

异构无人集群系统仿真平台使用指南

北京航空航天大学人工智能学院

1 仿真平台简介

XGC 异构无人集群系统仿真与试验平台，集成了多类型无人设备（包括无人车、四旋翼无人机及固定翼无人机）的协同算法验证功能。该平台基于 `swarm_sync_sim` 等数值仿真器，实现了高效的半实物仿真，并具备图形化参数配置与仿真/实物模式的一键切换功能。平台支持实时状态监控、算法托管、3D 场景可视化及数据记录与分析等核心功能。



图 1: 仿真平台主界面

2 平台部署方法

本课程采用 Docker 容器化部署仿真平台，建议同学们提前了解操作系统、虚拟机、镜像与容器等基本概念。操作系统可以选用 Ubuntu（推荐）或 Windows。系统要求及平台部署方法详见本章 2.1 节（Ubuntu）和 2.2 节（Windows）。课程资料下载链接如下

- <https://bhpan.buaa.edu.cn/link/AAA7E7E827C7514B2D80468406B57D731A>
- Folder Name: 2025 年春季学期-无人集群智能协同控制
- Pickup Code: xtyr

其中包括：

- `xgc.tar.gz`: 仿真平台及其运行环境的 Docker 镜像压缩文件，用于部署平台。
- `xgc_ws.zip`: ROS1 工作空间，用于存放编写的算法代码，包含了示例代码。
- `xgc_documents.zip`: 存放仿真平台配置文件，包含了示例配置文件。
- `docker-compose-linux-x86_64`: Docker Compose 的安装包。

2.1 Ubuntu 系统下部署方法（推荐）

系统软硬件要求见表 1。需要先安装 Docker，可选择安装 nvidia-container-toolkit 在容器内启用 GPU 硬件加速，再使用 Docker 部署仿真平台。

类别	要求
处理器架构	x86_64 或 amd64
内存	至少 4GB，建议 8GB 或更高
磁盘空间	预留至少 15GB
操作系统	Ubuntu 20.04（推荐）或更新版本

表 1: 部署仿真平台的系统硬件与软件要求

2.1.1 Docker 安装步骤

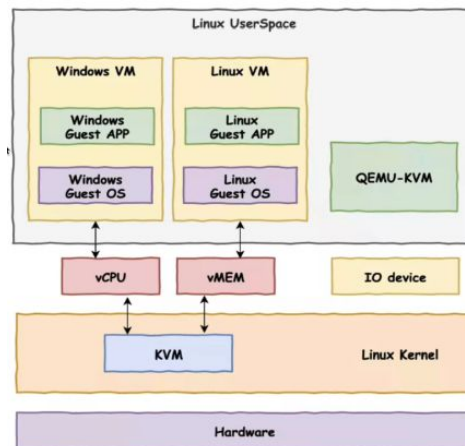


图 2: 操作系统、虚拟机与容器

参考官方说明文档 [1]，安装 Docker。如果因为网络问题配置失败，可以多次尝试或者使用加速器。接着，安装 Docker Compose，这是一个用于定义和管理多容器应用的工具，可以简化容器的创建、配置和编排。可以直接使用网盘提供的文件 docker-compose-linux-x86_64，也可以在官网仓库下载 [2]。下载后改名为 docker-compose。并移动到 /usr/local/bin/docker-compose，再赋予执行权限。

```
sudo chmod +x /usr/local/bin/docker-compose # 赋予执行权限，注意 +x 前有空格
```

在任意终端执行以下命令，测试是否安装成功。

```
lzk@lzk:~$ docker -v # 注意 -v 前有空格
Docker version 28.0.1, build 068a01e # 打印当前版本号，说明 docker 安装成功
lzk@lzk:~$ docker-compose -v
Docker Compose version v2.33.1 # 打印当前版本号，说明 docker-compose 安装成功
```

将当前用户加入 docker 组，之后无需使用 sudo 执行 docker 命令。

```
sudo usermod -aG docker $USER # 将当前用户加入docker组，注意 -aG 前有空格
```

```
sudo systemctl restart docker # 需要重启 docker 服务生效
```

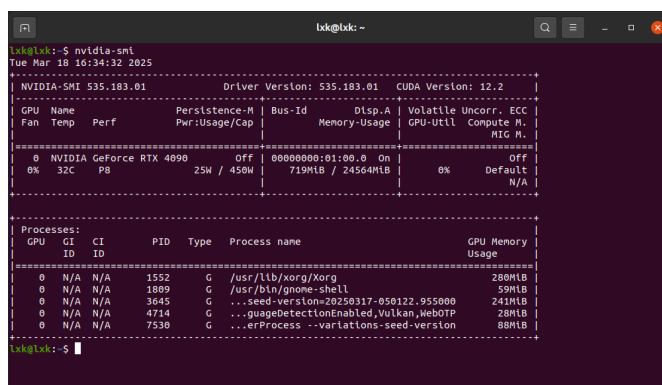
```
# 如果依旧提示权限不足，可以尝试重启电脑
```

如果在 Windows 下部署，在 WSL2 的 Ubuntu 系统中执行 systemctl 命令报错，可能是当前 wsl 版本过低。建议升级 Windows 系统到最新版本，然后升级 wsl，通过在 Windows 的 PowerShell 中执行 `wsl --shutdown` 与 `wsl --update`。

2.1.2 安装 nvidia-container-toolkit (可选)

如果想在 Docker 容器中使用 nvidia 显卡进行硬件加速，提高仿真平台可视化的渲染效率或使用显卡加速训练，可以按照 2.1.2 配置。Windows 系统目前不支持硬件加速，可以跳过此步骤。

首先，确认系统安装了 nvidia 驱动，执行 `nvidia-smi`，效果应该如图 3 所示。如果提示 Command 'nvidia-smi' not found，则需要选择适配自己显卡的驱动版本 [3]，参考网上教程，进行安装。



```
lsk@lsk:~$ nvidia-smi
Tue Mar 18 16:34:32 2025
+-----+
| NVIDIA-SMI 535.183.01                  Driver Version: 535.183.01   CUDA Version: 12.2     |
+-----+-----+
| GPU   Name                               Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap       Bus-Id        Memory Usage | GPU-Util  Compute M. |
|=====+=====+
| 0.    NVIDIA GeForce RTX 4090             Off | 00000000:01:00.0 On   |          Off         |
| 6%    32C    PB      25W / 450W           719MiB / 24564MiB |    6%      Default   |
+-----+-----+

Processes:
+-----+
| GPU   GI   CI        PID   Type   Process name                      GPU Memory |
| ID   ID   ID           |              |                     Usage         |
+-----+-----+
| 0.    N/A  N/A       1552    G   /usr/lib/xorg/Xorg                  280MiB |
| 0.    N/A  N/A       1809    G   /usr/bin/gnome-shell                 59MiB |
| 0.    N/A  N/A       3645    G   ...seed-version=20250317-050122.955000 243MiB |
| 0.    N/A  N/A       4714    G   ...guageDetectionEnabled,Vulkan,WebOTP   28MiB |
| 0.    N/A  N/A       7530    G   ...erProcess --variations-seed-version  88MiB |
+-----+-----+
lsk@lsk:~$
```

图 3: ubuntu 下安装显卡驱动，确保 nvidia-smi 有输出，可以打印显卡型号与驱动版本等信息

驱动安装完成后，要在 Docker 容器中使用显卡，需要按照官方说明 [4]，在 Ubuntu 宿主机（并非容器）安装 nvidia-container-toolkit，将显卡资源暴露给容器内部应用。注意 [4] 中 apt-get install nvidia-container-toolkit 之后，还要修改配置。

```
# 以下命令自动更改配置/etc/docker/daemon.json，指定使用 nvidia 的容器运行时
```

```
sudo nvidia-ctk runtime configure --runtime=docker
```

```
# 重启服务生效
```

```
sudo systemctl restart docker
```

```
# 检验有没有安装成功
```

```
docker info | grep -i nvidia
```

2.1.3 安装仿真平台

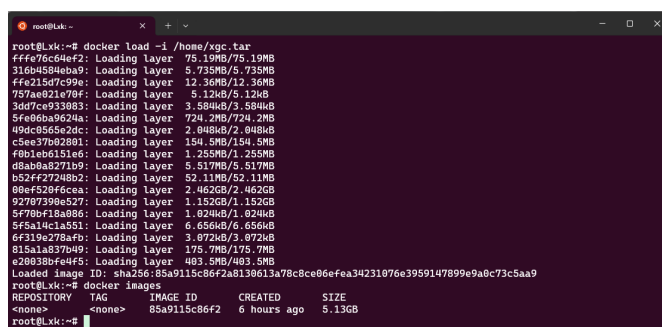
下载仿真平台的镜像压缩文件 `xgc.tar.gz`，使用命令加载镜像。如果在 `docker images` 的输出中，REPOSITORY 和 TAG 分别显示 `xgc` 与 `latest`，说明导入成功。如果 REPOSITORY 和 TAG 都显示为 `<none>`，说明出现悬空，则需要手动执行 `docker tag`。

```
# 在保存 xgc.tar.gz 的路径下打开终端，运行以下命令解压并加载镜像
gunzip -c xgc.tar.gz | docker load

# 等待加载完成，列出当前可用镜像，查看是否成功加载
docker images

# 如果出现悬空镜像则执行以下命令，重新打标签。注意，<IMAGE ID> 应该替换为悬空镜像的 id。
# docker tag <IMAGE ID> xgc

# 如果成功加载镜像，可以删除原始的压缩文件
# rm xgc.tar.gz
```



```
root@lxk:~# docker load -i /home/xgc.tar
ffff76c64ef2: Loading layer 75.19MB/75.19MB
316b4584eba9: Loading layer 5.735MB/5.735MB
ffe215d7c99e: Loading layer 12.36MB/12.36MB
757ae021e70f: Loading layer 5.124MB/5.124MB
3dd7ce933883: Loading layer 3.584MB/3.584MB
5fe06ba9624a: Loading layer 724.2MB/724.2MB
49dc9565e2dc: Loading layer 2.848MB/2.848MB
c5ae370d2801: Loading layer 159.5MB/159.5MB
f8b1eb6151e6: Loading layer 1.255MB/1.255MB
d8ab0a8271b9: Loading layer 5.517MB/5.517MB
b52ff27248b2: Loading layer 52.11MB/52.11MB
89eef5946c0a: Loading layer 2.462MB/2.462MB
92707390e527: Loading layer 1.152GB/1.152GB
5f70bf18a886: Loading layer 1.024MB/1.024MB
5f5a14c1a501: Loading layer 6.856MB/6.856MB
6f319c278a0b: Loading layer 3.972MB/3.972MB
815a1a837b49: Loading layer 175.7MB/175.7MB
e20838bf4f5: Loading layer 483.5MB/483.5MB
Loaded image ID: sha256:85a9115c86f2a8139613a78c8ce06efea3423107e3959147899e9a8c73c5aa9
root@lxk:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
<none>        <none>    85a9115c86f2   6 hours ago   5.13GB
```

图 4: 列出当前镜像

`xgc` 镜像不仅包含了 Ubuntu 操作系统的基本文件，还打包了仿真平台及编译与运行算法所需的 ROS1 工具链，确保在任何机器上运行容器时，都能获得一致的开发环境，而不依赖于宿主机的系统配置。需要注意的是，镜像是静态的，而容器是动态的。镜像仅为固定的模板，而容器则是基于镜像创建的独立运行实例。在运行时，容器相当于一台加载了镜像内容的“虚拟开发机”，可以在其中运行仿真平台，编译并运行算法。

容器的文件系统默认与宿主机隔离，这意味着容器内的文件操作不会影响宿主机，反之亦然。但是，仿真平台在运行时会产生数据文件，如保存无人机控制误差时间序列的 `rosbag`。为了避免删除容器时丢失这些重要数据，需要将容器内的部分目录映射到宿主机的指定目录。通过宿主机目录中的文件不会因容器删除而丢失的特性，实现重要数据的持久化。另一方面，也需要在容器环境中访问宿主机文件，如仿真平台配置文件与算法源码。

可以使用命令 `mkdir -p $HOME/xgc_documents/XGC $HOME/xgc_ws`，在宿主机上创建两个目录。变量 `$HOME` 是当前用户的主目录，比如当前用户是 `zhangsan`，`$HOME` 就是 `/home/zhangsan`；当前用户是根用户 `root`，`$HOME` 就是 `/root`。这两个目录的作用如下：

- `$HOME/xgc_ws`：用于存储仿真平台的配置文件和控制算法代码，后续将在该目录下进行算法编写。容器中目录 `/home/xgc_ws` 会被挂载到宿主机的这个目录上。

- `$HOME/xgc_documents`: 用于存储记录的仿真数据。容器中目录 `/home/xgc_documents` 会被挂载到这个目录上。

将下载的 `xgc_ws.zip` 其中内容解压到 `$HOME/xgc_ws` 中, `xgc_documents.zip` 其中内容解压到 `$HOME/xgc_documents` (如果 WSL2 的 Ubuntu 没创建普通用户, 就保存到 `/root/xgc_ws`)。使用以下命令启动容器 (注意选择自己对应的命令), 成功效果如图 5 所示。

```
# Windows 或者 Ubuntu 没有安装 nvidia-container-toolkit, 使用这个命令启动容器
cd $HOME/xgc_ws && docker-compose -f docker-compose-cpu.yml up

# Ubuntu 安装了 nvidia-container-toolkit, 使用这个命令启动容器
cd $HOME/xgc_ws && docker-compose -f docker-compose-gpu.yml up

# 列出正在运行的容器
docker ps

# 如果需要停止容器
# docker stop xgc

# 如果需要删除容器。
# docker rm xgc
```

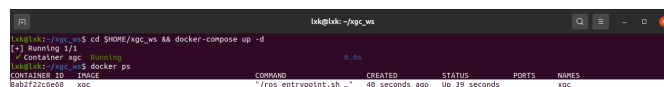


图 5: 列出当前运行容器, 仿真平台所在容器 `xgc` 正在运行, 已运行了 39 秒

如果容器运行成功, 通过以下命令, 进入容器并初始化示例代码所在的 ROS1 工作空间。

```
# 进入容器, 编译工作空间
docker exec -it xgc "/bin/bash"
cd /home/xgc_ws/ && catkin_make
```

注意, 如果后续代码都使用 Python 编写, 则无需再次 `catkin_make` 编译。但是需要进入容器给新建的 `py` 文件赋可执行权限, 否则无法通过仿真平台启动算法。如果代码用 C++ 编写, 则代码修改后需要再次执行上面的命令。

接下来, 通过以下命令启动仿真平台。如果关闭了仿真平台的界面, 则需要重新执行这个命令。

```
# 进入容器, 并启动仿真平台
docker exec -it xgc "/bin/bash"
/home/xgc/AppRun # 必须在容器中才能启动仿真平台!

# 以上两个命令等效
# docker exec -it xgc "/bin/bash" -ic /home/xgc/AppRun
```

看到如图 6 所示效果, 说明部署与启动成功。

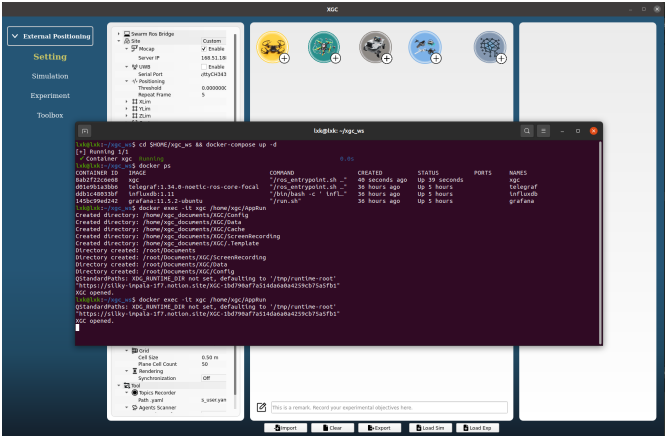


图 6: 仿真平台默认界面

2.2 Windows 系统下部署方法

系统软硬件要求见表 2。需要先安装 WSL2 虚拟机，在 WSL2 中安装 Ubuntu 系统，在 WSL2 模拟出的 Ubuntu 系统中使用 Docker 部署仿真平台。

类别	要求
处理器	x86_64 或 amd64，处理器支持虚拟化扩展
内存	至少 4GB，建议 8GB 或更高
磁盘空间	预留至少 30GB
操作系统	Windows 10 版本 16215.0 或更高版本 / Windows 11（推荐）

表 2: 部署仿真平台的系统硬件与软件要求

2.2.1 WSL2 安装步骤

WSL2 (Windows Subsystem for Linux 2) 由微软开发，提供了完整的 Linux 内核，使 Windows 能够高效运行 Linux 环境。它支持安装多个 Ubuntu 发行版以及其他 Linux 发行版，具备原生 Linux 文件系统、完整的系统调用兼容性和更快的 I/O 性能，适用于开发者运行 Linux 工具、Docker、AI 训练等任务，同时保持与 Windows 系统的无缝集成。

使用 WSL2 要求处理器支持虚拟化扩展，并且必须在 BIOS/UEFI 中启用，通常叫 Intel VT-x (Intel) 或 SVM Mode (AMD)。部分电脑默认关闭虚拟化扩展，需要手动开启，可以在任务管理器中检查是否开启，如图 7 所示。

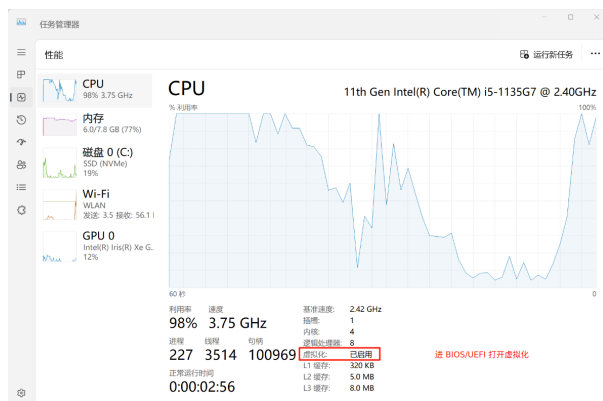


图 7: 查看是否启用虚拟化

在搜索栏找到**启用或关闭 Windows 功能**，勾选其中 Hyper-V、虚拟机平台与适用于 Linux 的 Windows 子系统这三项。部分 Windows 版本（如家庭版）没有安装 Hyper-V，列表中看不到该项，则需要手动安装 [5]。

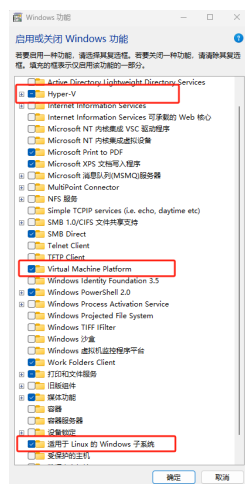


图 8: 安装 WSL2 的前置条件

Win+X 按 A，打开 Windows 的 PowerShell 终端（需要以管理员方式运行），开始安装 WSL2。

```
wsl --install
wsl --set-default-version 2 # 默认使用 WSL2

# 更新 wsl
wsl --update
```

微软应用商店搜索 Ubuntu 20.04，如图 9 所示，下载并安装。

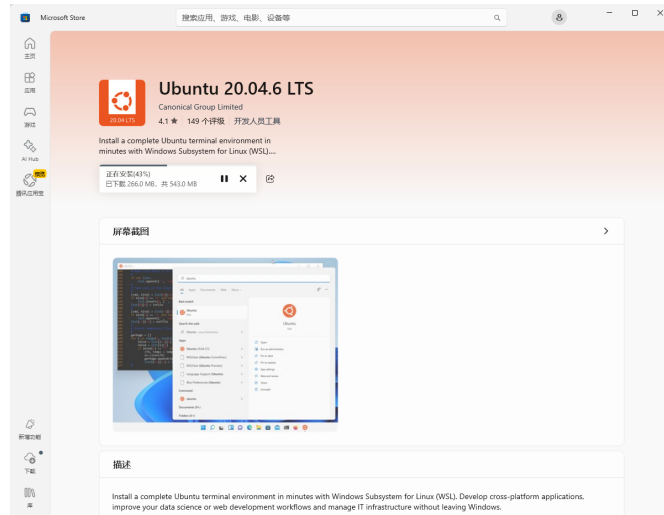


图 9: 应用商店安装 Ubuntu 20.04

默认安装位置在 C 盘，如果需要迁移到其他盘，可以在**安装的应用**中，找到 Ubuntu 20.04，点击**移动**，选择目标盘。



图 10: 选择安装的应用

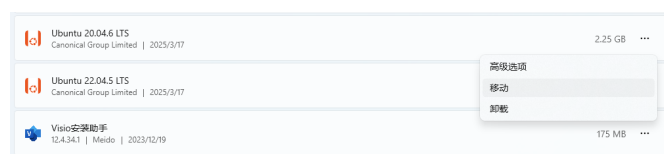


图 11: 应用迁移

安装完毕后，第一次打开 Ubuntu 终端时，会要求新建用户并设置密码，不要输入直接关闭终端，重新打开一个 Ubuntu 终端，就会跳过新建步骤进入正常终端。这里不创建普通用户的原因是，创建普通用户后，会自动取代根用户作为默认登录用户，导致无法通过 Windows 的文件资源管理器修改 Ubuntu 系统中的文件，提示权限不足。

有多种打开 Ubuntu 终端的方法，这里列举三种。之后的步骤注意区分命令是在 Windows PowerShell、Ubuntu 终端还是在 Docker 容器中运行。比如所有 docker 开头的命令，都应该在 Ubuntu 终端中运行；所有 wsl 开头的命令，应该在 Windows 的 PowerShell 中运行。Ubuntu 终端区别于 PowerShell，如果登录用户是根用户，输入命令前会有 # 提示，普通用户则是 \$。

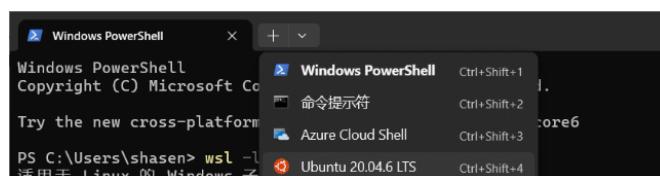


图 12: 方法一, 在 PowerShell 中点击向下的箭头, 展开菜单中找到 Ubuntu 20.04

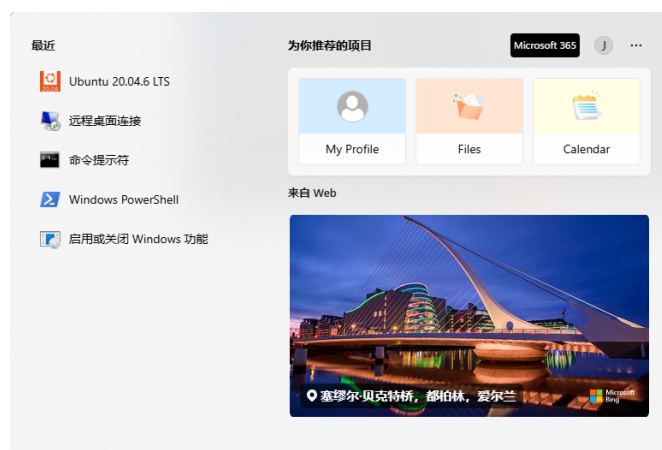


图 13: 方法二, 搜索栏输入 Ubuntu 20.04, 点击橙色图标

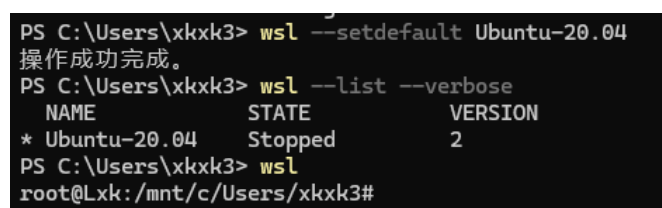


图 14: 方法三, 在 Windows PowerShell 中设置默认发行版, 之后直接输入 wsl, PowerShell 会变为 Ubuntu 终端

应用商店安装的 Ubuntu 发行版为服务器版本, 没有安装图形用户界面。可以借助 Windows 的文件资源管理器, 对 Ubuntu 系统下文件进行增删查改, 如图 15 所示。同时, Ubuntu 下运行的 GUI 应用 (比如仿真平台) 会自动作为 Windows 窗口弹出 [6]。没有 Ubuntu 自带的 GNOME 桌面环境对实验影响不大, 可以选择性安装。

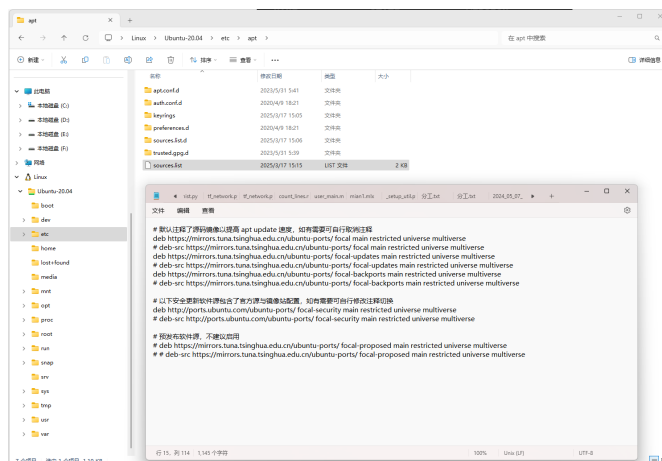


图 15: 可以在 Windows 文件管理器中修改 Ubuntu 系统文件

2.2.2 Docker 安装步骤

同 2.1.1。注意，安装 docker 的命令应该运行在 Ubuntu 终端（输入命令前要有 # 或 \$ 提示，终端打开方式参考图 12-14），而非 Windows 的 PowerShell 或者命令提示符中。相关资源下载与移动，可以使用 Windows 的浏览器下载，然后借助 Windows 的文件资源管理器移动到 Ubuntu 系统中（参考图 15）。如果 2.1.1 中安装创建了普通用户，移动文件时可能报错，需要修改 Ubuntu 默认登录用户。

2.2.3 安装仿真平台

同 2.1.3，如果出现如图 6 所示界面，说明部署与启动成功。

3 基本功能介绍

仿真平台可以实现无人机/无人车仿真器与算法的快速启动以及相关参数的图形化配置。其中，无人机/无人车控制输入的发布，以及位置与速度等状态的获取都是基于 ROS1（Noetic 版本）话题通信。建议算法以 ROS1 功能包的形式来组织与编写，方便使用 rosrn 或者 roslaunch 命令来启动，编程语言推荐 Python 或者 C++。建议同学们学习 ROS1 功能包创建与话题通信的方法 [7]（掌握 Python 与 C++ 其中一种即可）。注意，ROS1 已经包含在 Docker 容器中，无需在宿主机中重复安装，在 Ubuntu 的终端中使用 `docker exec -it xgc "/bin/bash"` 进入容器后，可以使用 ROS1 相关的功能。

3.1 增删无人机/无人车配置

在配置界面（左侧 Setting 按钮会高亮），分别点击中间上方绿色/灰色按钮添加无人机/无人车，可以添加任意数量的无人机与无人车组成异构集群系统。点击下方 Clear 按钮可以清除所有个体。中间上方最右侧的按钮可以添加算法，算法配置方法详见 Lab1。

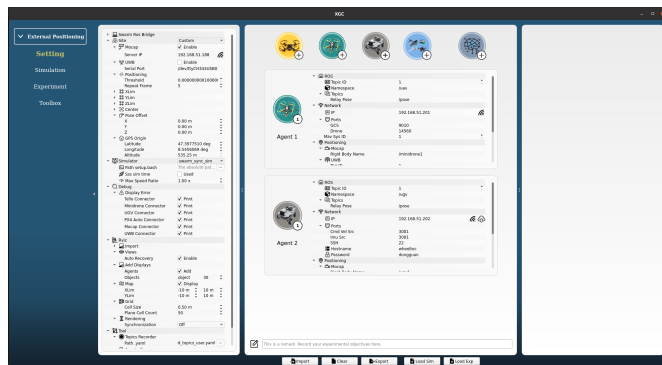
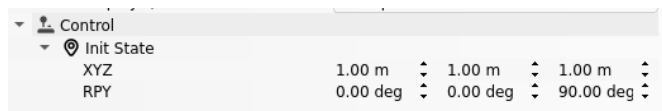


图 16: 配置界面

鼠标停留在某个无人机/无人车的配置框上，拖动滚轮找到 XYZ 如下图所示，可以设置该个体在全局惯性坐标系中的初始位置。

图 17: 初始状态在 $[1.0, 1.0, 1.0]$ ，单位米

3.2 导入/导出配置

首先演示如何导出当前集群系统的参数配置到文件，添加一架无人机。点击下方 Export 按钮导出配置，会弹出选择框，保持默认的存储路径不要修改（默认容器中的存储位置被挂载到了宿主机的目录 $\$HOME/xgc_documents$ ），可以修改文件名，比如重命名为 test.xconfig，点击选择框的 Save 按钮保存文件。

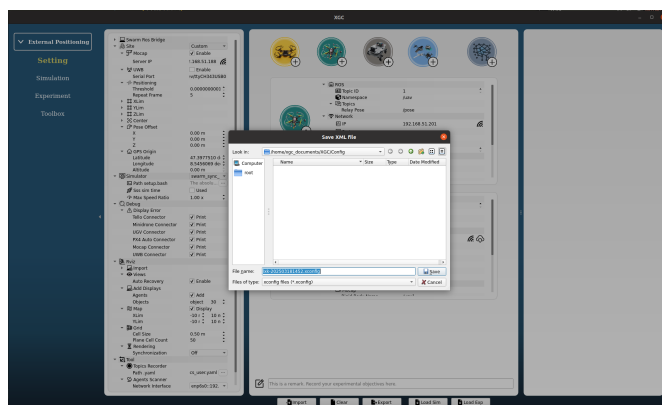


图 18: 导出配置

虽然仿真平台运行在容器中，但是挂载了保存配置文件的目录，所以在宿主机中可以看到刚才保存的配置文件。如果是 Ubuntu 系统就在 $\$HOME/xgc_documents/XGC/Config$ 路径下找。如果是 Windows 系统可使用文件资源管理器找到导出的文件，如图 19 所示。

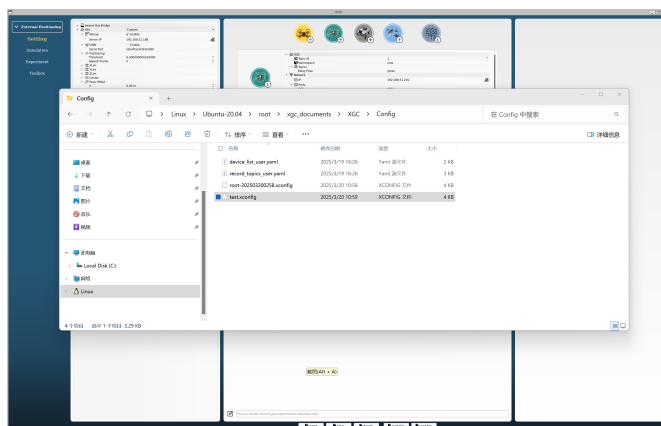


图 19: 在 Windows 文件资源管理器中查看导出的配置文件

点击下方 Clear 可以删除添加的无人机, 点击下方 Import 按钮可以导入刚才的配置。同样会弹出对话框, 默认路径下就可以看到 test.xconfig 文件, 选中并点击 Open 按钮, 可以重新添加无人机并恢复导出时参数。

3.3 启动仿真

添加一架无人机与一辆无人车的配置后, 点击下方 Load Sim 加载仿真按钮 (在 Export 右侧) 可以加载仿真, 每个无人机/无人车的仿真器都启动完毕后效果如图 20 所示, 左侧对应仪表变为蓝色并显示位置与姿态等数据。默认情况下, 所有无人机和无人车都会被选中 (选中的个体的仪表有黄色边框与对勾, 可点击仪表选中/取消选中), 先后点击下方 Disconnect 与 Connect 按钮, 可以停止与启动当前所有选中的无人机/无人车的仿真器 (默认已启动), 实现仿真初始状态的重置。

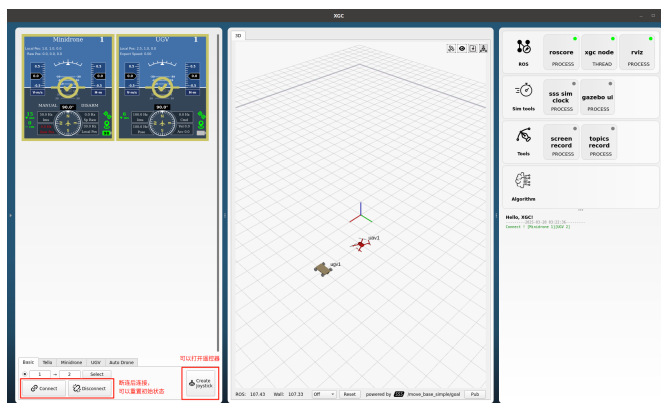


图 20: 加载仿真后主界面

3.4 无人机/无人车的运动控制

如图 20 所示, 左下方的控制面板可以对当前选中的个体发布一些基本的控制命令, 有多个标签页。无人机/无人车共用的控制命令在 Basic 页, 各自的控制命令分别在 Minidrone 页与 UGV 页中。基本控制命令除了关闭与打开选中无人机/无人车的仿真器, 还有一个简易的遥控器。如图 20 所示, 点击 Create Joystick 按钮创建一个遥控器, 如果当前选中了多个个体, 则这些个体共用这个遥控器。可以多次创建遥控器, 但是不能重复遥控, 在使用自己的算法控制时需要关闭遥控器。

创建遥控器后，无人车可以直接运动，无人机则模拟 PX4 飞控无人机，需要发布命令切换状态机进行起飞。如图 21 所示，依次点击三个按钮，并等待无人机起飞。之后可以使用遥控器控制无人机/无人车在机体坐标系下的 x 轴与 y 轴速度以及偏航角速度。

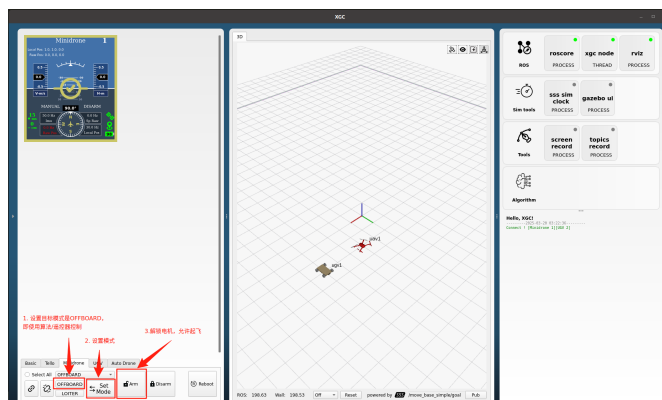


图 21: 无人机起飞步骤

3.5 数据记录与分析

如果无人机/无人车在遥控或者算法控制下，可以使用 Plotjuggler 可视化工具 [8] 分析数据，如无人机/无人车的控制输入、位置姿态与控制误差等。Plotjuggler 支持实时显示与离线分析（将数据记录到 bag 文件中，然后导入工具），实时显示的使用方法如图 22 所示，在左侧切换到 Toolbox 页，点击中间的 PlotJuggler 可以快速打开工具，然后按照图 23，将想要订阅的话题拖入到右侧显示的区域。

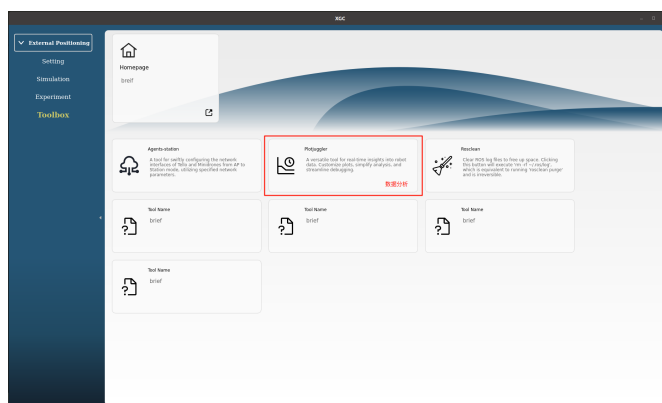


图 22: 无人机起飞步骤

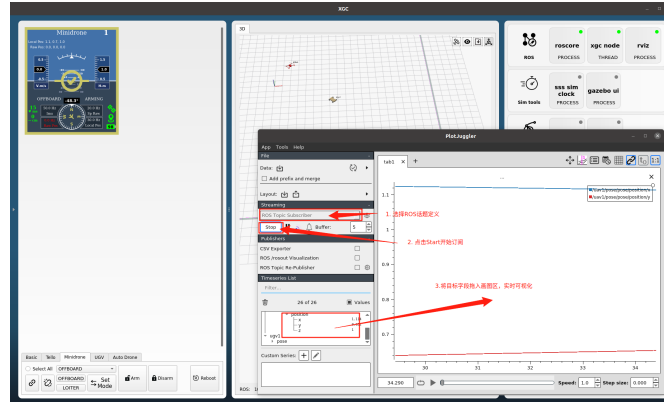


图 23: 无人机起飞步骤

可以遥控无人车与无人机的同时，尝试实时显示相关话题：

- `/uav1/mavros/setpoint_raw/local` 无人机的控制输入话题
- `/uav1/mavros/local_position/pose` 无人机位置与姿态话题
- `/uav1/mavros/local_position/velocity_local` 无人机速度话题
- `/ugv1/cmd_vel` 无人车控制输入话题
- `/ugv1/pose` 无人车位置与姿态话题

注意，遥控器向控制输入话题发布消息，例如期望的线速度与角速度，实现无人机/无人车的运动控制。同理，可以使用自己算法计算控制输入，并发布消息到这个话题。如果有多个无人机/无人车，编号为 1,2,3,... (车机分开编号)，更改话题中数字即可。

参考文献

- [1] Docker. *Docker Engine Installation Guide for Ubuntu*, 2024. URL <https://docs.docker.com/engine/install/ubuntu/>.
- [2] Docker. Docker compose v2.34.0, 2024. URL https://github.com/docker/compose/releases/download/v2.34.0/docker-compose-linux-x86_64.
- [3] NVIDIA. Nvidia 驱动程序查找, 2025. URL <https://www.nvidia.cn/drivers/lookup/>.
- [4] NVIDIA. Nvidia container toolkit installation guide, 2024. URL <https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html#with-apt-ubuntu-debian>.
- [5] 联想服务. Win11 家庭版如何开启 hyper-v, 2025. URL <https://newsupport.lenovo.com.cn/commonProblemsDetail.html?noteid=424688>.
- [6] Microsoft. 在适用于 linux 的 windows 子系统上运行 linux gui 应用, 2024. URL <https://learn.microsoft.com/zh-cn/windows/wsl/tutorials/gui-apps>.
- [7] Autolabor 官方. 【autolabor 初级教程】ros 机器人入门, 2024. URL <https://www.bilibili.com/video/BV1Ci4y1L7ZZ/>.

[8] Plotjuggler. Plotjuggler, 2024. URL <http://wiki.ros.org/plotjuggler>.