。OPEN-RMF 及其相关技术通过提供多机器人协同、基础设施集成和仿真验证的解决方案，在机器人领域展现了广泛的应用前景。它不仅提高了系统效率和灵活性，还为跨行业的自动化发展奠定了坚实基础。

未来，随着机器人技术在智能城市、工业 4.0 等领域的深入渗透，OPEN-RMF 有望通过引入更先进的算法（如强化学习）和硬件集成（如真实机器人平台的部署），进一步推动多机器人系统的智能化和规模化应用。

（三）实验目标以及创新点

1. 目的

结合 ROS 技术中的 RMF 框架，建立一个服务器，完成医生只需发送请求，系统自动处理，机器人群自动取药，实现一键取药的功能，有效提高医院取药的效率，减轻药剂科护士的工作量，解决以前的药房，取药时间长，人员在高峰期时紧缺的问题。

2. 创新点

（1）动态调整与实时调度能力：本系统具备出色的灵活性和可扩展性，能够根据医院的规模和实际需求，动态调整并实时调度相应数量的机器人及服务资源，以满足不断变化的服务需求。

（2）低成本与易部署性：相较于其他系统，本项目的引进成本更为经济。它无需对病房环境进行大规模改造，只需进行简单的路径规划后即可迅速投入使用，大大降低了部署难度和成本。

（3）群控制技术与智能调度：本系统采用先进的群控制技术，能够同时管理并协调多个机器人的运作。通过智能调度算法，实现药品的高效配送，显著提升药房的服务效率和配送准确性。

（4）优化人力资源分配：通过自动化和智能化的药品配送，本系统有效减轻了医护人员的工作负担，使得医院能够将宝贵的人力资源重新分配到更需要的医疗设备和仪器上，从而提升医院的整体服务水平和竞争力。

二、实验设计

（一）OPEN-RMF 框架概述

1. OPEN-RMF 的核心功能与架构

OpenRMF（Robotics Middleware Framework）是一个很有前途的中间件解决方案，旨在解决这些问题挑战。它简化了异构机器人模型的集成，提高了运营效率在预定义的环境中。鉴于涉及多机器人的应用越来越复杂和规模系统、OpenRMF 的实施和评估对于推动该领域发展和确保各种行业多机器人车队无缝运行。它通过资源分配和通过 RMF Core 防止共享资源冲突来为系统添加智能。

OpenRMF 可以帮助解决一些挑战，例如协调问题。系统中添加的任何任务都会通过竞标过程分配给一个舰队，每个舰队管理器会根据距离、电池电量以及可能影响任务的其他属性出价。它还能协助路径规划，但 RMF 的路径规划版本局限于用户在创建地图布局时预定义的一组有限路线，更复杂的实现需要用户自行完成。通过使用 RMF 附带的交通编辑器（Traffic Editor），系统可以自动解决碰撞或资源访问等问题，无需用户干预。

多机器人系统的开发和维护难度较大，因为需要处理不同类型的机器人，这些机器人可能来自不同制造商，拥有不同的通信协议和规格。使用 RMF 作为中间件，通过促进各种机器人模型的集成，可以开发复杂的解决方案。每个机器人需要开发一个舰队适配器（Fleet Adapter）以与 RMF 协同工作，一旦适配器完成，该机器人即可在任何 RMF 项目中使用。

所有这些特性使 RMF 成为开发简单和复杂多机器人系统的强大工具，但它受限于特定的布局。这意味着 RMF 无法在随机或动态的场所使用，它适用于建筑物或预定义区域。

2. 与 ROS2 的集成优势

OPEN-RMF 建立在 ROS2 之上，利用了后者的高性能数据交换和实时通信能力。其开源性质允许开发者根据具体需求定制功能，例如通过添加 GPS 支持扩展至室外应用，或集成机器学习算法优化任务调度。

与 ROS2 集成意味着 OPEN-RMF 可以直接使用 ROS2 社区提供的丰富软件包，例如导航（Navigation2）、运动规划（MoveIt）和仿真（Gazebo）。这减少了重复开发的工作量，开发者可以专注于特定应用逻辑而非底层实现。

ROS2 提供了强大的开发工具（如 colcon 构建系统）和广泛的社区支持。OPEN-RMF 的开发者可以利用这些工具快速构建、测试和部署系统，同时从 ROS2 的活跃社区中获取技术支持和资源，加速开发进程。

（二）药房自动化场景分析

1. 药房工作流程与需求

根据实验目的，需要发布指令然后，系统接受，然后分派机器人去完成取药的任务。流程图如下图 1 所示。

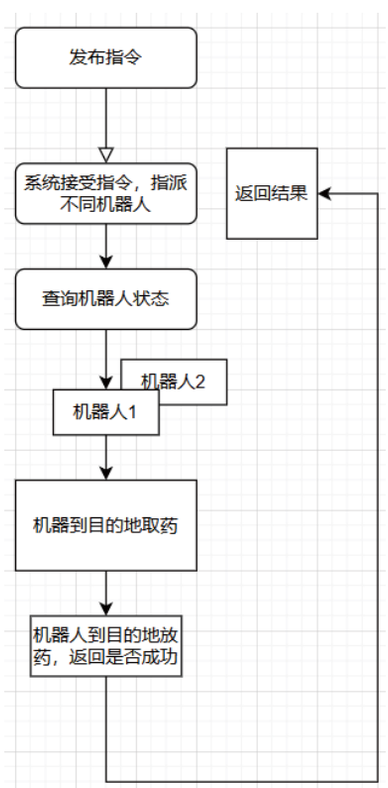

图 1 实验的设计逻辑

2. 多机器人任务类型（药品分拣、配送等）

将药房工作流程与需求进行分析，便知道机器人所需要的任务类型：药品分拣和配送。

（三）系统架构设计

1. 多机器人协同模块

多机器人协同模块包括控制 Robot 的 Controller Server；管理相同类型的 Robot 的 Controller Server 的 Fleet Manger；管理不同 Fleet Manger 的 Fleet Adapter 的 Fleet Adapter。其中，要在 Fleet Adapter 中完成与 OPEN-RMF 对接的 API（ClientCommand.py）。它们之间控制关系如图 2 所示。

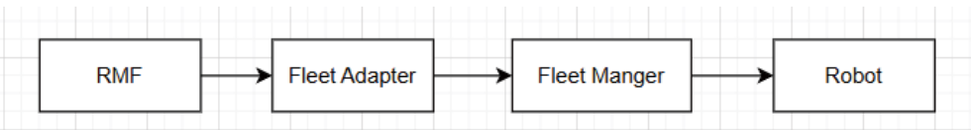

图 2　RMF 中的层级递进的控制关系

2. 任务分配与路径规划算法

由于使用 RMF 框架，RMF 简化了跨多队列系统的任务分配和管理。当用户提交新的任务请求时，RMF 会智能地将其分配给队列中最能执行该任务的机器人。RMF 支持三种类型的开箱即用任务请求，其中 Delivery 很符合需求，它适用于能够在设施内的各个位置之间配送物品的机器人。在此基础上添加一个到达取货点进行取货的动作，并给其接口命名为：GetMed 的一个 action（ROS2 进行消息转递的类型）。分配任务的流程图或者思路如图 3 所示。

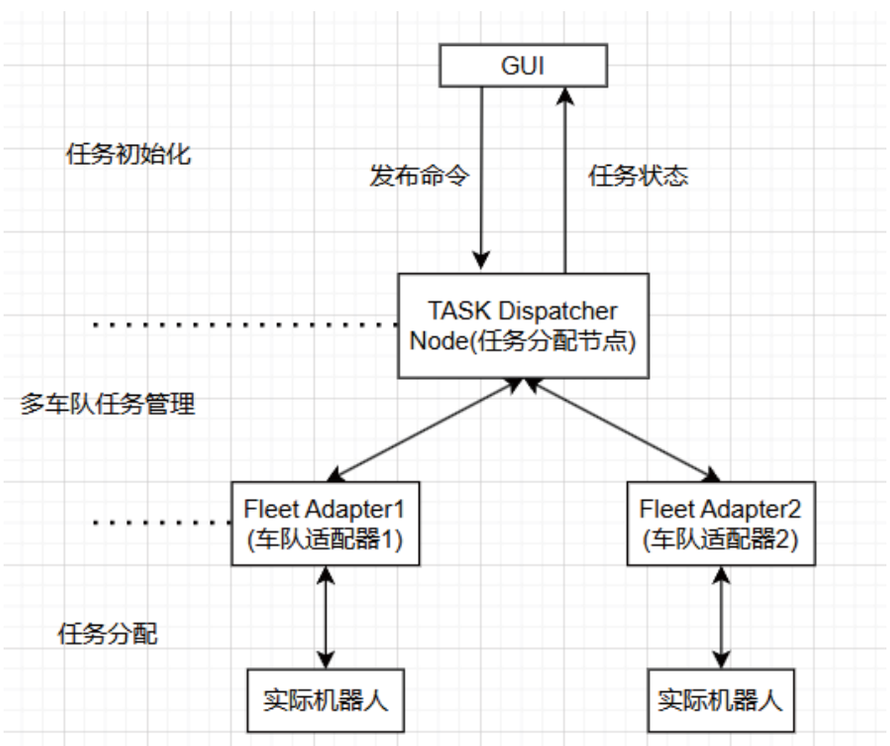

图 3　分配任务的流程图

在 RMF 中，路径规划实际上是由于 rmf_traffic_for_ros2 和 rmf_traffic 来完成的，在此基础上，仅仅只需要实现与其对接的接口，如：

（1）navigate：向机器人 API 发送导航命令。它接收来自 RMF 的目的地坐标、所需的地图名称和可选的速度限制。

（2）start_activity：向机器人发送命令以开始执行任务。此方法对于由 execute_action（）触发的自定义可执行作非常有用。

（3）stop：命令机器人停止移动。

3. 仿真环境搭建

首先，要使用一个名为 traffic_editor 的工具，用于从地图自动生成模拟世界，它可以规划机器人的路径，和关键资源。traffic-editor 的演示如图 4 所示。

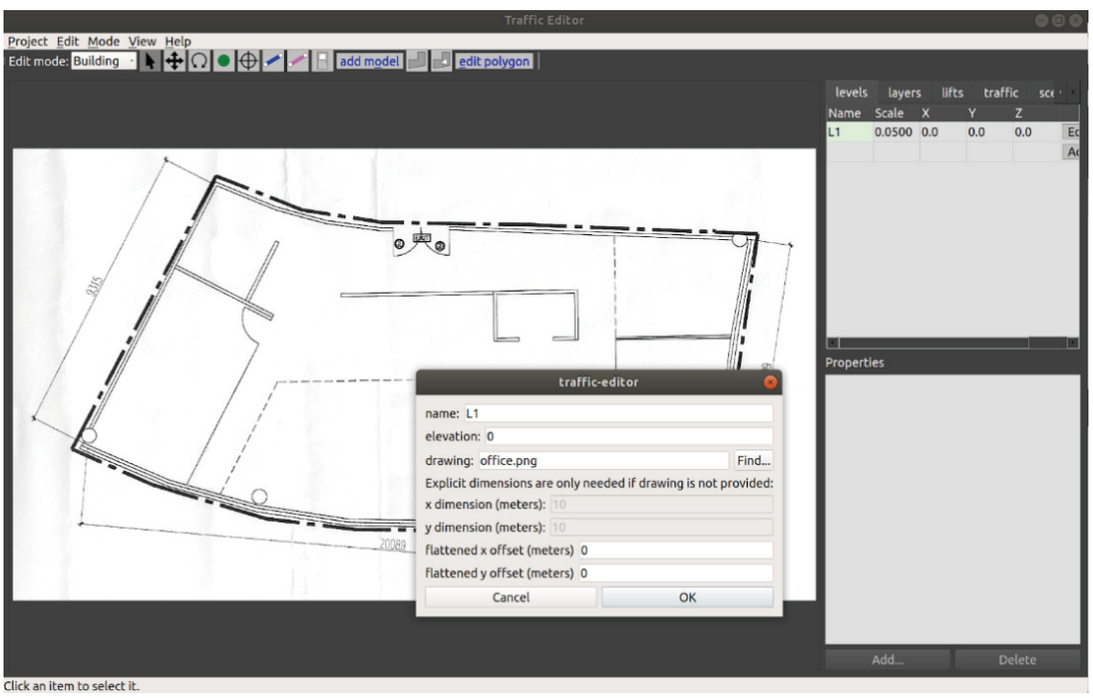

图 4 traffic-editor 的演示

手动创建，来完成地图搭建，只需手动的标记好关键资源和路径，便可以让机器人群在 RMF 框架下完成统一调度。接下来，在终端中输入以下指令来生成仿真环境，如图 5 所示。

```
1.  ##############################################################################
2.  # Generate Gz world and download Models
3.  ##############################################################################
4.
5.  ros2 run rmf_building_map_tools building_map_model_downloader ${map_path}
6.
7.  ros2 run rmf_building_map_tools building_map_generator gazebo ${map_path} ${output_world_path} ${output_model_dir}
8.
9.  ##############################################################################
10.  # generate the navmesh and required files for crowd simulation for gz
11.  ##############################################################################
12.
13.  ros2 run rmf_building_map_tools building_crowdsim ${map_path} ${crowd_sim_config_resource} ${output_world_path}
14.      DEPENDS ${output_world_path}
15.
16.
17.  ##############################################################################
18.  # Generate the nav graphs
19.  ##############################################################################
20.
21.  ros2 run rmf_building_map_tools building_map_generator nav ${map_path} ${output_nav_graphs_dir}
22.     DEPENDS ${map_path}
```

图 5 生成仿真环境指令

三、实验实现

（一）实验环境搭建

1. ROS2 与 Gazebo 的安装与配置

（1）安装 ROS2。

①确保 Ubuntu Universe 存储库已启用，如图 6 所示。

```
sudo apt install software-properties-common
sudo add-apt-repository universe
```

图 6　apt 指令添加存储库

②使用 apt 添加 ROS2 GPG 密钥，如图 7 所示。

```
sudo apt update && sudo apt install curl -y
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg
```

图 7　apt 指令添加密钥

③将存储库添加到您的源列表中，如图 8 所示。

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg]\
 http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME) main" \
 | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/nulla
```

图 8　添加存储库到 /etc/apt/sources.list.d/* 中

④在设置存储库后更新 apt 存储库缓存，如图 9 所示。

```
sudo apt update
```

图 9　更新 apt 库，确认添加是否成功

⑤更新系统，如图 10 所示。

```
sudo apt upgrade
```

图 10　更新当前软件到最新的代码示意图

⑥完全安装（含有完整的 GUI 工具，教程等），如图 11 所示。

```
sudo apt install ros-jazzy-desktop
```

图 11　安装 ros-jazzy 完全版的代码示意图

⑦然后启动环境，如图 12 所示。至此，ROS2 就完全安装完了。

```
# Replace ".bash" with your shell if you're not using bash
# Possible values are: setup.bash, setup.sh, setup.zsh
source /opt/ros/jazzy/setup.bash
```

图 12　添加启动脚本到 ~/.bashrc 以便开机自动启动的代码示意图

（2）安装 OPEN_RMF。

①安装 Open-RMF 包的所有非 ROS 依赖项，如图 13 所示。

```
sudo apt update && sudo apt install ros-dev-tools -y
```

图 13　安装 ros 工具包的代码示意图

② rosdep 帮助跨各种发行版安装 ROS 包的依赖项，并将与 ros-dev-tools 一起安装。安装完成后对其进行更新，如图 14 所示。

```
sudo rosdep init # run if first time using rosdep.
rosdep update
```

图 14　用初始化 rosdep 并且更新 rosdep 的代码示意图

③如果之前没有更新，还要更新 colcon mixin，如图 15 所示。

```
colcon mixin add default https://raw.githubusercontent.com/colcon/colcon-mixin-repository/master/index.yaml
colcon mixin update default
```

图 15 更新并且下载 mixin 的代码示意图

④进行二进制安装（直接下载包）。至此，完成了 OPEN_RMF 的安装，如图 16 所示。

```
sudo apt update && sudo apt install ros-jazzy-rmf-dev
```

图 16 用 apt 指令去下载 openrmf 的包

（3）安装 gazebo。

安装 gazebo 十分简单，只需要输入命令：sudo apt install gz-harmonic，即可完成安装。

2. 虚拟药房场景建模

（1）打开 traffic-editor。如图 17 所示。

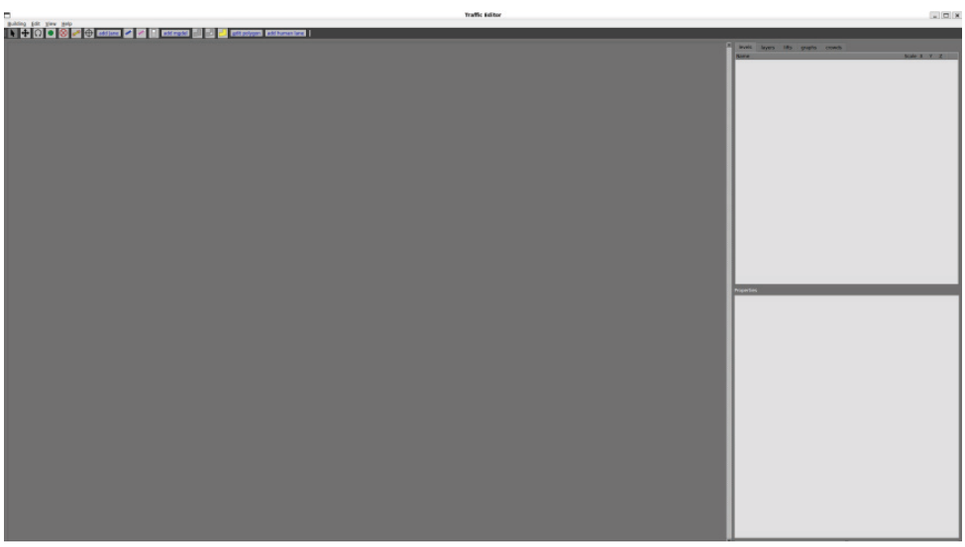

图 17 打开 traffic-editor 后，traffic-editor 的展示

（2）对地图进行设计，得到如图 18 的结果。

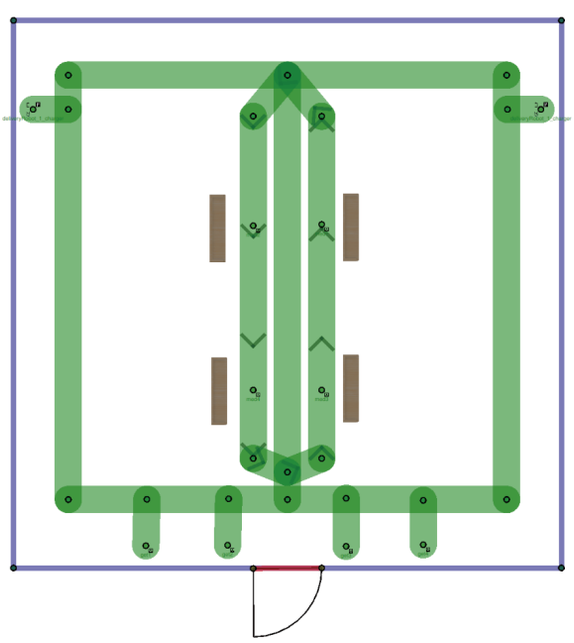

图 18 traffic editor 中药房的设计图

（3）对其进行编译，得到仿真地图和 RVIZ。如图 19、20 所示。

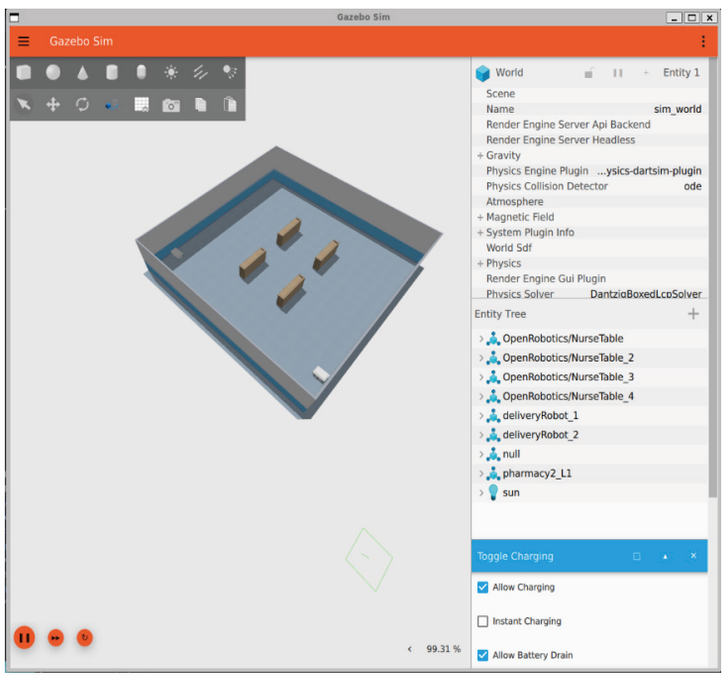

图 19　gazebo 搭建完成地图后的演示

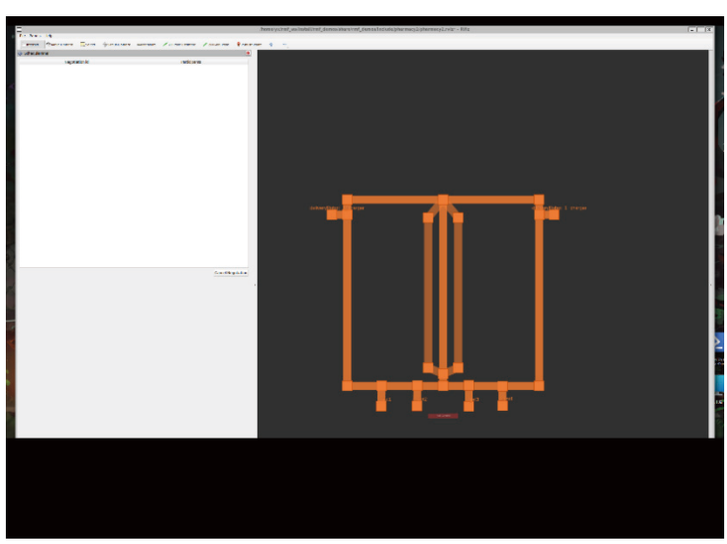

图 20　搭建完成之后 nav 生成的 RVIZ 的演示

（二）接口实现与机器人仿真配置的编写

1. 接口实现

设计要求作为接口实现的基础，需明确其具体内容。由于当前未提供详细的设计要求规范，假定该接口涉及机器学习模型的集成，用于支持用户通过前端界面与后端模型进行交互，典型场景包括数据输入、模型预测和结果展示。根据通用接口设计原则，需满足以下核心需求：

(1) 功能性：接口需支持用户输入数据，调用相应的 Fleet Manager 进行处理并返回结果。

(2) 用户体验：接口应具备直观性，确保用户能够轻松理解输入格式和输出结果。

(3) 可扩展性：接口设计需考虑未来新增模型或功能的兼容性。

(4) 性能：接口需保证低延迟响应，尤其在实时场景下。

(5) 鲁棒性：接口需包含错误处理机制，确保在异常输入或错误处理时提供清晰的反馈。

根据设计要求，按照 REST API 的技术约束去完成任务，接口的代码如图 21 所示。

```
def check_connection(self):
''''' Return True if connection to the robot API server is successful '''
        # ------------------------ #
        # IMPLEMENT YOUR CODE HERE #
        # ------------------------ #
        return True

    def navigate(
        self,
        robot_name: str,
        pose,
        map_name: str,
        speed_limit=0.0
    ):
        ''''' Request the robot to navigate to pose:[x,y,theta] where x, y and
            and theta are in the robot's coordinate convention. This function
            should return True if the robot has accepted the request,
            else False '''
        # ------------------------ #
        # IMPLEMENT YOUR CODE HERE #
        # ------------------------ #
        return False

    def start_activity(
        self,
        robot_name: str,
        activity: str,
        label: str
    ):
        ''''' Request the robot to begin a process. This is specific to the robot
        and the use case. For example, load/unload a cart for Deliverybot
        or begin cleaning a zone for a cleaning robot.
        Return True if process has started/is queued successfully, else
        return False '''
        # ------------------------ #
        # IMPLEMENT YOUR CODE HERE #
        # ------------------------ #
        return False

    def stop(self, robot_name: str):
        ''''' Command the robot to stop.
            Return True if robot has successfully stopped. Else False. '''
        # ------------------------ #
        # IMPLEMENT YOUR CODE HERE #
        # ------------------------ #
        return False

    def position(self, robot_name: str):
        ''''' Return [x, y, theta] expressed in the robot's coordinate frame or
        None if any errors are encountered '''
        # ------------------------ #
        # IMPLEMENT YOUR CODE HERE #
        # ------------------------ #
        return None

    def battery_soc(self, robot_name: str):
        ''''' Return the state of charge of the robot as a value between 0.0
        and 1.0. Else return None if any errors are encountered. '''
        # ------------------------ #
        # IMPLEMENT YOUR CODE HERE #
        # ------------------------ #
        return None

    def map(self, robot_name: str):
        ''''' Return the name of the map that the robot is currently on or
        None if any errors are encountered. '''
        # ------------------------ #
        # IMPLEMENT YOUR CODE HERE #
        # ------------------------ #
        return None

    def is_command_completed(self):
        ''''' Return True if the robot has completed its last command, else
        return False. '''
        # ------------------------ #
        # IMPLEMENT YOUR CODE HERE #
        # ------------------------ #
    return False
```

图 21 接口代码展示

图 21 所示是要填写的 API，其中的具体实现逻辑部分，可以采用 Fleet Manager 去完成，如图 22 所示。

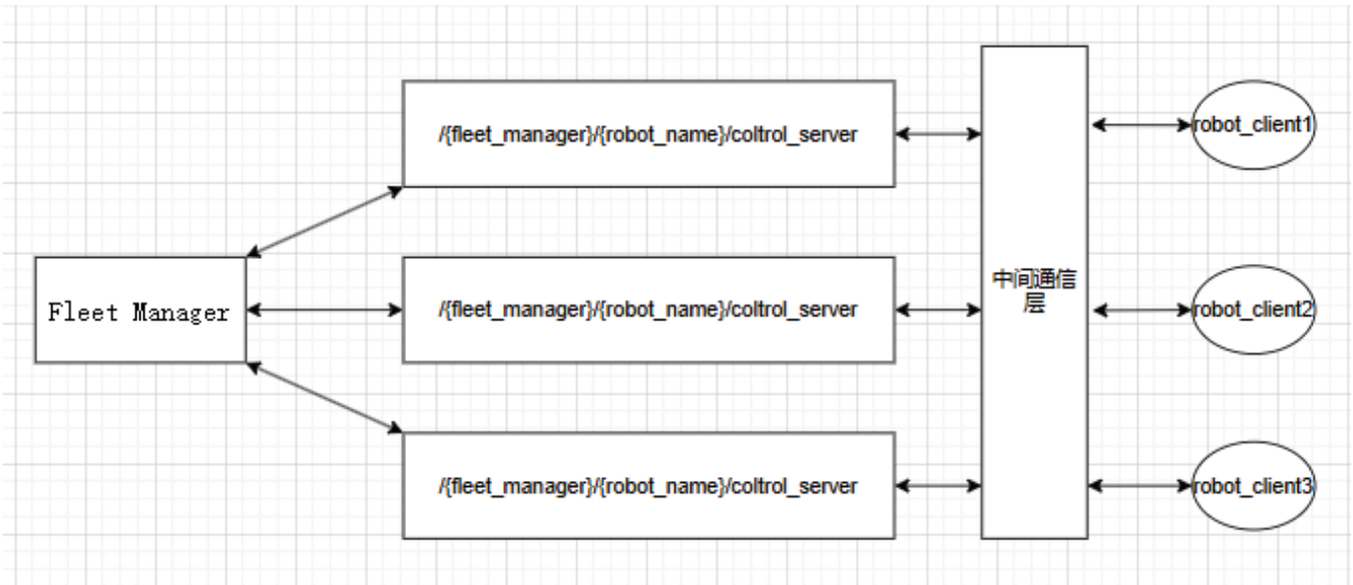

图 22 Fleet Manager 的设计思路

完成以上的 Fleet Manager 的设计后，便可以设计 coltol_server。根据对机器人的设计要求，机器人除了完成与 open-rmf 对接的接口外，还需完成运输、取药这两个动作的任务，加上健壮性和便于代码维护的需要，本文使用了 REST API 来完成任务的设计，设计思路如下图 23 所示。

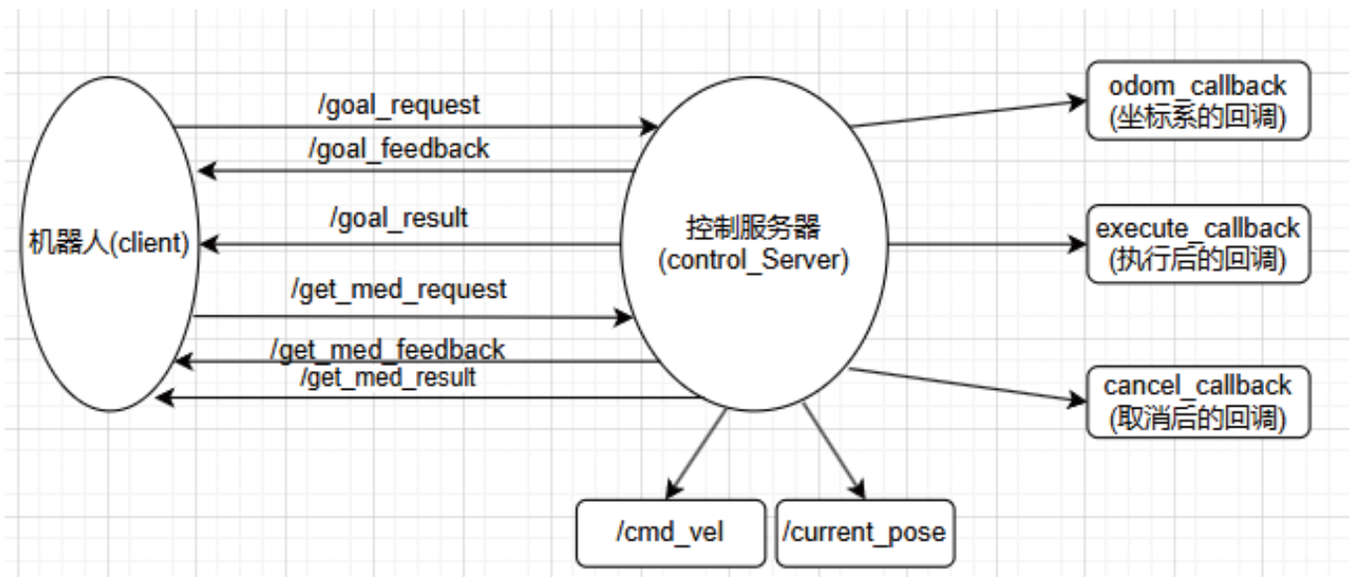

图 23 control_server 的设计思路

2. 机器人配置编写

在完成机器人接口的编写后，需要对机器人进行配置，具体分配逻辑如图 24 所示。

这里主要介绍对机器人 fleet_adapter.xml 的配置，下面是配置参数的说明。

(1)rmf_fleet：重要的 fleet 参数，包括车辆特征、任务能力和用于连接到 fleet manager 的用户信息。下面是 rmf_fleet 的参数：

① limits：线性和角度加速度以及速度的最大值。

② profile：此车队中车辆的足迹半径和个人附近区域。

③ reversible：在 robot 中启用 / 禁用反向遍历的标志。

④ battery_system：有关电池电压、容量和充电电流的信息。

⑤ recharge_threshold：设置最小电量值，低于该值时，机器人必须返回其充电器。

⑥ recharge_soc：机器人应充电至的总电池容量的分数。

⑦ task_capabilities：机器人可以在 loop、delivery 和 cleaning 之间执行的任务。

⑧ account_for_battery_drain：RMF 在调度任务之前是否应该考虑机器人的电池消耗。

⑨ action[可选]：队列的自定义可执行作列表。在此，需要选择 delivery。

⑩ finishing_request：机器人完成任务后应该做什么，可以设置为停车、充电或不充电。

⑪ responsive_wait[可选]：True # 默认情况下，整个队列的响应式等待是否应该打开 / 关闭？如果未指定，则为 False。

⑫ robots：有关队列中每个机器人的信息。本节中的每个项目都对应于队列中单个机器人的配置，包括 robotname1（机器人的名称）、charger（机器人的充电点名称）、responsive_wait（此特定机器人是否应打开 / 关闭其响应等待。覆盖队组范围的设置）。可以相应地添加更多机器人。

⑬ robot_state_update_frequency：机器人应多久更新一次队列。

(2)fleet_manager：可以配置为适合您选择 API。这些参数将用于设置与您的 Fleet Manager/ 机器人的连接。

(3)reference_coordinates[可选]：如果车队机器人不在与 RMF 相同的坐标系中运行，您可以提供两组（x，y）坐标，分别对应于每个系统中的相同位置。这有助于估计从一个帧到另一个帧的坐标转换。建议至少 4 个匹配的航点。

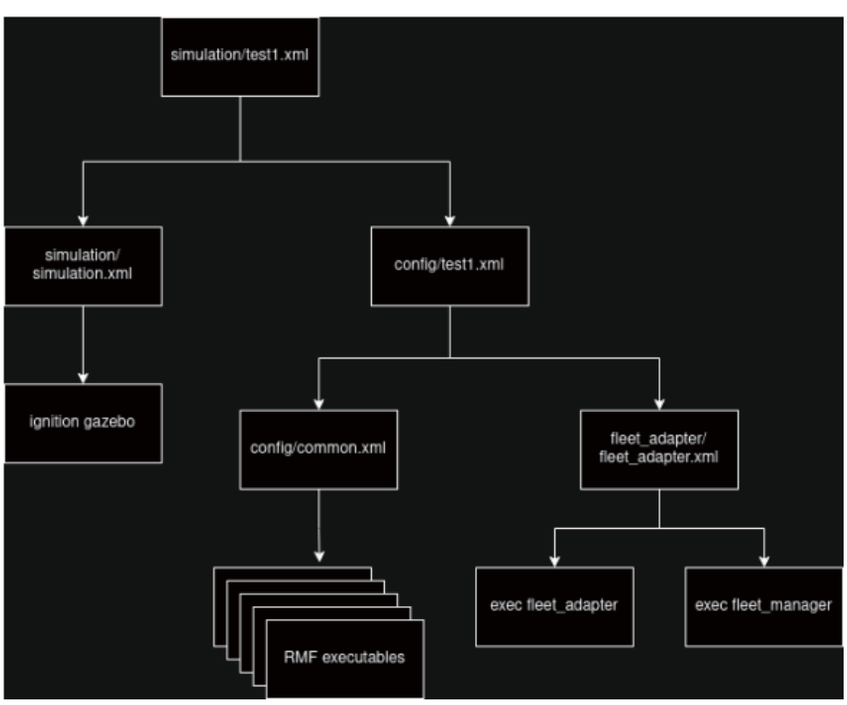

图 24 机器人配置的各项配置的关系

（三）仿真实验流程

1. 启动实验

在完成上述试验的配置后，就可以开始编译文件，如图 25 所示。在编译成功后，就可以启动文件，

启动流程如图 26 所示。

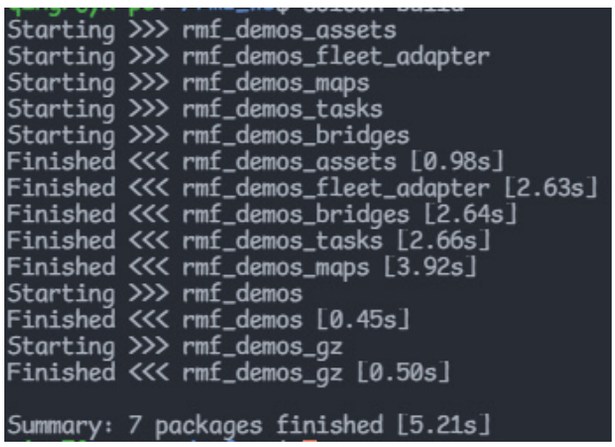

图 25　编译的展示

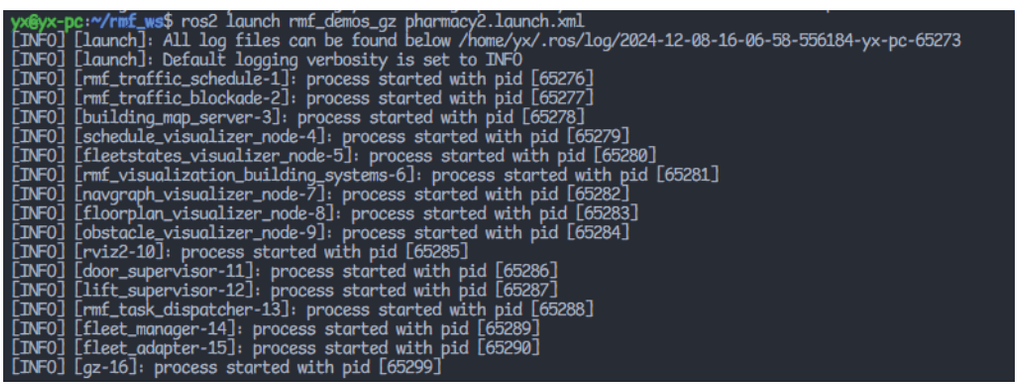

图 26　启动命令后的展示

四、实验数据分析

（一）实验结果展示

本次实验目的是医生只需发送请求，系统自动处理，机器人群自动取药，实现一键取药的功能。因此，当完成启动后，需要发出任务请求来查看返回的结果，用以判断是否能够达到目的。发送命令如图 27 所示。返回结果如图 28 所示。

```
ros2 run rmf_demos_tasks dispatch_delivery -p med1 -ph med -d get1 -dh med
--use_sim_time
```

图 27　发布取药命令

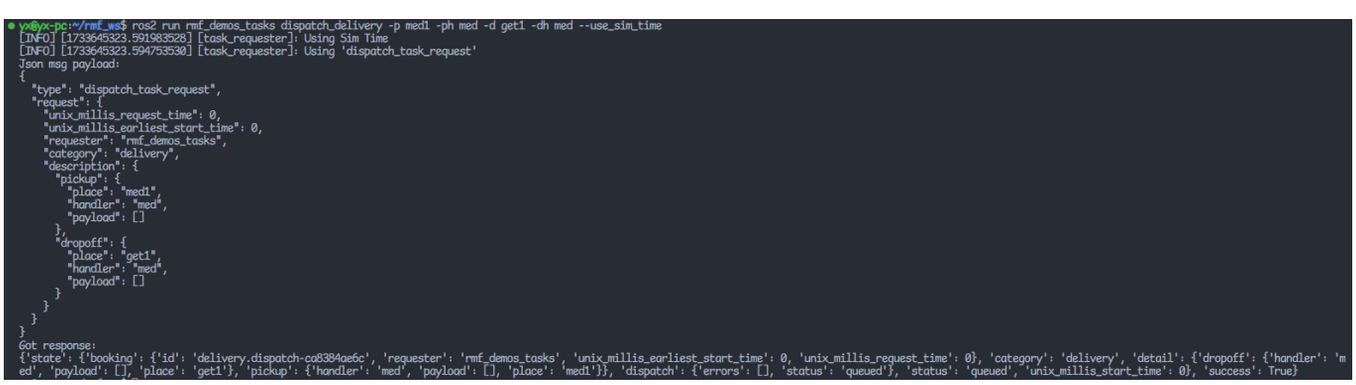

图 28　发布命令后的反馈展示

不同数量的机器人完成任务的时间统计。记录每项任务完成时间，采用 python 用来绘制图，如图 29 所示。

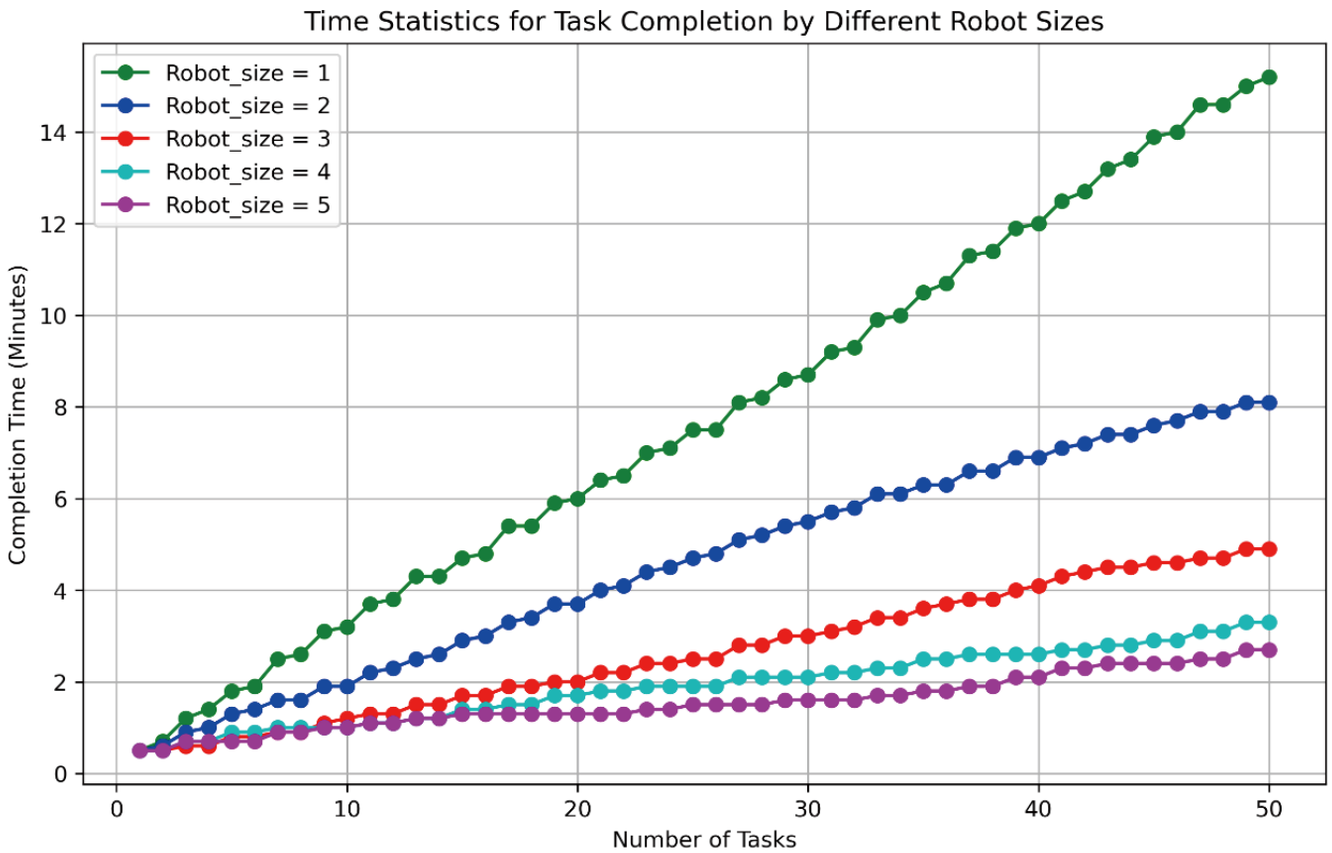

图 29 不同数量机器人完成任务的时间统计图

（二）实验数据分析

如图 29，本实验分别让 1—5 个机器人完成 50 个任务，并记录下了完成时间。实验结果分析如下：

（1）随着机器人数量的增加，任务的总完成时间呈现出下降趋势，这一现象在各种配置中均得到了验证。同时，当任务数量与机器人数量的比值向下取整（ceil（n/robot_size））保持恒定时，单机器人与多机器人配置在完成时间上的差异相对较小，显示出系统在任务分配上的稳定性。

（2）单机器人完成时间的增长趋势近似于一条斜率为 0.5 的直线，表明每个任务平均需要约 0.5 分钟的处理时间。当机器人数量为 2 时，完成时间的增长斜率显著降低至约 0.3，反映出多机器人协作带来的效率提升。进一步增加机器人数量至 3、4 和 5 个时，斜率分别下降至约 0.17、0.12 和 0.1。通过这些数据可以推断，增加机器人数量能够在一定程度上优化任务完成效率，尤其是在任务量较大的场景下，多机器人系统的优势明显。

（3）为了更深入地理解多机器人系统在动态环境中的表现，对实验数据进行了进一步的剖析。单个机器人完成时间的数据为一条斜率约为 0.5 的直线，虽然当机器人数量增加至 2 个时，完成时间的斜率降低至约 0.3，但这一斜率并未达到理论上的理想值（即机器人数量之比）0.25。这一差距提示我们，实际运行中可能存在某些效率损失。进一步分析 3 个、4 个和 5 个机器人配置的数据，其斜率分别为 0.17、0.12 和 0.1，虽然效率持续提升，但与理想状态的差距依然存在，且随着机器人数量的增加，效率增益呈现出边际递减的趋势。例如，4 个和 5 个机器人配置的斜率变化幅度较小，表明继

续增加机器人带来的时间节省逐渐减少。

（三）存在问题与优化方向

1. 算法局限性

由动态环境下的适应性分析可知，系统并没有充分的利用每个机器人，它的路径规划是带有一定代价的，且代价会在机器人数量越来越多的情况下变得越来越显著。未来，可以引入强化学习来优化任务分配，也可以将药房分区，在不同区配置不同速度的取药机器人，以实现取药配送的速度的最大化。

2. 系统扩展性讨论

（1）兼容性问题。本系统使用 RMF 架构，可以轻松的添加不同厂家的机器人。（2）机器人添加问题。通过修改配置文件，可以引入更多的机器。（3）药品引入问题。需要在 traffic-editor 中进行引入。

五、创新与应用前景

（一）研究的创新点

1.OPEN-RMF 在药房自动化中的低成本应用

采用机器人群控的思路和 RMF 框架，最大化利用现有的资源，更加节约产品成本。相较于国内外众多医院所采纳的取药系统，本系统不需要对医院进行改造，对场地的要求低，仅仅只需要投入机器人和服务器的费用，运营和维护成本较低，且在市场上尚未实现大规模商用。

2. 仿真验证的高效性与可移植性

在仿真验证上，本系统利用了一个名为 rmf_visualization 的包，它负责接受 rmf_core 的数据，并且将其输入到 gazebo 可视化，实时映射多机器人协同调度、任务分配及路径规划的全流程，显著缩短开发周期。

（1）高效性：可以快速迭代测试，通过仿真环境，可以在数分钟内完成物理测试需数个小时完成的任务场景验证（如高峰期药房取药的压力测试）；资源复用性高，仿真模型对真实机器人控制物理接口高度兼容，算法参数可以直接移植至实体机器人，节省重复开发成本；可以模拟复杂动态环境（如突发障碍物，机器人故障），验证系统健壮性。

（2）可移植性：跨平台兼容性好，RMF 支持不同厂商的机器人的统一调度，通过标准接口实现不同厂商设备的即插即用；环境的泛化能力好，医院地图配置文件可以快速适配其他场景，配置简单；开源生态支持；由于 RMF 建立在 ROS 上，依托 ROS 社区，可以灵活拓展功能或者测试模块。

（二）实际应用前景

1. 药房管理效率的提升

（1）精准性与时效性：通过多机器人协同调度，传统人工取药的时间大大降低，且错误率趋近于零。

（2）资源动态优化：基于实时数据分析（如药品库存、处方优先级），自动调整机器人任务队列，减少设备闲置率。

（3）每天 24 小时不间断服务：支持夜间处方预处理，缓解日间高峰期压力，提升患者满意度。

2. 扩展至其他自动化场景

（1）实现手术器械包、消毒物资的跨楼层配送，降低人工搬运时，污染药品或者感染的风险。

（2）通过温控机器人完成检验科与病房之间的样本输运，保障生物活性，避免污染。

（3）可以适配于电商分拣中心，实现仓储无人化管理，降低成本以及提高效率。

（4）加入挖矿的功能，使其代替工人在中高风险地区进行挖矿，降低工人遇险的几率。

（三）未来研究方向

（1）引入机器学习优化任务分配。动态负载均衡，基于历史任务数据训练预测模型，判断高峰期机器人的资源需求；优先级自适应，通过强化学习实现紧急任务的自动插队机制；能耗优化，结合机器人电量状态以及任务路径的复杂度，设计能效最优的调度策略。

（2）扩展至其他自动化场景真实硬件平台的部署验证。在医院（复杂人流）、检验室（干净环境）和仓库（高密度货架）等场景部署机器人；研究信息转递与边缘计算在跨区域多机器人通信中的低延迟保障方案；研究避障算法与紧急制动协议，满足 ISO 13482 服务机器人安全标准。

六、结论

本研究基于 OPEN-RMF 框架，成功构建了多机器人协同规划的药房自动化系统，通过 ROS2 与 Gazebo 仿真平台实现了动态任务分配、路径规划及药品配送的全流程验证。实验结果表明，系统在虚拟药房环境中可将单次取药时间明显缩短，且错误率趋近于零，能灵活应对突发障碍物、机器人故障等动态场景，验证了多机器人协同调度的高效性与鲁棒性。本研究的成果不仅为药房自动化提供了创新解决方案，更为多机器人系统在智慧医疗、智能物流等领域的规模化应用提供了技术参考，具有一定的学术价值与产业推广意义。亮点总结如下：

（1）技术整合性：通过开源框架降低商业化门槛，兼顾创新性与落地可行性。

（2）效率革命：从“人工主导”到“机器人群控”，重构药房服务流程。

（3）生态扩展性：以医疗场景为起点，向跨行业自动化领域辐射，推动机器人技术的普惠化发展。

参考文献

[1] 顾继红，缪丽燕．医院门诊药房自动化系统流程建设的实践[J]．中国医院药学杂志，2012.32(8):1225-1226.

[2] 吴军，徐昕，连传强，等．协作多机器人系统研究进展综述[J]．智能系统学报，2011,6(1):13-27.

[3] 王伟嘉，郑雅婷，林国政，等．集群机器人研究综述[J]．机器人，2020,42(2):129-144.

[4] 林艳，蔡志波，黄梦珊，等．国内门诊药房自动化发药系统发展现状及使用效果评价[J]．中国现代应用药学，2020,37(9):1131-1138.

[5] 陈井泉，刘燕．智慧门诊药房的建立与实践[J]．医药导报，2022,41(9):1393-1396.

[6] 周良，陈蓉．智能化门诊药房药品有效期闭环管理的探索与实践[J]．中国药房，2020,31(22):2796-2800.

[7] 朱磊，邹洁，朱宇．实践综合性医院门诊自动化药房的方法及成效[J]．临床合理用药杂志，2018,11(20):98-99.

[8] 林秀丽，洪亮亮，张小燕，等．智能化门诊药房存在的问题及其改进措施[J]．药学服务与研

究 ,2018,18(6):478-480.

[9] 德米斯·哈萨比斯 , 刘迪一 . 让 AlphaFold 的力量为全世界所用 [J]. 世界科学 ,2021(9):1.

[10] Stasevych M, Zvarych V. Innovative robotic technologies and artificial intelligence in pharmacy and medicine: paving the way for the future of health care—A review[J]. Big Data and Cognitive Computing, 2023, 7(3): 147.

[11] Khan O, Parvez M, Kumari P, et al. The future of pharmacy: How AI is revolutionizing the industry[J]. Intelligent Pharmacy, 2023, 1(1): 32-40.

[12] Salzillo, Angelo. Robot fleet control for mining applications using OpenRMF and OpenTCS[D]. Turin: Politecnico di Torino, 2024.

[13] Romero Mollá, Joel. Practical guide to implement OpenRMF in multi-robot environments[D]. Alicante: Universidad de Alicante, 2024.

Research on Multi-Robot Collaborative Planning in Pharmacy Automation Based on OPEN-RMF

YAN Yixin
Xuancheng Campus, Hefei University of Technology, Xuancheng, Anhui 242000, China

Abstract: With the growing demand for pharmacy automation, multi-robot collaborative technology has become crucial for improving efficiency and accuracy. This study investigates the practical application of multi-robot collaborative planning in pharmacy automation based on the OPEN-RMF (Open Robotics Middleware Framework). A virtual pharmacy environment was constructed using the ROS2 and Gazebo simulation platforms to achieve multi-robot task allocation, path planning, and medication delivery. Experimental results demonstrate that the proposed method significantly enhances drug distribution efficiency, reduces manual intervention, and adapts to dynamic environmental changes. The research provides innovative insights and technical references for the intelligent development of pharmacy automation, offering significant practical value.

Keywords: Multi-Robot Collaboration; Path Planning; Task Allocation; ROS2