

Table of Contents

| | |
|----------------------|---------|
| Introduction | 1.1 |
| 1 产品介绍 | 1.2 |
| 1.1 设计理念 | 1.2.1 |
| 1.2 适用人群 | 1.2.2 |
| 1.3 应用场景 | 1.2.3 |
| 1.4 周边配件 | 1.2.4 |
| 2 产品特性 | 1.3 |
| 2.1 机器规格参数 | 1.3.1 |
| 2.2 控制核心参数 | 1.3.2 |
| 2.3 机械结构参数 | 1.3.3 |
| 2.4 电气特性参数 | 1.3.4 |
| 2.5 笛卡尔坐标系 | 1.3.5 |
| 3 用户须知 | 1.4 |
| 3.1 安全须知 | 1.4.1 |
| 3.2 运输和储存 | 1.4.2 |
| 3.3 维护和保养 | 1.4.3 |
| 3.4 常见问题解决 | 1.4.4 |
| 4 首次安装使用 | 1.5 |
| 4.1 产品标准清单 | 1.5.1 |
| 4.2 产品开箱指南 | 1.5.2 |
| 4.3 开机检测指南 | 1.5.3 |
| 5 基础功能使用 | 1.6 |
| 5.1 系统使用说明 | 1.6.1 |
| 5.2 软件使用说明 | 1.6.2 |
| 5.3 固件功能说明 | 1.6.3 |
| 6 SDK 开发指南 | 1.7 |
| 6.1 基于python开发使用 | 1.7.1 |
| 6.2 机器人操作系统 1 (ROS1) | 1.7.2 |
| 1 环境搭建 | 1.7.2.1 |
| 2 ROS基础知识 | 1.7.2.2 |
| 3 Rviz使用 | 1.7.2.3 |
| 4 基本功能案例 | 1.7.2.4 |
| 6.3 机器人操作系统 2 (ROS2) | 1.7.3 |
| 1 环境搭建 | 1.7.3.1 |
| 2 ROS2基础知识 | 1.7.3.2 |

| | |
|---------------|---------|
| 3 Rviz2使用 | 1.7.3.3 |
| 4 基本功能案例 | 1.7.3.4 |
| 6.4 基于C++开发使用 | 1.7.4 |
| 6.5 基于JS开发使用 | 1.7.5 |
| 6.6 基于C#开发使用 | 1.7.6 |
| 6.7 基于APP开发使用 | 1.7.7 |
| 7 机械臂使用场景案例 | 1.8 |
| 7.1 3D无序分拣 | 1.8.1 |
| 7.2 2.5D二维码分拣 | 1.8.2 |
| 8 相关资料下载 | 1.9 |
| 8.1 产品资料 | 1.9.1 |
| 8.2 产品图纸 | 1.9.2 |
| 8.3 软件资料及源码 | 1.9.3 |
| 8.4 系统资料 | 1.9.4 |
| 8.5 宣传资料 | 1.9.5 |
| 9 关于我们 | 1.10 |
| 9.1 大象机器人 | 1.10.1 |
| 9.2 联系我们 | 1.10.2 |

Pro630_docs

[Pro630目录页](#)

4

1
2

1
2

1
2

1
2

1
2

10

11

12

13



用户须知



本章节是使用本产品的每一位用户必须仔细阅读的重要部分。它涵盖了关于产品使用、运输、储存及维护的关键须知，旨在确保用户在操作产品时的安全性和效率。此外，本章节也详细说明了因未遵循这些指南而可能导致的产品故障或损害的责任划分。用户须知分为几个小节，每个小节针对不同的主题提供详细的指导和建议：

- **安全须知**：包括责任划分、安全警告标志、通用安全准则、人身安全和紧急情况应对。
- **运输和储存**：说明产品的打包、运输和长期储存要求，以及相关的责任划分。
- **维护和保养**：提供日常维护和保养的指导，以延长产品的使用寿命。
- **常见问题解决**：为用户提供一个易于导航的问题解决指南，帮助他们快速解决常见问题。

通过仔细阅读本章节，用户将能够更好地理解如何安全、高效地使用产品，从而最大限度地提高产品性能和使用寿命。

[← 上一页](#) | [下一页 →](#)

安全须知

1 简介

本章详细介绍了有关对大象机器人执行安装、维护和维修工作的人员的常规安全信息。请在搬运、安装和使用前，先充分阅读和理解本章节的内容与注意事项。

2 危险识别

协作机器人的安全性建立在正确配置和使用机器人的前提上。并且，即使遵守所有的安全指示，操作者所造成的伤害或损伤依然有可能发生。因此，了解机器人使用的安全隐患是非常重要的，有利于防患于未然。以下表 1-1~3 是使用机器人的情境下可能存在的常见安全隐患: <!--

<!--

运输和储存

1 装箱打包



2 物流运输

在运输过程中，应使用原始包装运输mycobot pro630。在运输过程中，应确保mycobot pro630作为一个整体是稳定的，并通过适当的措施加以保护。在运输和长期储存过程中，环境温度应保持在-20至+55°C的范围内，湿度≤95%且无凝露。

由于机器人是精密机械，从包装中取出mycobot pro630时，应小心处理。在运输过程中，如果不能稳定放置，可能会引起振动并损坏机器人的内部部件。

3 设备储藏

运输完成后，原包装应妥善存放在干燥的地方，环境温度应保持在-20至+55°C的范围内，湿度≤95%且无凝露，以备将来重新包装和运输需要。不要将其他物品堆放到机械臂的原包装箱上，防止包装箱变形和机械臂损坏。

[← 上一页](#) | [下一页 →](#)

维护和保养

机器人维护和保养指南 作为一家机器人制造商，我们重视确保客户能够正确、安全地维护和升级他们的机器人设备。为此，我们提供以下详细的维护和保养指南，包括常见维护项目及维修或升级硬件的部分，请您认真阅读。

1 常见维护项目及推荐周期

| 维护项目 | 描述 | 推荐周期 |
|----------|--------------------------------|--------|
| 视觉检查 | 检查机器人有无明显的损坏、异物堆积或磨损。 | 日常 |
| 结构清洁 | 使用干净、干燥的布料清洁机器人结构部件，避免水分和侵蚀。 | 日常 |
| 紧固件检查 | 检查并紧固所有螺栓和连接件。 | 日常 |
| 电缆和接线检查 | 检查电缆和接线，确保无损坏或磨损。 | 每月 |
| 软件更新 | 检查并更新控制软件和应用程序。 | 每次有更新时 |
| 软件数据备份 | 定期备份关键软件配置和数据。 | 每月 |
| 固件更新 | 定期检查并更新固件，以获取最新的功能和安全补丁。 | 每次有更新时 |
| 紧急停止功能测试 | 定期测试紧急停止功能，确保其可靠性。 | 日常 |
| 安全配置复查 | 定期检查和确认机器人的安全配置，如限速和工作范围设定。 | 每月 |
| 环境条件监控 | 监控工作环境的温度、湿度、灰尘等，确保符合机器人的操作规格。 | 持续监控 |

2 独立更改机器人硬件的指南

我们理解客户可能会有自行升级或维修机器人硬件的需求。在进行任何升级操作之前，请务必详细阅读产品的相关参数，并与我们的官方人员确认是否被允许进行此类操作。未经官方允许的操作可能导致产品故障，且不在保修范围内。

物料要求 官方制造或推荐的物料：所有维修和升级所需的配件和组件必须是由我们官方制造或明确推荐的。这包括但不限于电子组件、传感器、电机、连接件和任何其他可更换部件。
物料获取：客户可通过我们的官方渠道购买所需的维修和升级物料。这确保了配件的质量和兼容性。

维修或升级流程 **客户自行维修**：客户应负责完成维修工作。我们将提供详细的维修指导和手册，以指导客户完成维修步骤。
遵循官方指导：维修操作应严格遵循我们提供的官方指导。任何偏离官方指导的操作都可能导致设备损坏。

3 责任和保修政策

责任划分

- **制造商:** 提供维修和升级的官方指导、官方制造或推荐的物料，并处理由制造缺陷导致的问题
- **客户:** 负责按照官方指导完成维修，使用官方配件。
- **保修政策:**
 - **保修有效:** 只有当维修操作完全遵循我们的指导，且使用官方配件时，保修才有效。
 - **保修无效:** 若客户未按官方指导操作，或使用非官方配件进行维修或升级，导致的任何损坏都将不在保修范围内。

4 注意事项

- **安全第一:** 在进行任何维修或升级操作前，请确保遵循所有安全指南，包括断电和使用适当的防护装备。
- **技术支持:** 如在维修过程中遇到问题，建议停止操作并联系我们的技术支持团队寻求帮助。

我们强烈建议客户严格遵循这些指南，以确保机器人设备的安全、有效运行。不当的维修操作可能导致设备损坏并影响保修状态。如需进一步的指导或支持，请及时联系我们的专业技术团队。

[← 上一页](#) | [下一页 →](#)

常见问题解决

章节内容

[← 上一页](#) | [下一页 →](#)

首次安装和使用



- 感谢您选择我们的产品

在开始之前，我们衷心感谢您选择我们的产品。我们致力于为您提供卓越的用户体验。

- 首次使用及问题处理

本章将详细介绍收到产品后的初次使用情况，并提供解决问题的相关信息，确保您在使用过程中无顾虑之忧。

- 跳转到每个章节
 - [4.1 产品清单](#)
 - [4.2 产品开箱指南](#)
 - [4.3 开机检测](#)

[← 上一页](#) | [下一页 →](#)

产品清单

1 产品清单图



每个产品都有编号和详细信息，以确保您可以准确地参考您的列表。

2 产品标准清单对照表

| 序号 | 产品 | 数量 |
|----|--------------------|----|
| 1 | Mycobot Pro630 机械臂 | 1 |
| 2 | 48V电源适配器 | 1 |
| 3 | 急停开关 | 1 |
| 4 | 安装底板 | 1 |
| 5 | G型夹 | 2 |
| 6 | HDMI线 | 1 |
| 7 | M8航插线 | 1 |
| 8 | 背头M6螺丝 | 4 |
| 9 | 背头M5螺丝 | 5 |
| 10 | M5扳手 | 1 |
| 11 | M4扳手 | 1 |
| 12 | M2扳手 | 1 |
| 13 | 8Pin插拔式接线端子 | 2 |
| 14 | 产品画册 | 1 |

注意：包装盒就位后，请确认机器人包装完好。如有损坏，请及时联系您所在地区的物流公司和供应商。开箱后，请根据物品清单清点箱内实际物品。

← 上一页 | 下一页 →

产品开箱指南

1 产品开箱图文引导

在本节中，我们强烈建议您按照指定的步骤删除该产品。这不仅有助于确保产品在运输过程中不会损坏，还可以最大限度地降低意外故障的风险。请仔细阅读以下图文指南，以确保您的产品在开箱过程中安全。

- 开箱前，检查盒子是否有损坏。如有损坏，请及时联系您所在地区的物流公司和供应商。
- 打开包装盒，取出产品说明书、海绵包装盖、myCobot机械臂、配套电源、急停开关、平板底座、配件包。
- 确保每个步骤都已完成，然后再进行下一步，以防止不必要的损坏或遗漏。

注意：取出产品后，请仔细检查每件物品的外观。请根据物品清单核对箱内的实际物品。

2 产品开箱视频指导



[← 上一页](#) | [下一页 →](#)

开机检测指南

1
2

11

11

11

11

11

ROS

ROS是用于机器人的开源的元操作系统。它提供了操作系统应有的服务，包括硬件抽象、低级设备控制、常用功能的实现、进程之间的消息传递以及包管理。它也提供用于获取、编译、编写、和跨计算机运行代码所需的工具和库函数。

ROS 运行时的“graph”是一种基于ROS通信基础结构的松耦合点对点进程网络。

ROS实现了几种不同的通信方式，包括基于同步RPC样式通信的服务（**services**）机制，基于异步流媒体数据的话题（**topics**）机制以及用于数据存储的参数服务器。

ROS并不是一个实时的框架，但ROS可以嵌入实时程序。Willow Garage的PR2机器人使用了一种叫做 pr2_etherCAT 的系统来实时发送或接收 ROS 消息。ROS还可以与Orococos 实时工具包无缝集成。

ROS 图标：



1 ros 的设计目标和特点

很多人都在问“ROS与其它机器人软件平台有什么不同？”这是一个很难解答的问题。因为ROS不是一个集成了大多数功能或特征的框架。事实上，ROS的主要目标是为机器人研究和开发提供代码复用的支持。ROS是一个分布式的进程（也就是节点）框架，这些进程被封装在易于被分享和发布的程序包和功能包集中。ROS也支持一种类似于代码储存库的联合系统，这个系统也可以实现工程的协作及发布。这个设计可以使一个工程的开发和实现从文件系统到用户接口完全独立决策（不受ROS限制）。同时，所有的工程都可以被ROS的基础工具整合在一起。

为了支持共享和协作这一主要目标，ROS 框架还有其他几个特点：

- 精简：ROS尽可能设计的精简，以便为ROS编写的代码可以与其他机器人软件框架一起使用。由此得出的必然结论是ROS可以轻松集成在其它机器人软件平台：ROS已经可以与OpenRAVE，Orococos和Player集成。
- ROS不敏感库：ROS的首选开发模型都是用不依赖ROS的干净的库函数编写而成。
- 语言独立：ROS框架可以简单地使用任何的现代编程语言实现。ros已经实现了Python版本，C++版本和Lisp版本。同时也拥有Java 和 Lua版本的实验库。
- 松耦合：ROS中功能模块封装于独立的功能包或元功能包，便于分享，功能包内的模块以节点为单位运行，以ROS标准的IO作为接口，开发者不需要关注模块内部实现，只要了解接口规则就能实现复用，实现了模块间点对点的松耦合连接
- 方便测试：ROS内建一个叫做rostest的单元/集成测试框架，可以轻松安装或卸载测试模块。
- 可扩展：ROS可以适用于大型运行时系统和大型开发进程。
- 免费且开源：开发者众多，功能包多

2 为什么使用ROS

通过ROS，我们能够在虚拟环境中实现对机械臂的仿真控制。

我们将通过 **rviz** 平台实现对机械臂的可视化，并使用多种方式对我们的机械臂进行操作；通过 **moveit** 平台进行机械臂行动路径的规划和执行，达到自由控制机械臂的效果。

我们将在接下来的章节中学习如何通过ros中的平台对我们产品的控制进行控制。

MoveIt

MoveIt 是目前针对机械臂移动操作的最先进的软件，已在 100 多个机器人上使用。它综合了运动规划、控制、3D 感知、运控学、控制和导航的最新成果，提供了开发先进机器人应用的易用平台，为工业、商业和研发等领域的机器人新产品的设计和集成体用评估提供了一个集成化软件平台。

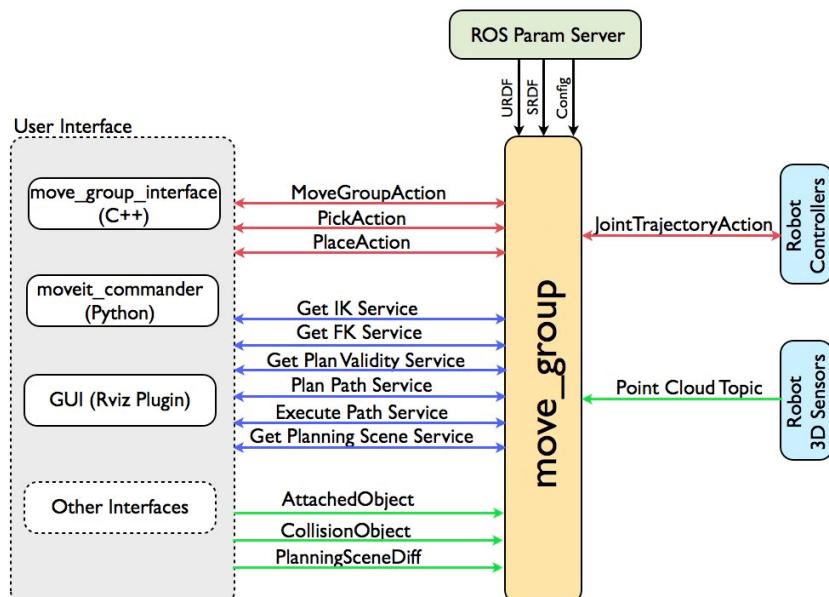
MoveIt 图标：



1 简介

MoveIt 是ROS中的一个集成开发平台，由多种用于操纵机械臂的功能包组成，包括：运动规划、操作、控制、逆运动学、3D感知和碰撞检测等。

下图所示为 Moveit 提供的主要节点 **move_group** 的高层结构，它像一个组合器：把所有单独的组件集成在一起，提供一系列的 **actions** 和 **services** 供用户使用。



2 用户界面

用户可以通过三种方式访问move_group提供的操作和服务：

- 在C++： 使用move_group_interface包可以方便实用move_group。
- 在 Python： 使用moveit_commander包。
- 通过 GUI： 使用 Motion——commander 的 Rviz（ROS可视化工具）。

move_group可以使用ROS参数服务器进行配置，从中还可以获取机器人的URDF和SRDF。

3 配置

move_group是一个 ROS 节点。它使用ROS参数服务器来获取三种信息：

1. URDF - move_group在ROS参数服务器上查找robot_description参数，以获取机器人的URDF。
2. SRDF - move_group在 ROS 参数服务器上查找robot_description_semantic参数，以获取机器人的 SRDF。SRDF 通常由用户使用 MoveIt 设置助理创建。
3. MoveIt 配置 - move_group将在 ROS 参数服务器上查找特定于 MoveIt 的其他配置，包括关节限制、运动学、运动规划、感知和其他信息。这些组件的配置文件由MoveIt设置助手自动生成，并存储在机器人的相应MoveIt配置包的配置目录中。配置助手的使用请参考：[MoveIt Setup Assistant](#)

← 上一节 | 下一页 →

在Linux中安装不同版本的ubuntu系统

1 虚拟机安装

前往[官方网站](#)下载虚拟机Virtual Box 或者前往[官方网站](#)下载虚拟机 VM ware

VirtualBox 安装包: [Windows hosts](#)

VirtualBox 拓展包: [VirtualBox 7.0.10 Oracle VM VirtualBox Extension Pack](#)

当然, 如果您已经拥有您的虚拟机, 您可以跳过该步骤。

我们选择下载Virtual box, 因为它是免费的。

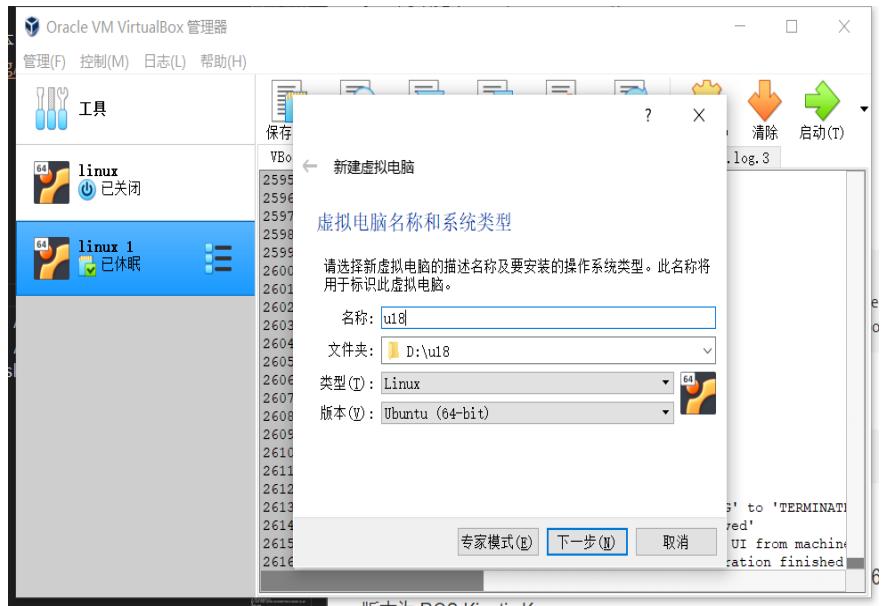


2 新建虚拟机

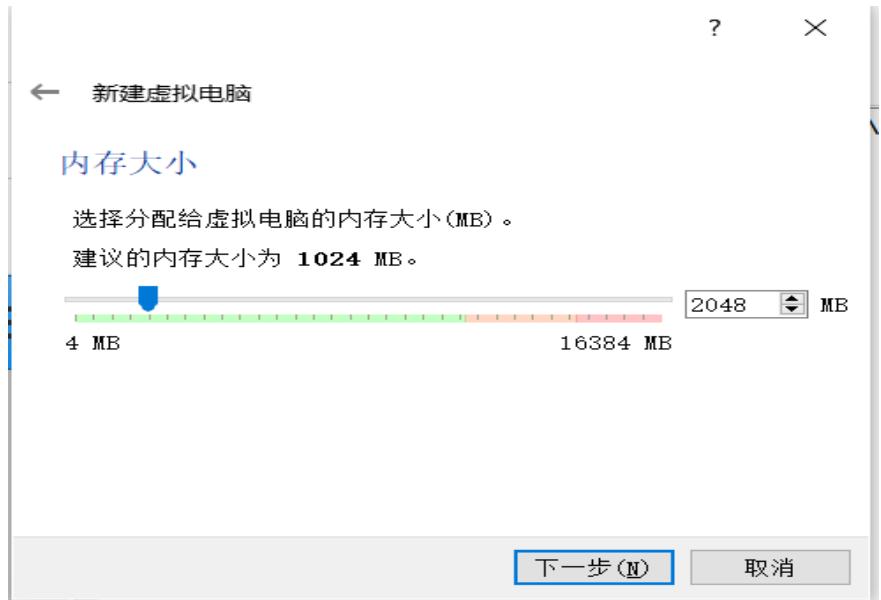
2.1 创建虚拟机

在控制中选择新建

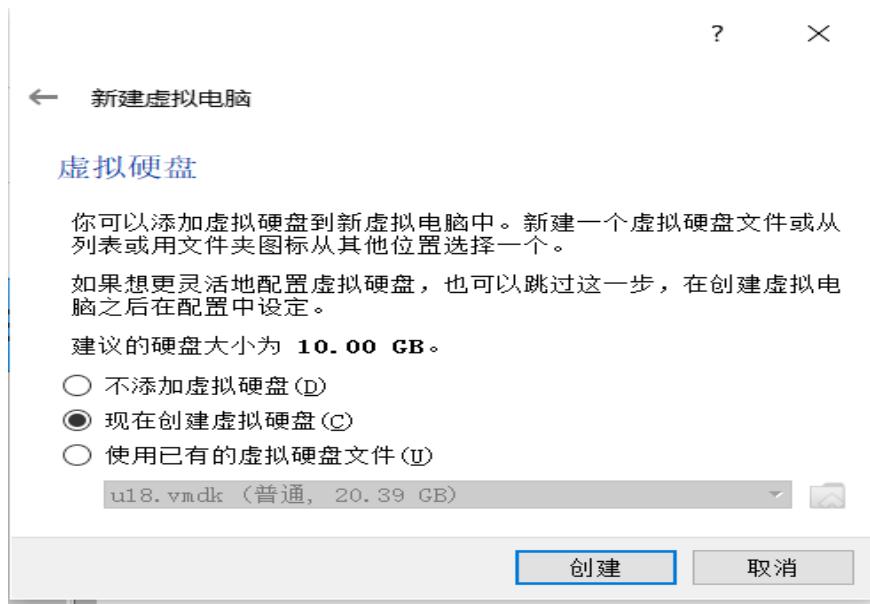
输入虚拟机名称和虚拟机存放的位置, 选择虚拟机类型为**Linux**, 选择ubuntu64位版本, 进行下一步。



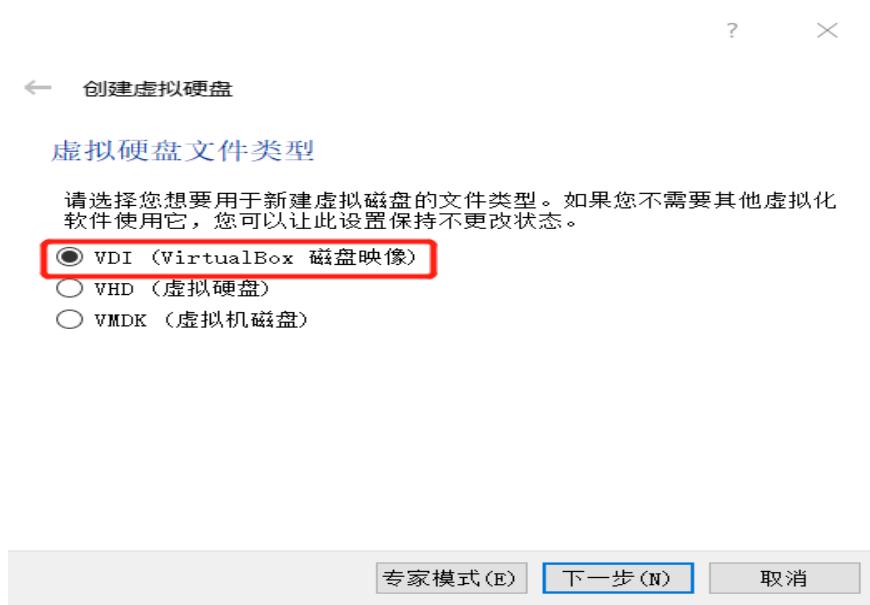
按照自己的需求配置内存大小，进行下一步。



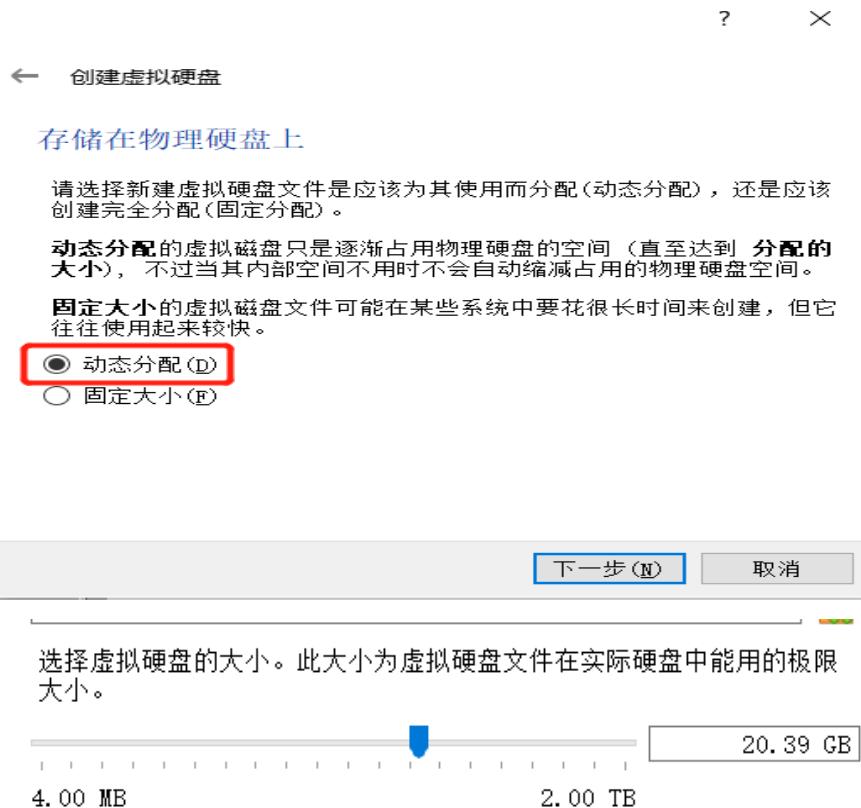
选择现在创建虚拟硬盘，进行创建。



虚拟硬盘类型选择**VDI**类型，进行下一步。



分配虚拟硬盘大小，由于需要安装ubuntu系统，而且还会在该系统中进行操作，建议大小不要低于20G。



2.2 导入ubuntu系统

2.2.1 下载ubuntu系统

请根据自己的需要选择ubuntu版本进行安装，推荐安装20.04版本。

- [16.04版本](#)
- [18.04版本](#)
- [20.04版本](#)

三种版本的安装方法和过程都是相同的，这里以**18.04**版本作为例子进行安装

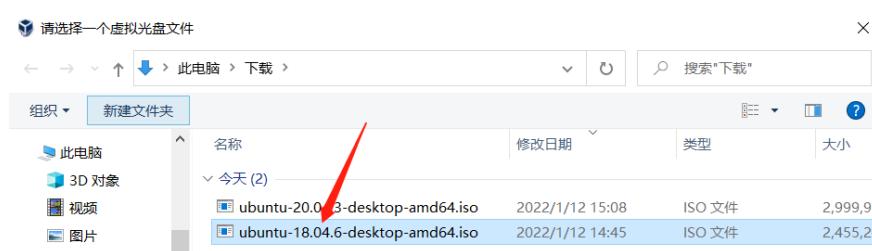
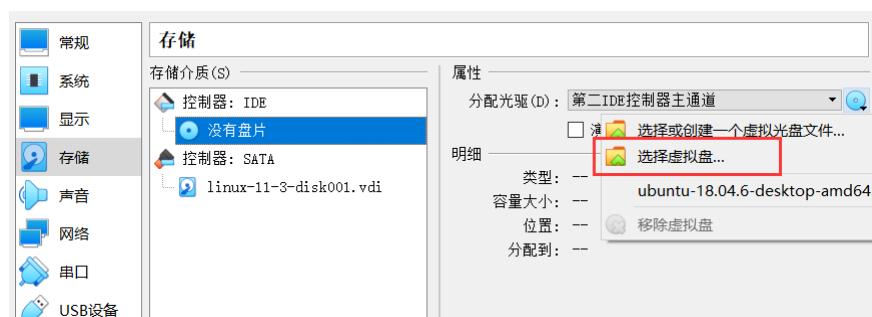
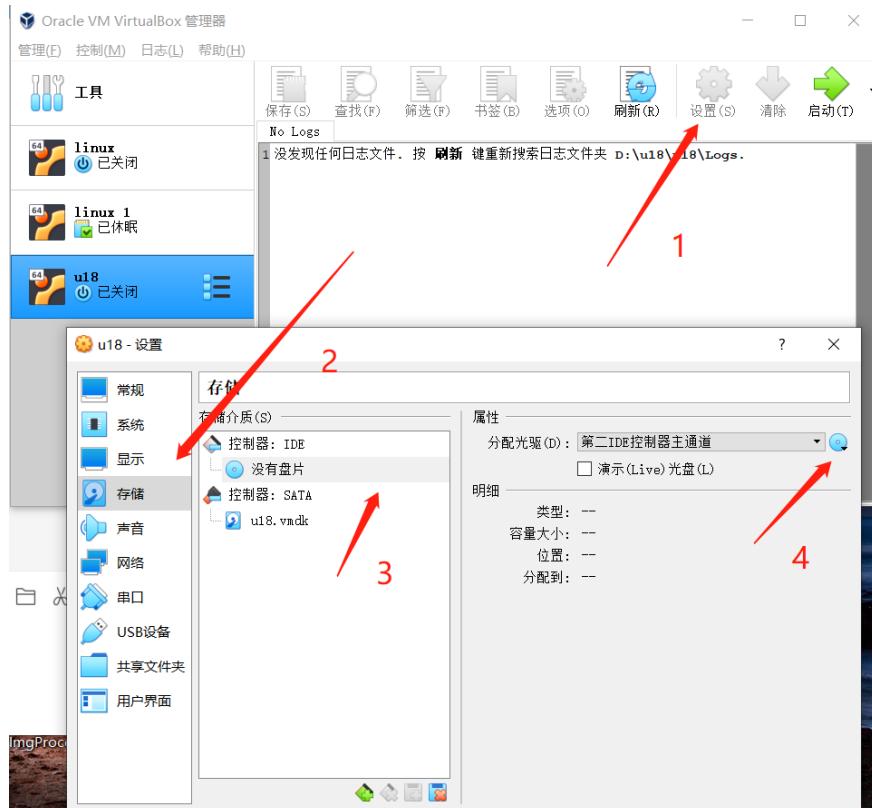
| If you need help burning these images to disk, see the Image Burning Guide. | | | | |
|---|------------------|------|---|--|
| Name | Last modified | Size | Description | |
| Parent Directory | | - | | |
| MD5SUMS-metalink | 2020-02-12 13:42 | 296 | | |
| MD5SUMS-metalink.gpg | 2020-02-12 13:42 | 916 | | |
| SHA256SUMS | 2021-09-16 21:58 | 202 | | |
| SHA256SUMS.gpg | 2021-09-16 21:58 | 833 | | |
| ubuntu-18.04.6-desktop-amd64.iso | 2021-09-15 20:42 | 2.3G | Desktop image for 64-bit PC (AMD64) computers (standard download) | |
| ubuntu-18.04.6-desktop-amd64.iso.torrent | 2021-09-16 21:46 | 188K | Desktop image for 64-bit PC (AMD64) computers (BitTorrent download) | |
| ubuntu-18.04.6-desktop-amd64.iso.zsync | 2021-09-16 21:46 | 4.7M | Desktop image for 64-bit PC (AMD64) computers (zsync metafile) | |
| ubuntu-18.04.6-desktop-amd64.list | 2021-09-15 20:42 | 7.8K | Desktop Image for 64-bit PC (AMD64) computers (file listing) | |
| ubuntu-18.04.6-desktop-amd64.manifest | 2021-09-15 20:36 | 59K | Desktop Image for 64-bit PC (AMD64) computers (contents of live filesystem) | |
| ubuntu-18.04.6-live-server-amd64.iso | 2021-09-15 20:42 | 969M | Server Install Image for 64-bit PC (AMD64) computers (standard download) | |

下载完成后有如图文件：

| 名称 | 修改日期 | 类型 | 大小 |
|----------------------------------|-----------------|--------|----------|
| 今天 (2) | | | |
| ubuntu-20.04.3-desktop-amd64.iso | 2022/1/12 15:08 | ISO 文件 | 2,999,93 |
| ubuntu-18.04.6-desktop-amd64.iso | 2022/1/12 14:45 | ISO 文件 | 2,455,20 |

2.2.2 导入ubuntu到虚拟机中

在Virtual box中找到之前安装的虚拟机，进入设置，并在存储中给控制器分配光盘：



然后打开虚拟机进行ubuntu安装，并点击启动。

2.2.3 ubuntu安装

等待系统启动，进入欢迎界面，选中“中文（简体）”，并点击“安装 Ubuntu”按钮；



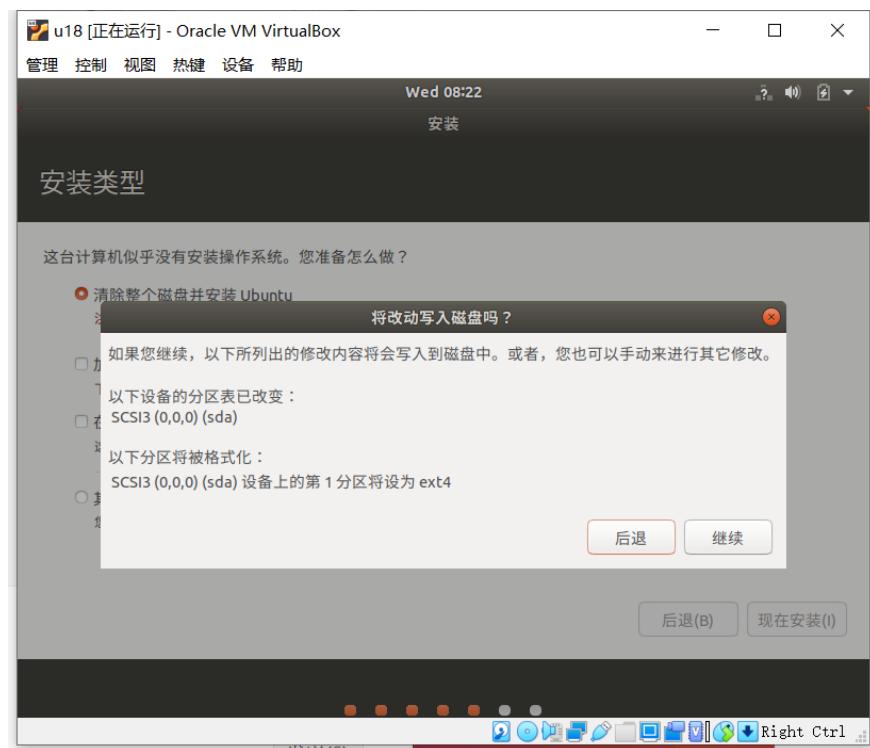
点击“继续”按钮；



选中“清除整个磁盘并安装 Ubuntu”选项，点击“现在安装”按钮；



在弹出的对话框中点击“继续”按钮；



设置地理位置，点击“继续”按钮；



设置用户名和密码，点击“继续”按钮；



进入系统安装界面，请耐心等待；



待安装完成，在弹出的对话框中，点击“现在重启”按钮，完成安装。



3 ROS 环境搭建

3.1 ROS 安装

基本的开发环境搭建需要安装机器人操作系统 ROS、MoveIt 以及 git 版本管理器，以下分别介绍其安装方法及流程。

3.1.1 版本选择

ROS 跟 ubuntu 有一一对应的关系，不同版本的 ubuntu 对应不同版本的 ROS，参考网站见下：<http://wiki.ros.org/Distributions>

- 这里给出对应Ubuntu支持的 ROS 版本：
 - Ubuntu 16.04 / ROS Kinetic
 - Ubuntu 18.04 / ROS Melodic
 - Ubuntu 20.04 / ROS Noetic

请根据自己安装的Ubuntu版本进行对应ROS版本的安装

如果版本不同，下载将会失败.在这里我们选择的系统为 Ubuntu 18.04, 对应 ROS 版本为 ROS Melodic

NOTE: 目前我们不提供 windows 安装 ROS 的任何参考, 若有需要请参考
<https://www.ros.org/install/>

3.1.2 开始安装

1 添加源

Ubuntu 本身的软件源列表中没有 ROS 的软件源, 所以需要先将 **ROS** 软件源配置到软件列表仓库中, 才能下载 ROS。打开一个控制台终端(快捷键 **Ctrl+Alt+T**), 输入如下指令:

- 官方源:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

- 若下载速度缓慢, 推荐就近选择一个镜像源替换上面的命令。例如, Tsinghua University为:

```
sudo sh -c '. /etc/lsb-release && echo "deb http://mirrors.tuna.tsinghua.edu.cn/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

这里会要求输入用户密码, 输入安装 Ubuntu 时设置的用户密码即可。

2 设置秘钥

配置公网秘钥, 这一步是为了让系统确认我们的路径是安全的的, 这样下载文件才没有问题, 不然下载后会被立刻删掉:

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

执行结果显示如下:

```
u18@u18-VirtualBox:~$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
Executing: /tmp/apt-key-gpghome.kD1wi001j3/gpg.1.sh --keyserver hkp://keyserver.ubuntu.com:80 --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
gpg: 密钥 F42ED6FBAB17C654: 公钥“Open Robotics <info@osrfoundation.org>”已导入
gpg: 合计被处理的数量: 1
gpg:           已导入: 1
u18@u18-VirtualBox:~$
```

3 安装

在加入了新的软件源后, 需要更新软件源列表, 打开一个控制台终端(快捷键 **Ctrl+Alt+T**), 输入如下指令:

```
sudo apt-get update
```

执行安装 **ROS**, 打开一个控制台终端(快捷键**Ctrl+Alt+T**), 请按照自己的Ubuntu 版本选择输入以下指令:

```
# Ubuntu 16.04  
sudo apt install ros-kinetic-desktop-full
```

```
# Ubuntu 18.04  
sudo apt install ros-melodic-desktop-full
```

```
# Ubuntu 20.04  
sudo apt install ros-noetic-desktop-full
```

这里推荐安装完整的 ROS，防止库和依赖的缺失。

安装过程耗时比较长，需要耐心等待

- 若安装过程中，控制台终端出现如下错误信息，则需要更换/etc/apt/sources.list中的软件源列表。

```
无法修复缺失的软件包。  
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/flann/libflann1.  
9_1.9.1+dfsg-2_amd64.deb 连接失败 [IP: 91.189.91.39 80]  
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/flann/libflann-d  
ev_1.9.1+dfsg-2_amd64.deb 连接失败 [IP: 91.189.91.39 80]  
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/fltk1.3/libfltk-  
cairo1.3_1.3.4-6_amd64.deb 连接失败 [IP: 91.189.91.39 80]  
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/fltk1.3/libfltk-  
gl1.3_1.3.4-6_amd64.deb 连接失败 [IP: 91.189.91.39 80]  
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/fltk1.3/libfltk1  
.3-dev_1.3.4-6_amd64.deb 连接失败 [IP: 91.189.91.38 80]  
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/fyba/libfyba-dev  
_4.1.1-3_amd64.deb 连接失败 [IP: 91.189.91.38 80]  
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/fonts-dejavu/ttf  
-dejavu-core_2.37-1_all.deb 连接失败 [IP: 91.189.91.39 80]  
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/libc/libccd/libccd  
-dev_2.0-1_amd64.deb 连接失败 [IP: 91.189.91.39 80]  
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/freexl/libfreexl  
-dev_1.0.5-1_amd64.deb 连接失败 [IP: 91.189.91.38 80]  
E: 中止安装。 [■]  
u182@u182-VirtualBox:~$
```

- 打开一个控制台终端(快捷键Ctrl+Alt+T)，输入如下指令：

```
sudo gedit /etc/apt/sources.list
```

- 将sources.list中的官方软件源全部替换成下面的阿里云软件源：

Ubuntu 16.04版本：

```
deb http://mirrors.aliyun.com/ubuntu/ xenial main  
deb-src http://mirrors.aliyun.com/ubuntu/ xenial main  
  
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates main  
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-updates main  
  
deb http://mirrors.aliyun.com/ubuntu/ xenial universe  
deb-src http://mirrors.aliyun.com/ubuntu/ xenial universe  
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates universe  
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-updates universe  
  
deb http://mirrors.aliyun.com/ubuntu/ xenial-security main  
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-security main  
deb http://mirrors.aliyun.com/ubuntu/ xenial-security universe  
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-security universe
```

Ubuntu 18.04版本：

```
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multive
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe mul

deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiver
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe mult

deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multive
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe mul

deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiv
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe mu
```

Ubuntu 20.04版本：

```
deb http://mirrors.aliyun.com/ubuntu/ focal main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ focal main restricted universe multiverse

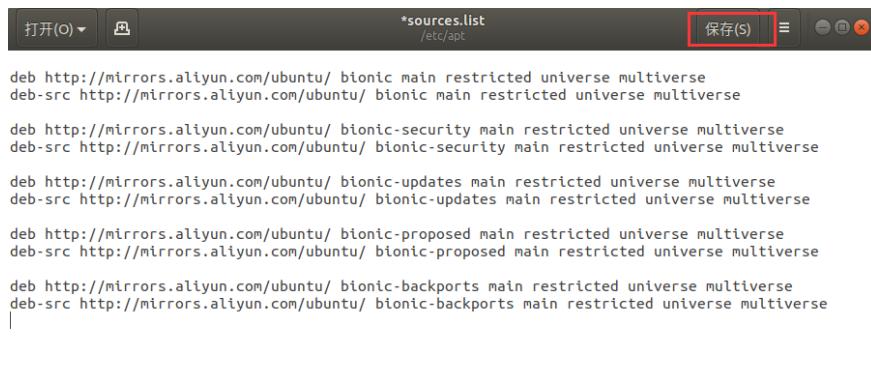
deb http://mirrors.aliyun.com/ubuntu/ focal-security main restricted universe multiver
deb-src http://mirrors.aliyun.com/ubuntu/ focal-security main restricted universe mult

deb http://mirrors.aliyun.com/ubuntu/ focal-updates main restricted universe multivers
deb-src http://mirrors.aliyun.com/ubuntu/ focal-updates main restricted universe multi

deb http://mirrors.aliyun.com/ubuntu/ focal-proposed main restricted universe multiver
deb-src http://mirrors.aliyun.com/ubuntu/ focal-proposed main restricted universe mult

deb http://mirrors.aliyun.com/ubuntu/ focal-backports main restricted universe multive
deb-src http://mirrors.aliyun.com/ubuntu/ focal-backports main restricted universe mul
```

- 配置完成后，sources.list文件内容如下所示，点击保存并退出。



```
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe mult

deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiver
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe mult

deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multive
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe mult

deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiv
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe mult
```

- 更新软件源列表，在控制台终端输入：

```
sudo apt-get update
```

- 在控制台终端输入安装ROS的指令：

```
# Ubuntu 16.04
sudo apt install ros-kinetic-desktop-full
```

```
# Ubuntu 18.04
sudo apt install ros-melodic-desktop-full
```

```
# Ubuntu 20.04
sudo apt install ros-noetic-desktop-full
```

安装过程耗时比较长，需要耐心等待

4 配置 ROS 环境到系统

rosdep 让你能够轻松地安装被想要编译的源代码，或被某些 ROS 核心组件需要的系统依赖，在终端依次执行以下命令，打开一个控制台终端(快捷键 Ctrl+Alt+T)。

如果您的系统没有安装rosdep,请使用命令 `sudo apt install python-rosdep` 进行安装。

如果您的安装的Ubuntu系统是20.04版本，请使用命令 `sudo apt install python3-rosdep` 进行安装，完成后执行rosdep初始化命令。

```
u18@u18-VirtualBox:~$ sudo rosdep init
sudo: rosdep: 找不到命令
u18@u18-VirtualBox:~$ sudo apt install python-rosdep
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
下列【新】软件包将被安装：
  python-rosdep
升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 1 个软件包未被升级。
。
需要下载 3,236 B 的归档。
解压缩后会消耗 21.5 kB 的额外空间。
获取:1 http://packages.ros.org/ros/ubuntu bionic/main amd64 python-rosdep all 0.
21.0-1 [3,236 B]
已下载 3,236 B, 耗时 1 秒 (4,120 B/s)
正在选中未选择的软件包 python-rosdep。
(正在读取数据库 ... 系统当前共安装有 247871 个文件和目录。)
正准备解包 .../python-rosdep_0.21.0-1_all.deb ...
正在解包 python-rosdep (0.21.0-1) ...
正在设置 python-rosdep (0.21.0-1) ...
u18@u18-VirtualBox:~$
```

初始化 rosdep:

```
sudo rosdep init
```

若出现如下图所示的错误提示：

```
u182@u182-VirtualBox:~$ sudo rosdep init
ERROR: cannot download default sources list from:
https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/sources.list.d/20-
default.list
Website may be down.
```

解决方法：修改hosts文件，控制台终端输入下面的指令：

```
sudo gedit /etc/hosts
```

在文件内容末端，加入以下两个网址的IP地址实现访问：

```
199.232.28.133 raw.githubusercontent.com
151.101.228.133 raw.github.com
```

```
hosts /etc  
127.0.0.1 localhost  
127.0.1.1 u182-VirtualBox  
  
# The following lines are desirable for IPv6 capable hosts  
::1 ip6-localhost ip6-loopback  
fe00::0 ip6-localnet  
ff00::0 ip6-mcastprefix  
ff02::1 ip6-allnodes  
ff02::2 ip6-allrouters  
199.232.28.133 raw.githubusercontent.com  
151.101.228.133 raw.github.com
```

修改完成后，在控制台终端执行：

```
sudo rosdep init
```

```
rosdep update
```

初始化完成后，为了避免每次关掉终端窗口后都需要重新生效 ROS 功能路径，我们可以把路径配置到环境变量中，这样在每次打开新的终端时便可自动生效
ROS 功能路径 在终端依次执行以下命令，打开一个控制台终端(快捷键
Ctrl+Alt+T)：

3.1.3 设置ros环境

1 Bash

执行以下命令：

```
# Ubuntu 16.04  
# 将 ros 环境加入到当前控制台的环境变量  
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 18.04  
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 20.04  
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

2 安装依赖项

在终端输入以下命令安装ROS额外依赖项，打开一个控制台终端(快捷键
Ctrl+Alt+T)：

```
sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

如果你的Unbutu系统是20.04版本，请执行以下命令安装：

```
sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
```

```
# Ubuntu 16.04  
sudo apt install ros-kinetic-joint-state-publisher-gui
```

```
# Ubuntu 18.04  
sudo apt install ros-melodic-joint-state-publisher-gui
```

```
# Ubuntu 20.04  
sudo apt install ros-noetic-joint-state-publisher-gui
```

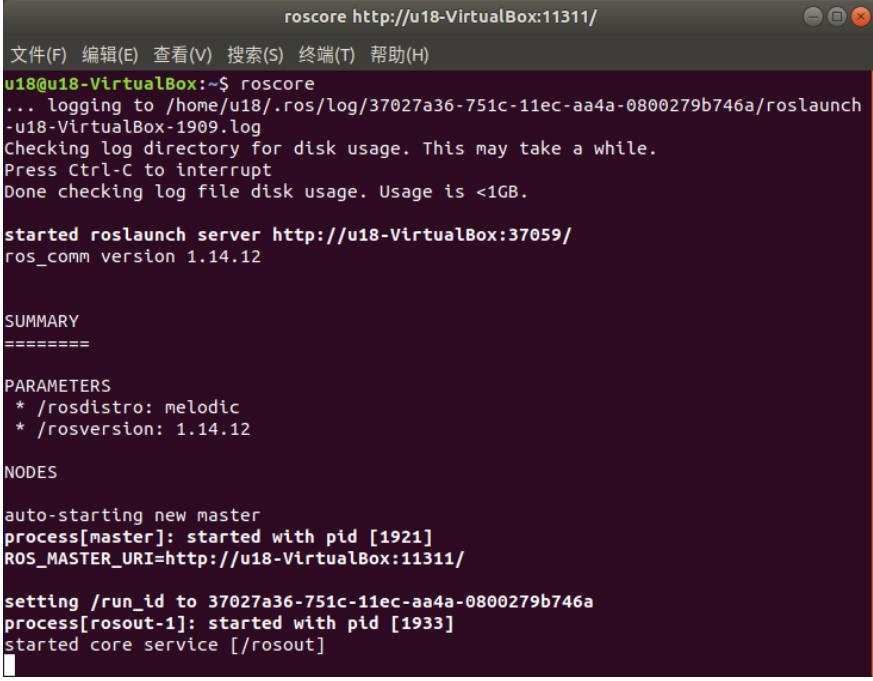
3.1.4 验证安装

ROS 系统的启动需要一个 **ROS Master**, 即节点管理器, 我们可以在终端输入 **roscore** 指令来启动 ROS Master。

为了验证 ROS 是否安装成功, 打开一个控制台终端(快捷键Ctrl+Alt+T), 在终端执行以下命令:

```
roscore
```

当显示如下界面, 则表示 ROS 安装成功



```
roscore http://u18-VirtualBox:11311/  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
u18@u18-VirtualBox:~$ roscore  
... logging to /home/u18/.ros/log/37027a36-751c-11ec-aa4a-0800279b746a/roslaunch  
-u18-VirtualBox-1909.log  
Checking log directory for disk usage. This may take a while.  
Press Ctrl-C to interrupt  
Done checking log file disk usage. Usage is <1GB.  
  
started roslaunch server http://u18-VirtualBox:37059/  
ros_comm version 1.14.12  
  
SUMMARY  
=====  
PARAMETERS  
* /rosdistro: melodic  
* /rosversion: 1.14.12  
NODES  
  
auto-starting new master  
process[master]: started with pid [1921]  
ROS_MASTER_URI=http://u18-VirtualBox:11311/  
  
setting /run_id to 37027a36-751c-11ec-aa4a-0800279b746a  
process[rosout-1]: started with pid [1933]  
started core service [/rosout]
```

roscore命令启动了一个节点管理器, 其作用就是用于节点管理, 在一个ros系统中, 有且只有一个, 它是ros节点运行的前提, 所以在执行启动ros节点前, 第一步都需要执行**roscore**。

更多更详细的安装指导, 可以参考官方的安装指导, 网址
<http://wiki.ros.org/ROS/Installation>

3.2 MoveIt 安装

MoveIt 是 ros 中一系列移动操作的功能包的组成，主要包含运动规划，碰撞检测，运动学，3D 感知，操作控制等功能。

1 更新软件源列表

打开一个控制台终端(快捷键Ctrl+Alt+T)，在终端窗口输入以下命令，以更新软件源列表：

```
sudo apt-get update
```

2 安装 MoveIt

打开一个控制台终端(快捷键Ctrl+Alt+T)，在终端窗口输入以下命令，执行 **MoveIt** 的安装：

```
# Ubuntu16.04  
sudo apt-get install ros-kinetic-moveit
```

```
# Ubuntu 18.04  
sudo apt-get install ros-melodic-moveit
```

```
# Ubuntu20.04  
sudo apt-get install ros-noetic-moveit
```

3.3 Git 安装

1 添加软件源

将 **git** 安装的软件源添加到 **ubuntu** 的软件源列表中，打开一个控制台终端(快捷键Ctrl+Alt+T)，在终端窗口输入以下命令：

```
sudo add-apt-repository ppa:git-core/ppa
```

2 更新软件源列表

打开一个控制台终端(快捷键Ctrl+Alt+T)，在终端窗口输入以下命令，以更新软件源列表：

```
sudo apt-get update
```

3 安装 git

打开一个控制台终端(快捷键Ctrl+Alt+T)，在终端窗口输入以下命令，执行 **git** 的安装：

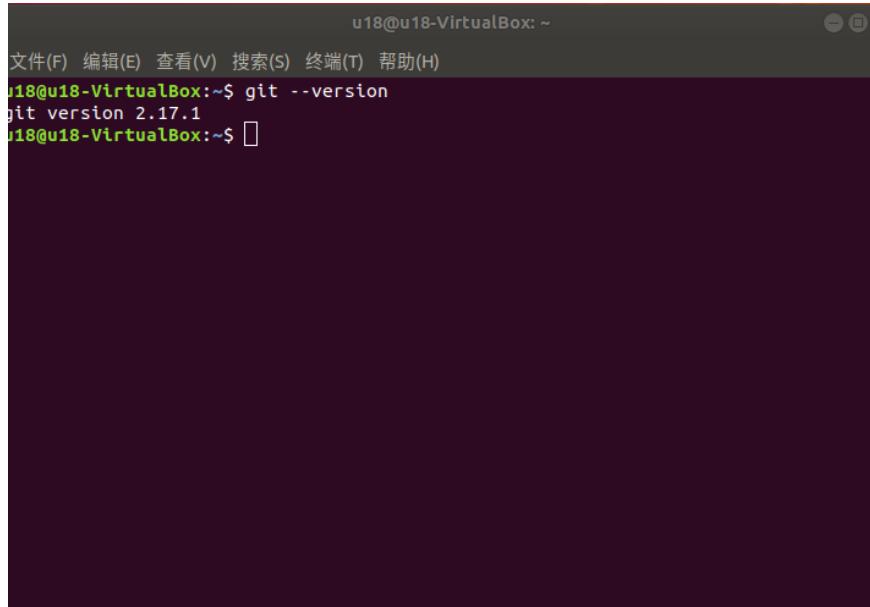
```
sudo apt-get install git
```

4 验证安装

读取 **git** 版本，打开一个控制台终端(快捷键Ctrl+Alt+T)，在终端窗口输入以下命令：

```
git --version
```

在终端中可以显示 git 版本号，如下，即为安装成功



The screenshot shows a terminal window titled 'u18@u18-VirtualBox: ~'. The window contains the following text:
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
u18@u18-VirtualBox:~\$ git --version
git version 2.17.1
u18@u18-VirtualBox:~\$

5 使用

在后续下载 **ros** 包需要用到git，git 的使用可以参考下面链接：

- <https://git-scm.com/book/zh/v2>
- <https://www.runoob.com/git/git-tutorial.html>

4.4 mycobot_ros 安装

mycobot_ros 是 ElephantRobotics 推出的，适配旗下桌面型六轴机械臂 mycobot 系列的ROS 包。

项目地址：http://github.com/elephantrobotics/mycobot_ros

1 前提

在安装包之前，请保证拥有 **ros** 工作空间。

这里我们给出创建工作空间的样例命令，默认为 **catkin_ws**，打开一个控制台终端(快捷键Ctrl+Alt+T)，在命令行输入以下命令：

```
mkdir -p ~/catkin_ws/src # 创建文件夹  
cd ~/catkin_ws/src # 进入文件夹  
catkin_init_workspace # 把当前目录初始化为一个ROS工作空间  
cd ~/catkin_ws # 返回工作空间  
catkin_make # 构建工作区中的代码。
```

添加工作空间的环境

官方默认的 ROS1 工作区是 `catkin_ws`。

```
# Ubuntu 16.04
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 18.04
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 20.04
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

2 安装

NOTE:

- 本包依赖于ROS和MoveIT，使用前确保已成功安装ROS和MoveIT。
- 本包与真实机械臂的交互依赖于PythonApi - `pymycobot`
- Api项目地为：<https://github.com/elephantrobotics/pymycobot>
- 快速安装：`pip install pymycobot --upgrade`

执行`pip install pymycobot --upgrade`命令时，若出现如下图错误提示：

```
u182@u182-VirtualBox:~$ pip install pymycobot --upgrade
Command 'pip' not found, but can be installed with:
sudo apt install python-pip
```

根据提示输入以下命令安装pip

```
sudo apt install python-pip
```

- 如果你的Ubuntu系统是20.04版本，请执行命令 `sudo apt install python3-pip` 安装pip pip安装完成后，终端再次执行

```
pip install pymycobot --upgrade
```

- 安装方式依赖于Git，请确保虚拟机系统中已安装Git。

官方默认的 ROS1 工作区是 `catkin_ws`。

```
cd ~/catkin_ws/src # 进入工作区的src文件夹中
# 克隆github上的代码
git clone https://github.com/elephantrobotics/mycobot_ros.git
cd ~/catkin_ws      # 返回工作区
catkin_make # 构建工作区中的代码
cd ..
source devel/setup.bash # 添加环境变量
```

导入Linux系统镜像

注意：为了降低环境搭建难度，我们将给出 Linux 系统镜像（Ubuntu 20.04）、Virtual Box 安装包以及其扩展包。接下来将教导大家如何安装 Virtual Box 以及导入 Linux 系统镜像（默认密码为 123）。已内置环境：ROS1 + Moveit + Git + pymycobot + mycobot_ros

1 安装虚拟机

前往[官方网站](#)下载虚拟机 Virtual Box 或者前往[官方网站](#)下载虚拟机 VM ware

VirtualBox 安装包：[Windows hosts](#)

VirtualBox 拓展包：[VirtualBox 7.0.10 Oracle VM VirtualBox Extension Pack](#)

当然，如果您已经拥有您的虚拟机，您可以跳过该步骤。

我们选择下载 Virtual box，因为它是免费的。



2 下载 Linux 系统镜像

点击下载：[Linux ubuntu20.04](#)

3 导入Linux系统镜像

在Virtual Box界面中点击 管理 -> 导入虚拟电脑 -> 选择虚拟镜像 -> 选择安装路径并进行导入，如下安装即可。





等待镜像导入即可，如下图即为安装成功。



然后启动系统即可， 默认密码为 **123**

3 更新 mycobot_ros 包

为了保证用户能及时使用最新的官方包，可以通过文件管理器进入 `~/catkin_ws/src` 文件夹，打开一个控制台终端(快捷键Ctrl+Alt+T)，在命令行输入命令进行更新：

```
# 克隆github上的代码  
cd ~/catkin_ws/src  
git clone https://github.com/elephantrobotics/mycobot_ros.git # 在决定是否执行此命令之前  
cd ~/catkin_ws      # 回到工作区  
catkin_make # 在工作区中构建代码  
source devel/setup.bash # 添加环境变量
```

注意：如果在 `~/catkin_ws/src` 目录下已经存在 `mycobot_ros` 文件夹，则需要先删除原来的 `mycobot_ros`，然后再执行上述命令。

至此ROS1环境搭建完成，您可以学习 [ROS的基础知识](#) 或者 [ROS使用案例](#)。

← 上一页 | 下一页 →

1 ROS工程结构

1.1 catkin工作空间

Catkin工作空间是创建、修改、编译catkin软件包的目录。catkin的工作空间，直观的形容就是一个仓库，里面装载着ROS的各种项目工程，便于系统组织管理调用。

- 创建工作空间：

```
mkdir -p ~/catkin_ws/src      # 创建文件夹  
cd ~/catkin_ws/src            # 进入文件夹  
catkin_init_workspace         # 把当前目录初始化为一个ROS工作空间  
cd ..                          # 返回上级目录  
catkin_make                    # 构建工作区中的代码。
```

catkin的结构十分清晰，它包括了src、build、devel三个路径，在有些编译选项下也可能包括其他。但这三个文件夹是catkin编译系统默认的。它们的具体作用如下：

```
src/: ROS的catkin软件包（源代码包）  
  
build/: catkin (CMake) 的缓存信息和中间文件  
  
devel/: 生成的目标文件（包括头文件，动态链接库，静态链接库，可执行文件等）、环境变量
```

一个简单的工作空间如下所示：

```
workspace_folder/          -- WORKSPACE  
src/                      -- SOURCE SPACE  
  CMakeLists.txt           -- 'Toplevel' CMake file, provided by catkin  
  package_1/  
    CMakeLists.txt        -- CMakeLists.txt file for package_1  
    package.xml            -- Package manifest for package_1  
    ...  
  package_n/  
    CMakeLists.txt        -- CMakeLists.txt file for package_n  
    package.xml            -- Package manifest for package_n
```

1.2 ROS软件包

Package不仅是Linux上的软件包，也是catkin编译的基本单元，我们使用catkin_make编译的对象就是每个ROS的package。

```
+-- PACKAGE
    +-- CMakeLists.txt
    +-- package.xml
    +-- src/
    +-- include/
    +-- scripts/
    +-- msg/
    +-- srv/
    +-- urdf/
    +-- launch/
```

- **CMakeLists.txt:** 定义package的包名、依赖、源文件、目标文件等编译规则，是package不可少的成分
- **package.xml:** 描述package的包名、版本号、作者、依赖等信息，是package不可少的成分
- **src/:** 存放ROS的源代码，包括C++的源码和(.cpp)以及Python的module(.py)
- **include/:** 存放C++源码对应的头文件
- **scripts/:** 存放可执行脚本，例如shell脚本(.sh)、Python脚本(.py)
- **msg/:** 存放自定义格式的消息(.msg)
- **srv/:** 存放自定义格式的服务(.srv)
- **urdf/:** 存放机器人的模型描述(.urdf或.xacro)、3D模型文件(.sda, .stl, .dae等)
- **launch/:** 存放launch文件(.launch或.xml)

创建自己的软件包：

- 指令格式：

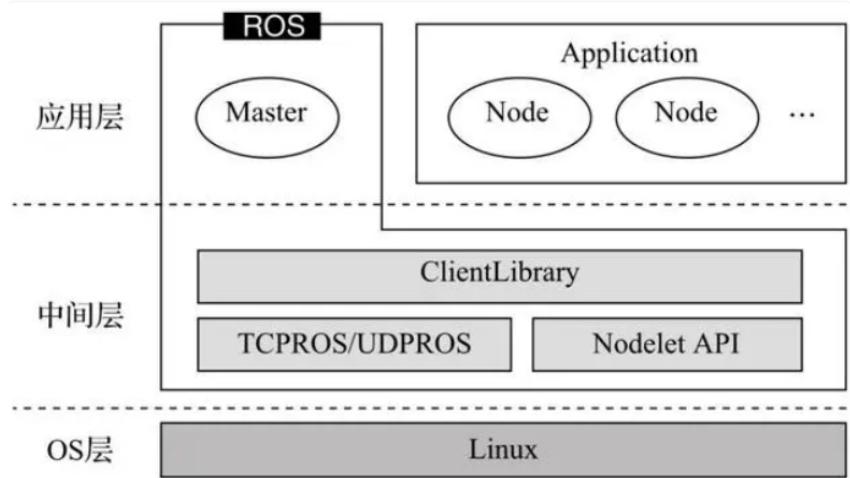
catkin_create_pkg命令会要求你输入package_name，如有需要还可以在后面添加一些需要依赖的其它软件包：

```
catkin_create_pkg <package_name> [depend1] [depend2] [depend3]
```

- 例如：

```
catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
```

2 ROS通信架构



2.1 Master与node

1 Master

节点管理器，每个node启动前要向master注册，管理node之间的通信。

2 roscore

启动master，也会启动rosout（日志管理）和parameter server（参数管理器）

3 node

ROS的进程、pkg里可执行文件运行的实例。

```
$rosrun [pkg_name] [node_name] #启动
$rosnode list #列出当前运行的node信息
$rosnode info [node_name] #显示某个node的详细信息
$rosnode kill [node_name] #结束某个node
```

4 launch

启动master和多个node。

```
$roslaunch [pkg_name] [file_name.launch]
```

2.2 Service 与 Topic

我们提供一些 service 和 topic，用以和 mycobot 交互。

1 Service

命令行中输入：

```
source ~/catkin_ws/devel/setup.bash # 添加环境变量  
roslaunch mycobot_320_communication communication_service.launch
```

支持参数:

- port: 连接串口字符串
- baud: 波特率

打开新的命令行:

```
# 显示活动的服务信息  
rosservice list  
  
#/get_joint_angles  
#/get_joint_coords  
#/set_joint_angles  
#/set_joint_coords  
#/switch_gripper_status  
#/switch_pump_status
```

相关的命令与说明:

| 命令 | 详细说明 |
|-----------------------------|----------------|
| rosservice list | 显示活动的服务信息 |
| rosservice info [服务名称] | 显示指定服务的信息 |
| rosservice type [服务名称] | 显示服务类型 |
| rosservice find [服务类型] | 查找指定服务类型的服务 |
| rosservice uri [服务名称] | 显示ROSRPC URI服务 |
| rosservice args [服务名称] | 显示服务参数 |
| rosservice call [服务名称] [参数] | 用输入的参数请求服务 |

2 Topic

命令行中输入:

```
source ~/catkin_ws/devel/setup.bash  
roslaunch mycobot_320_communication communication_topic.launch
```

支持参数:

- port: 连接串口字符串
- baud: 波特率

打开新的命令行:

```
# 显示活动的服务信息
rostopic list

#/mycobot/angles_goal
#/mycobot/coords_goal
#/mycobot/angles_real
#/mycobot/coords_real
#/mycobot/pump_status
#/mycobot/gripper_status
```

相关的命令与说明：

| 命令 | 详细说明 |
|--|----------------------------|
| <code>rostopic list</code> | 显示活动的话题目录 |
| <code>rostopic echo [话题名称]</code> | 实时显示指定话题的消息内容 |
| <code>rostopic find [类型名称]</code> | 显示使用指定类型的消息的话题 |
| <code>rostopic type [话题名称]</code> | 显示指定话题的消息类型 |
| <code>rostopic bw [话题名称]</code> | 显示指定话题的消息带宽 (bandwidth) |
| <code>rostopic hz [话题名称]</code> | 显示指定话题的消息数据发布周期 |
| <code>rostopic info [话题名称]</code> | 显示指定话题的信息 |
| <code>rostopic pub [话题名称] [消息类型] [参数]</code> | 用指定的话题名称发布消息 |

service与**topic**的区别：

| | service | topic |
|------|----------------|---------------|
| 同步性 | 异步 | 同步 |
| 通信模型 | 发布/订阅 | 服务器/客户端 |
| 底层协议 | ROSTCP/ROSUDP | ROSTCP/ROSUDP |
| 反馈机制 | 无 | 有 |
| 缓冲区 | 有 | 无 |
| 实时性 | 弱 | 强 |
| 节点关系 | 多对多 | 一对多 |
| 适用场景 | 数据传输 | 逻辑处理 |

您可以前往[service](#)和[topic](#)深入了解这两项功能的使用

2.3 msg和srv简介

- msg:** msg文件是描述ROS消息字段的简单文本文件。它们用于为不同语言（c++或者python等）的消息生成源代码。
- srv:** srv文件用来描述服务。它由两部分组成：请求（request）和响应（response）。msg文件存储在包的msg目录中，而srv文件存储在srv目录

中。

1 rosmsg

rosmsg是用于显示有关ROS消息类型的信息的命令行工具。

rosmsg 演示：

```
rosmsg show      # 显示消息描述
rosmsg info      # 显示消息信息
rosmsg list       # 列出所有消息
rosmsg md5        # 显示 md5 加密后的消息
rosmsg package    # 显示某个功能包下的所有消息
rosmsg packages   # 列出包含消息的功能包
```

- rosmsg list 会列出当前 ROS 中的所有 msg
- rosmsg packages 列出包含消息的所有包
- rosmsg package 列出某个包下的所有msg

```
//rosmsg package # 包名
rosmsg package turtlesim
```

- rosmsg show 显示消息描述

```
//rosmsg show # 消息名称
rosmsg show turtlesim/Pose
# 结果:
float32 x
float32 y
float32 theta
float32 linear_velocity
float32 angular_velocity
```

- rosmsg info 作用与 rosmsg show 一样
- rosmsg md5 一种校验算法，保证数据传输的一致性

2 rossrv

rossrv是用于显示有关ROS服务类型的信息的命令行工具，与 rosmsg 使用语法高度雷同。

```
rossrv show      # 显示服务消息详情
rossrv info       # 显示服务消息相关信息
rossrv list        # 列出所有服务信息
rossrv md5         # 显示 md5 加密后的服务消息
rossrv package     # 显示某个包下所有服务消息
rossrv packages    # 显示包含服务消息的所有包
```

- rossrv list 会列出当前 ROS 中的所有 srv 消息
- rossrv packages 列出包含服务消息的所有包
- rossrv package 列出某个包下的所有msg

```
//rossrv package # 包名  
rossrv package turtlesim
```

- **rossrv show** 显示消息描述

```
//rossrv show # 消息名称  
rossrv show turtlesim/Spawn  
# 结果:  
float32 x  
float32 y  
float32 theta  
string name  
---  
string name
```

- **rossrv info** 作用与 **rossrv show** 一致
- **rossrv md5** 对 service 数据使用 **md5** 校验(加密)

3 URDF介绍

- Unified Robot Description Format, 统一机器人描述格式, 简称为URDF。
ROS中的urdf功能包包含一个URDF的C++解析器, URDF文件使用XML格式
描述机器人模型。
- URDF 不能单独使用, 需要结合 Rviz 或 Gazebo, URDF 只是一个文件, 需
要在 Rviz 或 Gazebo 中渲染成图形化的机器人模型。

3.1 urdf文件描述

代码示例：

本处只截取部分代码进行展示：

```

<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="firefighter" >

<xacro:property name="width" value=".2" />

<link name="base">
  <visual>
    <geometry>
      <mesh filename="package://mercury_description/urdf/mercury_a1/base.dae"/>
    </geometry>
    <origin xyz = "0.0 0 0" rpy = "0 0 0"/>
  </visual>
  <collision>
    <geometry>
      <mesh filename="package://mercury_description/urdf/mercury_a1/base.dae"/>
    </geometry>
    <origin xyz = "0.0 0 0" rpy = "0 0 0"/>
  </collision>
</link>

<link name="joint1">
  <visual>
    <geometry>
      <mesh filename="package://mercury_description/urdf/mercury_a1/joint1.dae"/>
    </geometry>
    <origin xyz = "0.0 0 -0.08" rpy = "0 0 0"/>
  </visual>
  <collision>
    <geometry>
      <mesh filename="package://mercury_description/urdf/mercury_a1/joint1.dae"/>
    </geometry>
    <origin xyz = "0.0 0 -0.08" rpy = "0 0 0"/>
  </collision>
</link>

<link name="joint2">
  <visual>
    <geometry>
      <mesh filename="package://mercury_description/urdf/mercury_a1/joint2.dae"/>
    </geometry>
    <origin xyz = "0.0 0 0.0" rpy = "3.14159 0 -1.5708"/>
  </visual>
  <collision>
    <geometry>
      <mesh filename="package://mercury_description/urdf/mercury_a1/joint2.dae"/>
    </geometry>
    <origin xyz = "0.0 0 0.0" rpy = "3.14159 0 -1.5708"/>
  </collision>
</link>

<joint name="joint1_to_base" type="revolute">
  <axis xyz="0 0 1"/>
  <limit effort = "1000.0" lower = "-3.1066" upper = "3.1066" velocity = "0"/>
  <parent link="base"/>
  <child link="joint1"/>
  <origin xyz= "0 0 0.175" rpy = "0 0 0"/>
</joint>

<joint name="joint2_to_joint1" type="revolute">
  <axis xyz="0 0 1"/>
  <limit effort = "1000.0" lower = "-1.4311" upper = "2.2689" velocity = "0"/>
  <parent link="joint1"/>
  <child link="joint2"/>
  <!-- <origin xyz= "0 0 0" rpy = "1.5708 3.14159 0"/> -->

```

```

<origin xyz= "0 0 0" rpy = "-1.5708 0 0"/>
</joint>

</robot>

```

可以看出，urdf文件并不复杂，主要是由 `link` 和 `joint` 两个部分不断重复而成。

3.2 link部分

`link`元素描述具有惯性、可视特征和碰撞属性的刚体

3.2.1 属性

name: 用来描述链接本身的名字

3.2.2 元素

- `<inertial>` (可选)
 - 连杆的惯性特性
 - `<origin>` (可选, defaults to identity if not specified)
 - 定义相对于连杆坐标系的惯性参考系的参考坐标, 该坐标必需定义在连杆重心处, 其坐标轴可与惯性主轴不平行。
 - `xyz` (可选, 默认为零向量) 表示 x, y, z , x, y, z 方向的偏置, 单位为米。
 - `rpy` (可选: defaults to identity if not specified) 表示坐标轴在RPY方向上的旋转, 单位为弧度。
 - `<mass>` 连杆的质量属性
 - `<inertia>` 3×3旋转惯性矩阵, 由六个独立的量组成: $ixx, ixy, ixz, iyy, iyz, izz$ 。
- `<visual>` (可选)
 - 连杆的可视化属性。用于指定连杆显示的形状 (矩形、圆柱体等), 同一连杆可以存在多个`visual`元素, 连杆的形状为多个元素两个形成。一般情况下模型较为复杂可以通过solidwork绘制后生成stl调用, 简单的形状如添加末端执行器等可以直接编写。同时可以在此处可根据理论模型和实际模型差距调整几何形状的位置。
 - `<name>` (可选) 连杆几何形状的名字。
 - `<origin>` (可选, defaults to identity if not specified)
 - 相对于连杆的坐标系的几何形状坐标系。
 - `xyz` (optional: defaults to zero vector) 表示 x, y, z , x, y, z 方向的偏置, 单位为米。
 - `rpy` (optional: defaults to identity if not specified) 表示坐标轴在RPY方向上的旋转, 单位为弧度。
- `<geometry>` (必需)
 - 可视化对象的形状, 可以是下面的其中一种:
 - `<box>` 矩形, 元素包含长、宽、高。原点在中心。
 - `<cylinder>` 圆柱体, 元素包含半径、长度。原点中心。
 - `<sphere>` 球体, 元素包含半径。原点在中心。

- <mesh> 网格，由文件决定，同时提供 **scale**，用于界定其边界。推荐使用 Collada .dae 文件，也支持.stl文件，但必须为一个本地文件。
- <**material**> (可选)
 - 可视化组件的材料。可以在**link**标签外定义，但必需在**robot**标签内，在**link**标签外定义时，需引用**link**的名字。
 - <**color**> (可选) 颜色，由 red/green/blue/alpha 组成，大小范围在 [0,1] 内。
 - <**texture**> (可选) 材料属性，由文件定义。
- <**collision**> (可选)
 - 连杆的碰撞属性。碰撞属性和连杆的可视化属性不同，简单的碰撞模型经常用来简化计算。同一个连杆可以有多个碰撞属性标签，连杆的碰撞属性表示由其定义的几何图形集构成。
 - <**name**> (可选) 指定连杆几何形状的名称
 - <**origin**> (可选, defaults to identity if not specified)
 - 碰撞组件的参考坐标系相对于连杆坐标系的参考坐标系。
 - **xyz** (可选， 默认零向量) 表示x , y , z x,y,zx,y,z 方向的偏置，单位为米。
 - **rpy** (可选, defaults to identity if not specified) 表示坐标轴在RPY方向上的旋转，单位为弧度。
 - <**geometry**> 与上述**geometry**元素描述相同

详细元素以及各个元素的作用可以前往[官方文档](#)进行查看

3.3 joint部分

joint部分描述了关节的运动学和动力学，并指定了关节的安全限值。

3.3.1 joint的属性：

name:

指定关节的唯一名称

type:

指定关节的类型，其中类型可以是下列类型之一：

- **revolute** - 沿轴线旋转的铰链接头，其范围由上限和下限指定。
- 连续 - 一种连续铰链接头，围绕轴旋转，没有上限和下限。
- 棱柱形 - 沿轴滑动的滑动接头，其范围由上限和下限指定。
- 固定 - 这不是真正的关节，因为它不能移动。所有自由度都被锁定。这种类型的接头不需要轴，校准，动力学，极限或**safety_controller**。
- 浮动 - 此接头允许所有 6 个自由度的运动。
- 平面 - 此接头允许在垂直于轴的平面上运动。

3.3.2 joint的元素

- <**origin**> (可选, defaults to identity if not specified) 从parent link到child link 的变换，joint位于child link的原点，修改该参数可以调整连杆的位置，可用在调整实际模型与理论模型误差，但不建议大幅度修改，因为该参数影响连杆stl

的位置，容易影响碰撞检测效果。

- xyz (可选: 默认为零向量) 代表x , y , z x,y,zx,y,z轴方向上的偏移，单位米。
 - rpy (可选: 默认为零向量) 代表绕着固定轴旋转的角度: roll绕着x轴,pitch绕着y轴, yaw绕着z轴，用弧度表示。
- <parent> (必需)
 - parent link的名字是一个强制的属性。
 - link parent link的名字，是这个link在机器人结构树中的名字。
- <child> (必需)
 - child link的名字是一个强制的属性。
 - link child link的名字，是这个link在机器人结构树中的名字。
- <axis> (可选: 默认为(1,0,0))
 - joint的axis轴在joint的坐标系中。这是旋转轴(*revolute joint*)，*prismatic joint*移动的轴，是*planar joint*的标准平面。这个轴在joint坐标系中被指定。修改该参数可以调整关节的旋转所绕着的轴，常用于调整旋转方向，若模型旋向与实际相反，只需乘-1即可。固定(*fixed*)和浮动(*floating*)类型的joint不需要用到这个元素。
 - xyz(必需) 代表轴向量的x , y , z x,y,zx,y,z分量，为标准化的向量。
- <calibration> (可选)
 - joint的参考点，用来矫正joint的绝对位置。
 - rising (可选) 当joint正向运动时，参考点会触发一个上升沿。
 - falling (可选) 当joint正向运动时，参考点会触发一个下降沿。
- <dynamics> (可选)
 - 该元素用来指定joint的物理性能。它的值被用来描述joint的建模性能，尤其是在仿真的时候。

<limit> (当关节为旋转或移动关节时为必需)

- 该元素为关节运动学约束。
- lower (可选, 默认为0) 指定joint运动范围下界的属性(*revolute joint*的单位为弧度, *prismatic joint*的单位为米)，连续型的joint忽略该属性。
- upper (可选, 默认为0) 指定joint运动范围上界的属性(*revolute joint*的单位为弧度, *prismatic joint*的单位为米)，连续型的joint忽略该属性。
- effort (必需) 该属性指定了joint运行时的最大的力。
- velocity (required) 该属性指定了joint运行时的最大的速度。

<mimic> (可选)

- 这个标签用于指定已定义的 joint 来模仿已存在的 joint 。这个joint的值可以用以下公式计算: $value = multiplier * other_joint_value + offset$
- joint(必填) 需要模仿的joint的名字。
- multiplier(可选) 指定上述公式中的乘数因子。
- offset(可选) 指定上述公式中的偏移项。默认值为0

<safety_controller> (可选)

- 该元素为安全控制限制，此元素下数据会读入到move_group中，但实际上时无效，move_group会跳过此处限制直接读取limit下的参数内容，同时设置该

元素有几率会导致规划失败。

- **soft_lower_limit** (可选, 默认为0) 该属性指定了joint安全控制边界的下界, 是joint安全控制的起始限制点。这个值需要大于上述的limit中的lower值。
- **soft_upper_limit** (可选, 默认为0) 该属性指定了joint安全控制边界的上界, 是joint安全控制的起始限制点。这个值需要小于上述的limit中的upper值。
- **k_position**(可选, 默认为0) 本属性用于说明位置和速度之间的关系。
- **k_velocity**(必需) 本属性用于说明力和速度之间的关系。

详细元素以及各个元素的作用可以前往 <http://wiki.ros.org/urdf/XML/joint> 进行查看

4 常用命令工具

在ROS中, 有许多常用的命令行工具, 这些工具可以帮助你进行开发、调试、管理ROS节点等。以下是一些常用的ROS命令行工具:

4.1 编译工作空间

```
caktin_make
```

4.2 roscore

启动ROS主节点。在运行ROS节点之前, 通常需要先启动roscore

```
roscore
```

4.3 rosrun

运行指定的ROS节点。

```
rosrun package_name node_name
```

4.4 roslaunch

使用Launch文件启动一个或多个ROS节点。

```
roslaunch package_name launch_file.launch
```

4.5 rosnode

查看正在运行的ROS节点信息。

```
rosnode list  
rosnode info node_name
```

4.6 rostopic

查看正在运行的ROS话题信息。

```
rostopic list  
rostopic echo topic_name
```

4.7 rosservice

查看和调用ROS服务。

```
roservice list  
roservice call service_name
```

4.8 rosparam

获取和设置ROS参数。

```
rosparam get parameter_name  
rosparam set parameter_name value
```

4.9 rosmsg

查看ROS消息类型。

```
rosmsg show message_type
```

4.10 rosdep

安装ROS包的依赖项。

```
rosdep install package_name
```

4.11 环境变量

查看ROS_PACKAGE_PATH环境变量

```
echo $ROS_PACKAGE_PATH
```

← 上一页 | 下一页 →

rviz的简单介绍及使用

rviz是ROS中一款三维可视化平台，一方面能够实现对外部信息的图形化显示，另外还可以通过 rviz 给对象发布控制信息，从而实现对机器人的监测与控制。

1 rviz的安装及界面简介

在安装ros时，如果执行的完全安装，rviz已经安装好了，您可以直接尝试运行；如果没有完全安装，可单独安装rviz：

```
# Ubuntu20.04
sudo apt-get install ros-noetic-rviz
```

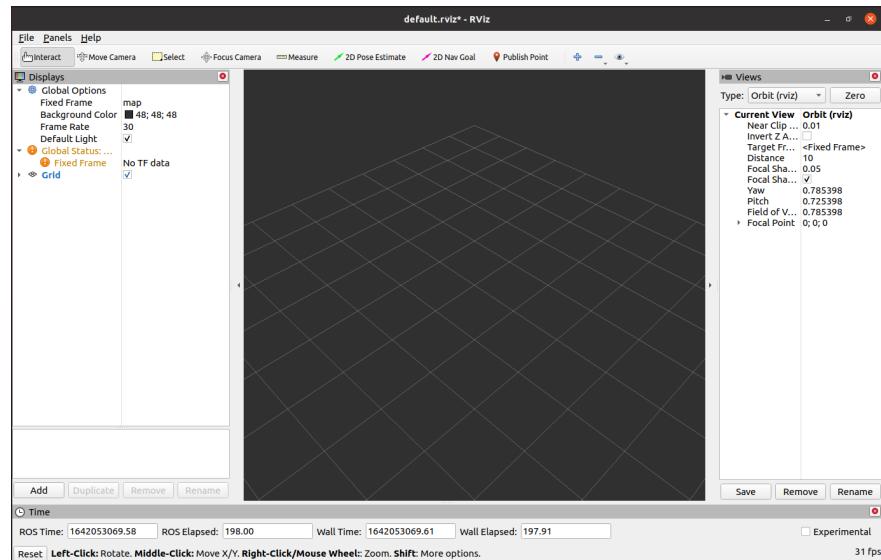
安装完成后，请先打开一个新的终端(快捷键Ctrl+Alt+T)，输入如下指令：

```
roscore
```

然后再打开一个一个新的终端(快捷键Ctrl+Alt+T)输入命令打开rviz

```
rosrun rviz rviz
# 或
rviz
```

打开rviz，显示如下界面：



1.1 各个区域介绍

- 左侧为显示器列表，显示器是在3D世界中绘制某些内容的东西，并且可能在显示列表中具有一些可用的选项。
- 上方是工具栏，允许用户用各种功能按键选择多种功能的工具

- 中间部分为3D视图: 它是可以用三维方式查看各种数据的主屏幕。3D视图的背景颜色、固定框架、网格等可以在左侧显示的全局选项 (Global Options) 和网格 (Grid) 项目中进行详细设置。
- 下方为时间显示区域，包括系统时间和ROS时间等。
- 右侧为观测视角设置区域，可以设置不同的观测视角。

本部分我们只进行粗略的介绍，如果您想了解更多详细的内容，可以前往[用户指南](#)进行查看。

2 简单使用

通过**launch**文件启动

本例子建立在您已经完成[环境搭建](#)，并成功将本公司的代码从GitHub上复制下来的基础上。

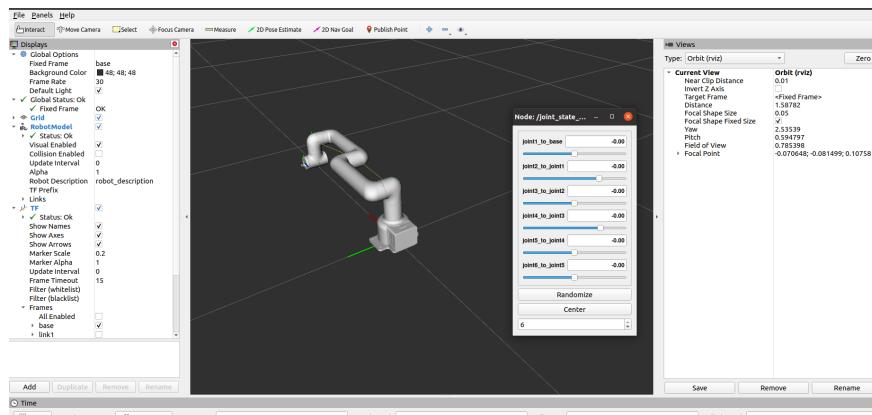
打开一个控制台终端(快捷键Ctrl+Alt+T)，在命令行输入以下命令：进行ROS的环境配置。

```
cd ~/catkin_ws/
source devel/setup.bash
```

再输入：

```
rosrun mycobot_630 test.launch
```

打开rviz，并得到如下结果：



如果您想了解更多rviz的相关资料信息，您可以前往[官方文档](#)进行查看

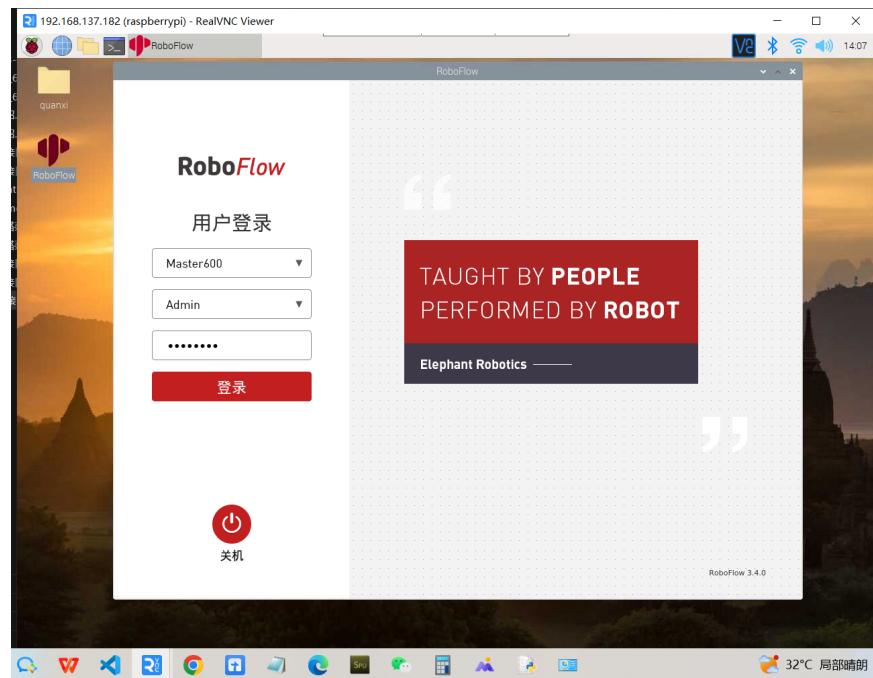
← 上一页 | 下一页 →

机械臂的控制

1 开启TCP服务器功能

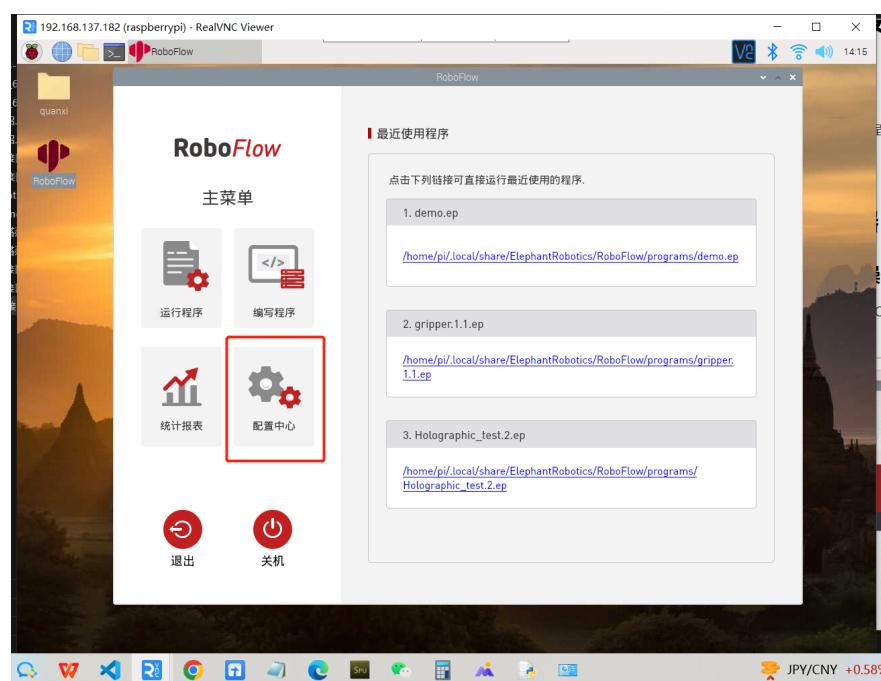
1.1 登录RoboFlow操作系统

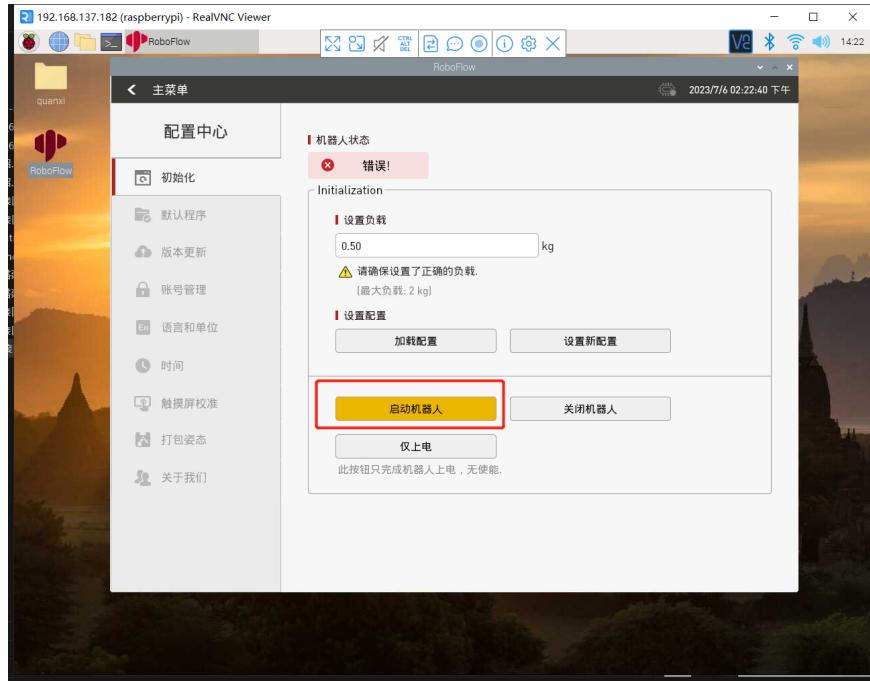
机器人上电开机后，使用VNC Viewer进入树莓派，登录RoboFlow操作系统



1.2 启动机器人

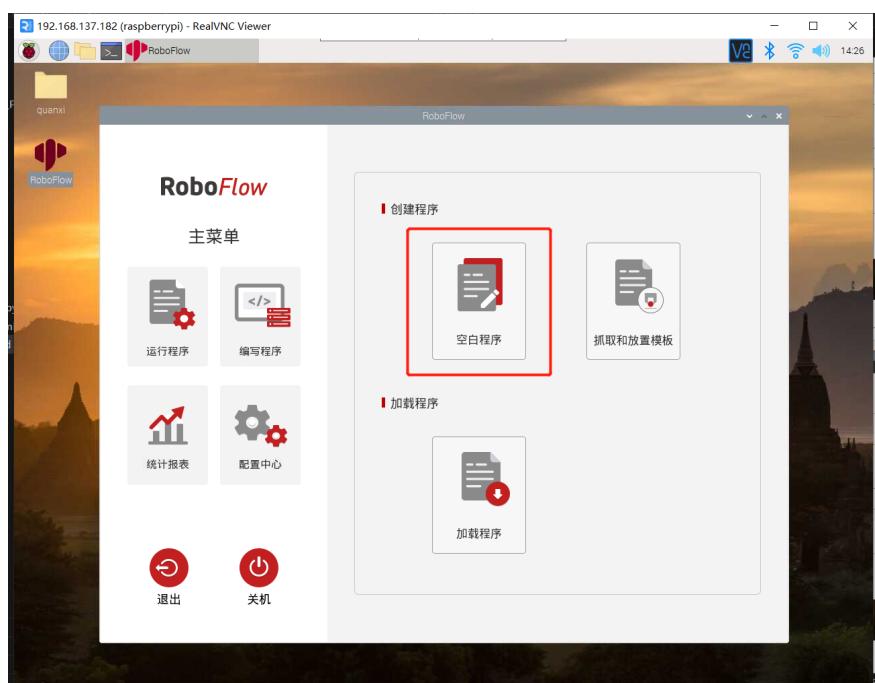
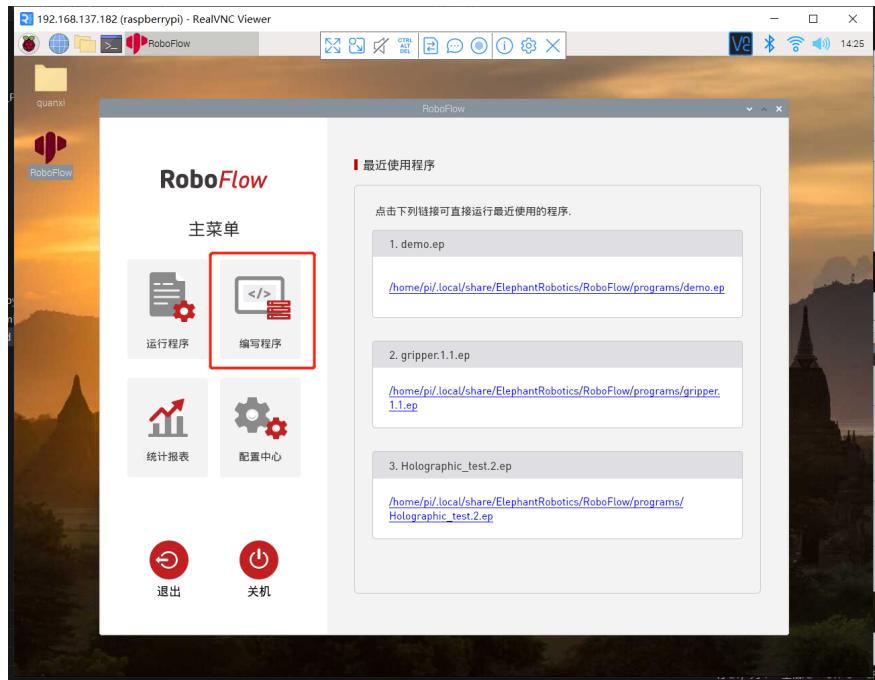
进入配置中心，点击启动机器人按钮

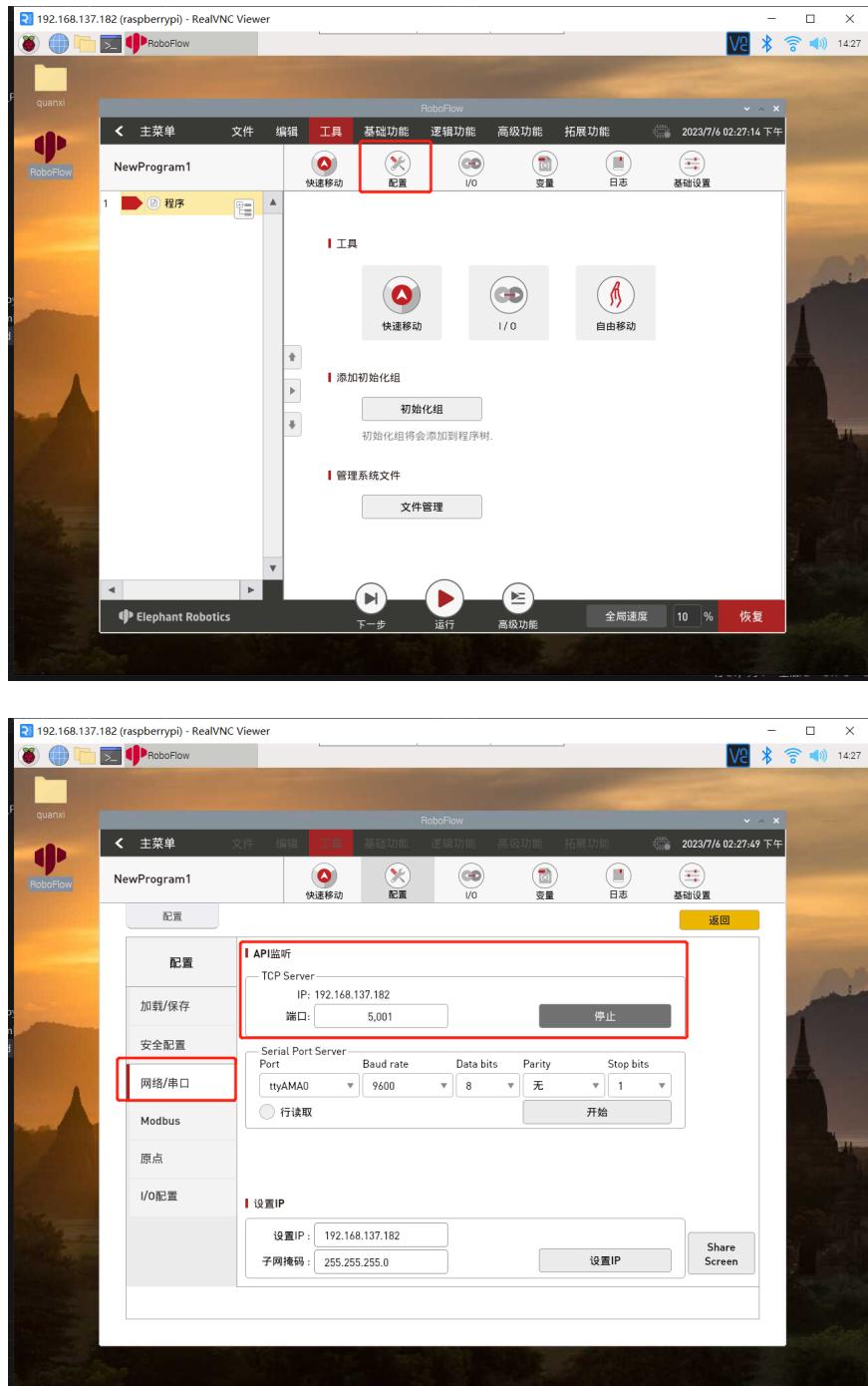




1.3 检查TCP服务器是否开启

返回主菜单，点击编写程序后，再点击空白程序，进入程序编辑界面后，点击配置按钮，点击网络/串口选项，检查TCP服务器是否开启，通常情况下，**TCP服务器是默认开启的**，若未开启，则需手动开启，也手动设置IP地址，如设置成**192.168.1.159**。





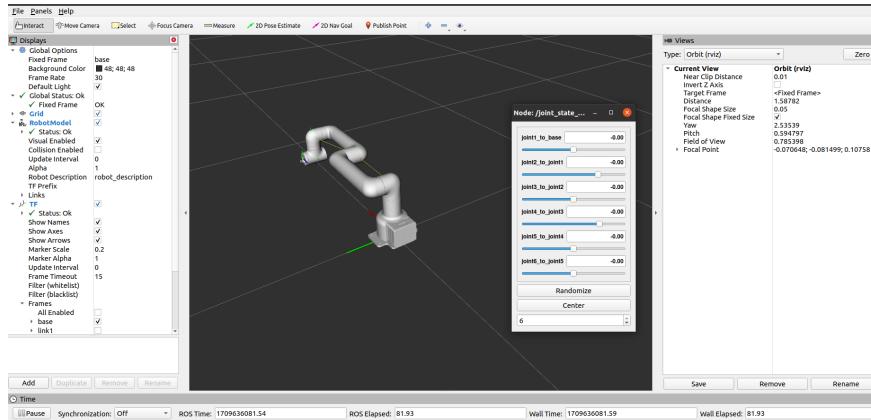
2 滑块控制

注意：TCP通信需确保本地虚拟机电脑与MyCobot Pro630系统使用同一网络，同一网段。

打开一个控制台终端，运行命令：

```
roslaunch mycobot_630 mycobot_630_slider.launch
```

它将打开 rviz 和一个滑块组件，你将看到如下画面：



接着你可以通过拖动滑块来控制 rviz 中的模型移动。如果你想让真实的 MyCobot Pr630 跟着一起运动，需要再打开一个控制台终端，运行命令：

```
# MyCobot Pro 630默认的Socket IP地址为192.168.1.159，端口号为5001，若不一致，可根据实际的IP
rosrun mycobot_630 mycobot_630_slider.py _ip:=192.168.1.159 _port:=5001
```

请注意：由于在命令输入的同时机械臂会移动到模型目前的位置，在您使用命令之前请确保rviz中的模型没有出现穿模现象。不要在连接机械臂后做出快速拖动滑块的行为，防止机械臂损坏。

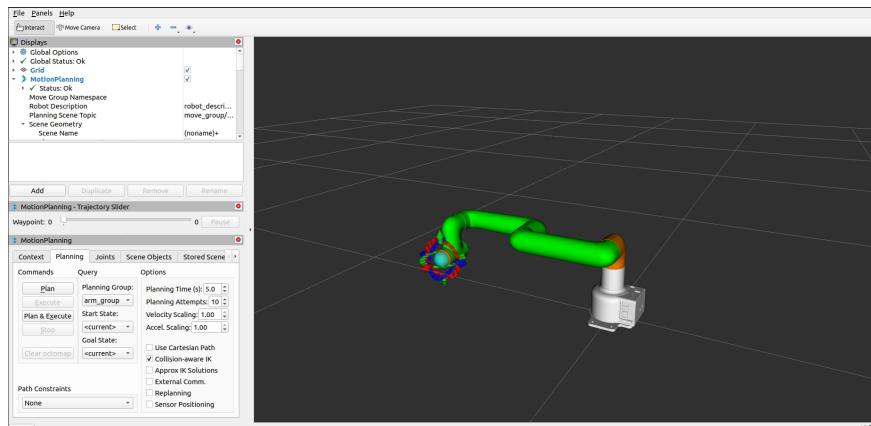
3 Moveit使用

`mycobot_630` 现已集成了 Moveit 部分。

打开一个控制台终端(快捷键Ctrl+Alt+T)，运行命令：

```
roslaunch mycobot_630_moveit mycobot630_moveit.launch
```

运行效果如下：

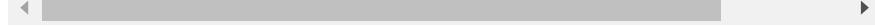


可以计划并执行，演示效果：

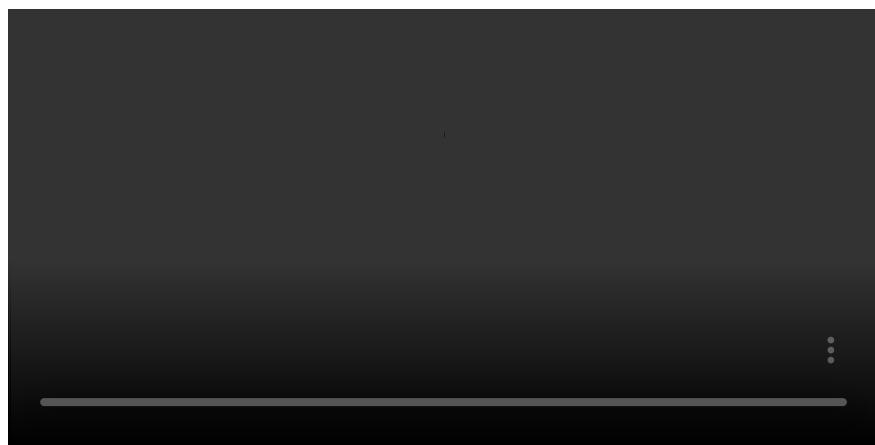


如果需要让真实的机械臂同步执行计划，需要再打开一个控制台终端，运行命令：

```
# MyCobot Pro 630默认的Socket IP地址为192.168.1.159，端口号为5001，若不一致，可根据实际的IP  
rosrun mycobot_630_moveit sync_plan.py _ip:=192.168.1.159 _port:=5001
```



然后再次计划并执行，演示效果：



← 上一页 | 下一节 →

在Linux中安装不同版本的ubuntu系统

1 安装虚拟机

注意：安装虚拟机系统时，请安装**Ubuntu 20.04**版本的系统，安装方法与ubuntu 18.04一致。

在Linux中安装不同版本的ubuntu系统，具体安装方法可参考 [6.2 ROS1环境搭建](#) 章节。

2 ROS2 环境搭建

2.1 ROS2 安装

基本的开发环境搭建需要安装机器人操作系统 ROS2 和 git 版本管理器，以下分别介绍其安装方法及流程。

2.1.1 版本选择

ROS2 跟 ubuntu 有一一对应的关系，不同版本的 ubuntu 对应不同版本的 ROS2，参考网站见下：<http://docs.ros.org/en/foxy/Releases.html>

这里给出对应Ubuntu支持的 ROS2 版本：

| ROS2版本 | 发布日期 | 维护截止日期 | Ubuntu版本 |
|----------|------------|----------|-------------------------------|
| Foxy | 2020年6月5日 | 2023年5月 | Ubuntu 20.04(Focal Fossa) |
| Galactic | 2021年5月23日 | 2022年11月 | Ubuntu 20.04(Focal Fossa) |
| Humble | 2022年5月23日 | 2027年5月 | Ubuntu 22.04(Jammy Jellyfish) |

请根据自己安装的Ubuntu版本进行对应ROS2版本的安装

如果版本不同，下载将会失败.在这里我们选择的系统为 Ubuntu 20.04 (推荐), 对应 ROS2 版本为 ROS2 Foxy

NOTE: 目前我们不提供 windows 安装 ROS2 的任何参考, 若有需要请参考
<http://docs.ros.org/en/foxy/Installation/Alternatives/Windows-Development-Setup.html>

2.1.2 开始安装

1 添加源

Ubuntu 本身的软件源列表中没有 ROS2 的软件源，所以需要先将 **ROS2** 软件源配置到软件列表仓库中，才能下载 ROS2。打开一个控制台终端(快捷键 Ctrl+Alt+T),输入如下指令：

- 官方源:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] https://packages.ros.org/ros2/ubuntu $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/ros2.list &gt;> /dev/null
```

- 若下载速度缓慢，推荐就近选择一个镜像源替换上面的命令。例如，huawei cloud为：

```
echo "deb [arch=$(dpkg --print-architecture)] https://repo.huaweicloud.com/ros2/ubuntu $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/ros2.list &gt;> /dev/null
```

2 设置秘钥

配置公网秘钥,这一步是为了让系统确认我们的路径是安全的的，这样下载文件才没有问题，不然下载后会被立刻删掉：

```
sudo apt install curl gnupg2 -y  
curl -s https://gitee.com/ohhuo/rosdistro/raw/master/ros.asc | sudo apt-key add -
```

3 安装

在加入了新的软件源后，需要更新软件源列表，打开一个控制台终端(快捷键 Ctrl+Alt+T),输入如下指令：

```
sudo apt-get update
```

执行安装 **ROS2**，打开一个控制台终端(快捷键Ctrl+Alt+T),请按照自己的Ubuntu 版本选择输入以下指令：

```
# Ubuntu 20.04 foxy版本  
sudo apt install ros-foxy-desktop
```

```
# Ubuntu 20.04 galactic版本  
sudo apt install ros-galactic-desktop
```

```
# Ubuntu 22.04 humble版本  
sudo apt install ros-humble-desktop
```

安装过程耗时比较长，需要耐心等待

安装完成后刷新环境变量：

```
source /opt/ros/foxy/setup.bash
```

2.1.3 设置**ros2**环境

为了避免每次关掉终端窗口后都需要重新生效 ROS2 功能路径，我们可以把路径配置到环境变量中，这样在每次打开新的终端便可自动生效 ROS2 功能路径，在终端依次执行以下命令，打开一个控制台终端(快捷键Ctrl+Alt+T)执行以下对应版本的命令：

```
# Ubuntu 20.04 foxy版本  
echo "source /opt/ros/foxy/setup.bash" >> ~/.bashrc  
  
# Ubuntu 20.04 galactic版本  
echo "source /opt/ros/galactic/setup.bash" >> ~/.bashrc  
  
# Ubuntu 22.04 humble版本  
echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc
```

然后运行：

```
source ~/.bashrc
```

安装 ROS2 额外依赖项

在终端输入以下命令安装**ROS2**额外依赖项，打开一个控制台终端(快捷键 Ctrl+Alt+T)：

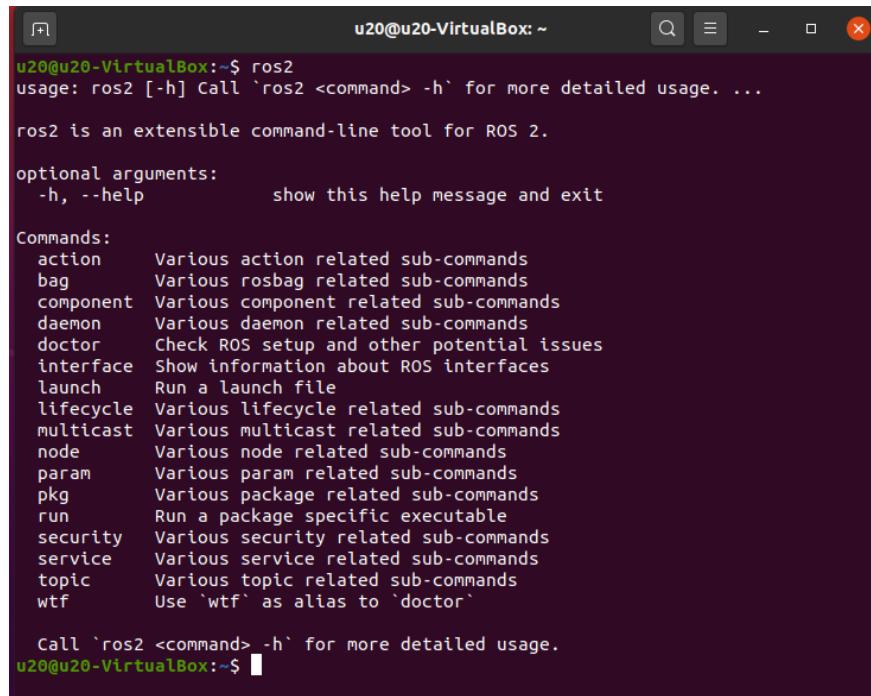
```
sudo apt install python3-argcomplete -y  
  
sudo apt install ros-foxy-xacro  
  
sudo apt-get install python3-colcon-common-extensions  
  
# Ubuntu 20.04 foxy版本  
sudo apt install ros-foxy-joint-state-publisher-gui  
  
# Ubuntu 20.04 galactic版本  
sudo apt install ros-galactic-joint-state-publisher-gui  
  
# Ubuntu 22.04 humble版本  
sudo apt install ros-humble-joint-state-publisher-gui
```

2.1.4 验证安装

为了验证 ROS2 是否安装成功，打开一个控制台终端(快捷键Ctrl+Alt+T)，在终端执行以下命令：

```
ros2
```

当显示如下界面，则表示 ROS2 安装成功



```
u20@u20-VirtualBox:~$ ros2
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...

ros2 is an extensible command-line tool for ROS 2.

optional arguments:
  -h, --help            show this help message and exit

Commands:
  action    Various action related sub-commands
  bag       Various rosbag related sub-commands
  component Various component related sub-commands
  daemon   Various daemon related sub-commands
  doctor   Check ROS setup and other potential issues
  interface Show information about ROS interfaces
  launch   Run a launch file
  lifecycle Various lifecycle related sub-commands
  multicast Various multicast related sub-commands
  node     Various node related sub-commands
  param    Various param related sub-commands
  pkg      Various package related sub-commands
  run      Run a package specific executable
  security Various security related sub-commands
  service  Various service related sub-commands
  topic    Various topic related sub-commands
  wtf      Use `wtf` as alias to `doctor`

Call `ros2 <command> -h` for more detailed usage.
u20@u20-VirtualBox:~$
```

2.2 git 安装

1 更新软件源列表

打开一个控制台终端(快捷键Ctrl+Alt+T)，在终端窗口输入以下命令，以更新软件源列表：

```
sudo apt-get update
```

2 安装 git

打开一个控制台终端(快捷键Ctrl+Alt+T)，在终端窗口输入以下命令，执行 **git** 的安装：

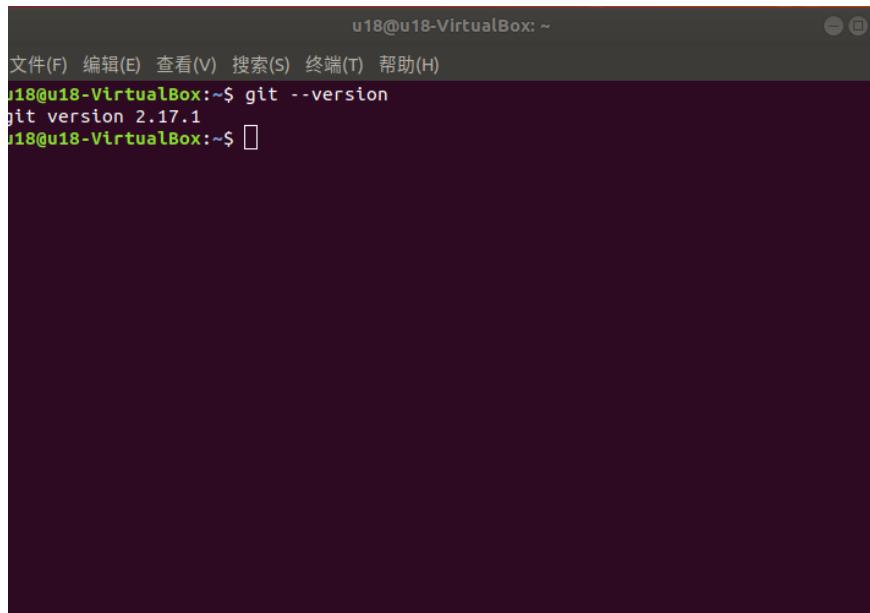
```
sudo apt-get install git
```

3 验证安装

读取 **git** 版本，打开一个控制台终端(快捷键Ctrl+Alt+T)，在终端窗口输入以下命令：

```
git --version
```

在终端中可以显示 **git** 版本号，如下，即为安装成功。



```
u18@u18-VirtualBox: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
u18@u18-VirtualBox:~$ git --version
git version 2.17.1
u18@u18-VirtualBox:~$ 
```

2.3 mycobot_ros2 安装

`mycobot_ros2` 是 ElephantRobotics 推出的，适配旗下桌面型六轴机械臂 mycobot 系列的ROS2 包。

项目地址：http://github.com/elephantrobotics/mycobot_ros2

1 前提

在安装包之前，请保证拥有 `ros2` 工作空间。

这里我们给出创建工作空间的样例命令，打开一个控制台终端(快捷键 `Ctrl+Alt+T`)，在命令行输入以下命令：

```
mkdir -p ~/colcon_ws/src # 创建文件夹
cd ~/colcon_ws
colcon build # 编译工作空间
```

添加工作空间的环境

官方默认的 ROS2 工作空间是 `colcon_ws`。

```
echo "source ~/colcon_ws/install/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

2 安装

NOTE:

- 本包依赖于ROS2和MoveIT2，使用前确保以成功安装ROS2。
- 本包与真实机械臂的交互依赖于PythonApi - `pymycobot`
- Api项目地为：<https://github.com/elephantrobotics/pymycobot>

- 快速安装: `pip install pymycobot --upgrade`

执行`pip install pymycobot --upgrade`命令时, 若出现如下图错误提示:

```
u182@u182-VirtualBox:~$ pip install pymycobot --upgrade
Command 'pip' not found, but can be installed with:
sudo apt install python-pip
```

根据提示输入以下命令安装pip

```
sudo apt install python3-pip
```

pip安装完成后, 终端再次执行

```
pip install pymycobot --upgrade
```

- 安装方式依赖于Git, 请确保电脑中已安装Git。

官方默认的ROS2工作区是`colcon_ws`。

```
cd colcon_ws/src # 进入工作区的src文件夹中
# 克隆github上的代码
git clone https://github.com/elephantrobotics/mycobot_ros2.git
cd ~/colcon_ws      # 返回工作区
colcon build --symlink-install # 构建工作区中的代码, --symlink-install: 避免每次调整 pyth
source install/setup.bash # 添加环境变量
```

至此ROS2环境搭建完成, 您可以学习[ROS2的基础知识](#)或者[ROS2使用案例](#)

← 上一页 | 下一页 →

1 ROS2工程结构

1.1 colcon工作空间

colcon工作空间是创建、修改、编译软件包的目录。colcon的工作空间，直观的形容就是一个仓库，里面装载着ROS的各种项目工程，便于系统组织管理调用。

- 创建工作空间：

```
mkdir -p ~/colcon_ws/src # 创建文件夹  
cd ~/colcon_ws/           # 进入文件夹  
colcon build               # 构建工作区中的代码。
```

注意： colcon 支持选项 `--symlink-install`。这允许通过更改 `source` 空间中的文件（例如 Python 文件或其他未编译的资源）来更改已安装的文件，以加快迭代速度。避免每次修改 python 脚本时都需要重新编译。

```
colcon build --symlink-install
```

ROS2工作空间是一个具有特定结构的目录。通常有一个 `src` 子目录。在该子目录中是 ROS2 包的源代码所在的位置。通常，目录以其他方式为空开始。

colcon 会进行源代码构建。默认情况下，它将创建以下目录作为 `src` 目录的同级目录：

```
src/: ROS2的colcon软件包（源代码包）  
build/: 存储中间文件的位置。对于每个包，将创建一个子文件夹，例如在其中调用 CMake。  
install/: 每个包的安装位置。默认情况下，每个包都将安装到单独的子目录中。  
log/: 包含有关每个 colcon 调用的各种日志记录信息。
```

一个ROS2工作空间目录结构如下所示：

```
WorkSpace --- 自定义的工作空间。
|--- build: 存储中间文件的目录，该目录下会为每一个功能包创建一个单独子目录。
|--- install: 安装目录，该目录下会为每一个功能包创建一个单独子目录。
|--- log: 日志目录，用于存储日志文件。
|--- src: 用于存储功能包源码的目录。
|   |-- C++功能包
|       |-- package.xml: 包信息，比如：包名、版本、作者、依赖项。
|       |-- CMakeLists.txt: 配置编译规则，比如源文件、依赖项、目标文件。
|       |-- src: C++源文件目录。
|       |-- include: 头文件目录。
|       |-- msg: 消息接口文件目录。
|       |-- srv: 服务接口文件目录。
|       |-- action: 动作接口文件目录。
|   |-- Python功能包
|       |-- package.xml: 包信息，比如：包名、版本、作者、依赖项。
|       |-- setup.py: 与C++功能包的CMakeLists.txt类似。
|       |-- setup.cfg: 功能包基本配置文件。
|       |-- resource: 资源目录。
|       |-- test: 存储测试相关文件。
|       |-- 功能包同名目录: Python源文件目录。
```

1.2 ROS2软件包

Package不仅是Linux上的软件包，也是colcon编译的基本单元，我们使用 `colcon build` 编译的对象就是每个ROS2的package。

创建自己的软件包：

- 使用Python创建软件包的命令语法为：

```
ros2 pkg create --build-type ament_python <package_name>
```

- 例如：

```
ros2 pkg create --build-type ament_python --node-name my_node my_package
```

2 基本工具命令

在本章中，您将了解ROS2的常用命令工具。

2.1 Topics

ROS 2 将复杂的系统分解为许多模块化节点。Topics是ROS图的重要元素，充当节点交换消息的总线。Topics是数据在节点之间移动的主要方式之一，因此在系统的不同部分之间移动。

具体参考：[官方教程](#)

- **topics** 帮助

```
ros2 topics -h
```

- 启动**turtlesim**和键盘控制

```
ros2 run turtlesim turtlesim_node  
ros2 run turtlesim turtle_teleop_key
```

- 节点关系图

```
rqt_graph
```

- 了解**topics**相关命令

```
ros2 topics -h
```

- 话题列表

```
ros2 topic list  
ros2 topic list -t # 显示相应的消息类型
```

- 查看话题内容

```
ros2 topic echo <topic_name>  
ros2 topic echo /turtle1/cmd_vel
```

- 显示话题相关信息，类型

```
ros2 topic info <topic_name>  
# 输出 /turtle1/cmd_vel 话题接口相关信息  
ros2 topic info /turtle1/cmd_vel
```

- 显示接口相关信息

```
ros2 interface show <msg_type>  
# 输出 geometry_msgs/msg/Twist接口相关信息  
ros2 interface show geometry_msgs/msg/Twist
```

- 发布命令

```
ros2 topic pub <topic_name> <msg_type> '<args>'  
# 发布速度命令  
ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}'  
# 按一定频率发布速度命令  
ros2 topic pub --rate 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}
```

- 查看话题发布的频率

```
ros2 topic hz <topic_name>  
#输出/turtle1/cmd_vel发布频率  
ros2 topic pub --rate 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}
```

2.2 Nodes

ROS 中的每个节点都应该负责一个单一的模块用途（例如，一个节点用于控制车轮电机，一个节点用于控制激光测距仪等）。每个节点都可以通过主题、服务、操作或参数向其他节点发送和接收数据。一个完整的机器人系统由许多协同工作的节点组成。在 ROS 2 中，单个可执行文件（C++ 程序、Python 程序等）可以包含一个或多个节点。

具体参考: [官方教程](#)

- **nodes** 帮助

```
ros2 nodes -h
```

- 启动**turtlesim**和键盘控制

```
ros2 run turtlesim turtlesim_node  
ros2 run turtlesim turtle_teleop_key
```

- 查看节点列表

```
ros2 node list
```

- 查看节点关系图

```
rqt_graph
```

- 重映射

```
ros2 run turtlesim turtlesim_node --ros-args --remap __node:=my_turtle  
ros2 node list
```

- 查看节点信息

```
ros2 node info <node_name>  
ros2 node info /my_turtle
```

2.3 Services

服务是 ROS 图中节点的另一种通信方法。服务基于调用和响应模型，而不是主题的发布者-订阅者模型。虽然主题允许节点订阅数据流并获得持续更新，但服务仅在客户端专门调用时才提供数据。

具体参考: [官方教程](#)

- **services** 帮助

```
ros2 service -h
```

- 启动**turtlesim**和键盘控制

```
ros2 run turtlesim turtlesim_node  
ros2 run turtlesim turtle_teleop_key
```

- 查看服务列表

```
ros2 service list  
# 显示服务列表及消息类型  
ros2 service list -t
```

- 查看服务接收到的消息类型

```
ros2 service type <service_name>  
ros2 service type /clear
```

- 找到使用某类消息类型的服务

```
ros2 service find <type_name>  
ros2 service find std_srvs/srv/Empty
```

- 查看服务消息类型定义

```
ros2 interface show <type_name>.srv  
ros2 interface show std_srvs/srv/Empty.srv
```

- 调用服务命令,清除行走轨迹

```
ros2 service call <service_name> <service_type>  
ros2 service call /clear std_srvs/srv/Empty
```

- 生成新乌龟

```
ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name: 'turtle2'}
```

2.4 Parameters

参数是节点的配置值。您可以将参数视为节点设置。节点可以将参数存储为整数、浮点数、布尔值、字符串和列表。在 ROS 2 中，每个节点都维护自己的参数。有关参数的更多背景信息，请参阅概念文档。

具体参考: [官方教程](#)

- parameters** 帮助

```
ros2 param -h
```

- 启动**turtlesim**和键盘控制

```
ros2 run turtlesim turtlesim_node  
ros2 run turtlesim turtle_teleop_key
```

- 查看服务列表

```
ros2 param list
```

- 获取参数值

```
ros2 param get <node_name> <parameter_name>  
ros2 param get /turtlesim background_g
```

- 设置参数值

```
ros2 param set <node_name> <parameter_name> <value>  
ros2 param set /turtlesim background_r 150
```

- 导出参数值

```
ros2 param dump <node_name>  
ros2 param dump /turtlesim
```

- 独立导入参数

```
ros2 param load <node_name> <parameter_file>  
ros2 param load /turtlesim ./turtlesim.yaml
```

- 启动节点同时导入参数

```
ros2 run <package_name> <executable_name> --ros-args --params-file <file_name>  
ros2 run turtlesim turtlesim_node --ros-args --params-file ./turtlesim.yaml
```

2.5 Actions

动作是 ROS 2 中的一种通信类型，用于长时间运行的任务。它们由三部分组成：
目标、反馈和结果。

操作基于主题和服务。它们的功能类似于服务，除了操作是可抢占的（您可以在执行时取消它们）。他们还提供稳定的反馈，而不是返回单一响应的服务。

操作使用客户端-服务器模型，类似于发布者-订阅者模型（在主题教程中描述）。“动作客户端”节点将目标发送到“动作服务器”节点，该节点确认目标并返回反馈流和结果。

具体参考: [官方教程](#)

- **action** 帮助

```
ros2 action -h
```

- 启动**turtlesim**和键盘控制

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

按G|B|V|C|D|E|R|T 实现旋转，按F键盘取消I

- 查看节点**action**的服务端和客户端

```
ros2 node info /turtlesim
```

- 查看动作列表

```
ros2 action list
ros2 action list -t # 显示动作类型
```

- 查看动作信息

```
ros2 action info <action>
ros2 action info /turtle1/rotate_absolute
```

- 查看动作消息内容

```
ros2 interface show turtlesim/action/RotateAbsolute
```

- 发送动作目标信息

```
ros2 action send_goal <action_name> <action_type>
ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 90.0}"
# 带反馈信息
ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 90.0}"
```

2.6 RQt

RQt 是一个图形用户界面框架，它以插件的形式实现各种工具和界面。可以将所有现有的 **GUI** 工具作为 **RQt** 中的可停靠窗口运行！这些工具仍然可以以传统的独立方式运行，但 **RQt** 可以更轻松地在单个屏幕布局中管理所有不同的窗口。

具体参考: [官方教程](#)

您可以通过以下方式轻松运行任何 **RQt** 工具/插件：

```
rqt
```

- rqt** 帮助

```
rqt -h
```

- 启动**turtlesim**和键盘控制

```
ros2 run turtlesim turtlesim_node  
ros2 run turtlesim turtle_teleop_key
```

- 动作浏览器: / Plugins -> Actions ->Action Type Browser
- 参数重配置: / Plugins -> configuration ->Parameter Reconfigure
- 节点图: /Node Graph
- 控制转向: /Plugins -> Robot Tools -> Robot Steering
- 服务调用: /Plugins -> Services -> Service Caller
- 服务类型浏览器: Plugins -> Services -> Service Type Browser
- 消息发布: Plugins -> Topics -> Message Publisher
- 消息类型浏览器: Plugins -> Topics -> Message Type Browser
- 话题列表: Plugins -> Topics -> Topic Monitor
- 绘制曲线图: Plugins -> Visualization -> Plot
- 查看日志: **rqt_console**

```
ros2 run rqt_console rqt_console  
ros2 run turtlesim turtlesim_node  
ros2 topic pub -r 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}
```

2.7 TF2

tf2 是变换库，它允许用户随着时间的推移跟踪多个坐标系。 tf2 以时间缓冲的树结构维护坐标系之间的关系，并让用户在任何需要的时间点在任意两个坐标系之间变换点、向量等。

具体参考: [官方教程](#)

让我们从安装演示包及其依赖项开始。

```
sudo apt-get install ros-foxy-turtle-tf2-py ros-foxy-tf2-tools ros-foxy-tf-transformat
```

- 跟随
- launch启动2个小乌龟，第一个小乌龟自动跟随第二个

```
ros2 launch turtle_tf2_py turtle_tf2_demo.launch.py
```

- 通过键盘控制第一个小乌龟移动

```
ros2 run turtlesim turtle_teleop_key
```

- 查看TF树

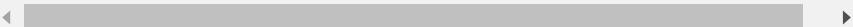
```
ros2 run tf2_tools view_frames.py  
evince frames.pdf
```

- 查看两个坐标系之间的关系

```
ros2 run tf2_ros tf2_echo [reference_frame] [target_frame]  
ros2 run tf2_ros tf2_echo turtle2 turtle1
```

- 在**rviz**上查看**TF**关系

```
ros2 run rviz2 rviz2 -d $(ros2 pkg prefix --share turtle_tf2_py)/rviz/turtle_rviz.rviz
```



2.8 URDF

URDF 是统一机器人描述格式，用于指定 ROS 中的机器人几何和组织。

具体参考: [官方教程](#)

- 完整语法

```

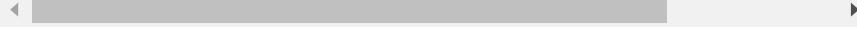
<robot>
  # describe:
  # Parameters: name=""
  # Child node:
  <link>
    # Description:
    # Parameters: name=""
    # Child node:
    <visual>
      # describe:
      # Parameters:
      # child nodes:
      <geometry>
        # description
        # parameters
        # Child node:
        <cylinder />
          # Description:
          # Parameters:
          # length="0.6"
          # radius="0.2"
      <box />
        # description
        # Parameters:size="0.6 0.1 0.2"
      <mesh />
        # Description
        #Parameters: filename="package://urdf_tutorial/mes
  <collision>
    # Description: collision element, prioritized
    # parameters
    # child node
    <geometry>
    <inertial>
      # description
      # parameters
      # Child nodes:
      <mass />
        # description: mass
        # Parameters: value=10
      <inertia />
        # Description: Inertia
        # Parameters: i+Cartesian product of xyz" (9 in t
  <origin />
    # Description:
    # Parameters:
    # rpy="0 1.5 0"
    # xyz="0 0 -0.3"
  <material />
    # Description
    # Parameters: name="blue"
  <joint>
    # Description
    # Parameters:
    # name=""
    # type=""
    # fixed
    # prismatic
  # child node
  <parent />
    # Description
    # Parameters: link=""
  <child />
    # Description:
    # Parameters: link=""
  <origin />

```

```

# Description:
# Parameters: xyz="0 -0.2 0.25"
<limit />
# Description
# Parameters:
#   effort="1000.0"      maximum effort
#   lower="-0.38"        Joint upper limit (radians)
#   upper="0"            Joint lower limit (radians)
#   velocity="0.5"       Maximum velocity
<axis />
# Description: Press ? axis rotation
# Parameters: xyz="0 0 1", along the Z axis
<material>
# Description:
# Parameters: name="blue"
# child node:
<color />
# description:
# Parameters: rgba="0 0 0.8 1"

```



- 安装依赖库

```

sudo apt install ros-foxy-joint-state-publisher-gui ros-foxy-joint-state-publisher
sudo apt install ros-foxy-xacro

```

- 下载源代码

```

cd ~/dev_ws
git clone -b ros2 https://github.com/ros/urdf_tutorial.git src/urdf_tutorial

```

- 编译源代码

```

colcon build --packages-select urdf_tutorial

```

- 运行示例

```

ros2 launch urdf_tutorial display.launch.py model:=urdf/01-myfirst.urdf

```

2.9 Launch

ROS 2 中的启动系统负责帮助用户描述他们系统的配置，然后按照描述执行。系统的配置包括要运行的程序、运行它们的位置、传递给它们的参数，以及 ROS 特定的约定，这些约定通过为每个组件提供不同的配置，使得在整个系统中重用组件变得容易。它还负责监视已启动流程的状态，并报告和/或响应这些流程状态的变化。

用 Python、XML 或 YAML 编写的launch文件可以启动和停止不同的节点，以及触发和处理各种事件。

具体参考: [官方教程](#)

Setup

创建一个新目录来存储您的launch文件：

```
mkdir launch
```

编写启动文件

让我们使用 `turtlesim` 包及其可执行文件将 ROS 2 启动文件放在一起。正如刚才提到的。

将完整代码复制并粘贴到 `launch/turtlesim_mimic_launch.py` 文件中：

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='turtlesim',
            namespace='turtlesim1',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            namespace='turtlesim2',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            executable='mimic',
            name='mimic',
            remappings=[
                ('/input/pose', '/turtlesim1/turtle1/pose'),
                ('/output/cmd_vel', '/turtlesim2/turtle1/cmd_vel')
            ]
        )
    ])
])
```

运行ros2启动文件

要运行上面创建的 `launch` 文件，请进入您之前创建的目录并运行以下命令：

语法格式为：

```
ros2 launch <package_name> <launch_file_name>
```

```
cd launch
ros2 launch turtlesim_mimic_launch.py
```

- **launch** 帮助

```
ros2 launch -h
```

- 运行节点

```
ros2 launch turtlesim_multisim.launch.py
```

- 查看**launch**文件有哪些参数

```
ros2 launch turtlebot3_fake_node turtlebot3_fake_node.launch.py -s  
ros2 launch turtlebot3_fake_node turtlebot3_fake_node.launch.py --show-arguments  
ros2 launch turtlebot3_bringup robot.launch.launch.py -s
```

- 运行**launch**文件带参数

```
ros2 launch turtlebot3_bringup robot.launch.launch.py usb_port:=/dev/openocr
```

- 运行节点并调试

```
ros2 launch turtlesim turtlesim_node.launch.py -d
```

- 只输出节点描述

```
ros2 launch turtlesim turtlesim_node.launch.py -p
```

- 运行组件

```
ros2 launch composition composition_demo.launch.py
```

2.10 Run

run用于运行单个节点，组件程序。

- **run** 帮助

```
ros2 run -h
```

- 运行节点

```
ros2 run turtlesim turtlesim_node
```

- 运行节点带参数

```
ros2 run turtlesim turtlesim_node --ros-args -r __node:=turtle2 -r __ns:=/ns2
```

- 运行组件容器

```
ros2 run rclcpp_components component_container
```

- 运行组件

```
ros2 run composition manual_composition
```

2.11 Package

一个包可以被认为是你的 ROS 2 代码的容器。如果您希望能够安装您的代码或与他人共享，那么您需要将其组织在一个包中。借助软件包，您可以发布您的 ROS 2 作品并允许其他人轻松构建和使用它。

ROS 2 中的包创建使用 `ament` 作为其构建系统，并使用 `colcon` 作为其构建工具。您可以使用官方支持的 CMake 或 Python 创建包，但确实存在其他构建类型。

具体参数: [官方教程](#)

创建工作空间

为每个新工作区创建一个新目录。名称并不重要，但它有助于表明工作区的用途。让我们为“开发工作区”选择目录名称 `ros2_ws`:

```
mkdir -p ~/ros2_ws/src  
cd ~/ros2_ws/src
```

- **pkg** 帮助

```
ros2 pkg -h
```

- 列出功能包

```
ros2 pkg executable turtlesim
```

- 输出某个功能包可执行程序

```
ros2 pkg executable turtlesim
```

- 创建**Python**功能包

在运行包创建命令之前，请确保您位于 `src` 文件夹中。

```
cd ~/ros2_ws/src
```

在 ROS 2 中创建新包的命令语法是：

```
ros2 pkg create --build-type ament_python <package_name>  
# 您将使用可选参数 --node-name 在包中创建一个简单的 Hello World 类型可执行文件。  
ros2 pkg create --build-type ament_python --node-name my_node my_package
```

- 构建**package**

将包放在工作区中特别有价值，因为您可以通过在工作区根目录中运行 `colcon build` 来一次构建许多包。否则，您将不得不单独构建每个包。

```
# 返回到工作区目录:  
cd ~/ros2_ws  
# 现在你可以构建你的包:  
colcon build
```

- **Source setup**文件

要使用您的新包和可执行文件，首先打开一个新终端并获取您的主要 ROS 2 安装源。

然后，从 `ros2_ws` 目录中，运行以下命令来获取您的工作空间：

```
source install/setup.bash
```

现在您的工作区已添加到您的路径中，您将能够使用新包的可执行文件。

- 使用 **package**

要运行您在包创建期间使用 `--node-name` 参数创建的可执行文件，请输入以下命令：

```
ros2 run my_package my_node
```

← 上一页 | 下一页 →

rviz2的简单介绍及使用

Rviz2是一个可视化工具，用于显示机器人环境中的消息，提供3D视角来查看机器人的状态和活动。它可以帮助开发者更好地理解机器人当前的状态和活动，以及其他可视化消息。Rviz2提供了一系列的可视化工具，可以帮助开发者更好地理解机器人的状态和活动，比如可视化坐标系、激光扫描消息、点云消息、机器人模型等等。使用Rviz2，可以轻松地查看和调试机器人系统，从而更好地实现机器人目标。

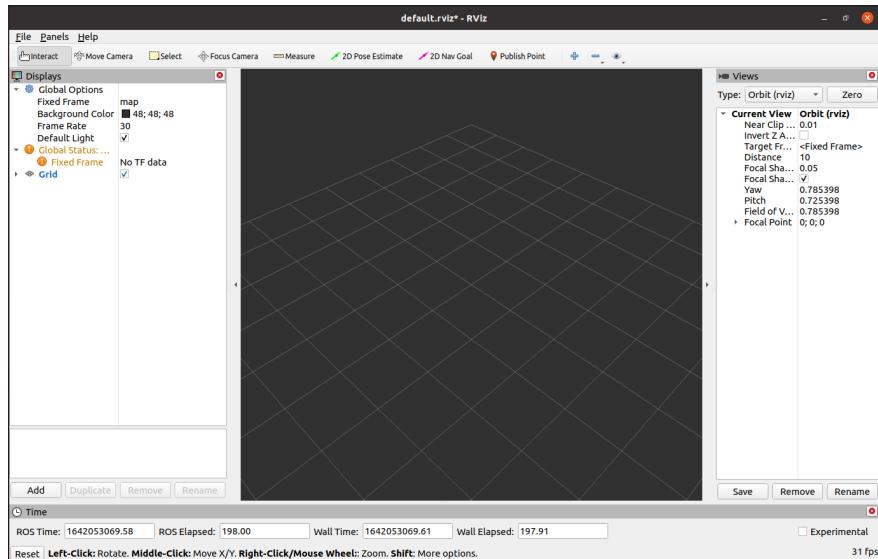
1 rviz2的安装及界面简介

ros2安装成功表明rviz2也一起安装成功了，因为ros2的安装包含了rviz2。

打开一个控制台终端(快捷键Ctrl+Alt+T)，输入命令打开rviz2：

```
ros2 run rviz2 rviz2
# 或
rviz2
```

打开rviz2，显示如下界面：



1.1 各个区域介绍

- 左侧为显示器列表，显示器是在3D世界中绘制某些内容的东西，并且可能在显示器列表中具有一些可用的选项。包括添加、删除、复制、重命名插件，显示插件，以及设置插件属性等功能。
- 上方是工具栏，允许用户用各种功能按键选择多种功能的工具
- 中间部分为3D视图：它是可以用三维方式查看各种数据的主屏幕。3D视图的背景颜色、固定框架、网格等可以在左侧显示的全局选项（Global Options）和网格（Grid）项目中进行详细设置。
- 下方为时间显示区域，包括系统时间和ROS时间等。
- 右侧为观测视角设置区域，可以设置不同的观测视角。

本部分我们只进行粗略的介绍，如果您想了解更多详细的内容，可以前往[用户指南](#)进行查看。

2 简单使用

通过**launch.py**文件启动

本例子建立在您已经完成[环境搭建](#)，并成功将本公司的代码从[GitHub](#)上复制下来的基础上。

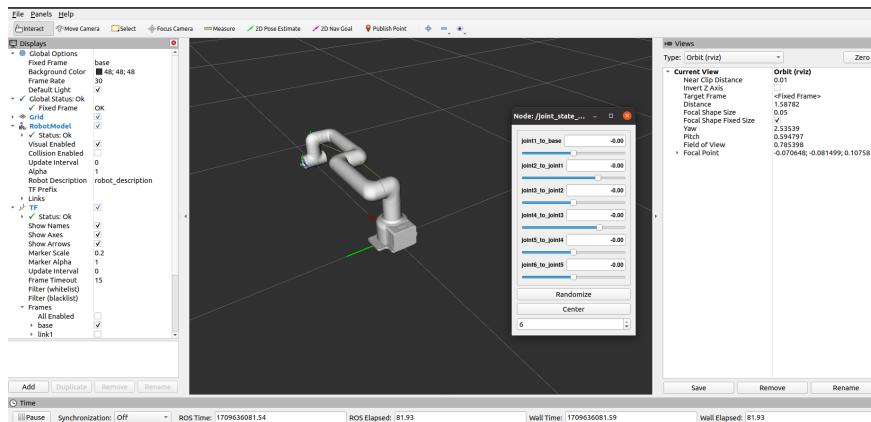
打开一个控制台终端(快捷键Ctrl+Alt+T)，输入以下命令进行**ROS2**的环境配置。

```
cd ~/colcon_ws/
colcon build --symlink-install
source install/setup.bash
```

再输入：

```
ros2 launch mycobot_630 test.launch.py
```

打开**rviz**，并得到如下结果：



如果您想了解更多**rviz**的相关资料信息，您可以前往[官方文档](#)进行查看

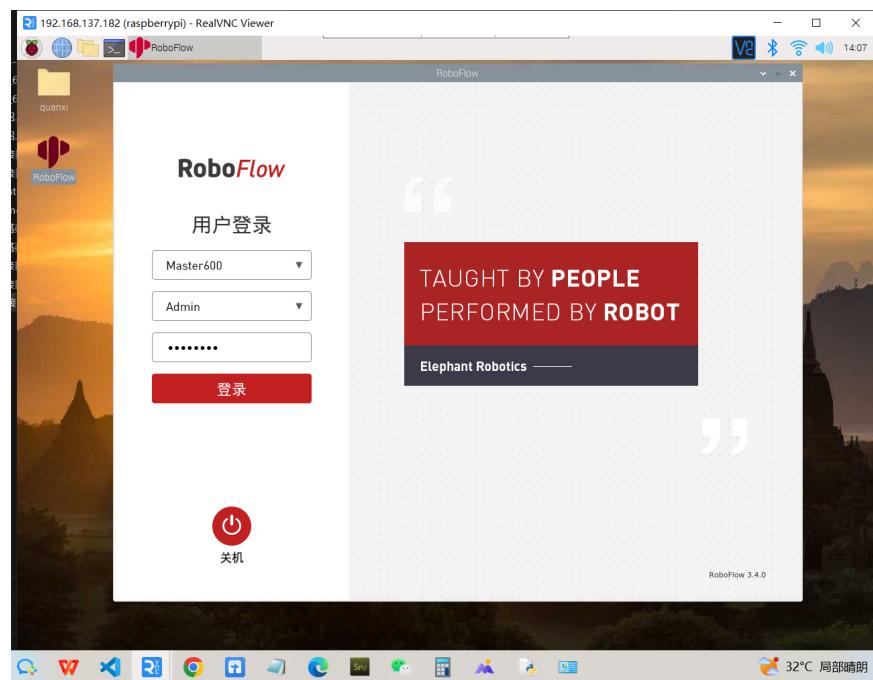
← 上一页 | 下一页 →

机械臂的控制

1 开启TCP服务器功能

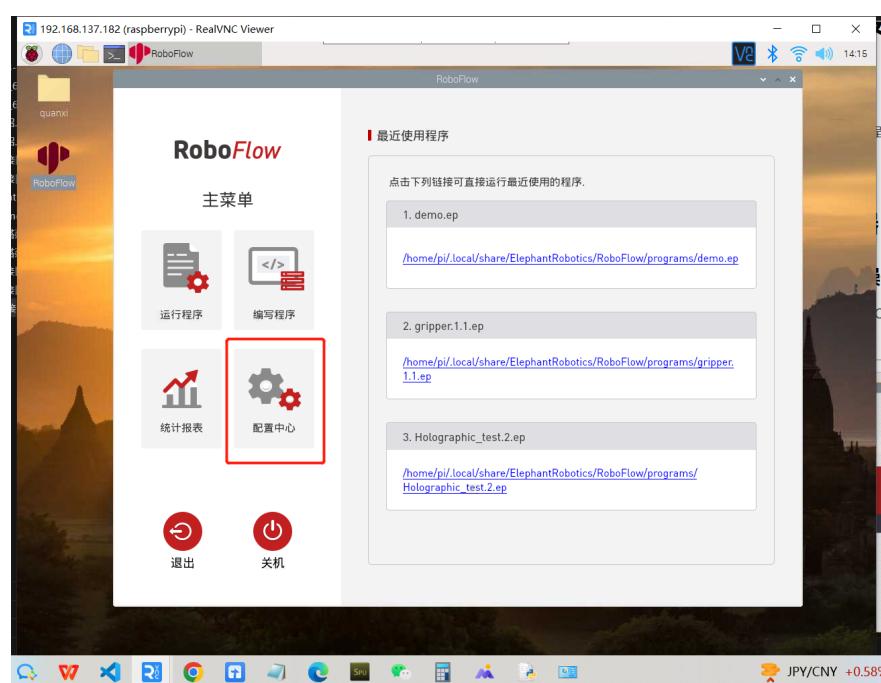
1.1 登录RoboFlow操作系统

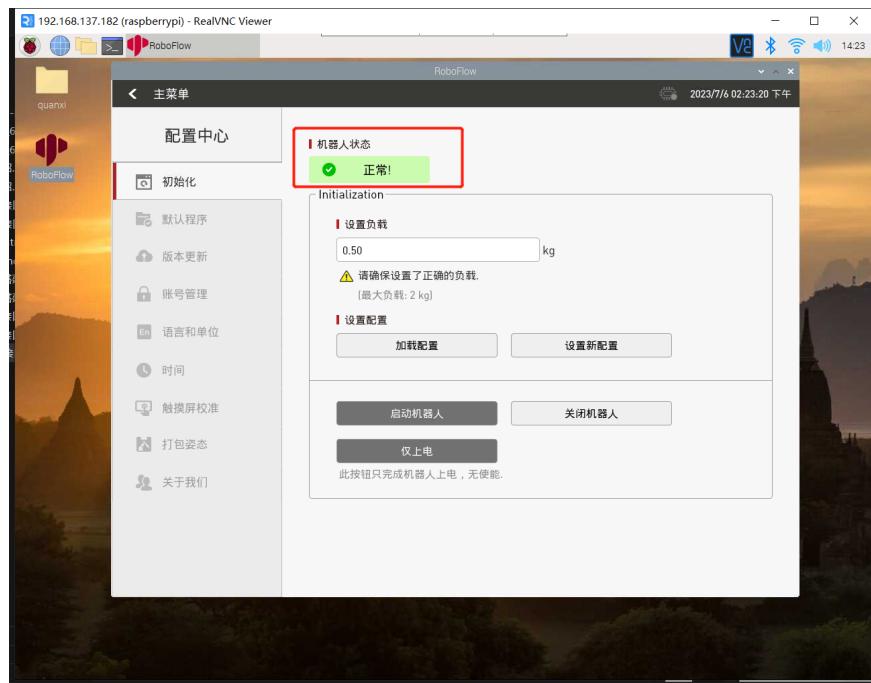
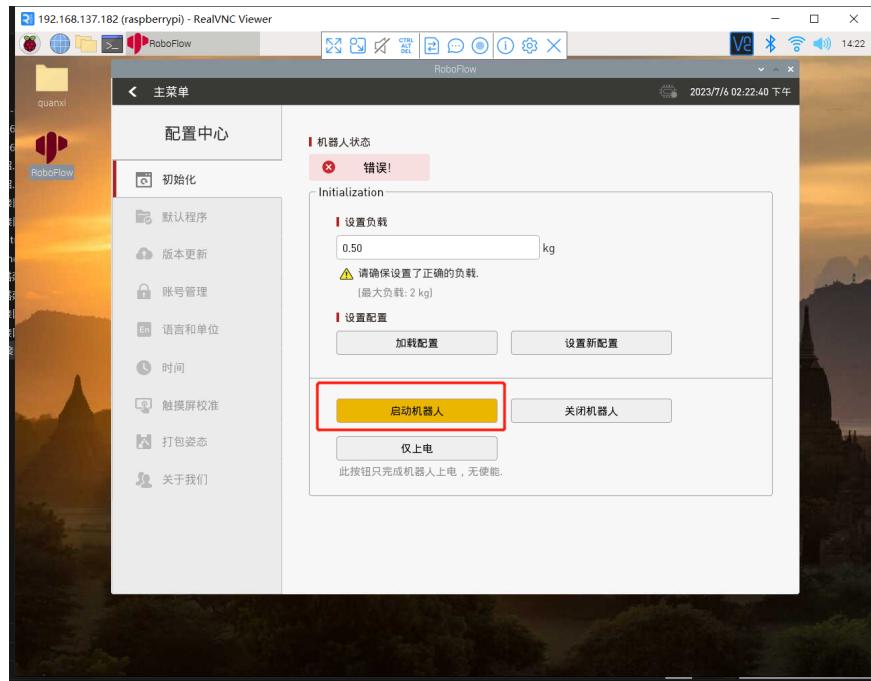
机器人上电开机后，使用VNC Viewer进入树莓派，登录RoboFlow操作系统



1.2 启动机器人

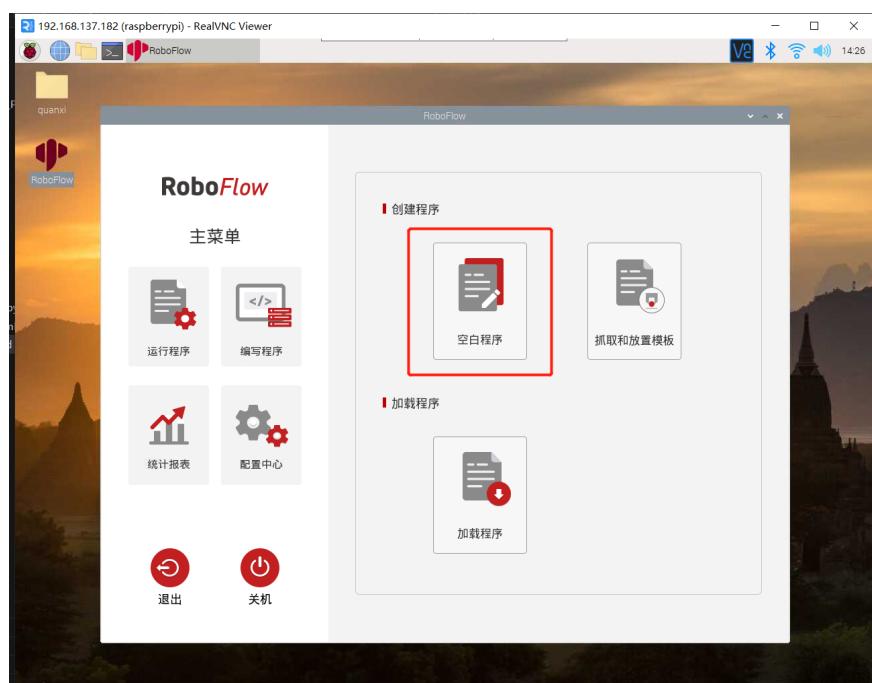
进入配置中心，点击启动机器人按钮

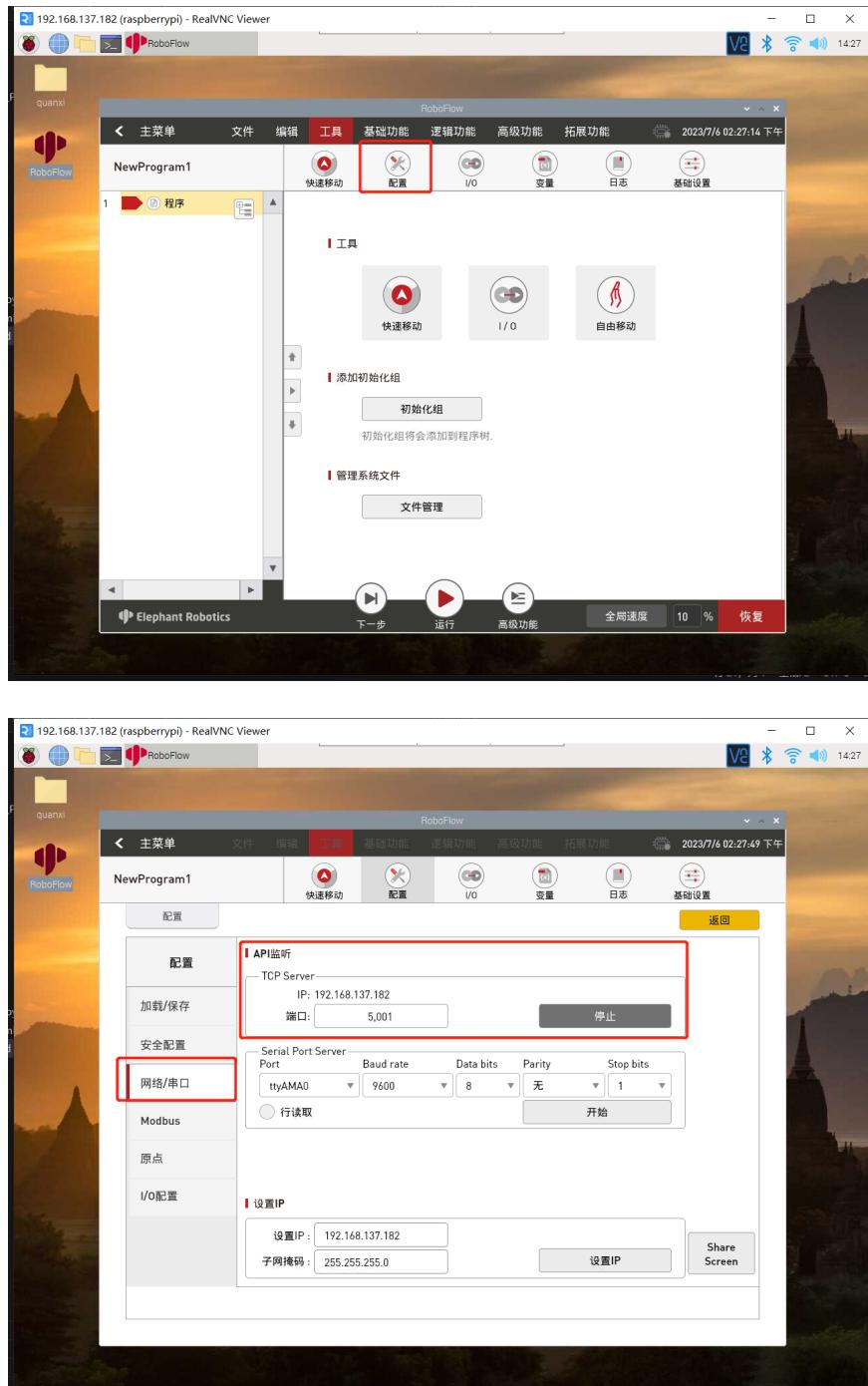




1.3 检查TCP服务器是否开启

返回主菜单，点击编写程序后，再点击空白程序，进入程序编辑界面后，点击配置按钮，点击网络/串口选项，检查TCP服务器是否开启，通常情况下，**TCP服务器是默认开启的**，若未开启，则需手动开启，也手动设置IP地址，如设置成**192.168.1.159**。





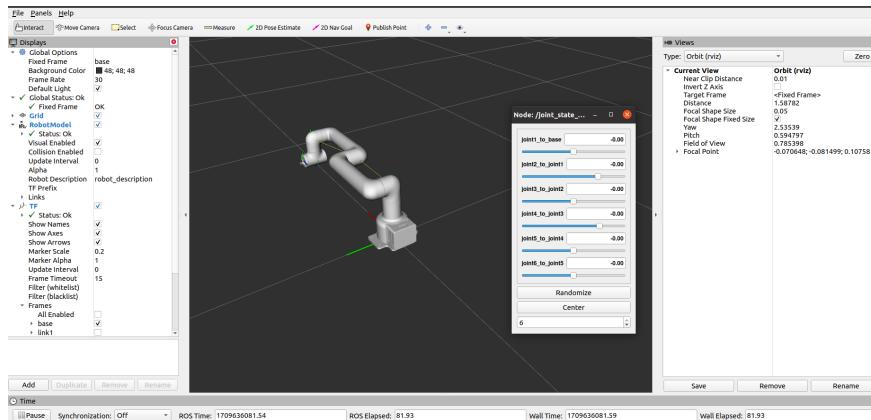
2 滑块控制

注意：TCP通信需确保本地虚拟机电脑与MyCobot Pro630系统使用同一网络，同一网段。

打开一个控制台终端，运行命令：

```
# MyCobot Pro 630默认的Socket IP地址为192.168.1.159，端口号为5001，若不一致，可根据实际的I
ros2 launch mycobot_630 slider_control.launch.py ip:=192.168.1.159 port:=5001
```

它将打开 rviz2 和一个滑块组件，你将看到如下画面：



接着你可以通过拖动滑块来控制 rviz 中的模型移动。真实的 MyCobot Pr630 将跟着一起运动。

请注意：由于在命令输入的同时机械臂会移动到模型目前的位置，在您使用命令之前请确保 rviz 中的模型没有出现穿模现象。不要在连接机械臂后做出快速拖动滑块的行为，防止机械臂损坏。

[← 上一页](#) | [下一节 →](#)

105

106

107

108

109

3D无序抓取套件



1 套件硬件介绍

1.1 MyCobot Pro630机械臂



1.2 图漾 FS820-E1



2 图漾 FS820-E1基本使用说明

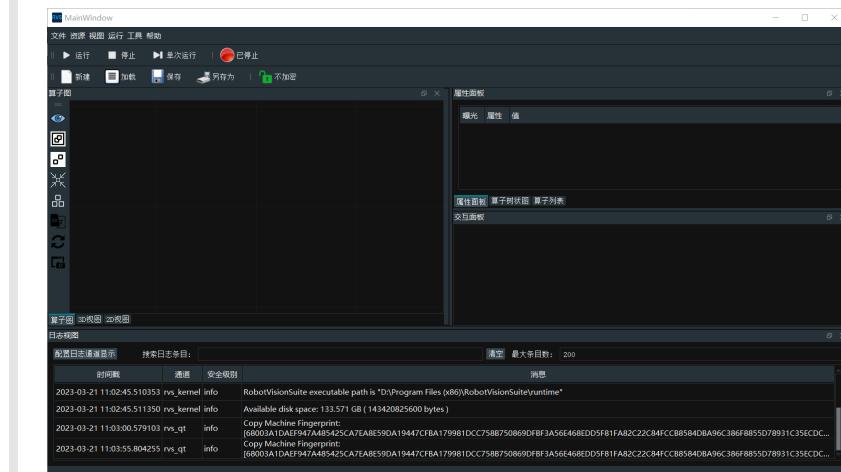
2.1 摄像头软件环境搭建

图漾提供两种开发者模式，SDK和RVS模式，这里使用的是RVS进行开发，可以利用RVS中的视觉算子写成节点(Node)快速搭建抓取功能。根据[安装手册](#)完成RVS安装。

官网提供两种版本，两种版本只在AI算子上有差别，FULL版本就是使用GPU进行AI训练，其他一样，套件选择FULL版本 CPU 版文件名为：[RobotVisionSuite-WIN-XX-CPU.zip](#) FULL 版文件名为：[RobotVisionSuite-WIN-XX-GPU.zip](#)

注意：

- 安装后需要复制机器码发给官网邮箱或询问相关人员，得到激活码
创建license.txt放在license文件夹里进行激活
- 安装路径选择在(D:\)方便算子路径读取文件



2.2 摄像头硬件连接

在官网查找相机供电方式，这里使用[FS820-E1相机](#)型号，使用**DC 12V~24V**供电。连接方式[参考这里](#)。这里使用方式**1**

网络连接方式1

通过千兆以太网线缆直接将相机接入到计算机千兆以太网接口。

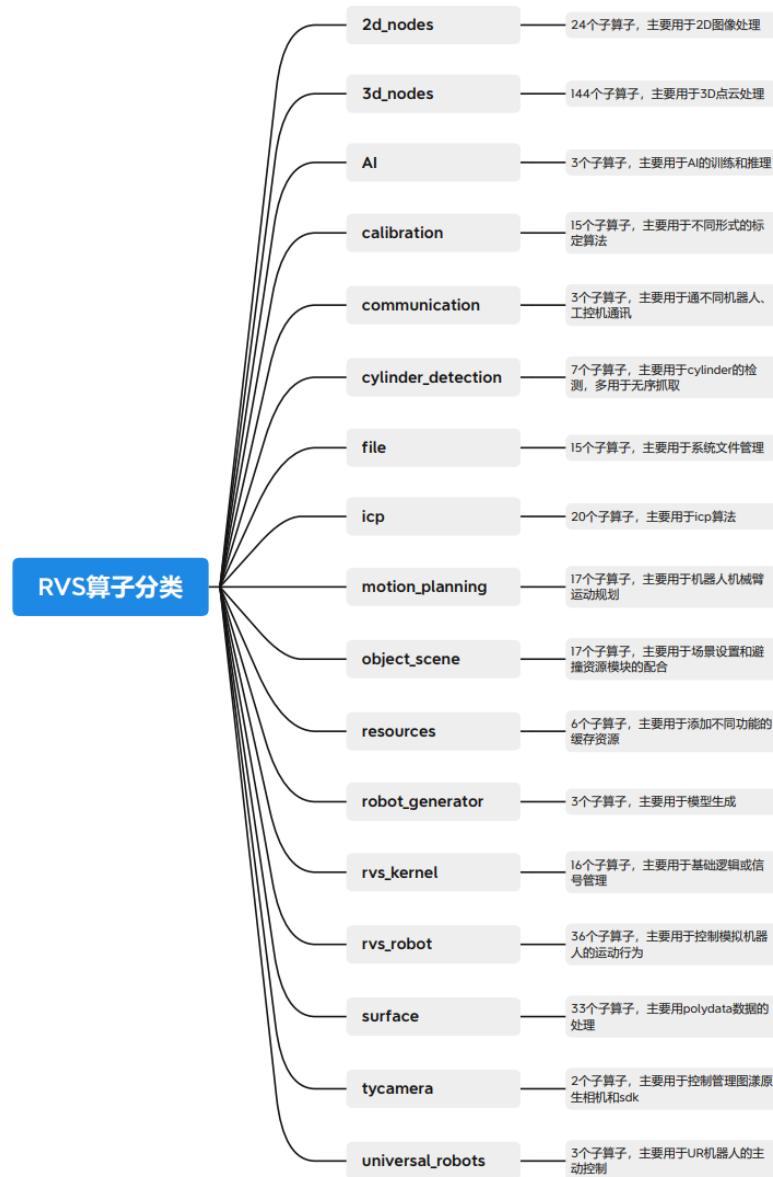
相机上电启动后约1分钟，计算机和相机可以成功协商获得 169.254.xx.xx 网段的 IP 地址。

通过 SDK 示例程序 ListDevices 确认相机是否已经获得 IP 地址和设备号后，然后运行 SimpleView_FetchFrame.exe -id <设备号> 查看图像。设备号可从设备标签上获得，也可从枚举结果中获得。



2.3 算子总览

详情[参考这里](#)



2.4 图漾相机数据实时采集

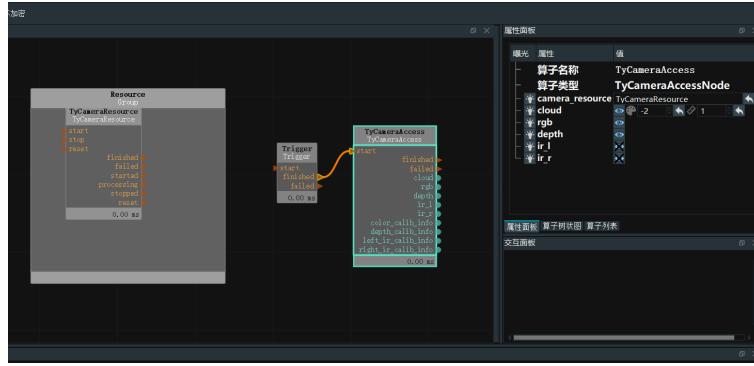
在左上角窗口资源中，找到TyCameraResource 算子添加到算子图中的 ResourceGroup 中，在算子列表搜索 **TyCameraAccess,trigger** 算子添加到算子图中，并根据需要调整算子参数。然后点击运行和属性面板 **Trigger->ture** 即可查看可视化数据。没有报错能正常显示即可进行下一步。

TyCameraResource 算子

- start 以及 stop 分别用于开启、关闭资源算子的线程。auto_start 也是用于开启资源算子，如果勾选，则仅在打开 RVS 软件后第一次进入运行状态时自动开启资源线程。
- reset：在打开资源线程后如果需要更改属性参数，需要选中该选项进行重置。

TyCameraAccess 算子

- 打开cloud、rgb、depth可视化属性，将 cloud_color设置为-2，代表真实颜色



3 手眼标定

准备棋盘格，算好棋盘格行列数，以及棋盘格边长(**mm**) 注意：每一次的手眼标定开始前需要把RVS安装目录下的runtime\handeye_data 文件夹里面的文件全部清空

3.1 数据录制

第一步：点击左上角加载，打开 unstacking_runtime/HandEyeCalibration/HandEyeCalibration.xml，根据下图标注操作



第二步：启动机械臂

在终端运行 HandInEyeCalib.py 文件

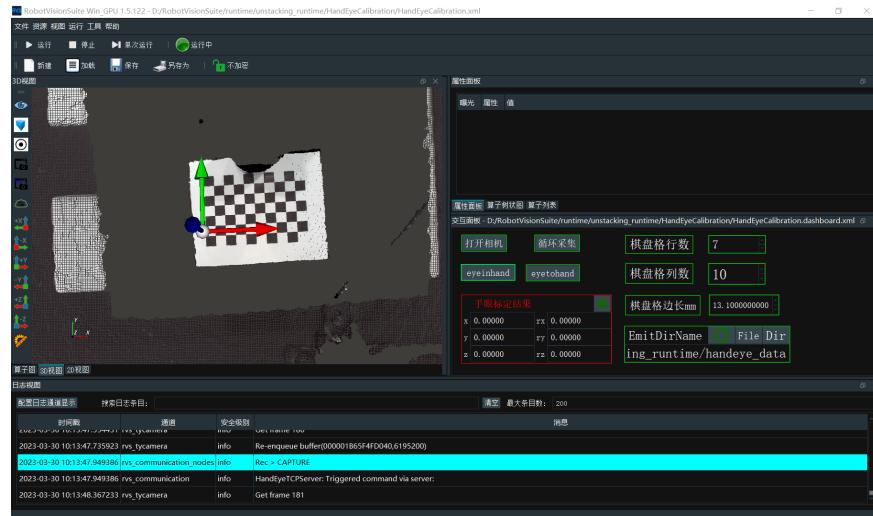
```

283     robot_speed = 10
284     robot_speed_fast = 50
285
286     erobot = eclient.robota('192.168.11.121', 5902) #连接proc680机械臂
287     erobot.start_client()
288
289     consSuc_rvs, sock_rvs=connectRvsServer(rvs_ip, rvs_port)#连接RVS
290
291
292     while exec_index < len(point):
293         exec_index = exec_index + 1
294         print("第 %d 次执行" % exec_index)
295         print("成功到拍照位")
296
297         erobot.write_angles(point[exec_index-1], 1000)
298         erobot.command_wait_done()
299         time.sleep(3)
300
301
302         get_send_TCP(sock_rvs)#获取机器人当前位置TCP并发送给RVS
303         #get_send_jpoint(sock_rvs)#获取机器人当前位置关节值并发送给RVS
304
305         #@CAPTURE
306         print("*****%s*****" % time.ctime())
307         capture_cmd = "CAPTURE #"
308         capture_bytes=bytes(capture_cmd,encoding='utf-8')
309         erobot.send_command(capture_bytes)
310
311
312     PS C:\Users\Elephant\Desktop\PVC_3D>

```

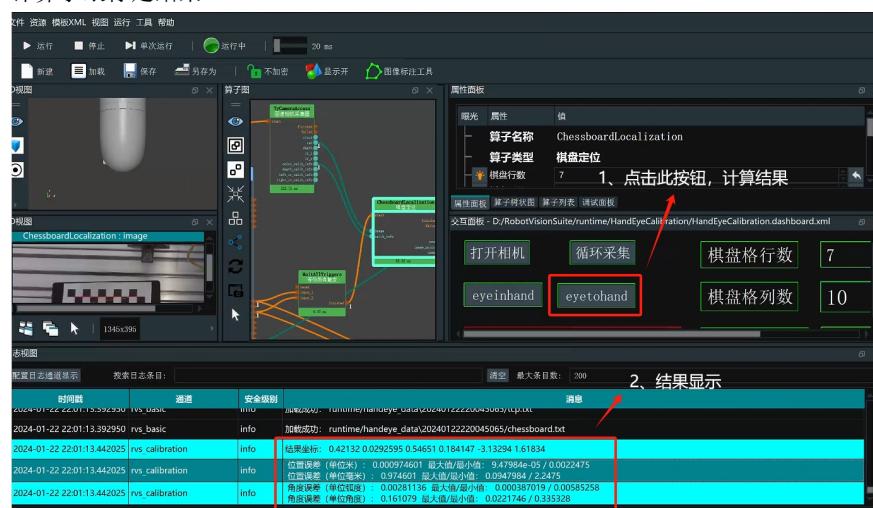
第三步：启动机械臂脚本后，就会开始自动标定

标定前确保相机能完整识别完整的棋盘格，以及标定过程中，棋盘格是固定的，不能发生移动。运行完成会得到20组数据。

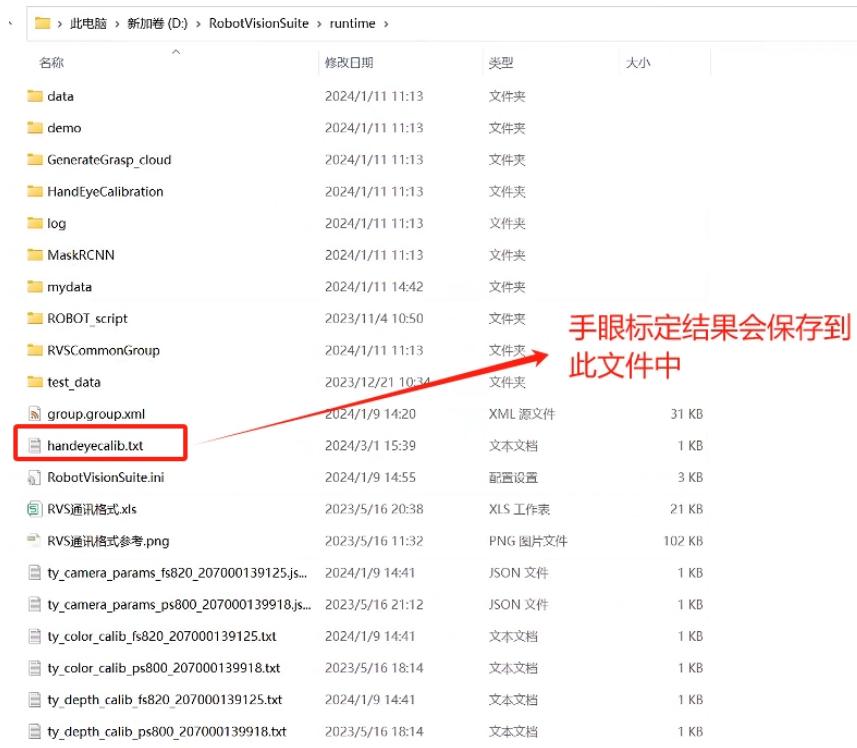


3.2 计算标定结果

计算手眼标定结果



位置误差在 0.005 (5毫米) 以内，则比较理想；如果误差很大，请检查前述步骤是否正确执行



| 名称 | 修改日期 | 类型 | 大小 |
|---|------------------|----------|--------|
| data | 2024/1/11 11:13 | 文件夹 | |
| demo | 2024/1/11 11:13 | 文件夹 | |
| GenerateGrasp_cloud | 2024/1/11 11:13 | 文件夹 | |
| HandEyeCalibration | 2024/1/11 11:13 | 文件夹 | |
| log | 2024/1/11 11:13 | 文件夹 | |
| MaskRCNN | 2024/1/11 11:13 | 文件夹 | |
| mydata | 2024/1/11 14:42 | 文件夹 | |
| ROBOT_script | 2023/11/4 10:50 | 文件夹 | |
| RVSCommonGroup | 2024/1/11 11:13 | 文件夹 | |
| test_data | 2023/12/21 10:34 | 文件夹 | |
| group.group.xml | 2024/1/9 14:20 | XML 源文件 | 31 KB |
| handeyecalib.txt | 2024/3/1 15:39 | 文本文档 | 1 KB |
| RobotVisionSuite.ini | 2024/1/9 14:55 | 配置设置 | 3 KB |
| RVS通讯格式.xls | 2023/5/16 20:38 | XLS 工作表 | 21 KB |
| RVS通讯格式参考.png | 2023/5/16 11:32 | PNG 图片文件 | 102 KB |
| ty_camera_params_f820_207000139125.js... | 2024/1/9 14:41 | JSON 文件 | 1 KB |
| ty_camera_params_ps800_207000139918.js... | 2023/5/16 21:12 | JSON 文件 | 1 KB |
| ty_color_calib_fs820_207000139125.txt | 2024/1/9 14:41 | 文本文档 | 1 KB |
| ty_color_calib_ps800_207000139918.txt | 2023/5/16 18:14 | 文本文档 | 1 KB |
| ty_depth_calib_fs820_207000139125.txt | 2024/1/9 14:41 | 文本文档 | 1 KB |
| ty_depth_calib_ps800_207000139918.txt | 2023/5/16 18:14 | 文本文档 | 1 KB |

4 模型训练

注意：模型已训练好，客户直接使用，无需再训练，直接阅读案列运行章节即可。若客户想自己训练，可以把安装目录下的runtime/MaskRCNN/train data的文件夹里面的文件全部删除即可，根据下面内容操作。

4.1 采集图像

打开RVS加载unstacking_runtime/MaskRCNN/ty_ai_savedata.xml文件

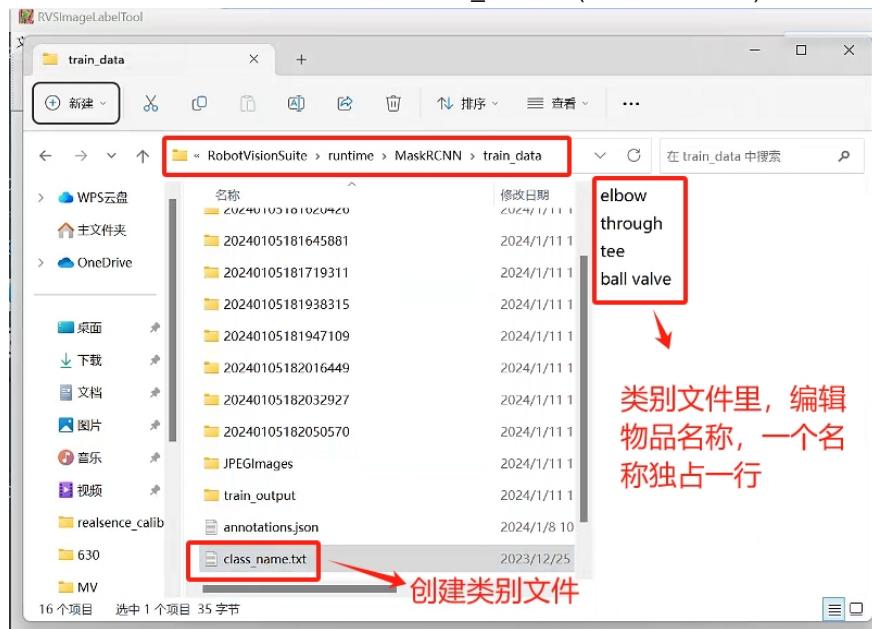


在图像采集时，我们应注意以下几点：

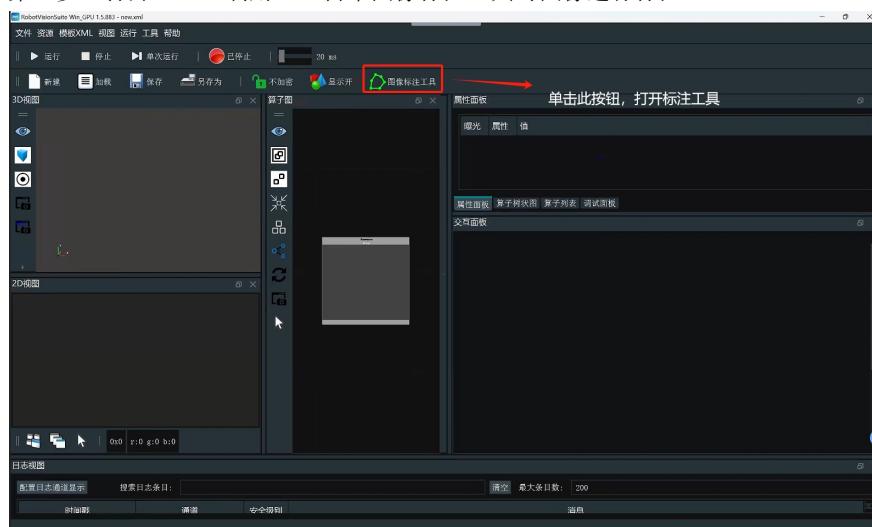
- 背景光照
 - 单一稳定光源，亮暗适宜且没有过多反光
 - 户外光照情况变化过大，不建议
- 复杂工况考虑
 - 尽量使得采样样本对实际运行的全局样本有足够的代表性而不仅仅是全局样本的一种特例
- 图像质量
 - 人眼需清晰可见目标边缘，尤其是距离相机最远的那层目标，否则考虑更换相机具体图像录制可以参考unstacking_runtime/MaskRCNN/train_data中的rgb图像

4.2 图像标注

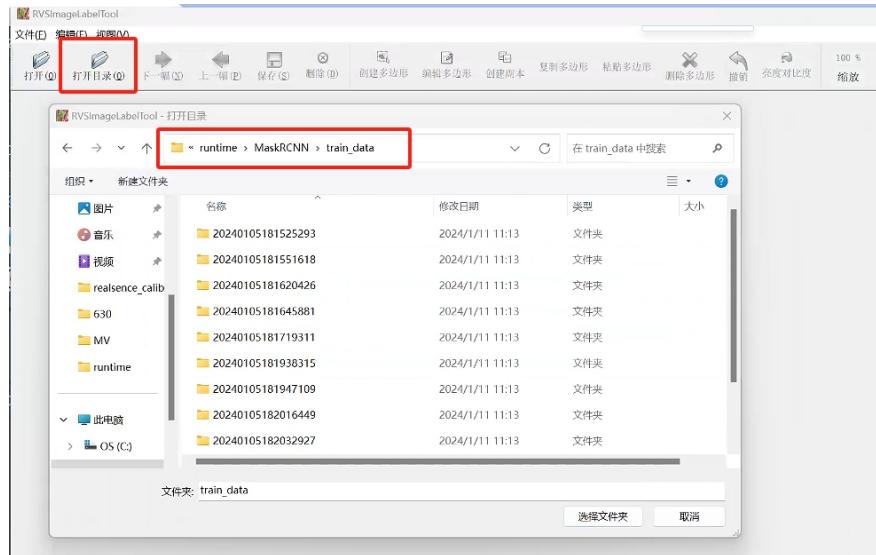
第一步：在需要标注图片的文件夹新建class_name.txt(文件名不能更改)



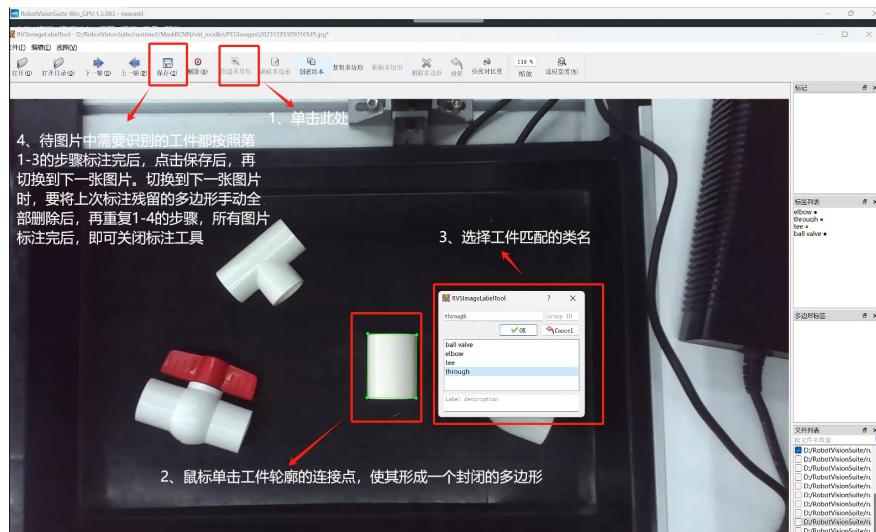
第二步：打开RVS，利用RVS自带图像标注工具对图像进行标注



第三步：点击打开目录，选择要标注的图片的所在目录



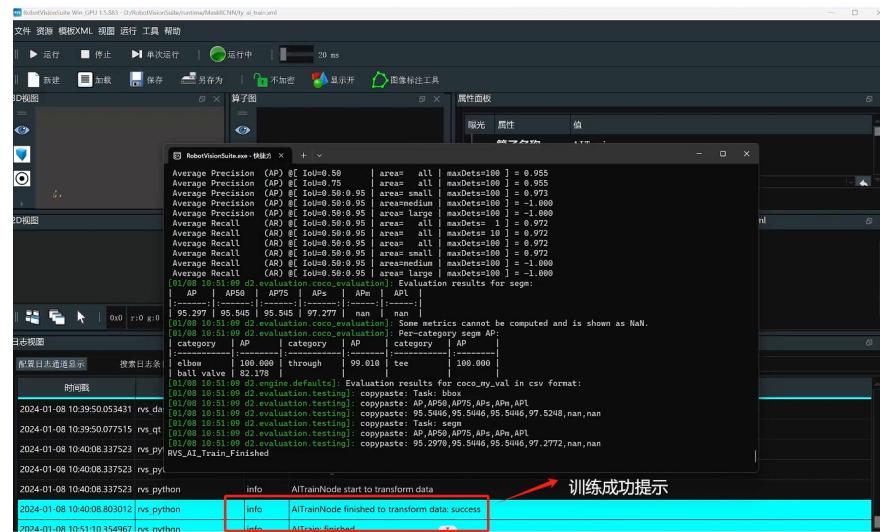
第四步：标注图片



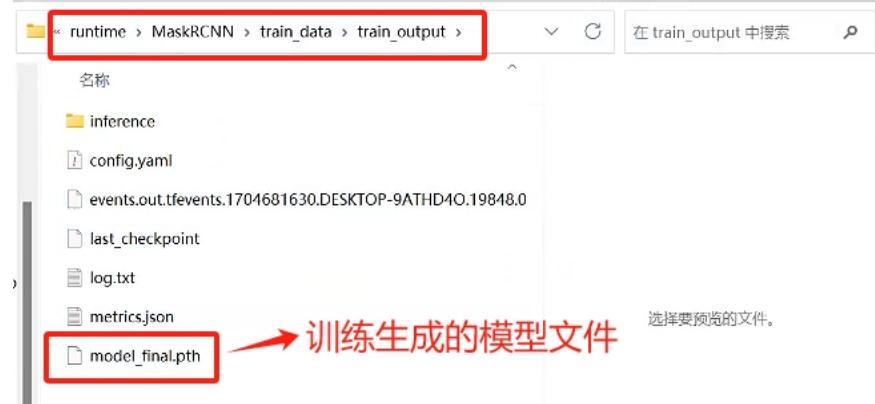
4.3 数据训练

第一步：加载unstacking_runtime/MaskRCNN/ty_ai_train.xml工程，进行数据训练





第二步：训练完成后，会生成一个train_output文件夹，里面会包含一个模型权重文件



5 案例运行

第一步：打开unstacking_runtime/demo/demo.xml工程后，检查相机标定文件路径是否正常，保持默认即可



第二步：检查手眼标定文件路径是否正常，保持默认即可



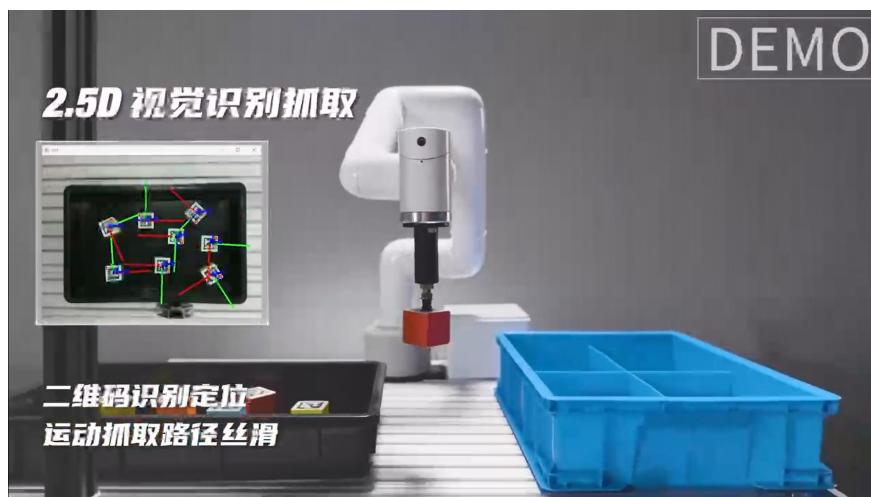
第三步：打开计算抓取点算子



第四步：检查AI推理算子的类名文件、权重文件、配置文件是否正常；根据实际情况调节得分阈值，上述配置保持默认即可



2.5D二维码分拣套件



1 套件硬件介绍

1.1 MyCobot Pro630机械臂



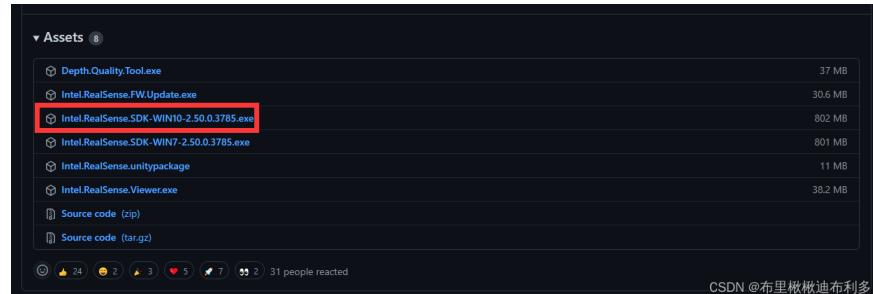
1.1 RealSense D435



2 realsense软件环境搭建

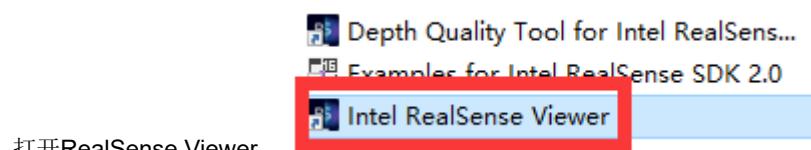
2.1 下载RealSense SDK 2.0

进入网址



下载完成后文件夹内有如下图所示软件，直接安装即可。

2.2 测试RealSense D435



打开RealSense Viewer。

将赠送的usb3.0线连接到电脑上，一定要是usb3.0的插口！！！如果是3.0插口和线，那么在Viewer界面左侧就如下图所示。接下来只要将Stereo Module和RGB Camera的off设置成on即可。 将off设置成on后，就会出现如下图所示界面。测试成功后，即可以关闭该软件

3 Python 环境搭建

3.1 下载python

下载地址



选择你要安装的版本,推荐安装3.7以上版本

- Download Windows x86 embeddable zip file
- Download Windows x86 executable installer
- Download Windows x86 web-based installer
- Python 3.5.10 - Sept. 5, 2020

Note that Python 3.5.10 cannot be used on Windows XP or earlier.

- No files for this release.

▪ Python 3.7.9 - Aug. 17, 2020 找到你需要下载和安装的版本

Note that Python 3.7.9 cannot be used on Windows XP or earlier.

- Download Windows help file
- Download Windows x86-64 embeddable zip file
- Download Windows x86-64 executable installer 电脑是64位的,
下载 *.exe (可执行文件)
- Download Windows x86-64 web-based installer 这个需要联网才能完成安装
- Download Windows x86 embeddable zip file
- Download Windows x86 executable installer
- Download Windows x86 web-based installer

CSDN @weixin_74

名称

 python-3.7.9-amd64.exe

 未确认 989784.crdownload

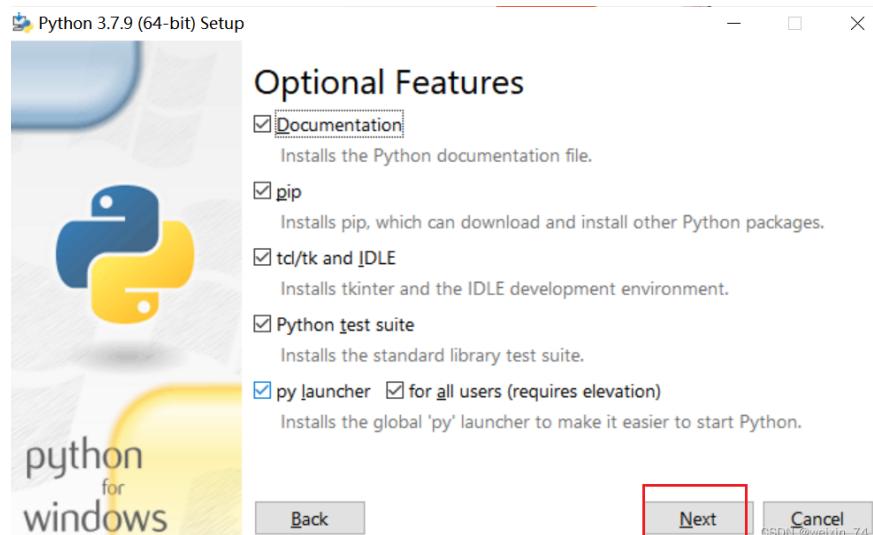
 副.pdf

CSDN @weixin_74

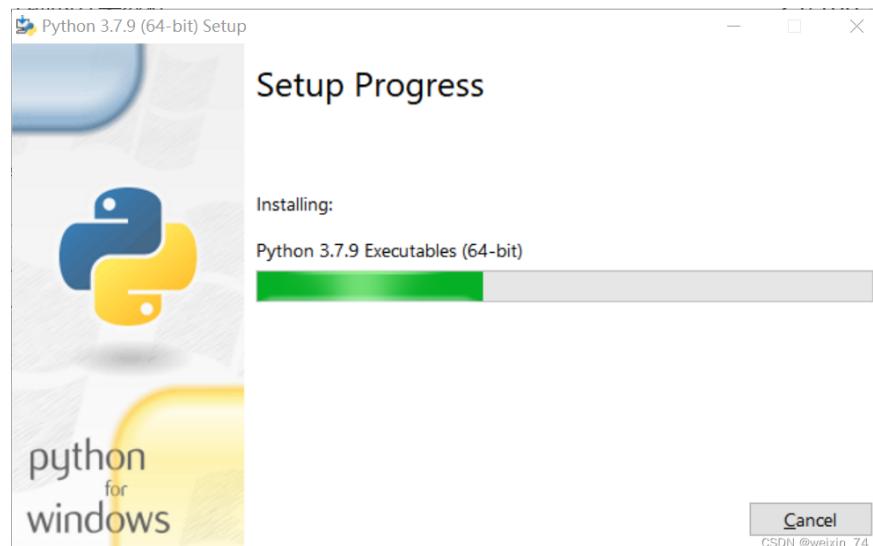
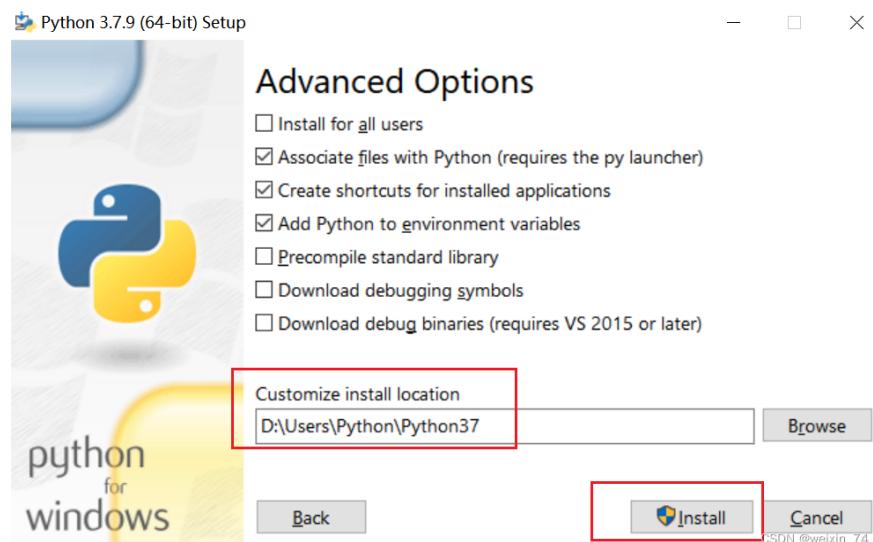
双击安装包后

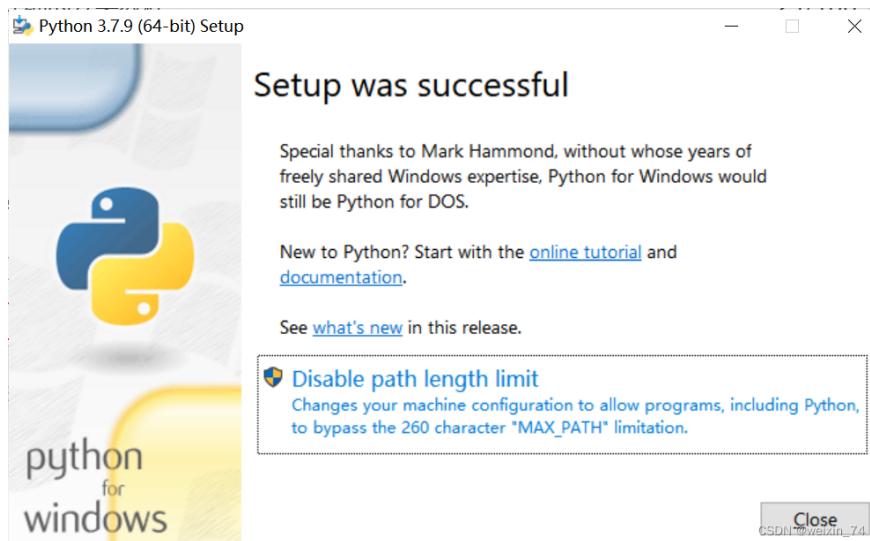


保持默认配置，直接点击**Next**

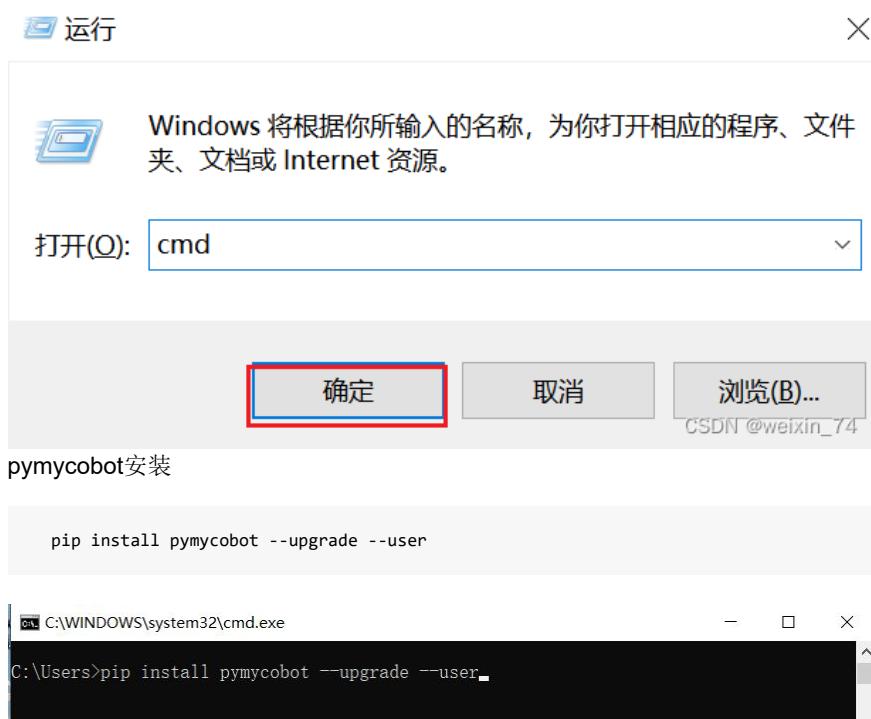


若选择自定义软件安装位置后，点击**Install**（建议安装的文件路径是全英路径，因为有些软件安装遇到中文会打不开）

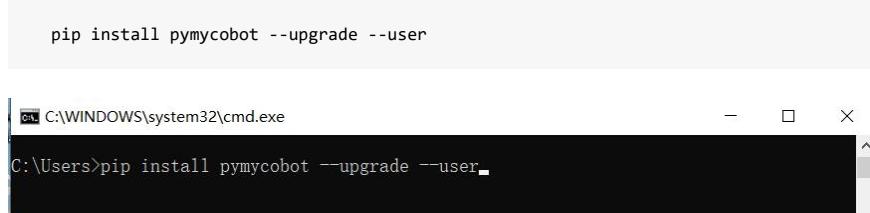




安装完成后，可以完毕当前页面，然后按键盘的 Windows键 + r键，并输入 cmd



pymycobot安装



opencv安装



四

127

11

11

130

131

关于我们



本章主要提

供有关公司简介和发展历史的信息，为用户提供团队和机器人的额外背景。“联系我们”部分提供了详细的联系信息，包括电子邮件格式、描述问题的说明和复制步骤的模板。这使用户能够与团队进行有效沟通并解决问题。总的来说，本章的目标是在展示公司和团队信息的同时，与用户建立高效的沟通渠道。

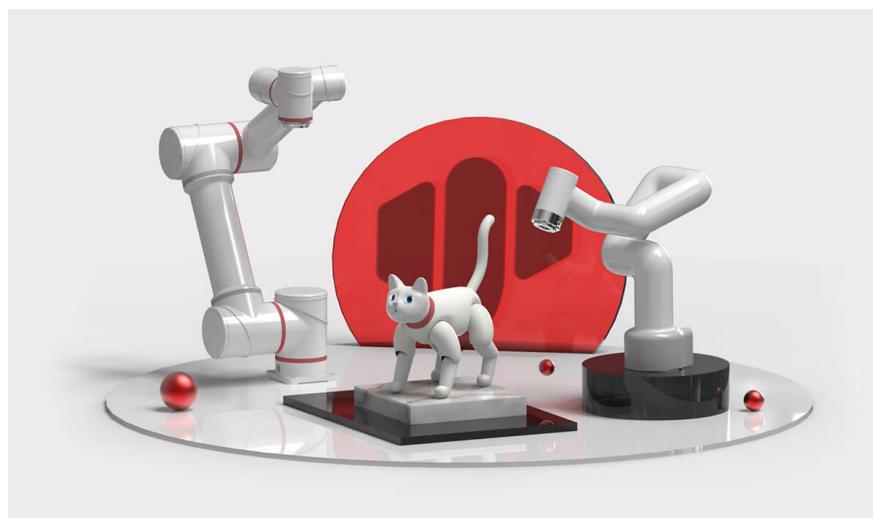
[← 上一页](#) | [下一页 →](#)

大象机器人



1 公司简介

大象机器人(Elephant Robotics)立足于中国·深圳，是一家专注于机器人研发设计及自动化解决方案的高新科技企业。我们致力于为机器人教育及科研机构、商业场景、工业生产提供高柔性的协作机器人、简单易学的操作系统以及智能的自动化解决方案。其产品质量及智慧方案备受韩国、日本、美国、德国、意大利、希腊等数家来自世界500强名企工厂的一致认可与好评。大象机器人秉持“Enjoy Robots World”的愿景，倡导人与机器人的协同工作，让机器人成为人类工作生活的好帮手，帮助人们从简单、重复、枯燥的工作中解放出来，充分发挥人机协同优势，进而提高工作效率，帮助人类缔造美好新生活。未来，大象机器人希望通过新一代尖端科技推动机器人产业发展，携手与客户伙伴们共同开启自动化智能化新时代。



2 发展历程

2016.08 -----大象机器人有限公司正式成立 2016.08 -----进入 HAX 孵化器，获得 SOSV 种子轮投资 2016.08 ----- 开始研发 Elephant S 工业协作机器人 2017.01 ----- 荣获“CES 中国最具创新企业 Top10” 2017.04 -----出席汉诺威工业博览会及韩国

自动化展览会 2017.07 -----两位创始人入选福布斯亚洲评选的“30 位 30 岁以下商业精英” 2017.10 -----第五代单臂工业协作机器人 Elephant S 问世 2018.04 -----获得“云天使基金”天使轮投资 2018.06 -----首次公开亮相 2018 年汉诺威世界工业博览会 2018.06 -----获得长江商学院“智造创业 MBA 奖” 2018.06 -----获得清华经管“创业加速器 X-elerator 奖” 2018.11 -----获得亚洲智能硬件大赛深圳赛区第二名 2018.11 -----获得高工金球奖“最具投资企业奖” 2019.03 -----获得高工金球奖“领军人物奖” 2019.04 -----2019年3月 Catbot获“工业机器人创新奖” 2019.09 -----出席华为欧洲生态大会(HCE)，正式成为华为生态伙伴一员 2019.11 -----大象机器人携手哈工大出席IROS国际智能机器人与系统大会 2019.12 -----大象机器人-华南理工大学“智能机器人联合开发实验室”正式揭牌 2019.12 -----荣获高工2019年度“创新技术奖” 2019.12 -----荣获高工2019年度“十大快速成长企业” 2019.12 -----荣获深圳装备工业-工业机器人细分领域-“新锐企业奖” 2019.12 -----世界首款仿生机器猫MarsCat问世 2020.05 -----创始人获得2019年度深圳市机器人新锐人物奖 2020.10 -----全球最轻最小的六轴协作机器人myCobot问世 2021.03 -----面向科研的最小协作机器人 myCobotPro 320问世 2021.05 -----火星仿生猫MarsCat获得新华财经、中国日报、南京日报、哈尔滨日报等多家媒体的竞相报道 2021.07 -----发布最小的复合机器人底盘 - 小象移动机器人myAGV 2021.09 -----全球首款全包裹式的四轴机械臂-小象码垛机械臂myPalletizer问世

3 相关链接

- 官网: <https://www.elephantrobotics.com>
- 购买链接
 - 淘宝: <https://shop504055678.taobao.com>
 - shopify: <https://shop.elephantrobotics.com/>
- 视频
 - bilibili: <https://space.bilibili.com/2126215657>
 - youtube: <https://www.youtube.com/c/Elephantrobotics>

← 上一页 | 下一页 →

9.2 联系我们

如果您还有其他问题，可以通过以下方式联系我们

- 邮箱 E-mail : sales@elephantrobotics.com

我们会在1~2个工作日内回复



- 微信

我们只对已经购买过myCobot的用户进行微信1对1服务。

← 上一页 | 下一页 →