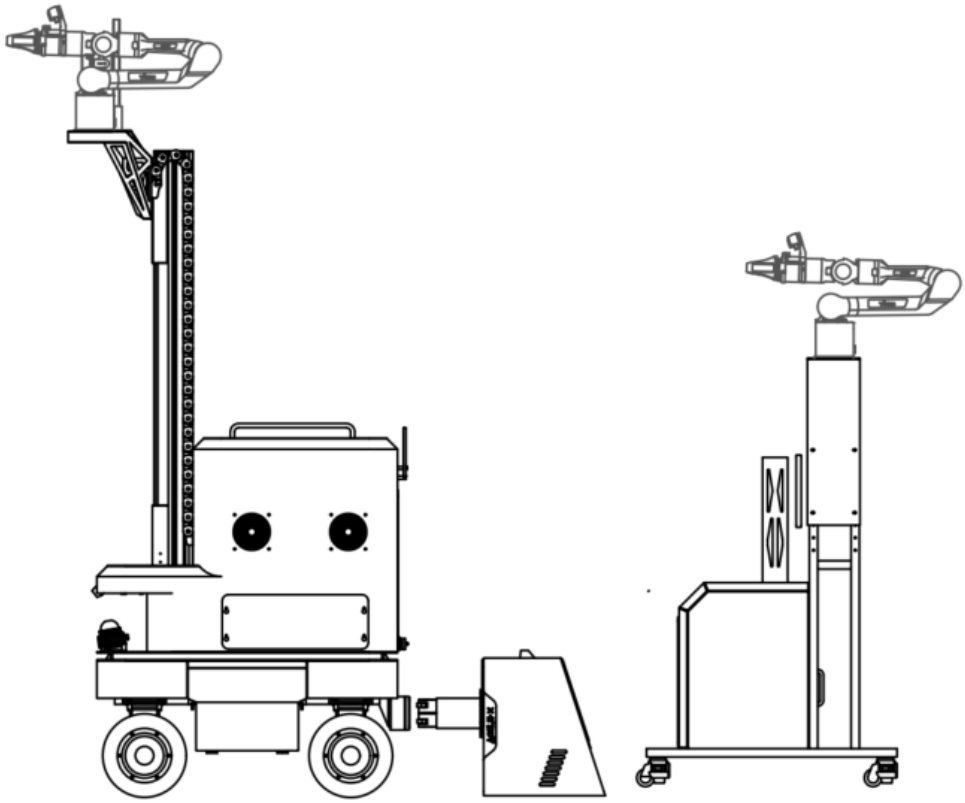
Modular Aloha Product User Manual

Version	Description	Author	Reviewer	Date
V1.0.0	Document Creation	Dennis		2025-05-30
V1.0.1	• Add charging instructions • Post-processing method for modification of robotic arm USD port • Enhanced Manual Instructions for Power-off and Restart Procedures of Manipulator Arms	Dennis		2025-06-04



Device description：

The split-type ALOHA system comes in two versions. The differences between the open-source navigation version and the NAVIS navigation version are as follows:

Device Name	Difference Explanation
Open-Sources Navigation Modular Aloha	The navigation section utilizes an open-source configuration. For navigation instructions, please refer to the Open-Source Navigation User Manual; the teleoperation software remains the same.
NAVIS Navigation Modular Aloha	The navigation section utilizes a closed-source NAVIS configuration. For navigation instructions, please refer to the NAVIS Navigation User Manual; the teleoperation software remains the same.

Important Notes！！

- Industrial PC (IPC - RTX 4090/4060 GPU Versions): Supports robotic arm teleoperation, data collection, model training, and inference.
- Training Recommendations: High-compute servers with large VRAM are recommended for model training. For the 4060 GPU version, training a dataset of 50 episodes and 500 timesteps may trigger "Out of Memory" errors even with a batch size of 4. The 4060 is primarily recommended for inference.
- Troubleshooting: Please consult the Q&A section first if any issues arise.
- System Environment: The software environment is pre-configured at the factory. Do not modify system settings unless necessary.
- Ethernet Port Binding: For units equipped with LiDAR, do not change the Ethernet port connections, as they are bound to specific hardware addresses.
- USB Port Binding: The USB ports for the left arm, right arm, and chassis are bound at

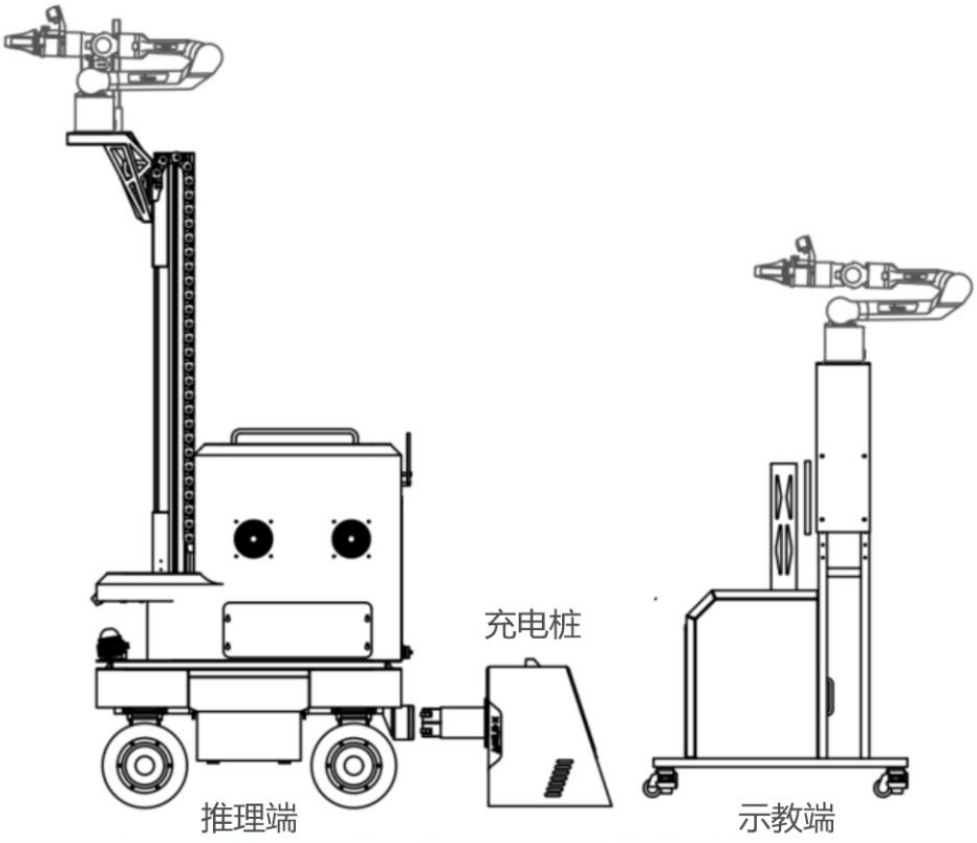
the device level. Each USB cable is labeled; do not swap their positions, as this will cause CAN enablement failures.

- Power On: The IPC will automatically boot upon receiving power. The system is ready once the status indicator on the left side of the IPC lights up.
- Display Output: Ensure the HDMI or DP cable is plugged directly into the GPU ports, not the motherboard ports.

1. Product Introduction

Product Overview

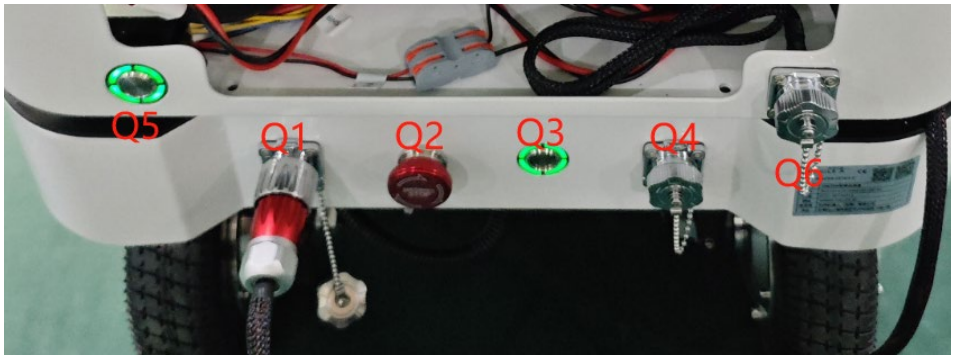
Built upon the Stanford Mobile Aloha open-source project, this comprehensive navigation system features lift-and-drop ROS drivers that balance autonomous mobility and grasping capabilities, while enabling more flexible remote operation for data acquisition and inference validation.

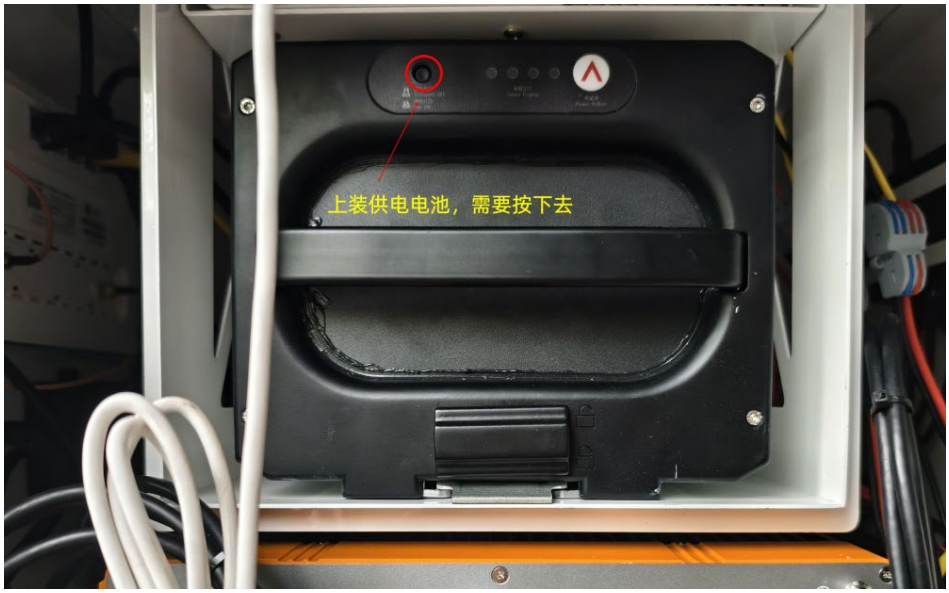


2. Hardware Operation：

2.1 Power On/Off Operation：

2.1.1 Power on/off of the inferences-end devices：





The rear panel of the mobile device is shown in the figure above, with the ports illustrated as follows:

Q1	Chassis external power supply and communication ports	Q4	Chassis charging port
Q2	Chassis emergency stop	Q5	Battery-powered switch for upper section

Q3	Open-source chassis power supply	Q6	Upper battery charging port
● Q3 and Q5 are push-button switches that activate when pressed and deactivate when released. The green circular light strip indicates current battery level, with four rings representing 100% capacity. Each ring decreases by 25% as the battery level drops. ● Q3 controls power supply to the mobile chassis, navigation system, lift column module, and Q5 switch power supply unit ● Q5 controls the industrial control computer and supplies power to two slave arms (Q3 must be turned on before using Q5) ● Pressing the Q2 emergency stop button activates the chassis in parking mode, halting movement			

Power-on procedure: First activate Q3, then turn on Q5 (the Q5 indicator light will illuminate simultaneously after Q3 activation).

Power-off procedure: First turn off Q5, then Q3.

Charging: Charge the upper and lower modules separately. Q4 and Q6 correspond to the bottom and top charging ports respectively. Use the matching charger for charging.

Note：

Before powering on or off, place the robotic arm horizontally and manually reset the gripper and manipulator to the closed position. Each time you power on, first return the robotic arm to the correct position before powering on. Powering on without positioning it correctly may cause the robotic arm's zero point to shift.

2.1.2 Power On/Off of Teaching Terminal Equipment:

Open the cabinet door-Turn on the power switch-Turn on the power strip switch

2.2 Remote Control of the Chassis:

Device Power-On： Turn on Q3. Ensure the Emergency Stop switch is in the released position (rotate clockwise to pop it up).

Remote Controller Power-On: Toggle all switches (SW1, SW2, SW3, SW4) to the uppermost position. Press and hold both POWER buttons simultaneously to turn on the controller.

Remote Control Operation：

- **SWB:** Toggle to the middle or bottom position to enter Remote Control mode.
- **SWD (UP):** Enables Dual-Ackermann and Rotation modes.

①Use the **Left Stick** to control velocity and the **Right Stick** to control the steering angle.

②Keep the **Left Stick** centered and move the **Right Stick** left or right to control self-rotation.

Remote Controller Power-Off： Toggle all switches (SW1, SW2, SW3, SW4) to the uppermost position. Press and hold both POWER buttons simultaneously to power off.

Remote Controller Overview：

As shown in the figure above, the button functions are defined as:

SWA	Light control switch: Set to the lower position to turn off the lights (SWB must first enter remote control mode).
SWB	Turn the control mode lever to the top for command control mode, and to the middle or bottom for remote control mode.
SWC	When the lever is moved downward, the vehicle enters parking mode, with all four wheels locked in a X configuration.
SWD	Chassis motion mode setting switch: The top allocation is ① before and after Ackermann and ② spin mode

	① The left joystick controls speed, while the right joystick controls rotation angle; ② The left joystick remains stationary, while the right joystick controls rotation in lateral direction; SWD offset to the lower diagonal mode: The left joystick controls speed, while the right joystick controls rotation angle (with a maximum of 90° for lateral movement).
POWER	Power button: Hold to power on/off
KEY1	Press KEY1 under any circumstances to forcibly clear all chassis errors **Note!! Use only under special safety conditions**
KEY2	Remote control configuration

Instructions for battery replacement in the remote control:

The Fux remote control uses AA batteries as its power source. When the Remoter display level shows low battery status, this indicates insufficient power. In such cases, open the battery cover on the back of the remote control to replace the batteries.

For more detailed instructions, refer to the Ranger Mini 3.0 user manual at https://agilexsupport.yuque.com/staff-hso6mo/peoot3/ucr0xfc7a0uqqflo?singleDoc#

2.3 Master-Follower CAN BUS Connection：

1. On the inference side, cut open the reserved black rolled cable harness, leaving approximately 3 meters of cable. Connect the CAN lines of the master and slave arms in

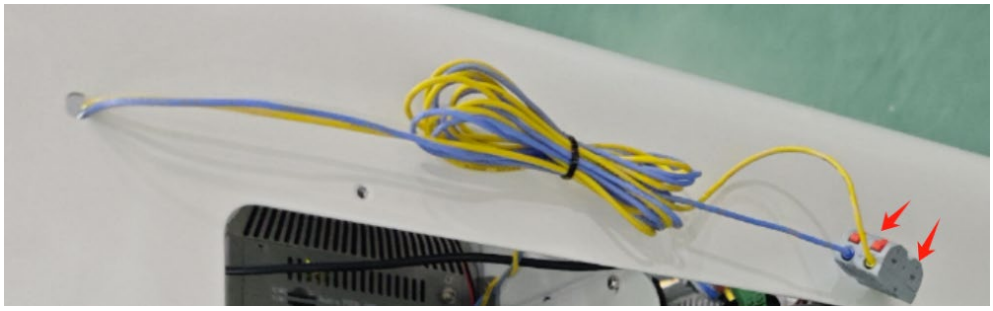
parallel, with left-side connections at both front and rear ends, and right-side connections at the same positions.

2. Unplug the reserved wiring harness on the teaching end. By default, the harness remains intact upon receipt and can be cut open with simple tools.

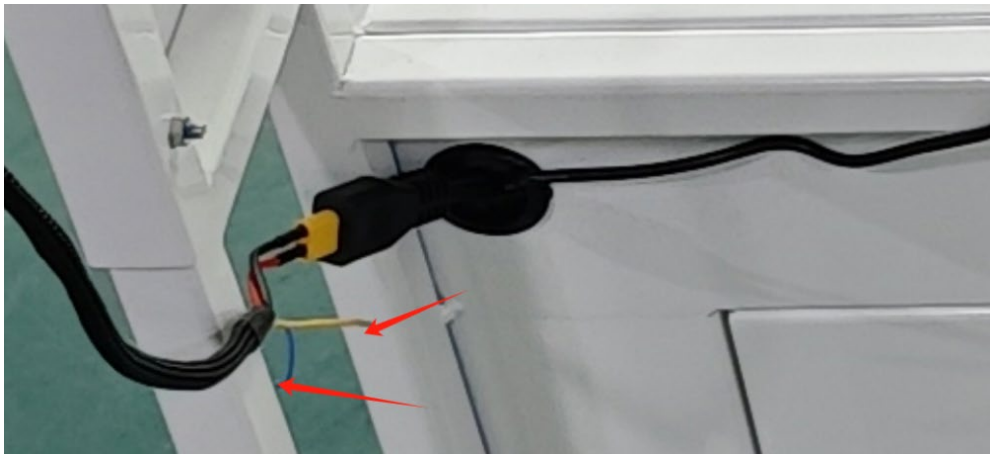
3. Yellow cable harness corresponds to yellow cable harness connectors; blue cable harness corresponds to blue cable harness connectors

When using master-slave joystick control, first establish wired connections between the inference terminal and teaching terminal. This primarily involves connecting the CAN lines of the master arm and slave arm.

Reserve wiring harness for the inference module:

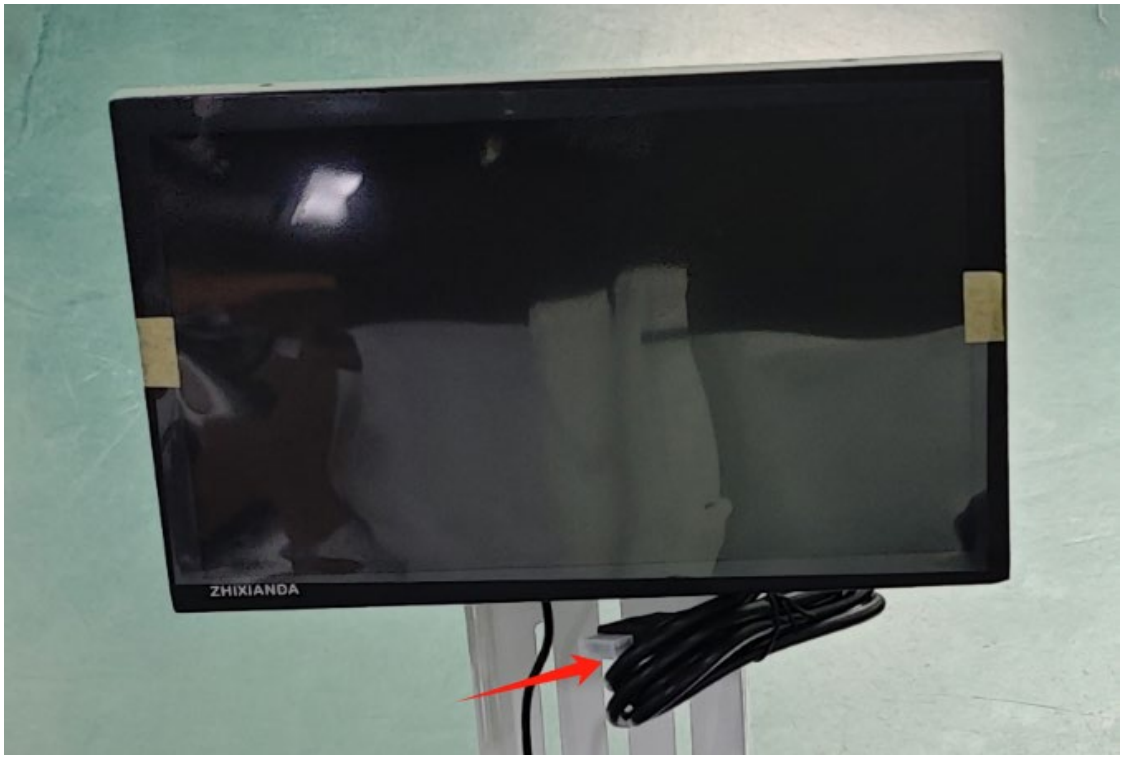


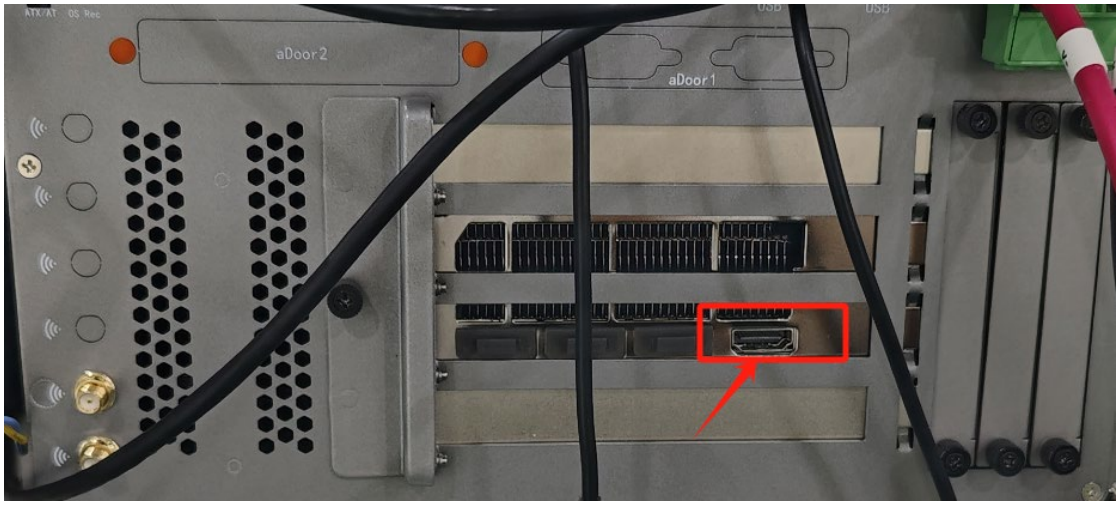
Reserve wiring harness for the demonstration terminal:



2.4 Display screen harness connection：

The display screen on the teaching terminal has an HDMI cable compartment reserved behind it, which connects to the HDMI port of the dedicated graphics card on the teaching terminal's industrial control computer.





All hardware operations have been completed.

3. Software Operation:：

3.1 Environmental Configuration Description：

Pre-configured before shipment. No reconfiguration required.

Industrial control computers are available in two configurations: 4060 and 4090 graphics card models. The system image includes default packages such as Ubuntu-20.04, ROS1-Noetic, CUDA-11.3, Torch-1.10, Conda, and Python-3.8.

1. Basic configuration for data collection: ubuntu-20.04, ros1-noetic

2. Model training inference configuration: Ubuntu-20.04, ROS1-Noetic, CUDA-11.3, Torch-1.10 (CUDA-11.8, Torch-2.1.1), Conda, Python-3.8. The configuration has been successfully tested. For other CUDA/Torch versions, please conduct independent testing.

3. Mechanical arm SDK version: piper_sdk=0.3.0

3.2 System Login：

Industrial PC login：

● Host name: agilex

● User name: agilex

● Password: agx

Backend login for router：

● Router network segment：192.168.1.1

● User name: admin

● Password: admin

Router WiFi name: Log in to the admin panel to view

Password: 12345678

3.3 Camera Startup：

● Search for camera serial number connected to the industrial PC

Note: Configuration is correct upon factory release. Proceed with this step.

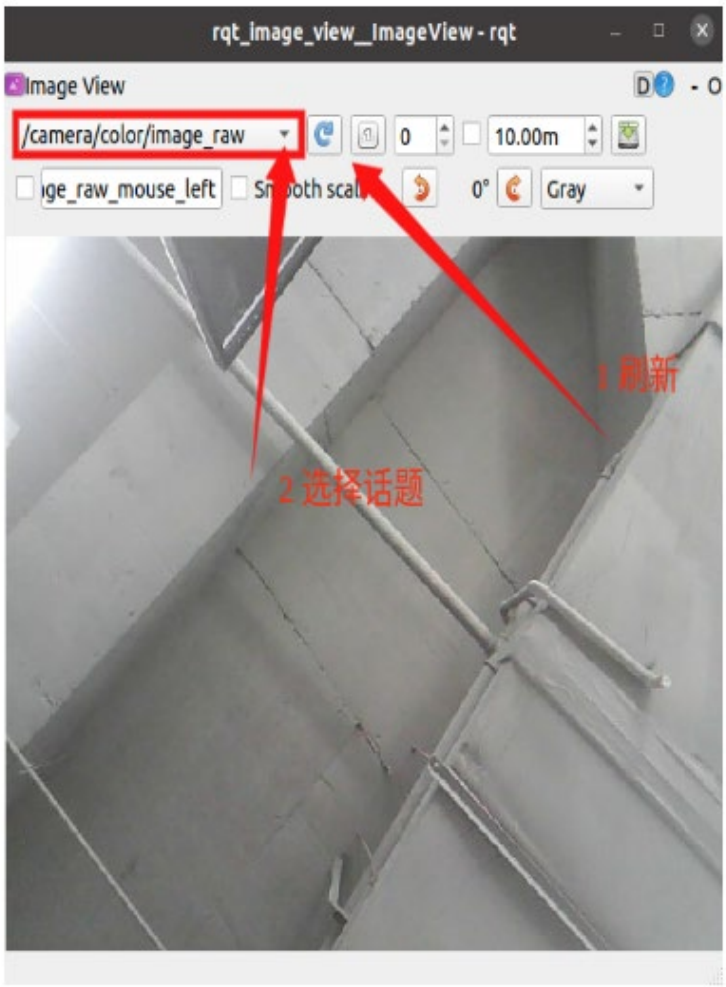
```
# dabai
cd cobot_magic
./tools/camera_serial.sh
Change camera serial number
gedit camera_ws/src/ros_astra_camera/launch/multi_camera.launch
```

● Start cameras (three cameras will start simultaneously)

```
# dabai
roslaunch astra_camera multi_camera.launch
```

● Open rqt_image_view to view the camera output image, as shown below. Note: Refresh the screen first before selecting the corresponding topic to display the image.

```
rqt_image_view
```



3.4 Mechanical Arm Startup

Open a new terminal, activate the robotic arm node, and read master-slave robotic arm messages

```
cd ~/cobot_magic/Piper_ros_private-ros-noetic/
bash can_multi_activate.sh
# Note: The USB port of the robotic arm is pre-configured with port binding. Do not modify the interface position arbitrarily.
source devel/setup.bash
# Use the following command to start the robotic arm node in read-only mode without automatic enabling:
roslaunch piper start_ms_piper.launch mode:=0 auto_enable:=false
```

Note：

If you have replaced the USB port interface before activating the robotic arm here, follow these steps:

① Enable the CAN module. Note that this CAN module only supports the built-in CAN module of the robotic arm and does not support other CAN module file paths. ~/cobot_magic/Piper_ros_private-ros-noetic/

② To locate CAN modules

- launch a terminal and execute the: bash script find_all_can_port.sh.

- After entering the password, if one CAN module is inserted into the computer and detected, the output will resemble:

Both ethtool and can-utils are installed.

Interface can0 is connected to USB port 3-1.4:1.0.

- If multiple CAN modules are inserted, the output will appear as:

Both ethtool and can-utils are installed.

Interface can0 is connected to USB port 3-1.4:1.0.

Interface can1 is connected to USB port 3-1.1:1.0.

- The number of CAN modules corresponds to the number of lines containing output like Interface can1 is connected to USB port 3-1.1:1.0. Here, can1 represents the system-detected CAN module name, while 3-1.1:1.0 indicates the USB port address of the module. If a CAN module has been previously activated and renamed, Assuming the name is can_piper, the output would be:

Both ethtool and can-utils are installed.

Interface can_piper is connected to USB port 3-1.4:1.0.

Interface can0 is connected to USB port 3-1.1:1.0.

- If no CAN module is detected, only the following output will appear: Both ethtool and can-utils are installed.

③ Activating multiple CAN modules simultaneously: Here, the can_muti_activate.sh script is used to first determine how many official CAN modules are inserted into the computer (assuming 2 modules). Note: If 5 CAN modules are currently inserted, only the specified 2 modules can be activated.

④ Recording USB port hardware addresses for each CAN module: Insert CAN modules sequentially while recording the corresponding USB port hardware addresses in the can_muti_activate.sh script. The number of elements in the USB_ports parameter corresponds to the pre-activated CAN module count (assumed as 2). (1) Insert one CAN module into the PC separately, then run bash find_all_can_port.sh in a new terminal and record the USB port values. For example, configure 3-1.4:1.0 (2). Insert the next CAN module, ensuring it is not connected to the same USB port as the previous CAN module. Then launch a new terminal to execute bash find_all_can_port.sh and record the USB port values of the second CAN module, such as 3-1.1:1.0. Note: If the system has not been activated, the first inserted CAN module will default to CAN0 and the second to CAN1. If previously activated, the names will retain their original configurations

⑤ Predefine USB ports, target interface names, and their bit rates:

1) Edit the can_muti_activate.sh file by launching a new terminal to execute gedit can_muti_activate.sh.

2) Assuming the recorded USB port values are 3-1.4:1.0 and 3-1.1:1.0, populate the USB_ports["xxxx"] field in the can_muti_activate.sh file with the corresponding values, where "xxxx" represents the specific content. The final configuration results: USB_ports["3-1.4:1.0"]= "can_left:1000000" USB_ports["3-1.1:1.0"]= "can_right:1000000" Explanation: The USB port with hardware code 3-1.4:1.0 connects to CAN devices, renamed as can_left with a baud rate of 1000000 and activated port 3.

3) Save configuration file.

⑥ Execute bash can_muti_activate.sh in a new terminal to activate multiple CAN modules.

⑦ Check configuration status by running ifconfig in a new terminal to verify if can_left and can_right names correspond to active CAN devices.

- Execute ip link show in a new terminal to confirm device status.

3.6 Data Collection

3.6.1 Initiate data acquisition program

Note：Before data collection, ensure that the data cables of the four robotic arms are properly connected and that the green indicator light is illuminated.

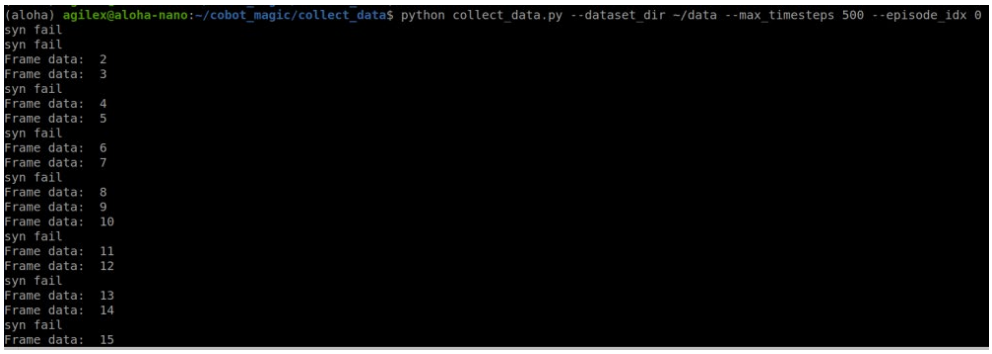
```
# 1 Start roscore
roscore
# 2 Activate the virtual environment
conda activate aloha
# 3  Start data collection
## 3.1 Enter the collect_data directory
cd collect_data
## 3.2 View the configuration parameters of collect_data.py
python collect_data.py -h
# View parameters
## 3.3 Data Collection
python collect_data.py --dataset_dir ~/data --max_timesteps 500 --episode_idx 0
```

- The terminal displays the following during data collection:

● A 'sync fail' status on the terminal is normal, indicating no sensor data synchronization at this moment. As long as the terminal doesn't consistently display' sync fail' without Frame data output, this remains a normal occurrence.

● As long as the terminal continuously receives Frame data: xxx print information, it indicates that the dataset is being recorded.

● If the print sync fails and no subsequent output appears, it indicates sensor data was not received. Use the rostopic echo topic name to verify if the ROS topic was successfully published.



● Data storage path description

```
#  Data collection will be saved to the ${dataset_dir}/{task_name} directory after completion. # Run the above code with default parameters to generate the dataset episode_0.hdf5. The project directory is as follows:
collect_data
├── collect_data.py
└── data                    # --dataset_dir   Data set storage path
    └── aloha_mobile_dummy   # --task_name
        ├── episode_0.hdf5   # Location for generating dataset files
        ├── episode_idx.hdf5 # idx 由--episode_idx  Parameter
        └── ...
        ├── readme.md
        ├── replay_data.py
        ├── requiredments.txt
        └── visualize_episodes.pycollect_data.py
```

Note: This code does not support depth map acquisition. The code is open source and can be modified to adapt depth map acquisition.

Detailed parameter description:

--dataset_dir Data set save path--task_name Task name, used as the dataset file name

--episode_idx Action chunk index number--max_timesteps Maximum number of time steps for action chunks

--camera_names Camera names, default ['cam_high', 'cam_left_wrist', 'cam_right_wrist']

--img_front_topic Camera 1 Color Image Topic

--img_left_topic Camera 2 Color Image Topic

--img_right_topic Camera 3 color image topic--use_depth_image Use depth information

This code does not support depth map acquisition. The code is open source and can be modified to adapt.

--depth_front_topic Camera 1 depth map topic--depth_left_topic Camera 2 depth map topic

--depth_right_topic Camera 3 depth map topic--master_arm_left_topic Left main arm topic

--master_arm_right_topic Right main arm topic--puppet_arm_left_topic Left slave arm topic

--puppet_arm_right_topic Right arm topic--use_robot_base Use chassis information

--robot_base_topic chassis topic--frame_rate frame rate acquisition, the default frame rate is 30 fps due to the camera's image stabilization setting of 30 fps

Data collection topic description:

Topic Name	Meaning
/master/joint_left	Data from the left upper arm joint
/master/joint_right	Data from the right upper limb joint
/puppet/joint_left	Left from the arm joint data
/puppet/joint_right	Data from the right arm joint

/puppet/end_left	Left from the end-of-arm pose data
/puppet/end_right	Right side from the end-of-arm pose data
/camera_f/color/image_raw	Top camera RGB image
/camera_l/color/image_raw	RGB image from the left wrist camera
/camera_r/color/image_raw	RGB image from the right wrist camera
/camera_f/aligned_depth_to_color/image_raw	Depth image from the top camera
/camera_l/aligned_depth_to_color/image_raw	Depth image from the left wrist camera
/camera_r/aligned_depth_to_color/image_raw	Depth image of the right wrist camera
/camera_f/color/camera_info	Top camera RGB intrinsic parameters
/camera_l/color/ camera_info	RGB internal parameters of the left wrist camera
/camera_r/color/ camera_info	RGB internal parameters of the right wrist camera
/camera_f/aligned_depth_to_color/camera_info	Top camera depth parameters
/camera_l/aligned_depth_to_color/camera_info	Depth reference parameters for left wrist camera
/camera_r/aligned_depth_to_color/camera_info	Depth reference parameters for right wrist camera
/odom	Chassis odometer (only when chassis data collection is enabled)

/tracer_states	Chassis status (shown as tracer, only when chassis data collection is enabled)
/cmd_vel	Chassis control (only when chassis data acquisition is enabled)

3.6.2 Visualized Datasets

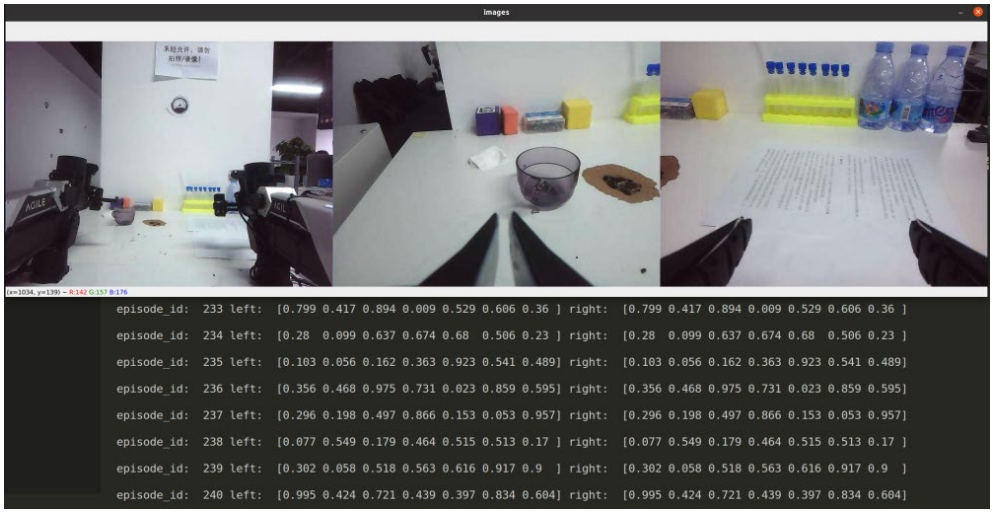
```
# 1 Activate virtual environment
conda activate aloha
# 2 Run visualize_episodes.py
python visualize_episodes.py --dataset_dir ~/data --task_name aloha_mobile_dummy  --episode_idx 0
```

● Visualize the collected data by running the above code.

The--dataset_dir, --task_name, and--episode_idx parameters must match those used during data collection.

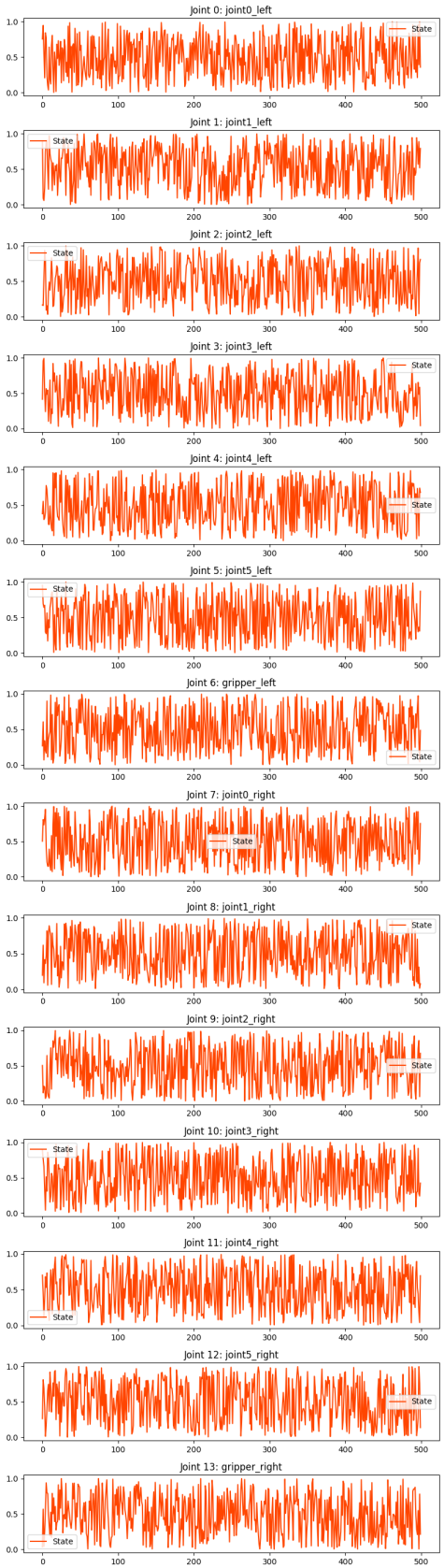
● Run the above code to visualize the results as follows:

The terminal will print the action and display a color image window.

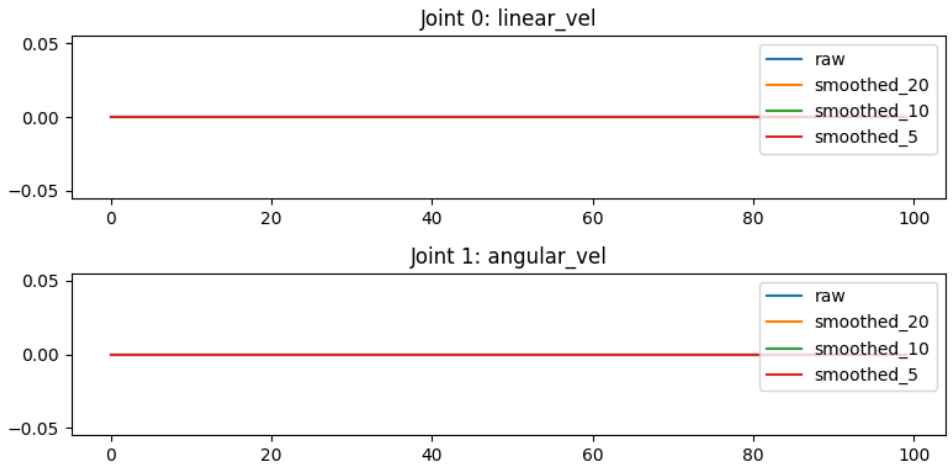


● After execution, the following files will be generated in ${dataset_dir}/{task_name}: episode_${idx}_qpos.png, episode_${idx}_base_action.png, and episode_${idx}_video.mp4, with the directory structure as follows:

```
collect_data
  ├── data
  ├── aloha_mobile_dummy
  └── episode_0.hdf5
  ├── episode_0_base_action.png # base_action image
  ├── episode_0_qpos.png # qpos image
  └── episode_0_video.mp4 # Colored Image Video Stream
  episode_${idx}_qpos.png
```



episode_${idx}_base_action.png



3.6.3 Parameter Description

● --datset_dir #Dataset save path

● --task_name #Task name, used as the dataset file name

● --episode_idx a #Action block index number

3.6.4 Dataset Replay

Note:

- **Before proceeding with this section, ensure all terminals are powered off.**
- **Power down and restart the robotic arm, and first remove the main arm's aviation plug!!!**
- **When restarting the robotic arm after power loss, make sure it has returned to the zero position!!!**
- **Then simply plug and unplug the aviation plug of the robotic arm, which will cause power disconnection and arm drop.**

● Package the collected dataset and publish the color image of the dataset along with the robotic arm joint attitude using ROS.

● After releasing this data packet, cobot_magic can subscribe to the message, and the teaching arm will move according to the dataset.

```
# 0 Start roscore roscore **requires unplugging the main arm aviation plug**

# 0.1 activate CAN

bash can_multi_activate.sh # 1 Start roscore to activate the slave arm

roslaunch piper start_ms_piper.launch mode:1 auto_enable:true # 2 Launch a new terminal and activate the virtual environment

conda activate aloha #3 releases color images, main arm, and slave arm messages ## 3.1 Access the collect_data directory

cd cobot_magic/collect_data/ ## 3.2 Replay dataset, recording data from repeated arm execution

python replay_data.py--dataset_dir ~/data--task_name aloha_mobile_dummy--episode_idx 0 # 4 Publish color images and messages from the arm

python replay_data.py --dataset_dir ~/data --task_name aloha_mobile_dummy  --only_pub_master --episode_idx 0
```

Parameter Declaration

- --dataset_dir #Data set save path
- --task_name #Task name, used as the dataset filename
- --episode_idx #Action block index number
- --only_pub_puppet #specifies whether to publish joint pose messages only for the main arm

3.7 Data Training

- ubuntu-20.04,cuda-11.8,cudnn-8.6.0,torch-2.1.1, python-3.8 Testing Passed
- ubuntu-20.04,cuda-11.3,cudnn-8.6.0,torch-1.10.0,python-3.8 Testing Passed

```
# 1 Activate virtual environment
conda activate aloha
```

```
# 2 Training
## 2.1 Enter the aloha-devel directory
cd aloha-devel
## 2.2 Initiation Training
python act/train.py --dataset_dir ~/data0314/ --ckpt_dir train --batch_size 4 --num_epochs 5000 --num_episodes 50
```

Key Parameters and descriptions: currently only ACT model training is supported.

- --dataset_dir #Dataset directory
- --ckpt_dir #Directory for training model save files
- --batch_size #Training batch size
- --num_epochs #training epochs
- --task_name #Task name
- --pretrain_ckpt #Path to pre-trained model
- --ckpt_name #Model name
- --num_episodes #Number of data sets

3.8 Inference Execution

Use the following command for replay and deduction

Note that power supply must be disconnected from all four robotic arms before recharging them.

```
1 Launch roscore

roscore
```

```
2 Start a new terminal to initiate the connection from the arm to the camera

## 2.1 Access the robotic arm directory

cd cobot_magic/Piper_ros_private-ros-noetic/ source devel/setup.bash

## 2.2 Start two slave arms. Before startup, disconnect and restart the robotic arm.

roslaunch piper start_ms_piper.launch mode:=1 auto_enable:=true

# 3 Reasoning ## 3.1 Activating Virtual Environment

conda activate aloha

## 3.2 Execute Inference

##3.2.1 Access the aloha-devel directory

cd aloha-devel

##3.2 Reasoning

python inference.py --ckpt_dir train
```

Note: Exercise caution during reasoning. If abnormal reasoning behavior is detected, immediately terminate the code or disconnect the robotic arm power supply to prevent damage to the robotic arm.

4. Lift Column Control

Note: Before running the lift column node, ensure the robot is in an open area.

Github：GitHub - agilexrobotics/lifting_ws

● File location：/home/agilex/cobot_magic/lifting_ws

● Turn on the terminal and activate the lift column:

```
# source
source devel/setup.bash
# start node
roslaunch lifting_ctrl start_850pro_motor.launch
```

```
#service call
# Initialization (the lift column is initialized only once during each startup)
rosservice call /lifter_1/LiftingMotorService "{val: 0, mode: 1}"
```

● Control lifting operations to ensure cable safety

○ The parameter val in rosservice specifies the lift column height, with a range of [0, -800].

【The top position is set to 0 points, and the bottom limit position is-800】

```
rosservice call /lifter_1/LiftingMotorService "{val: -300, mode: 0}"
```

○ Feedback is provided after normal ascent and descent to the target position. If not reached, feedback-1 is displayed.

```
state:1


rostopic echo /lifter_1/LiftMotorStatePub #50hz
```

Modify configuration file (pre-set by factory):

Accessory file path (json): lifting_ws/lifting_ctrl/scripts/config

```
{
   "portName": "/dev/lifter_1",//串口名字,如果不写,程序会自动寻找
   "initMode":0,//开始上电时的模式,0为位置控制,1为上电开始初始化,建议测试好功能后,将电机装到模组上面并测试好
   "reductionRatio":20000,//执行器末端执行1mm，电机运行多少脉冲(1圈10000脉冲)，单位脉冲数
   "responseTimeout":70,//应答超时时间，单位s
   "motorSpd":800,//电机最大速度设定，单位rpm
   //电机加减速时间(梯形曲线)，
   //(830ABS):单位 1 为 65ms，假设设定为10,则加减速时间为650ms
   //(850pro):单位 1 为 1ms，假设设定为10,则加减速时间为10ms,这个同时还会设置加减速时间
   "motorTime":0,
   "upLimitVal":305,//电机正方向运行最大值，单位mm
   "downLimitVal":0,//电机负方向运行最小值，单位mm
   "initSpd":-800,//初始化时电机运行速度，单位rpm
   "initPos":0,//初始化后执行器要到达的位置，单位mm
   "motorStallCurrent":17,//电机堵转电流限幅,单位A
}
```

5. Navigation Section

Navigation is optional；

For Navis navigation, refer to Appendix 3 "Navis User Manual".

https://agilexsupport.yuque.com/staff-hso6mo/peoot3/svogy1ekpbzieqm8?singleDoc# Attachment 3: NAVIS Navigation Usage

For open-source navigation systems purchased, please refer to Attachment 4: Open-Source Navigation User Manual.

https://agilexsupport.yuque.com/staff-hso6mo/peoot3/wotxkwlo2sow6t6x?singleDoc# Attachment 4-Open-Source Navigation User Guide

6. Q&A

> Q1: When running the ROS node, an unknown error occurred, or the camera name was incorrect, or the robotic arm message was abnormal.
>
> A: Run pkill-f ros to terminate all nodes first, then restart ROS and scripts according to the above procedure.
>
> Q2: During remote operation, violent or incorrect maneuvers may cause a robotic arm to reach its limit position, resulting in malfunction.
>
> A: For a normally functioning robotic arm in zero position, manually support the non-functional arm, then disconnect all terminals of the robotic arm or power it off, followed by a restart.
>
> Q3: After starting the chassis, the odometer displays readings, but when moving the vehicle, the odometer data remains at zero.
>
> A: The chassis can be enabled, followed by restarting the chassis program.
>
> Q4: How to check the topic frequency in ROS
>
> A: Enter the topic name 'rostopic hz' in the terminal to print its frequency.
>
> Q5: Run the collect_data.py script. If the code encounters an error
>
> A: Use the built-in image environment of cobot.magic

Q6: When running the collect_data.py script, the terminal keeps displaying 'sync fail'

A: Check if the ROS topics collected by collect_data.py are generating normal outputs. If not, verify whether each topic name corresponds to its actual content.

Q7: The terminal error message is as follows:

RLException: 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

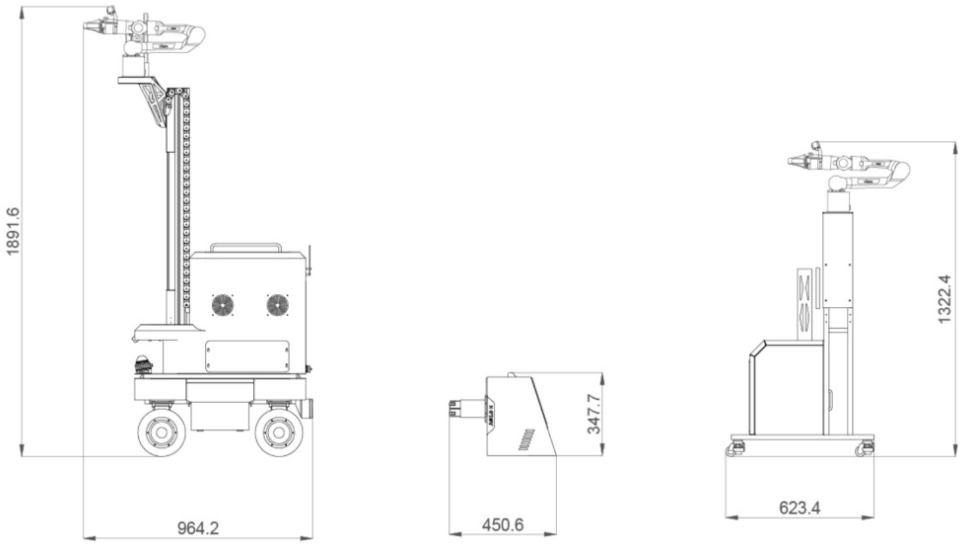
A: This issue is caused by repeated ROSCORE startup attempts. Restart the computer or log out of the system.

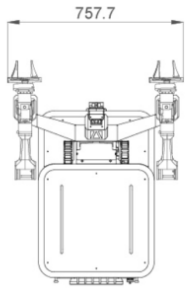
Q8: The odometer message/ODOM data printed by the chassis ROS shows no change in the numbers.

A: Verify whether the chassis CAN2USB cable is properly connected and whether the chassis CAN bus is enabled.

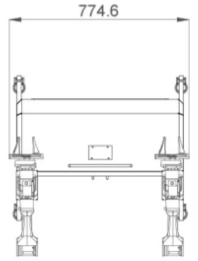
7. System Image Link：

Appendix I: Dimensional Drawings




757.7


210.0


774.6