

- [cobot\\_magic](#)详见使用说明文档

- [1 上电配置](#)

- [1.1 工控机上电配置](#)
  - [1.2 机械臂上电配置](#)
  - [1.3 相机上电配置](#)

- [2 软件环境配置](#)

- [2.1 相机配置](#)
  - [2.2 机械臂配置](#)
    - [2.2.1 环境安装](#)
    - [2.2.2 绑定机械臂can口](#)
    - [2.2.3 启动机械臂](#)

- [3 数据采集](#)

- [3.1 环境依赖](#)
  - [3.2 运行](#)
    - [3.2.1 采集数据](#)
    - [3.2.2 可视化数据集](#)
    - [3.2.3 重播数据集](#)

- [4 ACT训练推理](#)

- [4.1 环境配置](#)
  - [4.2 数据集采集](#)
  - [4.3 训练](#)
  - [4.4 推理](#)

- [5 Q&A](#)

# cobot\_magic详见使用说明文档

---

[Cobot\\_Magic-github地址](#)

---

- **jetson-orin-nano:** 只支持机械臂遥操作、采集数据、不支持模型训练、不支持模型推理
  - 工控机(**4090显卡版,4060显卡版**): 支持机械臂遥操作、采集数据、支持模型训练、支持模型推理
  - ACT模型训练建议使用高算力大显存服务器, 4060显卡训练 50 episode、500 timesteps的数据集,bitch-size给4都会出现显存不足。4060可用于推理
  - 遇到问题请先参考第[5\(Q&A\)](#)章节
-

# 1 上电配置

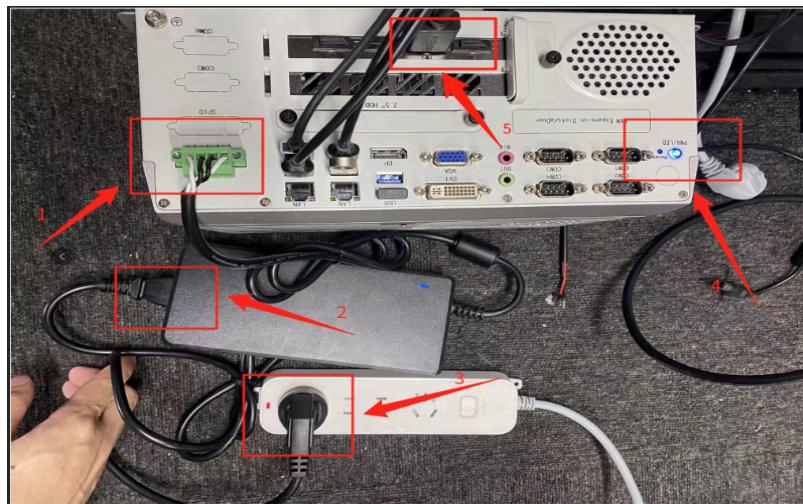
## 1.1 工控机上电配置

### 1. 工控机上电

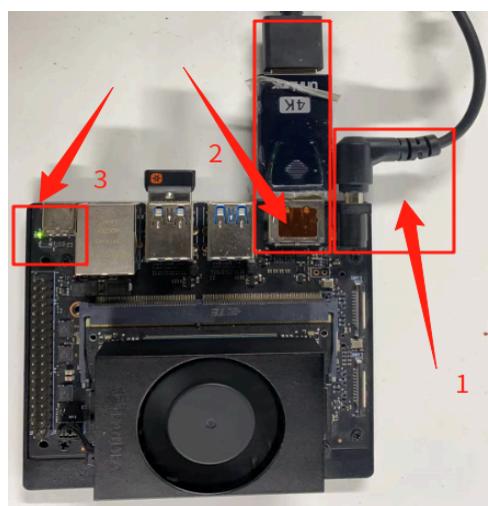
工控机开机密码为: **agx**

路由器WiFi名称: **aloha** 密码: **12345678**

- 工控机上电如下图所示, 按图中箭头1、2、3接通电源, 通电后工控机会自动开机, 箭头4处开机指示灯会亮起即可。
- 如果电脑关机后未断开电源, 需要重启工控机按图中箭头4处开机按钮即可。
- 显示器输出HDMI(DP)线请插入显卡上的HDMI或者DP接口处, 如图中5所示。



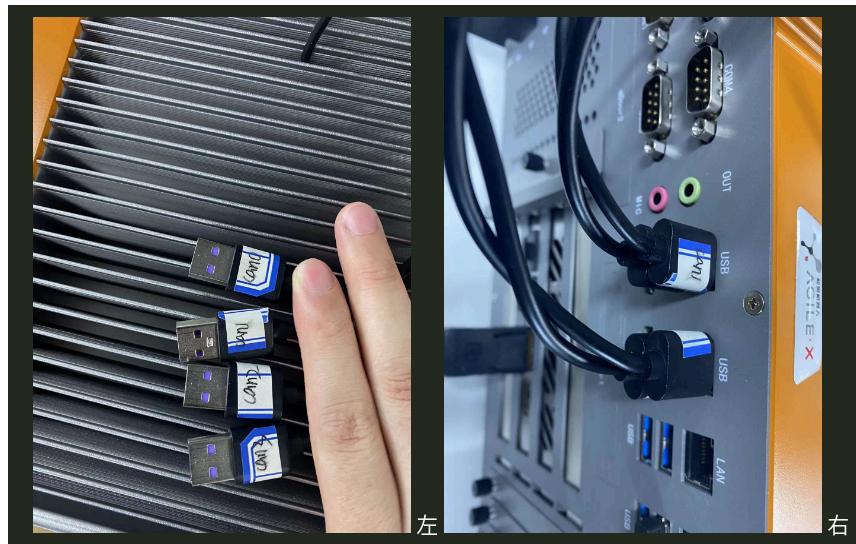
### 2. jetson-orin-nano上电



如上图所示,2处连接DP线供显示器输出, 1处接通电源,orin-nano会自动开机,3处开机指示灯会亮起即可

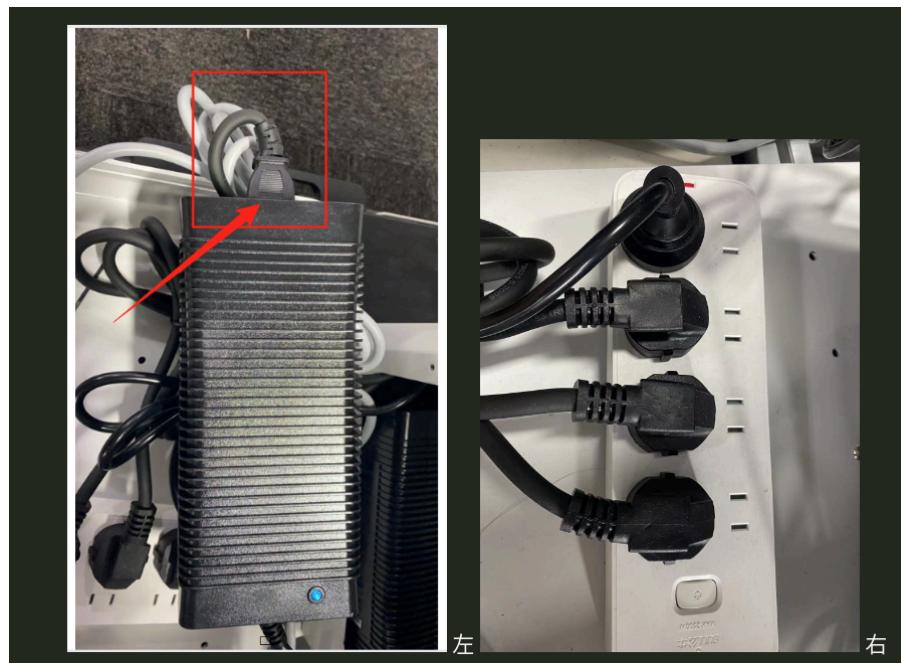
## 1.2 机械臂上电配置

### 1. 机械臂can2usb连线



如上图所示,左图分别是4条机械臂的4条can2usb线,按右图所示分别接入工控机的usb口处(或者usb拓展坞上)即可。

### 2. 机械臂上电



如上图所示,分别按左图所示,接上适配器的插头;然后按右图所示,4条机械臂的适配器插头接上电源即可。

接通电源后,如下图机械臂有4处指示灯亮即表示通电成功,如果不是4处指示灯亮,请检查机械臂的连线。



## 1.3 相机上电配置

### 1. 相机usb连线



如上图所示,左图分别是3个相机的3条usb线,按右图所示分别接入工控机的usb口处(或者usb拓展坞上)即可。

### 2. 相机上电

相机供电由usb供电,无需额外供电,连接好usb线即可。

## 2 软件环境配置

- cobot\_magic的工控机(4060显卡、4090显卡版本),镜像自带ubuntu-20.04、ros1-noetic、cuda-11.8、torch-2.1.1、conda、python-3.8, 镜像下载地址
- jetson-orin-nano板载,镜像自带ubuntu-20.04、ros1-noetic

1. 数据采集基础配置: ubuntu-20.04、ros1-noetic
  2. 模型训练推理配置: 1. ubuntu-20.04、ros1-noetic、cuda-11.3、torch-1.11(cuda-11.8、torch-2.1.1), conda,python-3.8, 已测试通过, cuda、torch其他版本请用户自己测试
- 

- 下载cobot\_magic源码

```
# 1 下载源码  
git clone https://github.com/agilexrobotics/cobot_magic.git  
  
# 2 进入cobot_magic工作目录  
cd cobot_magic
```

## 2.1 相机配置

---

- 相机参数

Baseline	40mm
深度距离	0.3-3m
深度图分辨率	640x400x30fps、320x200x30fps
彩色图分辨率	1920x1080x30fps、1280x720x30fps、640x480x30fps
精度	6mm@1m (81%FOV区域参与精度计算)
深度FOV	H 67.9° V 45.3°
彩色FOV	H 71° V 43.7° @ 1920x1080
延迟	30-45ms
数据传输	USB2.0或以上
工作温度	10°C~40°C
尺寸	长59.5x宽17.4x厚11.1 mm

---

### 1. 编译相机驱动

```

# 1 进入ros_astra_camera目录
cd camera_ws

# 2 安装依赖
sudo apt install libgflags-dev libgoogle-glog-dev libusb-1.0-0-dev libeigen3-dev
ros-$ROS_DISTRO-image-geometry ros-$ROS_DISTRO-camera-info-manager ros-$ROS_DISTRO-
image-transport ros-$ROS_DISTRO-image-publisher ros-$ROS_DISTRO-libuvc-ros -y

# 3 编译
catkin_make

# 4 Install udev rules.
source devel/setup.bash && rospack list
roscd astra_camera
./scripts/create_udev_rules
sudo udevadm control --reload && sudo udevadm trigger

# 5 增加永久ros_astra_camera包的环境变量
## 5.1 进入astra_ws目录
cd ../..
## 5.2 永久增加环境变量
echo "source $(pwd)/devel/setup.bash" >> ~/.bashrc
## 5.3 环境变量生效
source ~/.bashrc

```

## 2. 运行

工控机接上[astra](#)相机的usb

```

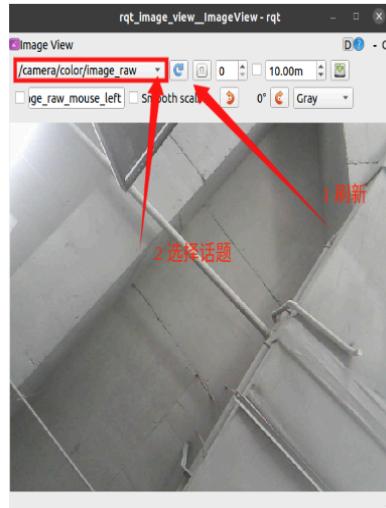
# 1 检测相机序列号
rosrun astra_camera list_devices_node
# 终端显示
## 1.1 显示如下： 表示相机未连接,请重新拔插usb
# [ INFO] [1710298393.524932942]: Found 0 devices

## 1.2 显示如下： 一个Serial number,即表示相机已经成功连接
# [ INFO] [1710298585.382430351]: Found 1 devices
# Device connected: Astra
# URI: 2bc5/0657@1/11
# Serial number: CC1GA35002L

# 2 启动相机
roslaunch astra_camera dabai.launch

# 3 打开rqt_image_view查看图像
rqt_image_view
## 显示如下图，注意先刷新再选择对应的话题即可显示图像

```



### 3. 重新编译

```
catkin_make clean && catkin_make
```

### 4. 配置多相机

- 这里测试3个相机, 工控机接上3个[astra](#)相机的usb,

```
# 1 检测相机序列号
## 1.1 相机序列号查询,方法1
rosrun astra_camera list_devices_node
## 终端显示如下:
# [ INFO] [1710299519.592966536]: Found 3 devices
# Device connected: Astra
# URI: 2bc5/060e@1/29
# Serial number: AU1231201GE
# Device connected: Astra
# URI: 2bc5/060e@1/35
# Serial number: AU1953304F2
# Device connected: Astra
# URI: 2bc5/060e@1/24
# Serial number: AU1P32201SA

## 发现3个devices, 即表示相机连接成功, 记住这3个Serial number, 后续会使用
## 未发现3个devices, 请检测相机usb线是否成功连接

## 1.2 相机序列号查询,方法2
## cobot_magic/tools目录下camera_serial.sh脚本, 注意路径,如果没有该文件就使用方法1
## 1.2.1 进入cobot_magic目录
cd cobot_magic
## 1.2.2 运行该脚本
./tools/camera_serial.sh

# 2 修改multi_camera.launch配置文件
## 修改camera_ws/src/ros_astra_camera/launch/multi_camera.launch第13左右相机的Serial
## number值
## 将1中输出的3个Serial number值,分别填入下面代码中即可
```

```
<arg name="camera1_serila_number" default="AU1231201GE"/>
<arg name="camera2_serila_number" default="AU1953304F2"/>
<arg name="camera3_serila_number" default="AU1P32201SA"/>

# 3 启动多相机
roslaunch astra_camera multi_camera.launch
## 使用rqt_image_view即可查看相应的图像话题
```

## 2.2 机械臂配置

### 2.2.1 环境安装

can2usb模块若未刷固件, 请在[固件升级canable地址](#)升级固件

#### 1. 安装依赖

```
sudo apt install can-utils net-tools libkdl-parser-dev -y
```

#### 2. 查看机械臂的serial号

机械臂按顺序依次进行以下操作：

- 机械臂上电
- 机械臂usb插入工控机
- 查看serial号
- 机械臂**serial号(右后0, 右前1, 左后2, 左前3)**顺序对应**arx\_can.rules**中顺序  
**(canable0,canable1,canable2,canable3)**, 切记务必对应上顺序

```
# 1 查看ttyACM
ls /dev/ttyACM*
## 终端显示ttyACM0, 下面就用/dev/ttyACM0查看serial序列号
## 插上机械臂的can2usb才会有终端显示, 如果终端没有输出, 请重新拔插

# 2 查看serial号
udevadm info -a -n /dev/ttyACM0 | grep serial

## 具体操作终端显示如下图
```

- 拔掉usb, 换下一条臂继续上面操作, 记录下每条臂的serial号

```

(base) agillex@ubuntu:~$ ls /dev/ttyACM*
/dev/ttyACM0  /dev/ttyACM1  /dev/ttyACM2  /dev/ttyACM3
(base) agillex@ubuntu:~$ udevadm info -a -n /dev/ttyACM0 | grep serial
ATTRS{serial}=="206E3894D4D"
ATTRS{serial}=="000000000"
ATTRS{serial}=="3610000.xhci"
(base) agillex@ubuntu:~$ udevadm info -a -n /dev/ttyACM1 | grep serial
ATTRS{serial}=="206E3894D4D"
ATTRS{serial}=="000000000"
ATTRS{serial}=="000000000"
ATTRS{serial}=="3610000.xhci"
(base) agillex@ubuntu:~$ udevadm info -a -n /dev/ttyACM2 | grep serial
ATTRS{serial}=="207338614D40"
ATTRS{serial}=="000000000"
ATTRS{serial}=="000000000"
ATTRS{serial}=="3610000.xhci"
(base) agillex@ubuntu:~$ udevadm info -a -n /dev/ttyACM3 | grep serial
ATTRS{serial}=="206C38954D4D"
ATTRS{serial}=="000000000"
ATTRS{serial}=="000000000"
ATTRS{serial}=="3610000.xhci"
(base) agillex@ubuntu:~$
```

### 3. 构建规则

新建arx\_can.rules文件,内容如下

```
# 当前目录下新建arx_can.rules文件，并进去编辑模式
vim arx_can.rules
```

- 需要修改serial号内容
- 机械臂serial号(右后0, 右前1, 左后2, 左前3)顺序对应arx\_can.rules中顺序  
**(canable0,canable1,canable2,canable3)**, 切记务必对应上顺序

```
# ATTRS{serial}=="自己对应的串口号内容"
SUBSYSTEM=="tty", ATTRS{idVendor}=="16d0", ATTRS{idProduct}=="117e",
ATTRS{serial}=="207D38544D4D", SYMLINK+="canable0"
SUBSYSTEM=="tty", ATTRS{idVendor}=="16d0", ATTRS{idProduct}=="117e",
ATTRS{serial}=="207938A14D4D", SYMLINK+="canable1"
SUBSYSTEM=="tty", ATTRS{idVendor}=="16d0", ATTRS{idProduct}=="117e",
ATTRS{serial}=="2068385D4D4D", SYMLINK+="canable2"
SUBSYSTEM=="tty", ATTRS{idVendor}=="16d0", ATTRS{idProduct}=="117e",
ATTRS{serial}=="208E386A4D4D", SYMLINK+="canable3"
```

### 4. 更新rule

```
# 1 添加权限
chmod +x arx_can.rules

# 2 将arx_can.rules拷贝到/etc/udev/rules.d/目录下
sudo cp arx_can.rules /etc/udev/rules.d/

# 3 更新rule
## 只要更新arx_can.rules文件中serial的值都需要运行一次下面的代码
sudo udevadm control --reload-rules && sudo udevadm trigger
```

## 2.2.2 绑定机械臂can口

1. 绑定can口 开机上电, 电脑插上4条机械臂的can转usb口接线
2. 新建can.sh文件

```
# 当前目录下新建can.sh文件, 并进行编辑  
vim can.sh
```

can.sh内容如下

- 需要将password设置成本机的密码

```
#!/bin/bash  
source ~/.bashrc  
password=agx    # 修改这一行即可  
  
gnome-terminal -t "can" -x bash -c "source ~/.bashrc;source  
/opt/ros/noetic/setup.bash;echo $password | sudo -S slcand -o -f -s8 /dev/canable0  
can0;sudo ifconfig can0 up;exec bash;"  
  
gnome-terminal -t "can" -x bash -c "source ~/.bashrc;source  
/opt/ros/noetic/setup.bash;echo $password | sudo -S slcand -o -f -s8 /dev/canable1  
can1;sudo ifconfig can1 up;exec bash;"  
  
gnome-terminal -t "can" -x bash -c "source ~/.bashrc;source  
/opt/ros/noetic/setup.bash;echo $password | sudo -S slcand -o -f -s8 /dev/canable2  
can2;sudo ifconfig can2 up;exec bash;"  
  
gnome-terminal -t "can" -x bash -c "source ~/.bashrc;source  
/opt/ros/noetic/setup.bash;echo $password | sudo -S slcand -o -f -s8 /dev/canable3  
can3;sudo ifconfig can3 up;exec bash;"
```

- can.sh文件加权限

```
chmod +x can.sh
```

3. 执行can.sh

```
./can.sh
```

#### 4. 查看can口信息 执行上面代码后,查看can口是否生效

```
# 1 查看can口信息
ifconfig | grep can
# 显示内容如下即can口, 即可以正常启动机械臂:
can0: flags=193<UP,RUNNING,NOARP> mtu 16
can1: flags=193<UP,RUNNING,NOARP> mtu 16
can2: flags=193<UP,RUNNING,NOARP> mtu 16
can3: flags=193<UP,RUNNING,NOARP> mtu 16
## 如果只出现can1到can3, can0无显示, 再运行一次can.sh脚本

# 2 复制./can.sh脚本
## 将can.sh脚本复制到remote_control/tools目录下便于启动
cp ./can.sh remote_control/tools
```

### 2.2.3 启动机械臂

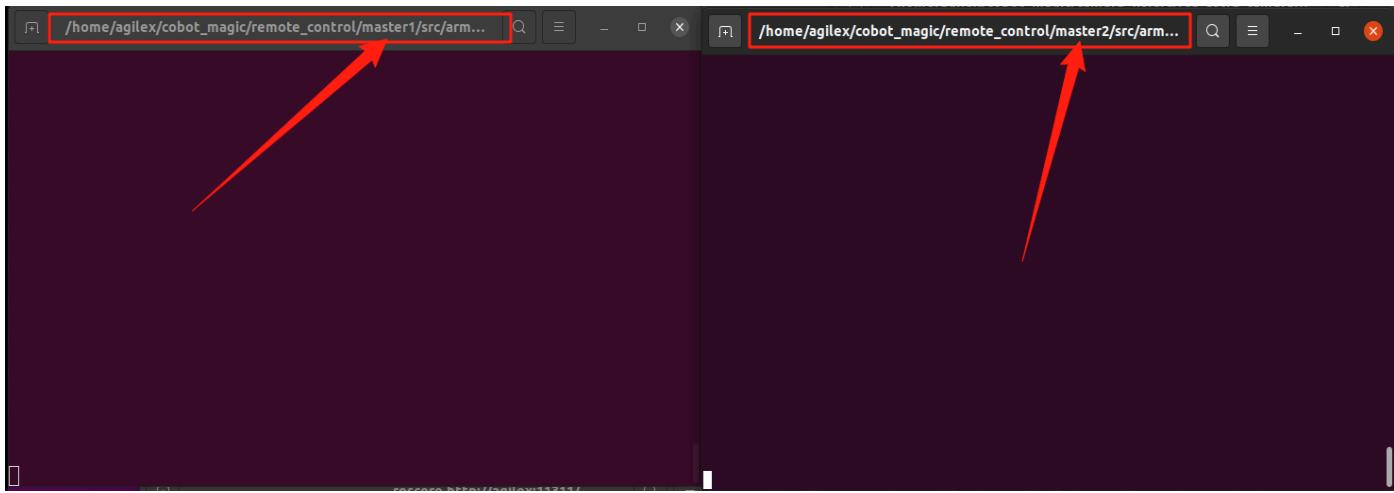
```
# 1 启动roscore
roscore

# 2 新开一个终端, 进入remote_control目录
cd remote_control

# 3 机械臂上电,can使能
./tools/can.sh
## 出现的4个窗口都无报错才能进行下一步, 注意检查每个窗口的输出, 如果只有can0不行,就重新运行一次该脚本
## 如果can都不行,检查usb,重新插入

## 3.1 检测机械臂的can是否通讯上
# 按照本文第2.2.2(绑定机械臂can口)章节的第4点(查看can口信息), 有同样输出即可

# 4 启动4条机械臂
./tools/remote.sh
## 在master1和master2窗口(如下图所示)中按1次i进入遥操作模式,臂能正常遥操作即验证成功
## 如果重启./tools/remote.sh请先关闭机械臂的所有窗口
```



## 3 数据采集

- 默认ubuntu20.04-noetic环境已经配置完成。

### 3.1 环境依赖

```
# 1 ros依赖
sudo apt install ros-$ROS_DISTRO-sensor-msgs ros-$ROS_DISTRO-nav-msgs
ros-$ROS_DISTRO-cv-bridge

# 2 python依赖
## 2.1 激活python环境
conda activate aloha

## 2.2 安装torch
pip install torch==2.1.1 torchvision==0.16.1 torchaudio==2.1.1 --index-url
https://download.pytorch.org/whl/cu118

## 2.3 进入当前工作空间目录,安装requirements.txt文件中的依赖即可
cd collect_data

## 2.4 安装requirements.txt
pip install -r requirements.txt

## 2.5 安装detr
cd act/detr && pip install -v -e .
```

### 3.2 运行

### 3.2.1 采集数据

- 采集数据前需要启动机械臂遥操作模式
- 检测机械臂、相机是否运行成功

#### 1. 启动机械臂、相机

- 启动硬件前, 请检查机械臂电源、通讯线, 相机通讯线是否成功连接
- 如果之前已经启动了机械臂遥操作代码, 这里就不需要启动机械臂遥操作, 只用启动相机即可

```
# 1 启动硬件
## 1.1 启动
roscore

# 1.2 启动遥操作
## 1.2.1 机械臂can使能
./tools/can.sh
## 电脑每次上电后记得运行./tools/can.sh
## 出现的4个窗口都无报错才能进行下一步, 注意检查每个窗口的输出, 如果只有can0不行, 就重新运行一次该脚本
## 后续只要不断开电源和断开通讯线就不用重复执行; 不放心, 可以每次启动臂前执行一次该脚本

### 1.2.1 进入remote_control目录
cd remote_control
### 1.2.2 启动遥操作脚本
### 启动后按在master1和master2窗口下按一次i键进入遥操作模式
./tools/remote.sh

## 1.3 启动相机
roslaunch astra_camera multi_camera.launch

# 1.4 终端运行rostopic list查看ros话题
rostopic list
## 显示如下图, 有机械臂4个、相机3个话题即可采集数据

# 1.5 一次打印上面ros话题的内容, 保证采集数据前, 传感器数据正常
rostopic echo 话题名
## 终端会按频率打印传感器数据, 传感器有数据变化即可
```

```
(base) agilex@agilex:~/cobot_magic$ rostopic list
/camera_f/color/camera_info
/camera_f/color/image_raw
/camera_f/depth/camera_info
/camera_f/depth/image_raw
/camera_f/depth/points
/camera_f/ir/camera_info
/camera_l/color/camera_info
/camera_l/color/image_raw
/camera_l/depth/camera_info
/camera_l/depth/image_raw
/camera_l/depth/points
/camera_l/ir/camera_info
/camera_r/color/camera_info
/camera_r/color/image_raw
/camera_r/depth/camera_info
/camera_r/depth/image_raw
/camera_r/depth/points
/camera_r/ir/camera_info
/joy
/master/joint_left
/master/joint_right
/puppet/joint_left
/puppet/joint_right
/rosout
```

## 2. 采集数据

```
# 1 启动roscore
roscore

# 2 激活虚拟环境
conda activate aloha

# 3 启动采集数据
## 3.1 进入collect_data目录
cd collect_data

## 3.2 查看collect_data.py配置参数
python collect_data.py -h # 查看参数

## 3.3 采集数据
python collect_data.py --dataset_dir ~/data --max_timesteps 500 --episode_idx 0
```

- 数据采集时终端显示如下图所示：

终端打印 **sync fail** 是正常的, 表示当前时刻没同步传感器数据, 只要终端不是一直 **sync fail** 而没有 **Frame data: xxx** 输出, 都是正常现象

只要终端一直有 **Frame data: xxx** 打印信息, 即表示正在记录数据集

如果打印 **sync fail**, 后续没有其他输出, 证明传感器数据没接收到, 请使用 **rostopic echo 话题名** 检查 **ros** 话题是否成功发布

```
(aloha) agilex@aloha-nano:~/cobot_magic/collect_data$ python collect_data.py --dataset_dir ~/data --max_timesteps 500 --episode_idx 0
syn fail
Frame data: 2
Frame data: 3
syn fail
Frame data: 4
Frame data: 5
syn fail
Frame data: 6
Frame data: 7
syn fail
Frame data: 8
Frame data: 9
Frame data: 10
syn fail
Frame data: 11
Frame data: 12
syn fail
Frame data: 13
Frame data: 14
syn fail
Frame data: 15
```

- 数据保存路径说明

```
# 数据采集完成后会保存到` ${dataset_dir} / ${task_name}` 目录下
# 运行上面这句代码, 其他参数采用默认值。产生数据集episode_0.hdf5, 该工程目录如下
collect_data
    ├── collect_data.py
    ├── data           # --dataset_dir 数据集保存路径
        └── aloha_mobile_dummy # --task_name 任务名
            ├── episode_0.hdf5 # 产生数据集文件的位置
            ├── episode_idx.hdf5 # idx由--episode_idx参数决定
            └── ...
    ├── readme.md
    ├── replay_data.py
    ├── requirements.txt
    └── visualize_episodes.py
```

- collect\_data.py参数详细介绍

本代码暂时不支持深度图采集, 代码已开源, 可自行修改源码适配深度图采集

--dataset_dir	数据集保存路径
--task_name	任务名, 作为数据集的文件名
--episode_idx	动作分块索引号
--max_timesteps	最大动作分块的时间步数
--camera_names	相机名称, 默认['cam_high', 'cam_left_wrist', 'cam_right_wrist']
--img_front_topic	相机1彩色图话题
--img_left_topic	相机2彩色图话题
--img_right_topic	相机3彩色图话题
--use_depth_image	是否使用深度信息 # 本代码暂时不支持深度图采集, 代码已开源, 可自行 修改源码适配
--depth_front_topic	相机1深度图话题
--depth_left_topic	相机2深度图话题
--depth_right_topic	相机3深度图话题
--master_arm_left_topic	左主臂话题

```
--master_arm_right_topic 右主臂话题
--puppet_arm_left_topic 左从臂话题
--puppet_arm_right_topic 右从臂话题
--use_robot_base 是否使用底盘信息
--robot_base_topic 底盘话题
--frame_rate 采集帧率 因相机图像稳定值为30帧,默认30帧
```

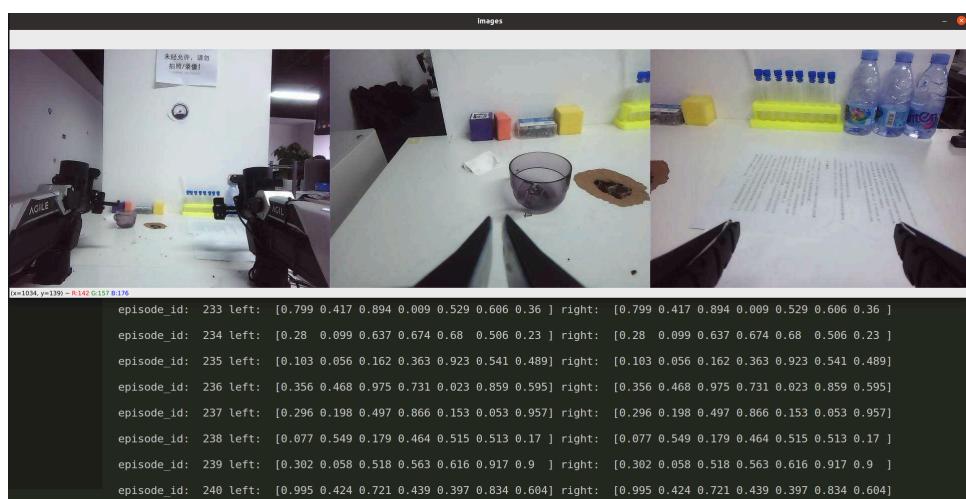
## 3.2.2 可视化数据集

### 1. 运行

```
# 1 激活虚拟环境
conda activate aloha

# 2 运行visualize_episodes.py
python visualize_episodes.py --dataset_dir ./data --task_name aloha_mobile_dummy --
episode_idx 0
```

- 将3.2.1小节采集数据进行可视化运行上面代码。--dataset\_dir、--task\_name与--episode\_idx参数需要与3.2.1小节采集数据时相同
- 运行上面代码,可视化结果如下: 终端会打印action,并显示一个彩色图像窗口



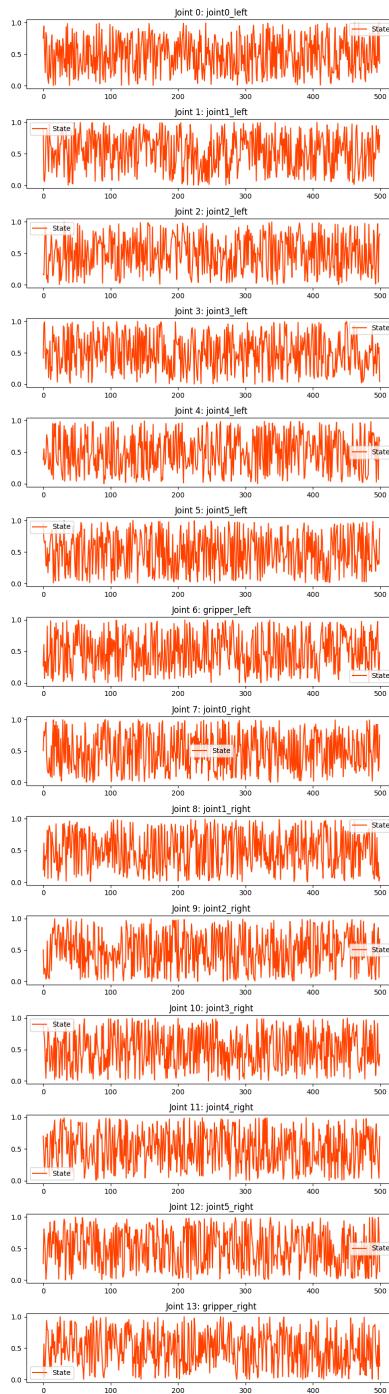
- 运行完成后,会在\${dataset\_dir}/\${task\_name}下产生episode\_\${idx}\_qpos.png、episode\_\${idx}\_base\_action.png与episode\_\${idx}\_video.mp4文件,目录结构如下:

```
collect_data
├── data
│   ├── aloha_mobile_dummy
│   │   └── episode_0.hdf5
│   └── episode_0_base_action.png # base_action图
```

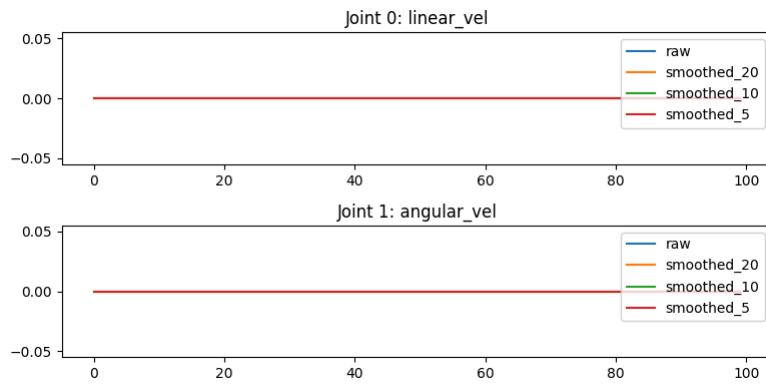
```
|   └── episode_0_qpos.png  
|   └── episode_0_video.mp4
```

```
# qpos图  
# 彩图图视频流
```

- `episode_${idx}_qpos.png`



- `episode_${idx}_base_action.png`



- `episode_{idx}_video.mp4`



## 2. 参数说明

- `--dataset_dir` 数据集保存路径
- `--task_name` 任务名,作为数据集的文件名
- `--episode_idx` 动作分块索引号

### 3.2.3 重播数据集

#### 1. 运行

- 将采集的数据集包,使用ros发布该数据包的彩色图和机械臂关节姿态
- 发布该数据包后,cobot\_magic可订阅该消息进行示教模式跟随运动
- 运行`replay_data.py`请关闭遥控操作代码窗口, 只启动`puppet.sh`脚本即可

```
# 1 启动roscore
roscore

# 2 新启动一个终端, 激活虚拟环境
conda activate aloha

# 3 发布彩色图、主臂、从臂消息
## 3.1 进入collect_data目录
cd collect_data

## 3.2 启动replay_data.py脚本
```

```

## replay_data.py运行前, 请关闭遥操作代码窗口, 只启动puppet.sh脚本即可
## 如果不 启动puppet.sh脚本启动从臂, 从臂不会随动
python replay_data.py --dataset_dir ./data --task_name aloha_mobile_dummy --
episode_idx 0
# 4 发布彩色图、主臂消息
python replay_data.py --dataset_dir ./data --task_name aloha_mobile_dummy --
only_pub_master --episode_idx 0
## replay_data.py运行前, 请关闭遥操作代码窗口, 只启动puppet.sh脚本即可

```

不管发不发布从臂姿态信息，只要发布主臂的姿态信息，从臂都可以动。

## 2. 参数说明

- **--dataset\_dir** 数据集保存路径
- **--task\_name** 任务名,作为数据集的文件名
- **--episode\_idx** 动作分块索引号
- **--only\_pub\_master** 是否只发布主臂的关节姿态消息

# 4 ACT训练推理

## 4.1 环境配置

- ubuntu-20.04,cuda-11.8,cudnn-8.6.0,torch-2.1.1, python-3.8已测试通过
- ubuntu-20.04,cuda-11.3,cudnn-8.6.0,torch-1.10.0,python-3.8已测试通过

```

# 1 创建conda虚拟环境
conda create -n aloha python=3.8

# 2 激活虚拟环境
conda activate aloha

# 3 安装torch-2.1.1
pip install torch==2.1.1 torchvision==0.16.1 torchaudio==2.1.1 --index-url
https://download.pytorch.org/whl/cu118

# 4 安装requirements.txt
pip install -r requirements.txt

# 5 安装detr
cd aloha-devel/act/detr && pip install -v -e .

```

详情请参考[mobile-aloha](#)、[act-plus-plus](#)

## 4.2 数据集采集

- 参考本文3.2.1(采集数据)章节

## 4.3 训练

```
# 1 激活虚拟环境  
conda activate aloha  
  
# 2 训练  
## 2.1 进入aloha-devel目录  
cd aloha-devel  
  
## 2.2 启动训练  
python act/train.py --dataset_dir ~/data0314/ --ckpt_dir train --batch_size 4 --  
num_epochs 5000 --num_episodes 50
```

### 1. 主要参数说明 目前仅支持ACT模型训练

- --dataset\_dir 数据集目录
- --ckpt\_dir 训练模型保存目录
- --batch\_size 训练批量大小
- --num\_epochs 训练周期
- --task\_name 任务名称
- --pretrain\_ckpt 预训练模型路径
- --ckpt\_name 模型名称
- --num\_episodes 多少组数据

## 4.4 推理

```
# 1 启动roscore  
roscore  
  
# 2 新启动一个终端，启动从臂与相机  
## 2.1 进入remote_control目录  
cd remote_control  
  
## 2.2 执行puppet.sh脚本，该脚本功能只是启动2条从臂，启动前需要关闭机械臂遥操作代码  
../tools/puppet.sh
```

```
# 3 推理
## 3.1 激活虚拟环境
conda activate aloha

## 3.2 执行推理
## 3.2.1 进入aloha-devel目录
cd aloha-devel
## 3.2.2 推理
python inference.py --ckpt_dir train_dir
## 推理时注意安全，如果发现推理表现不正常，请立即中断代码或者断开机械臂电源，以免损伤机械臂
```

## 5 Q&A

1.

**Q:**启动遥操作remote.sh脚本程序, 手操作主臂, 但是主臂锁住

**A:** 在master1与master2终端窗口按一次i即可, 切记不要按i多次

2.

**Q:** 启动remote.sh脚本时, 四条臂或者单臂无法进入初始零位位置。

**A:** 检查无法正常归零位的臂的电源、网口是否松动

3.

**Q:** 遥操作时,由于暴力操作或者误操作导致某条机械臂限位, 无法正常工作。

**A:** 正常的臂归零位状态, 无法正常运作的臂用手托着, 然后关闭机械臂所有终端或者断电, 重新启动即可

4.

**Q:** 启动底盘后, 有里程计/odom消息, 但是移动车,/odom数据始终处于0数据。

**A:** 底盘can使能, 然后重新启动底盘程序

5.

**Q:** 如何查看ros话题频率

**A:** 终端输入`rostopic hz` 话题名称即可打印该话题频率

6.

Q: 运行collect\_data.py脚本，如果代码报错

A: 请使用cobot\_magic自带的镜像环境

7.

Q: 运行collect\_data.py脚本，如果终端一直打印sync fail

A: 请检查collect\_data.py收集的ros话题是否有正常输出，没有正常输出请检查各话题名是否对应

8.

Q: 终端报错信息如下：

```
RLEexception: roscore cannot run as another roscore/master is already running.  
Please kill other roscore/master processes before relaunching.  
The ROS_MASTER_URI is http://PC:11311/  
The traceback for the exception was written to the log file
```

A: 由于重复启动roscore导致，可以重启或者注销电脑

9.

Q: 编译remote\_control、camera\_ws代码时。报empty,rospkg,catkin\_pkg相关的错误

A:

```
pip install empty==3.3.4 catkin-pkg rospkg -i  
https://pypi.tuna.tsinghua.edu.cn/simple/
```

10.

Q: 底盘ros里程计消息/odom打印，数字没有变化

A: 确定底盘can2usb线是否成功连接和底盘can是否使能

11.

Q: 录制数据终端打印