Dobot X-Trainer (Lightweight Base) User Guide

Issue: V1.1
Date: 2024-07-31

Shenzhen Yuejiang Technology Co., Ltd.

Disclaimer

To the maximum extent permitted by applicable law, the products described (including its hardware, software and firmware, etc.) in this document are provided **AS IS**, which may have flaws, errors or faults. Dobot makes no warranties of any kind, express or implied, including but not limited to, merchantability, satisfaction of quality, fitness for a particular purpose and non-infringement of third party rights. In no event will Dobot be liable for any special, incidental, consequential or indirect damages resulting from the use of our products and documents.

Before using our product, please thoroughly read and understand the contents of this document and related technical documents that are published online, to ensure that the robot arm is used on the premise of fully understanding the robot arm and related knowledge. Please use this document with technical guidance from professionals. Even if follow this document or any other related instructions, damages or losses may happen in the using process. Dobot shall not be considered as a guarantee regarding all security information contained in this document.

The user has the responsibility to make sure following the relevant practical laws and regulations of the country, in order that there is no significant danger in the use of the robot arm.

Shenzhen Yuejiang Technology Co., Ltd.

Address: Room 1003, Building 2, Chongwen Garden, Nanshan iPark, Liuxian Blvd, Nanshan District, Shenzhen, Guangdong Province, China

Website: www.dobot-robots.com

Preface

Purpose

This document is used to guide users to install and configure the Dobot X-Trainer AI Robot Operating Platform (Lightweight Base).

Intended audience

This document is intended for:

- Customer
- Sales Engineer
- Installation and Commissioning Engineer
- Technical Support Engineer

Revision history

Date	Version	Revised content
2024/07/31	V1.1	The second release, content optimization
2024/05/24	V1.0	The first release

Symbol conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
DANGER	Indicates a hazard with a high level of risk which, if not avoided, could result in death or serious injury.
WARNING	Indicates a hazard with a medium level or low level of risk which, if not avoided, could result in minor or moderate injury, robot arm damage.
NOTICE	Indicates a potentially hazardous situation which, if not avoided, could result in robot arm damage, data loss, or unanticipated result.
NOTE	Provides additional information to emphasize or supplement important points in the main text.

Contents

1. Product Introduction

1.1 Overview

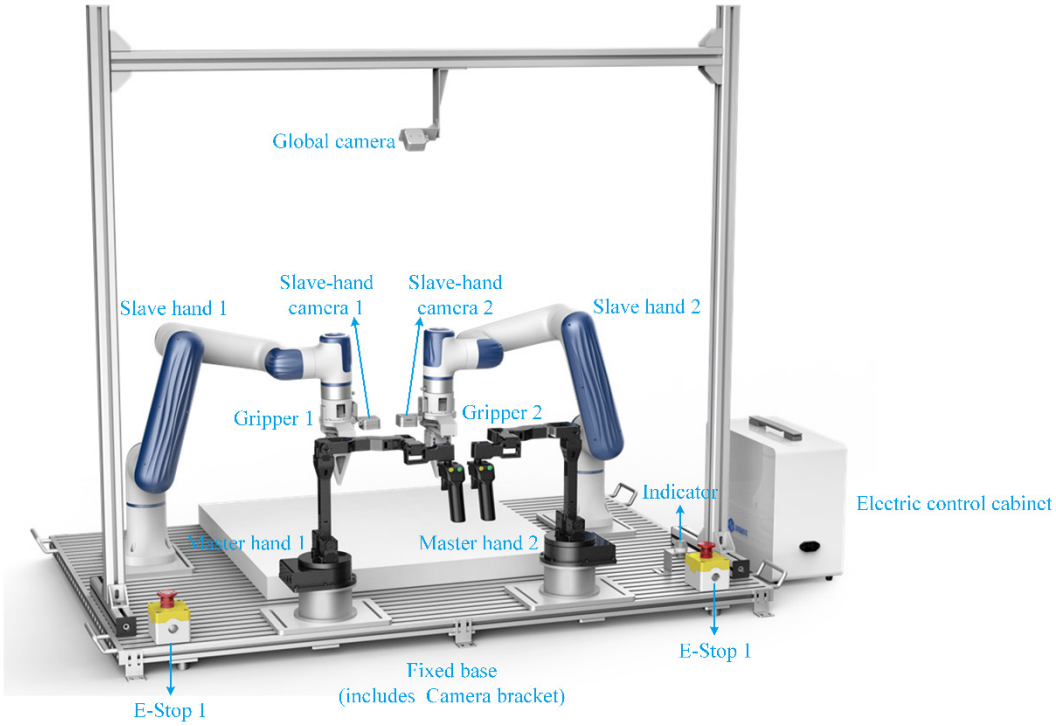
At present, research on Artificial General Intelligence (AGI) at home and abroad has entered a new upsurge, with Embodied AI recognized as a crucial step towards AGI. A promising approach to developing Embodied AI is imitation learning from human demonstrations. This kind of “behavior cloning” allows robots to learn a variety of primitive skills, from simple pick-and-place tasks to more sophisticated operations.

The kit offers a platform for bimanual teleoperation, enabling intuitive, dexterous, and precise control of slave hands by directly manipulating the master hands to achieve reliable, efficient and high-quality demonstration data collection. The kit also provides basic demonstration program for data collection, algorithm training, and inference, along with a comprehensive API for the entire platform. This API includes interfaces for:

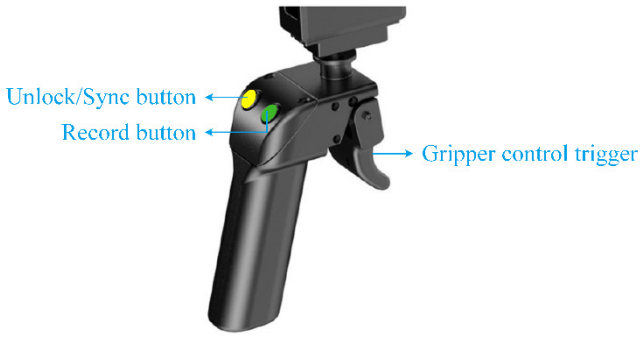
- Master-hand joint angle reading;
- Master-hand locking control;
- Slave-hand joint angle and Cartesian coordinate reading/control;
- Gripper stroke control;
- Camera RGB/depth image acquisition.

With this platform and secondary development interfaces, flexible and extensive research on robot applications can be conducted. For example, Household service robots help people with daily chores such as cooking, cleaning and organizing. Industrial automation robots perform tasks such as assembly and quality inspection.

1.2 Components



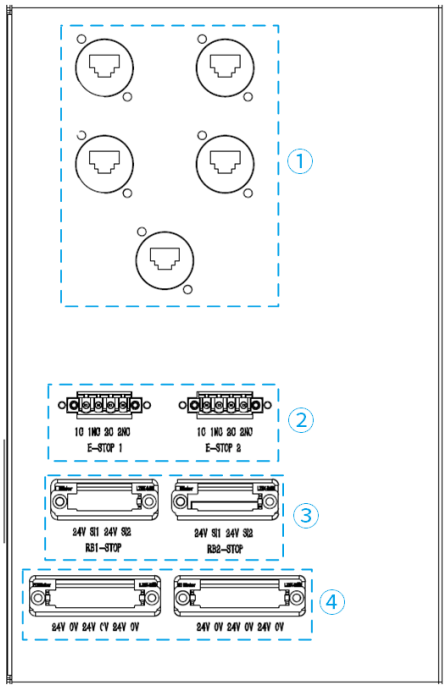
- **Master hands**: High-precision 6-DOF (degrees of freedom) master hands, one for each side, each equipped with an operation handle (see the figure bellow) at the end.



- **Slave hands**: Nova 2 robot arms (controller version 3.5.7 or higher), one for each side. The salve hand will follow the master hand after synchronizing with it.
- **Grippers**: One at the end of each slave hand with a maximum stroke of 95mm. The gripper is controlled by the trigger at the end of the master hand after synchronizing with it.
- **Cameras**: Three Intel RealSense D405 cameras. One is mounted on the camera bracket as a global camera, and each slave hand has one camera mounted at its end.
- **E-Stop switches**: The emergency stop switch of the slave hand. Pressing any one will brake both slave hands in an emergency.
- **Indicator**: Tricolor indicator light, with the following meanings.
  - Light-off: Master and slave hands are not synchronized.
  - Yellow: Master and slave hands are synchronized but not recording data.
  - Green: Data recording.
  - Red: Error.
- **Electric control cabinet**: It uses mains power as the power input. For details on the interfaces at the rear of the cabinet, see Electric control cabinet interface.

- **Fixed base**: The mounting base for X-Trainer, including the camera bracket.

1.3 Electric control cabinet interface



- ①: 5 network interfaces, for connecting the slave-hand controller (CCBOX) and computer.
- ②: 2 E-Stop switch interfaces.
- ③: 2 robot E-Stop output interfaces, for connecting to the E-Stop input interfaces of the slave-hand controller.
- ④: 24V output interfaces (2 rows, 6 sets in total), mainly used to power the slave-hand grippers.

2. Hardware Installation

2.1 Installation environment

To maintain the X-Trainer performance and to ensure the safety, please place the X-Trainer in an environment with the following conditions.

- Install indoors with good ventilation.
- Keep away from excessive vibration and shock.
- Keep away from direct sunlight and radiant heat.
- Keep away from dust, oily smoke, salinity, metal powder, corrosive gases, and other contaminants.
- Do not use in a closed environment. A closed environment may cause high temperature of the controller and shorten its service life.
- Keep away from flammable.
- Keep away from cutting and grinding fluids.
- Keep away from sources of electromagnetic interference.

2.2 Unpacking

Before unpacking, please carefully check the outer packaging for signs of damage in transportation. If any damage is found, please unpack in the presence of shipping company staff.

When unpacking, please check the attached shipping list to ensure that all contents are included and intact. If anything is missing or damaged, please contact your supplier.

After unpacking, please save all packaging materials for future transportation.

In addition to the standard parts provided with the X-Trainer, users need to prepare the following parts.

- Prepare a computer for teleoperation and algorithm training (hereafter referred to as the **training host**), with the following specifications (lower specifications are acceptable for teleoperation only):

Configuration item	**Recommended**	
	Teleoperation Only	**Algorithm training**
Graphics card	GTX1050	RTX4090, 24GB VRAM (minimum: RTX2080 Ti, 11GB VRAM)
CPU	Intel i5-9300H	Intel i9-14900KF
RAM	16 GB	32 GB
Hard drive	1 TB	2TB SSD
Operating system	Ubuntu 20.04	

- Prepare another computer (Windows 7/10/11 operating system, you can also install dual systems on the training host) or a tablet (Android 10 or above, or iOS 10 or above) to install the control software for the slave hand (Nova2) (hereafter referred to as the **slave-hand operation terminal**), with the following specifications:

Terminal type	PC	Android tablet	iPad
Operating system	Windows7/10/11	Android 10 and above	iOS 10 and above
Control software	DobotStudio Pro	Dobot CRStudio	Dobot CRStudio
Minimum	CPU: Intel Core i3	CPU: 4-core	iPad 5th-generation

configuration	RAM: 4G Hard drive space: 128GB Network card: Gigabit network card VRAM: 1G	RAM: 2G Storage space: 32G Display: 8-inch	and above
Communication mode	LAN/WiFi	WiFi	WiFi

* If WiFi communication is required, please select the WiFi module for Nova when purchasing the X-Trainer.

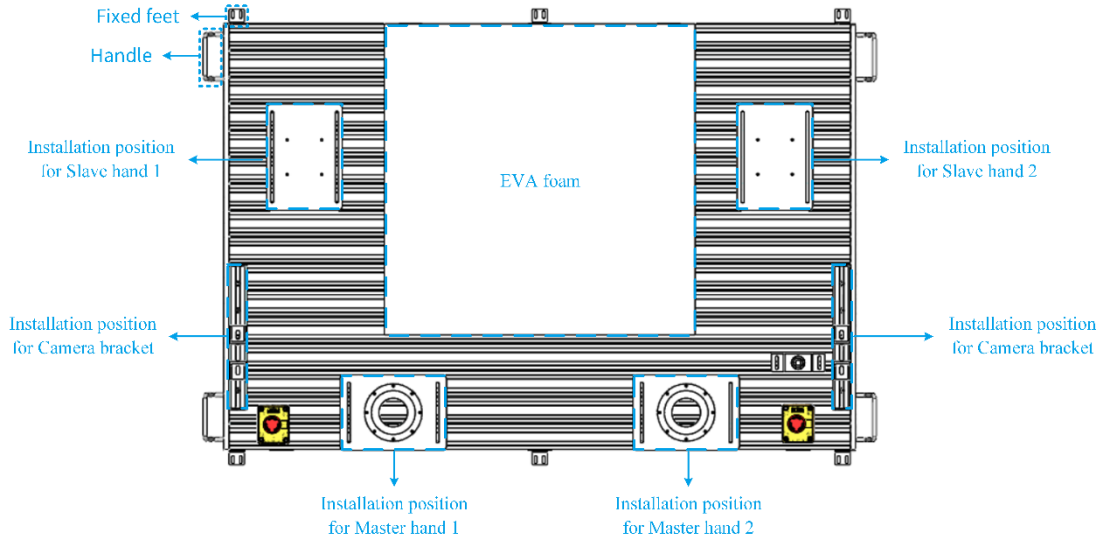
- Prepare a power strip that meets local standards, with at least 5 additional sockets available for the X-Trainer in addition to the socket for computer.

2.3 Installation steps

2.3.1 Overall layout

The lightweight-base X-Trainer base measures 1400mm x 1000mm (excluding handles), and the overall layout is shown below.

The EVA foam is included in the shipping list and should be arranged according to the figure below. The 6 fixed feet on both sides are shipped as separate parts. If the user needs to fix the base to the tabletop, use M5*10 screws to install the fixed feet.



2.3.2 Installing electric control cabinet

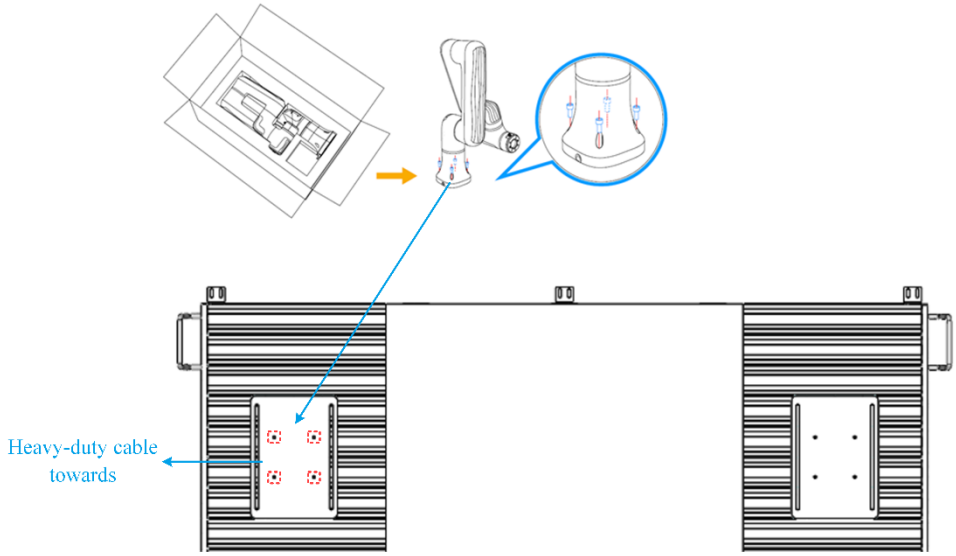
Take out the electric control cabinet from its packaging box and place it next to the base. Refer to the subsequent instructions for the wiring of the electric control cabinet.

2.3.3 Installing slave hand

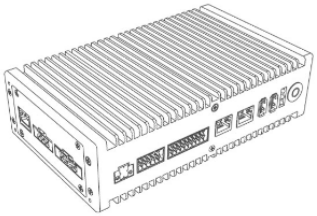
Tools needed: Allen wrench * 2 (M3/M6), M3*8 screws.

1. Remove 4 M6 hex screws from the mounting plate of **slave hand 1** (highlighted in red in the figure below). Take out one slave hand (Nova 2) from its packaging and use the removed screws to fasten it onto the mounting plate, **with the heavy-duty cable facing towards the left**. When moving the robot from its packaging box to the mounting

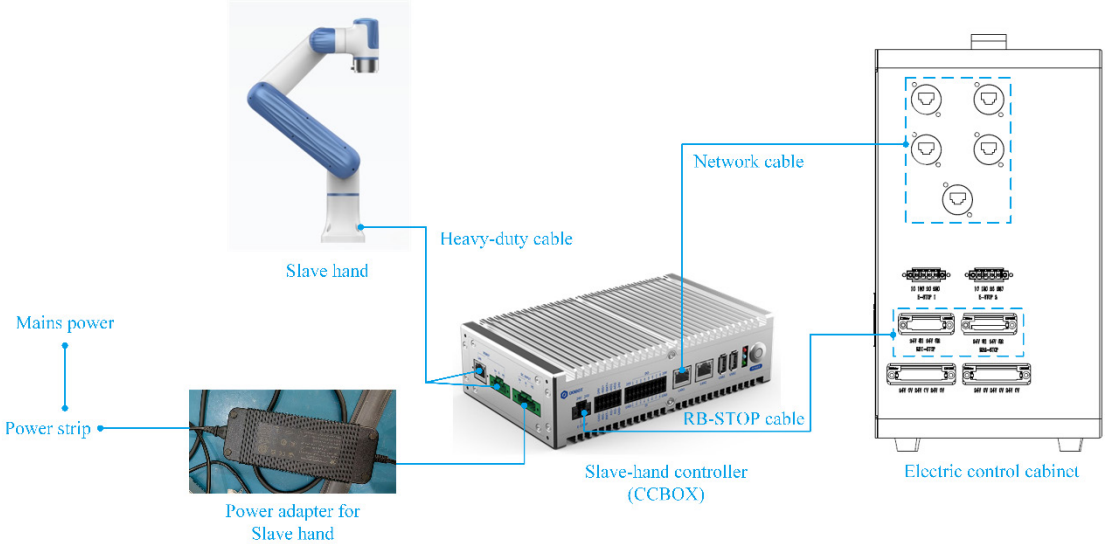
position, hold the robot until all bolts on the robot base are fastened.



2. Place the slave-hand controller (CCBOX, as shown in the figure below) near the slave hand, leaving at least 50mm of space on each side for heat dissipation.



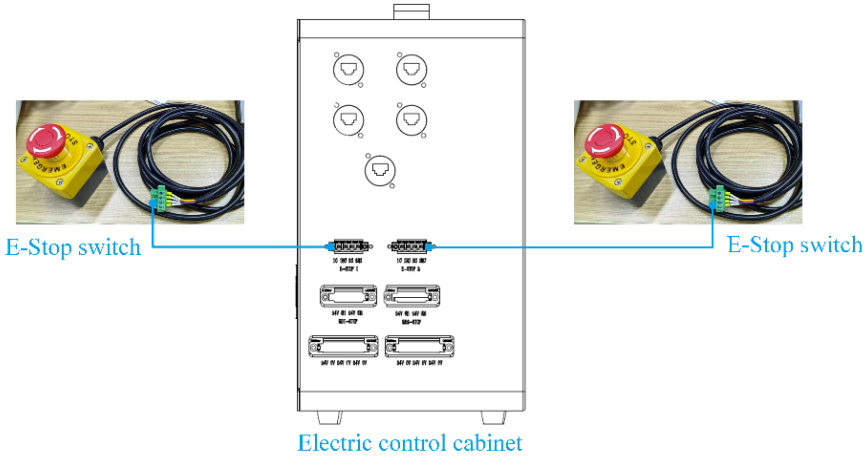
3. Refer to the above steps to install the other slave hand to the installation position for Slave hand 2, **with the heavy-duty cable facing towards the right**. Both slave hands should be symmetrically mirrored.
4. Refer to the figure below to connect the cables for the slave hands (both slave hands are connected in the same way).



The RB-STOP cable is shown below, with the black terminal connected to the CCBOX and the green terminal connected to the electric control cabinet.

2.3.4 Installing E-Stop switch

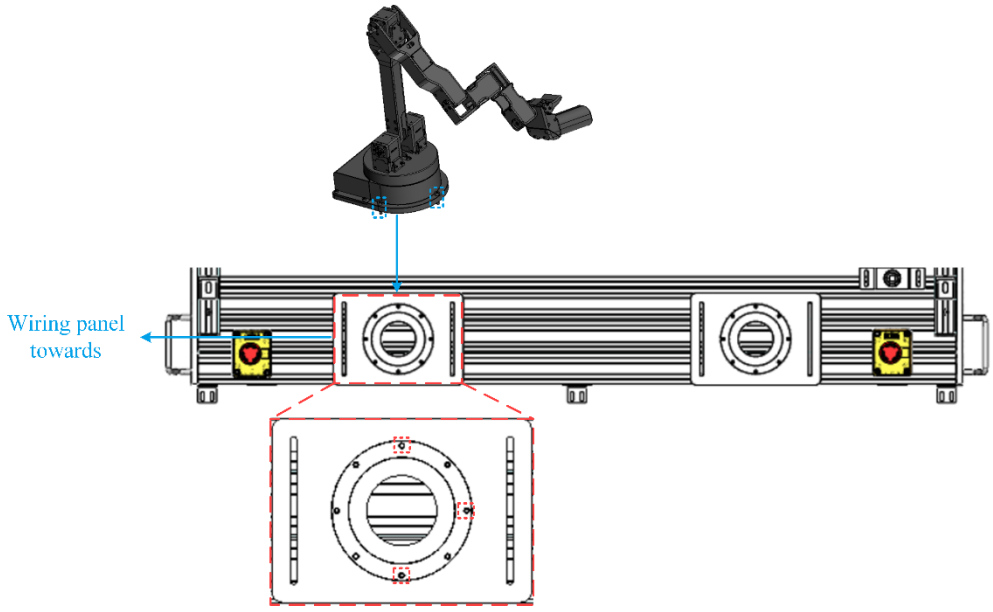
Place the E-Stop switch on the mounting base and connect the cable to the E-Stop switch interface of the electric control cabinet.



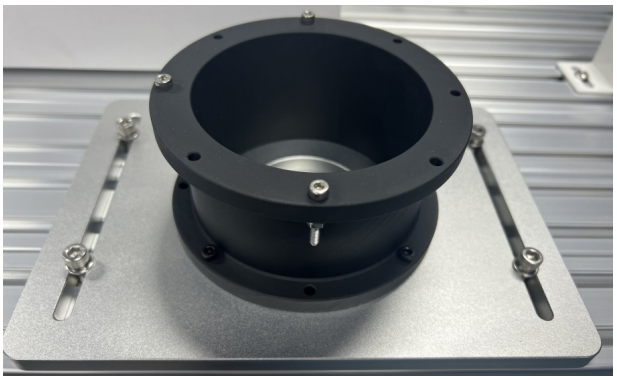
2.3.5 Installing master hand

Tools needed: Allen wrench * 1 (M5).

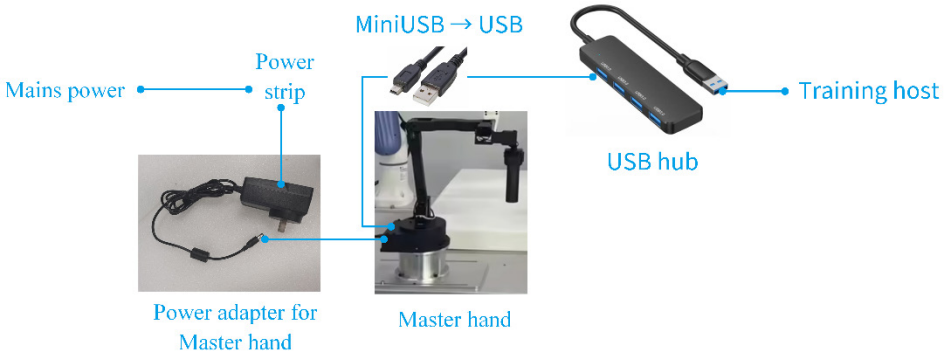
1. Remove 3 M5 hex screws from the mounting base of Master hand 1 (highlighted in red in the figure below). Take out one master hand from its packaging box and use the removed screws to fasten it onto the mounting plate, **with the wiring panel facing towards the left**.



The cylindrical base shown in the figure below is pre-installed on the master hand mounting plate. Depending on the site requirements (e.g., if the operator's seating height is lower), it can be removed, and the master hand can be directly mounted onto the lower mounting plate.



2. Refer to the above steps to install the other master hand to the installation position for Master hand 2, **with the wiring panel facing towards the right**. Both master hands should be symmetrically mirrored.
3. Refer to the figure below to connect the power cable and data cable for the master hands (both master hands are connected in the same way).

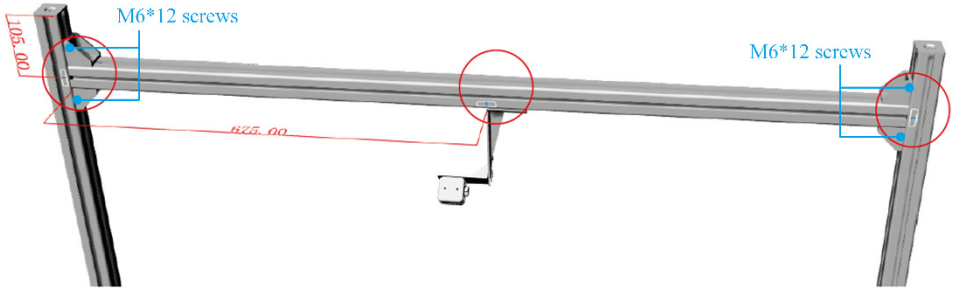


2.3.6 Installing global camera

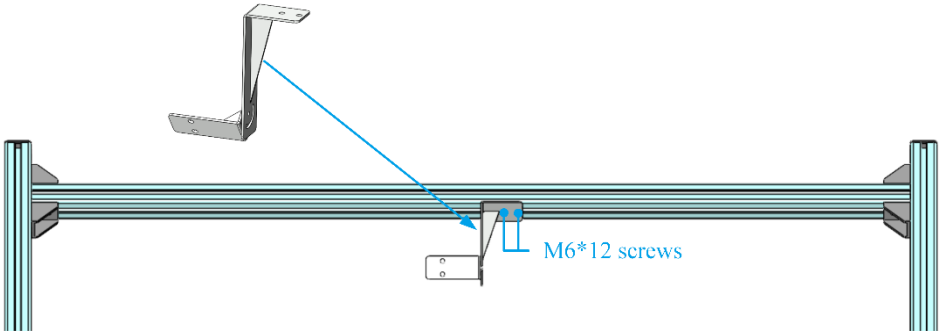
Tools needed: Allen wrench * 2 (M4/M6), M6*12 screws, M6*8 screws, M6 flat gaskets.

Before installing the camera, it is recommended to record the ID (as shown in the figure below) and installation position of each camera for future configuration.

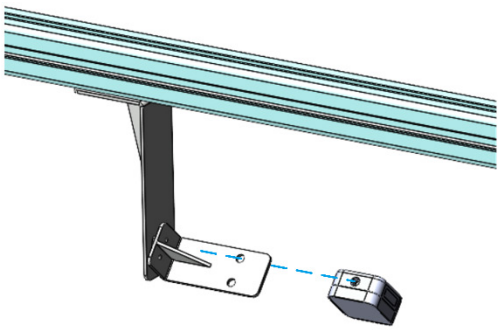
1. The camera bracket consists of two vertical bars (1.2m each) and one horizontal bar (1.32m). Use 4 M6*12 screws to assemble the horizontal and vertical bars. During assembly, align the stickers as shown in the figure below. Both the horizontal and vertical bars have built-in T-nuts. When assembling, slide the nuts to the appropriate positions to fix the screws.

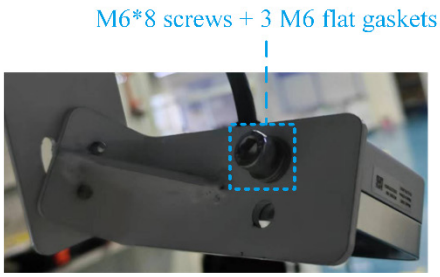


2. Take out the camera mounting kit (as shown in the figure below). Align the vertical sheet metal with the sticker on the horizontal bar (as shown in the figure above), and use 2 M6*12 screws to fix it.

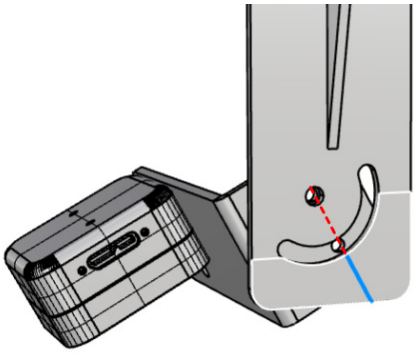


3. Use 1 M6*8 screws and 3 M6 flat gaskets to install the global camera.

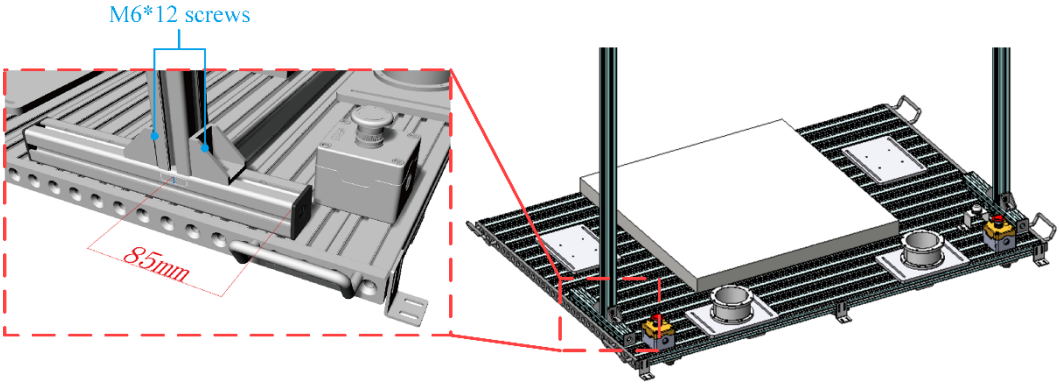




After installation, refer to the figure below to verify the angle of the camera bracket (with the center of the two screw holes located on the red line). If not aligned, adjust it manually. The screw specification here is M4*10.

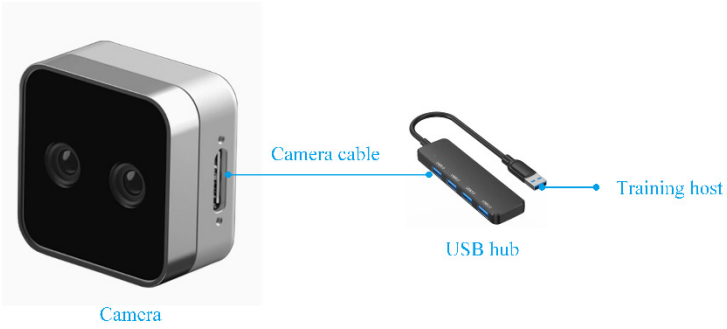


4. Stand up the camera bracket and insert it between the two L-shaped parts at the installation position for camera bracket. The centerline of the vertical bar should align with the sticker on the L-shaped part. Then, use 2 M6*12 screws to fix it.



5. Connect the camera cable as shown in the figure below.

   Ensure that the camera and master hands are connected to different USB hubs, and the USB hub for the camera is connected to a USB 3.0 interface on the computer. Make sure the camera cable is securely connected to prevent any data collection issues due to a poor connection.

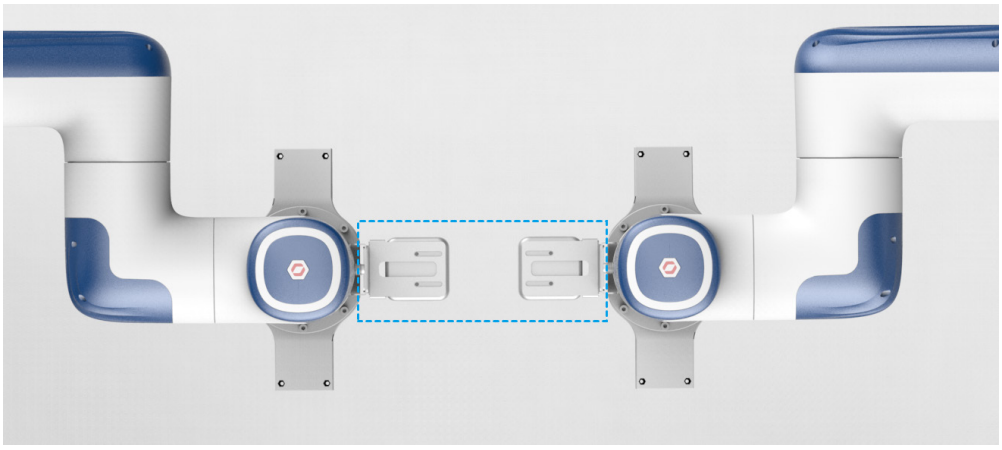


The camera cable is shown below.

2.3.7 Installing slave-hand gripper and camera

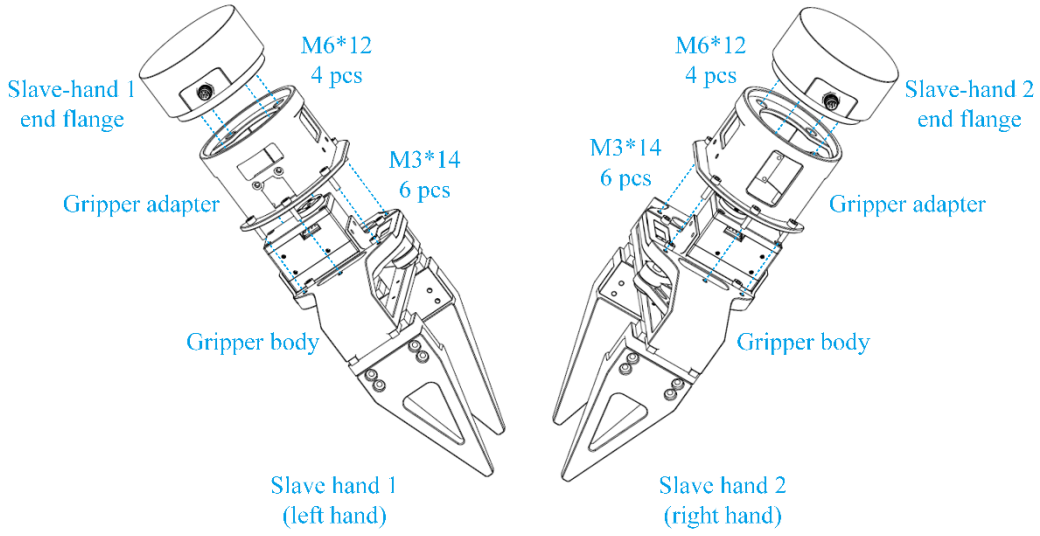
Tools needed: Allen wrench * 2 (M3/M6), M6*12 screws, M3*6 screws, camera calibration card * 1.

Before installing the slave-hand gripper and camera, refer to Powering on slave hand and Slave hand settings. Move **slave hand 1** and **slave hand 2** to their initial positions, then power them off before proceeding with the installation. After installation, the two cameras should be positioned as shown in the figure below.

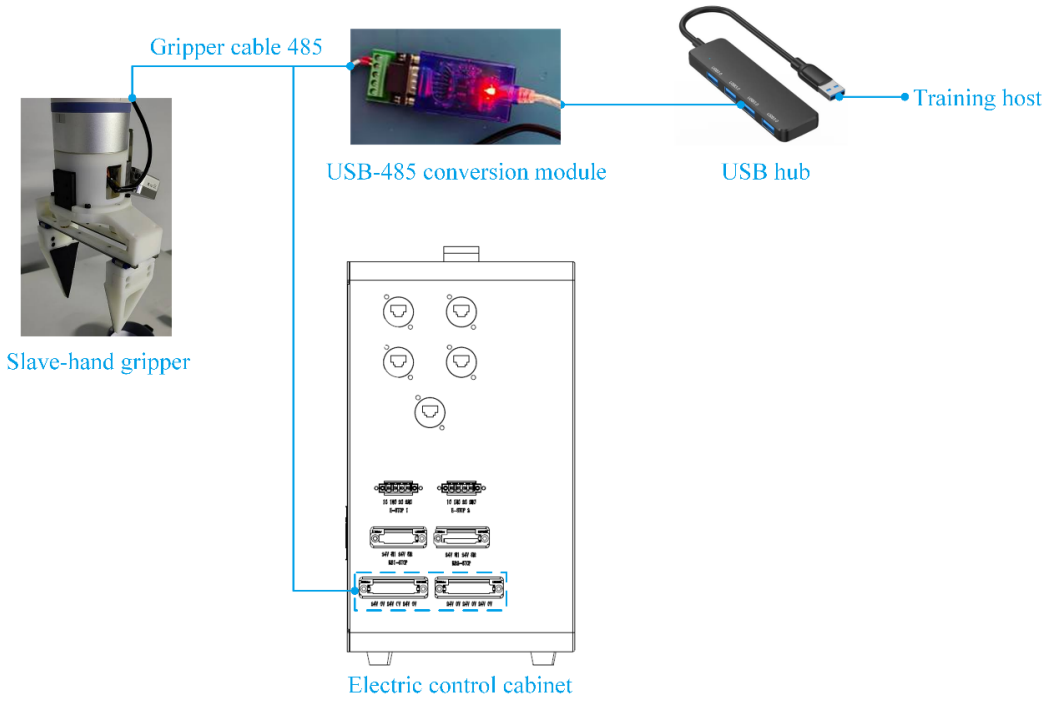


Installing slave-hand gripper

1. Take out the gripper from its packaging box, remove 6 M3*14 screws, and separate the gripper body from the adapter.
2. Use 4 M6*12 screws to secure the gripper adapter to the end flange of the slave hand.
3. Use 6 M3*14 screws to fix the gripper body to the adapter. **The two slave-hand grippers are installed in different orientations. Please refer to the figure below and pay attention to the position of the end-flange aviation socket, the gripper adapter, and the gripper body.**



4. Connect the gripper cables as shown in the figure below.

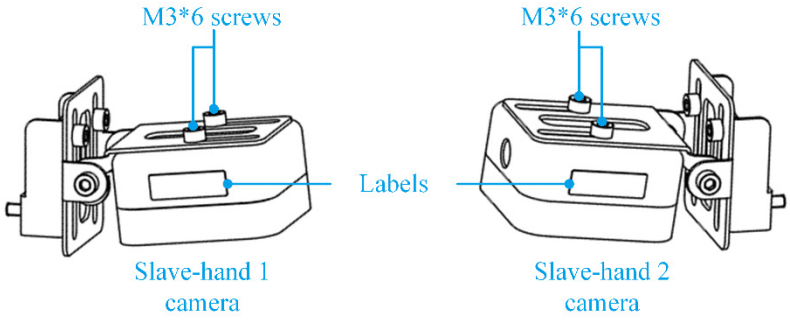


The gripper cable 485 is shown in the figure below, and the wiring instructions are listed in the table below.

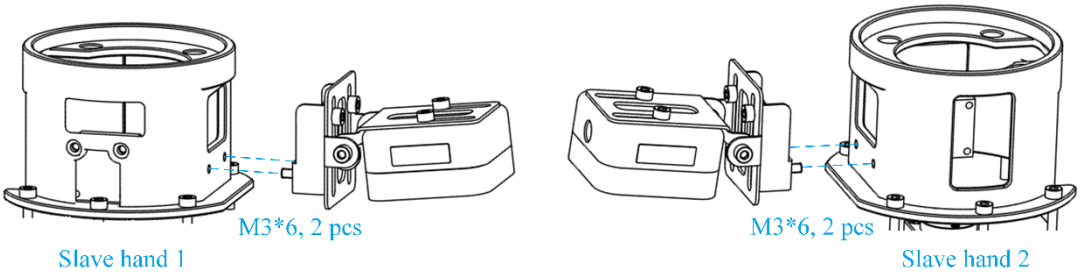
Cable label (Color)	Peer interface	Peer device
A (Green)	T/R+	USB-485 conversion module
B (White)	T/R-	
24V (Red)	24V	Electric control cabinet
GND (Black)	0V	
PE (Black)	N/A	

Installing slave-hand camera

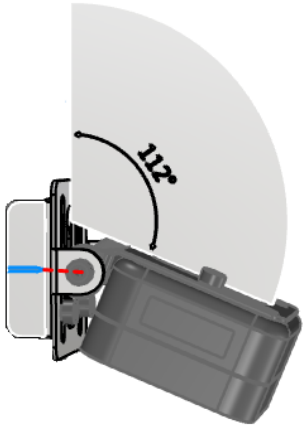
1. Use 2 M3*6 screws to mount the camera onto the camera bracket. **The cameras on the two slave hands need to be installed in opposite orientations. Please refer to the figure below and ensure that when the two cameras are facing each other, their labels are oriented towards the user.**



2. Use 2 M3*6 screws to mount the camera bracket onto the gripper adapter.



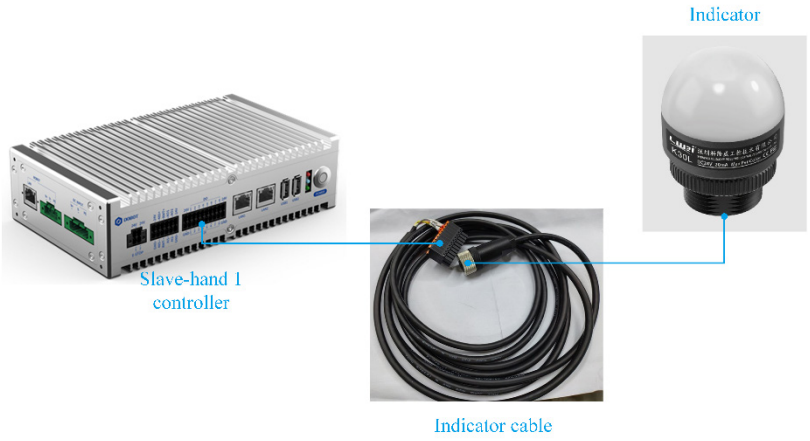
After installation, adjust the bracket so that the center of the screw hole on the side of the bracket aligns with the red line (see figure below). Use the provided calibration card to adjust the angle, ensuring the upper angle is 112°, as shown below.



3. Connect the camera cables. The connection method is the same as for the global camera. **Ensure that the camera and master hands are connected to different USB hubs, and the USB hub for the camera is connected to a USB 3.0 interface on the computer. Make sure the camera cable is securely connected to prevent any data collection issues due to a poor connection.**

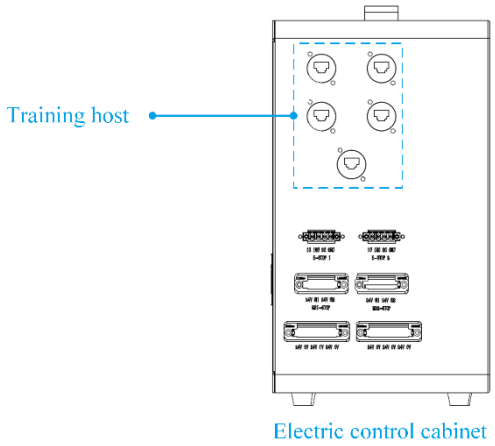
2.3.8 Connecting the indicator

The indicator is factory installed on the base. Use the indicator cable to connect the Indicator and the Slave-hand 1 controller. Ensure the cable labels align with the silkscreens on the controller, and organize the cables to the left side of the terminal after connecting.



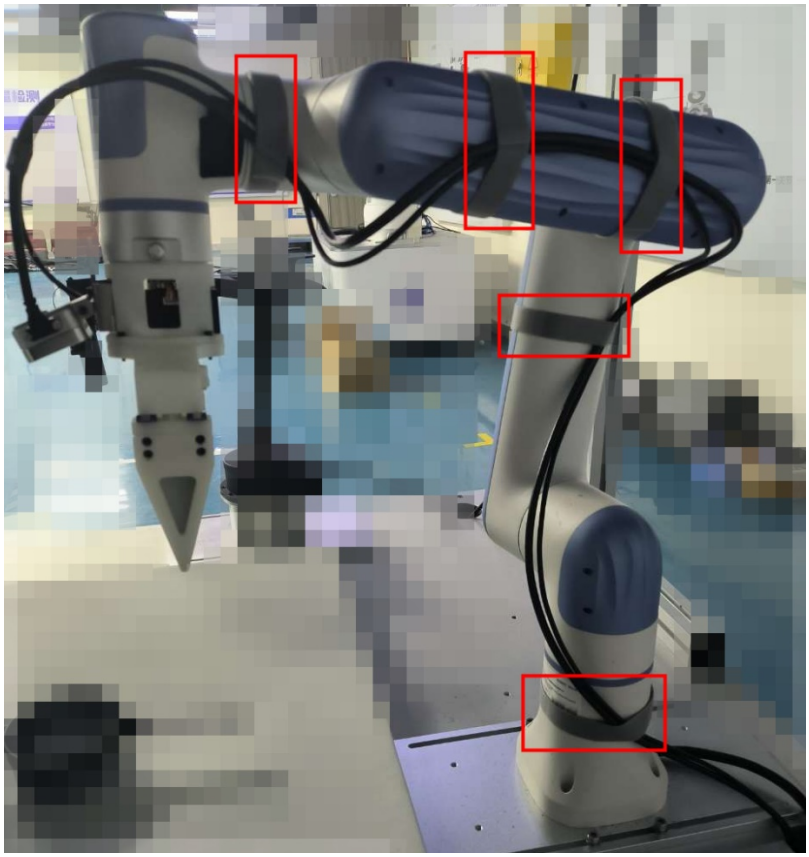
2.3.9 Connecting the training host

Use the network cable to connect the training host to the electric control cabinet.



2.3.10 Organizing the cables

After completing the installation and connection, organize the cables as shown below. Use the Velcro provided (shipped as a whole roll, to be cut by the user) to secure the gripper and camera cables to the slave hand, preventing cable strain when the slave hand is moving.

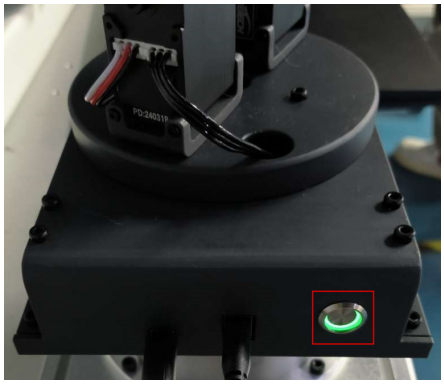


The shipping list also includes a pack of plastic ties for organizing additional cables.

2.3.11 Powering on device

Powering on master hand

The power switch for the master hand is located on the base. Press the switch to power the master hand on, press again to power the master hand off.



Powering on slave hand

The power switch for the slave hand is located on the controller (CCBOX). Short-press to power the slave hand on, long-press (over 3 seconds) to power the slave hand off.

Powering on slave-hand gripper

The slave-hand gripper is powered by the electric control cabinet. Connect the electric control cabinet to the power supply to power it on. The power interface of the electric control cabinet is located on the front side (opposite to the network interface).

Powering on camera

The cameras are powered by the training host via USB. The camera powers on automatically after the training host is powered on.

3. Operating Environment Configuration

Unless otherwise specified, the following operations are performed on the training host. If you encounter any issues during configuration and use that are not covered in this document, please refer to the *Dobot X-Trainer Operation Guide and FAQ* for solutions.

3.1 Operating system

It is recommended to use a Linux system: either Ubuntu 20.04 or 23.10.

NOTE

- Dobot has not verified the configuration and use of the training and inference environment on other versions of Linux or on Windows systems, so we cannot provide corresponding guidance.
- Compared to Linux systems, master-slave teleoperation has higher latency on Windows systems (approximately an additional 10ms).
- Avoid using Chinese characters when naming folders. It is recommended to use English letters and numbers.

3.2 Installing CUDA and cuDNN

NOTE

You can skip this section if only the master-slave teleoperation function is required.

Before installing CUDA, ensure that you have the latest graphics driver installed.

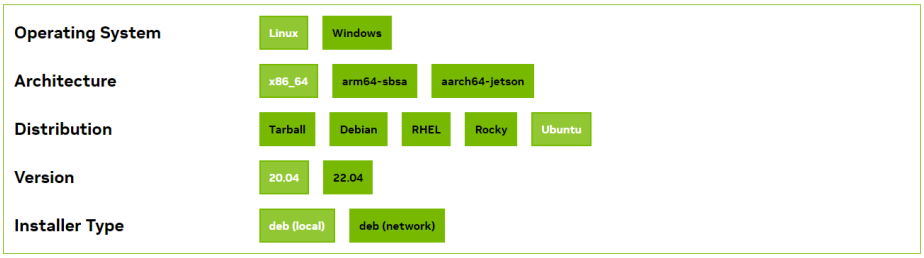
Here takes downloading and installing CUDA 11.8 on Ubuntu 20.04 as an example. If your graphics card does not support CUDA 11.8, install the appropriate versions of the graphics driver, CUDA, cuDNN, and torch based on your graphics card specifications.

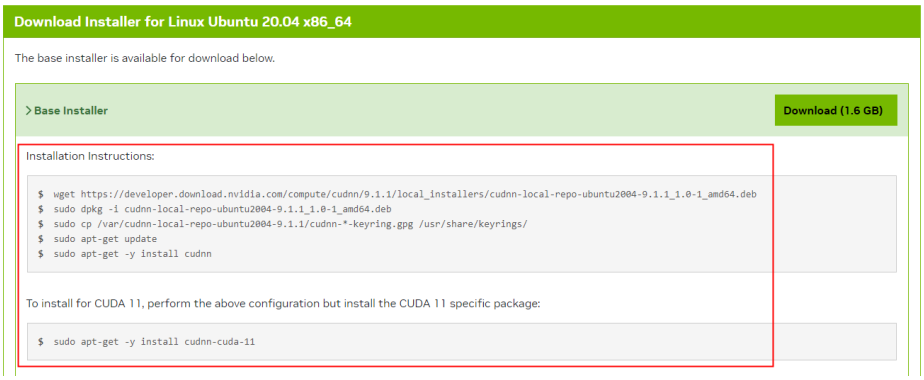
1. Visit the CUDA website (https://developer.nvidia.com/cuda-11-8-0-download-archive), select based on your operating system, and obtain the download and installation commands for CUDA runfile.

2. Open a terminal and enter the download command obtained in the previous step.

```
wget https://developer.download.nvidia.com/compute/cuda/11.8.0/local_installers/cuda_11.8.0_520.61.05_linux.run
```


3. After the download is completed, execute the installation command.
   sudo sh cuda_11.8.0_520.61.05_linux.run
4. Follow the on-screen prompts to complete the installation. (Since the GPU driver version included with CUDA is quite old, please uncheck the option to install the GPU driver when installing CUDA; otherwise, the GPU may not function properly).
5. Use gedit to open the user configuration file in the terminal (if gedit is not installed, please install it and learn how to use it, or use your preferred text editor, such as vim).
   sudo gedit ~/.bashrc
6. Add the following content to the file and save it. The example below uses the default installation path. If you have modified the the installation path for CUDA, modify the path in the command accordingly.
   export PATH=/usr/local/cuda-11.8/bin:$PATH
   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-11.8/lib64
7. Restart the operating system.
8. Verify whether CUDA has been successfully installed. If the version information of nvcc is output after executing the command, the installation is successful; otherwise, reinstall CUDA.
   nvcc -V
9. Visit the cuDNN website (https://developer.nvidia.com/cudnn-downloads), select based on your operating system, and obtain the download and installation commands for cuDNN.





10. Follow the commands provided on the cuDNN website to complete the download and installation of cuDNN.

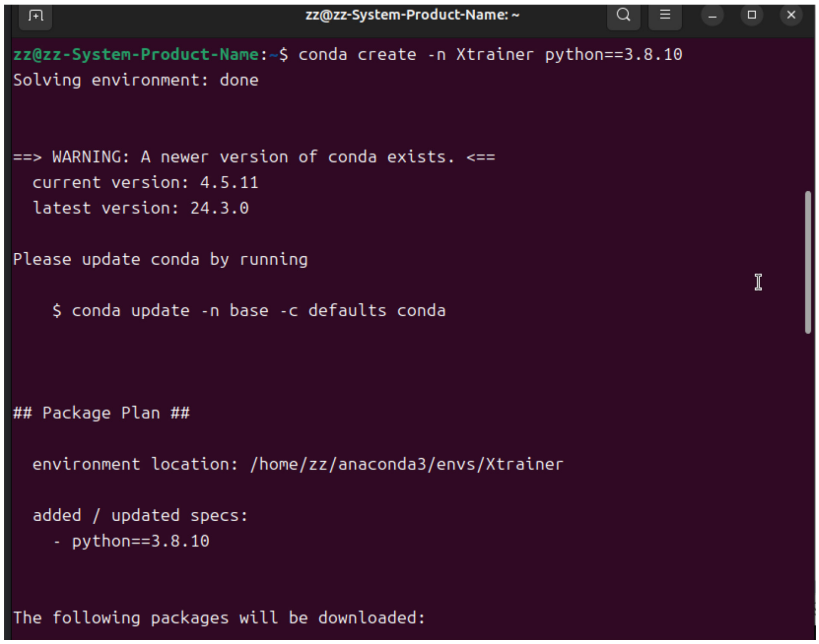
3.3 Installing Anaconda

1. Download the Anaconda installer from the official website (https://www.anaconda.com/download#download-section) or mirror site (https://mirrors.tuna.tsinghua.edu.cn/help/anaconda/).
2. Go to your download directory, open a terminal, and run the installer package (the command here is an example only).
   bash Anaconda3-2020.07-Linux-x86_64.sh
3. Follow the on-screen prompts to complete the installation.

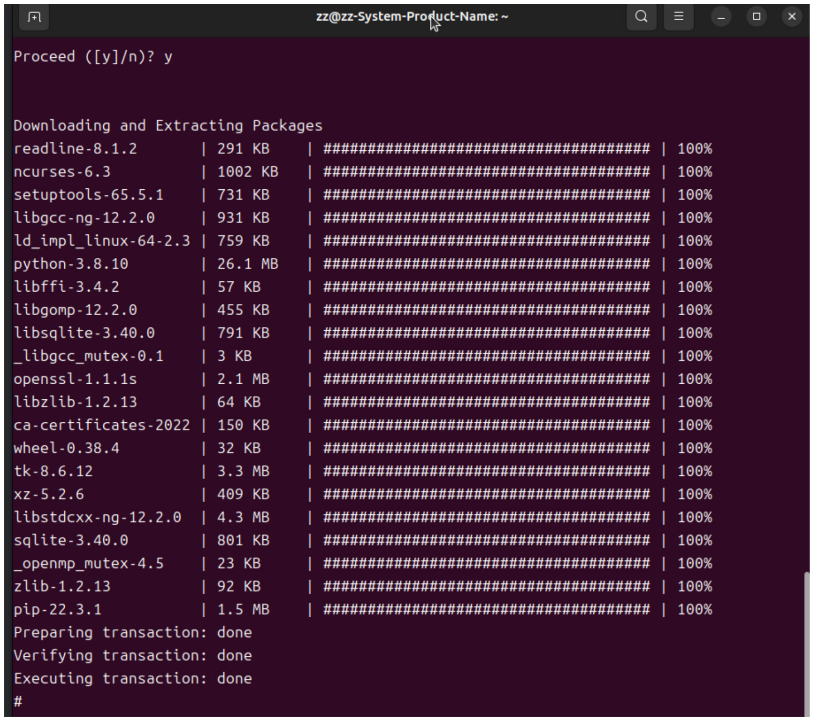
4. Use gedit to open the user configuration file in the terminal.
   `sudo gedit ~/.bashrc`
5. Add the following content to the file and save it. Modify the path in the command to the actual installation path of Anaconda.
   `export PATH="/home/dobot/anaconda3/bin:$PATH"`
6. Execute the following command in the terminal and apply the environment variables.
   `source ~/.bashrc`
7. Verify whether Anaconda has been successfully installed. If the version information of conda is output after executing the command, the installation is successful; otherwise, reinstall Anaconda.
   `conda -V`

3.4 Configuring the virtual environment

1. Open a terminal or Anaconda command window and create a new virtual environment with Python 3.8.10.
   For example, create a virtual environment named “Xtrainer”.
   `conda create -n Xtrainer python==3.8.10`

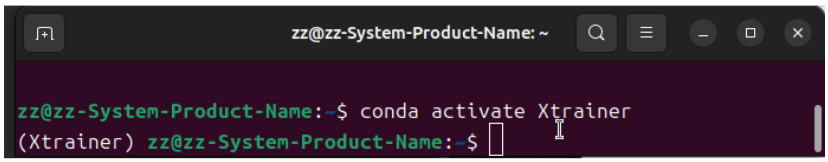


   Confirm the download by typing “y” when prompted.



2. Activate and enter the virtual environment. After executing the command, (Xtrainer) should appear, indicating that you have entered the virtual environment.

```
conda activate Xtrainer
```



3. Install pytorch 2.0.1 and torchvision 0.15.2.

> **NOTE**
>
> You can skip this step if only the master-slave teleoperation function is required.

```
pip install torch==2.0.1 torchvision==0.15.2 torchaudio==2.0.2 --index-url https://download.pytorch.org/whl/cu118
```

After installation, run the following commands in turn to verify whether torch/torchvision has been successfully installed.

```
python
import torch
import torchvision
torch.cuda.is_available()
exit()
```

If `torch.cuda.is_available()` returns `True`, the installation is successful. If not, check if CUDA and cuDNN are installed and if the correct version of torch/torchvision is installed. If you installed the wrong version of torch/torchvision, use the pip uninstall command to remove the incorrect version and reinstall the correct one.

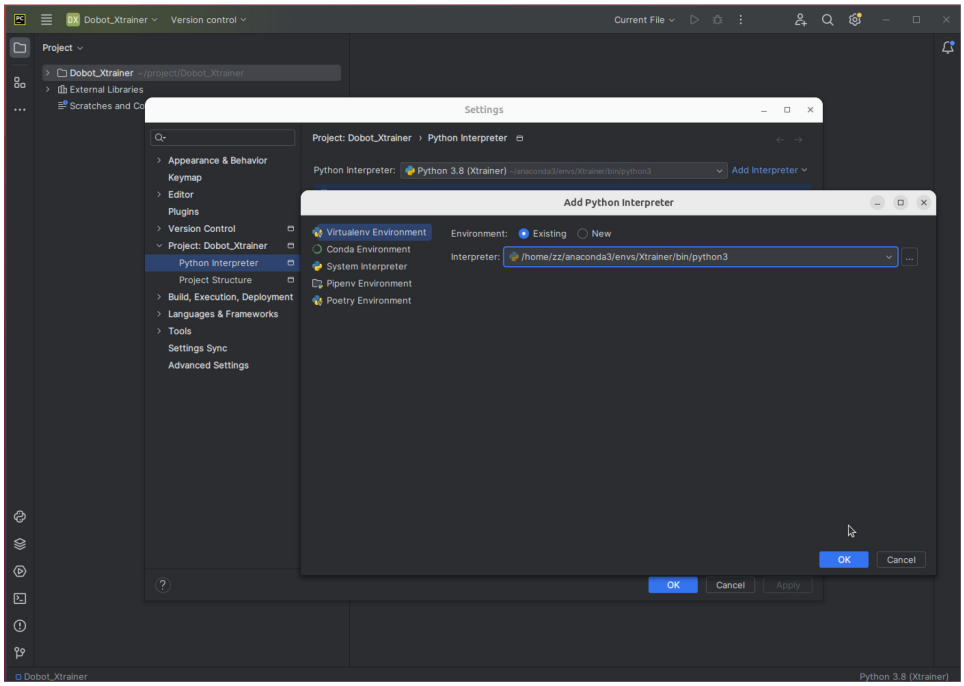
4. Open a terminal in the Dobot_Xtrainer (demo project provided by Dobot) folder and activate the virtual environment (see Step 2). Execute the following commands in turn to install the dependent package.

```
pip install -r requirements.txt
pip install -e third_party/DynamixelSDK/python
```


> **NOTE**
>
> You have no need to execute the following three commands if only the master-slave teleoperation function is required.

```
pip install -e ModelTrain/detr
pip install -e robomimic-r2d2
sudo apt install libxcb-cursor0
```

5. Set and use the virtual environment (Xtrainer) in Your IDE. Take PyCharm as a n example: Open the Dobot_Xtrainer folder in PyCharm, go to **Files > Settings > Python Interpreter > Add Interpreter > Existing > xxxx/anaconda3/envs/Xtrainer/bin/python3** (xxx is the actual installation path for Anaconda), click **OK** and **Apply** to complete the settings.

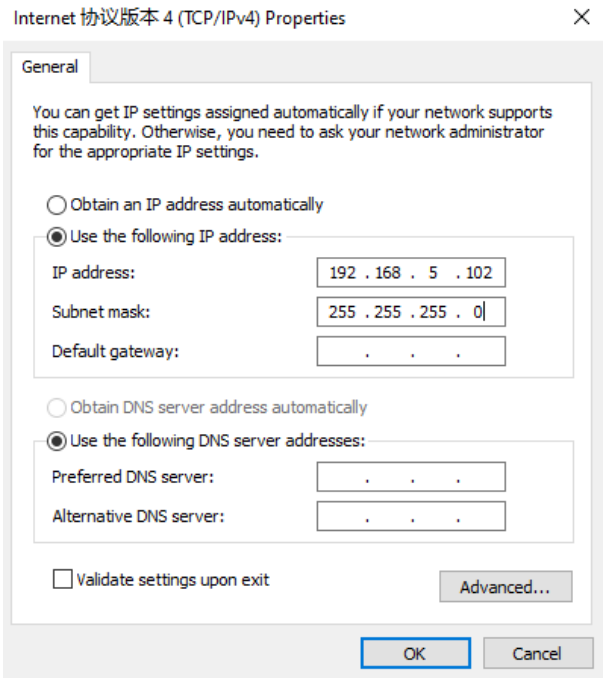


3.5 Slave hand settings

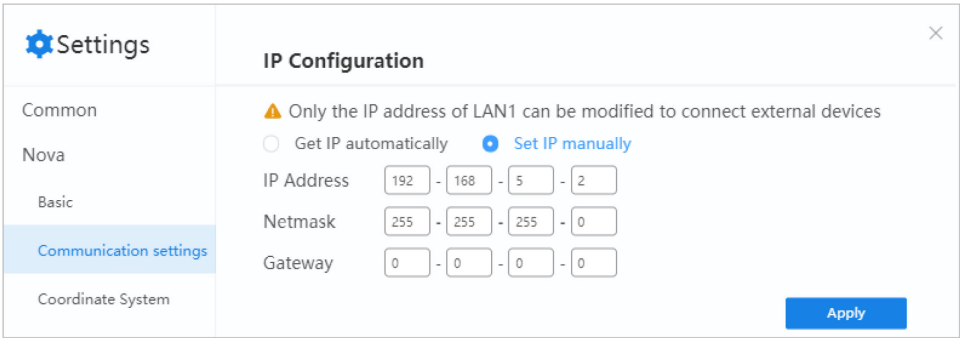
3.5.1 Modifying the IP of slave hand 2

Both slave hand 1 and slave hand 2 have the same default IP address for their LAN1 interfere (192.158.5.1). To avoid conflicts, you need to modify the IP address of slave hand 2 (right hand) to 192.168.5.2. Follow the steps below to modify the IP address, with specific operations referenced in the DobotStudio Pro help documentation.

1. Use the network cable to connect the CCBOX of slave hand 2 to the **slave-hand operation terminal (e.g., a computer)**.
2. Modify the IP address of the computer used for connecting to the slave hand to the 192.168.5.xxx subnet (subnet mask 255.255.255.0) without conflicting with the IPs of the two slave hands. For example, set it to 192.168.5.102.

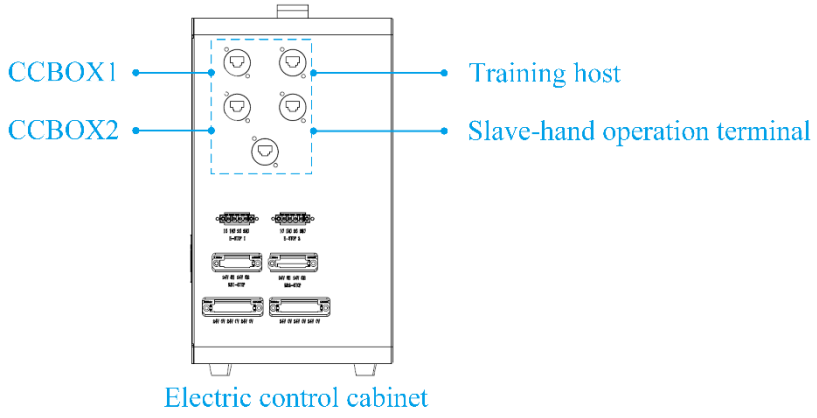


3. Open the control software (DobotStudio Pro 2.x.x) on the computer and connect to the detected Nova 2.
4. Go to **Settings > Communication settings**. Select **Set IP manually** on the **IP Configuration** page, and modify the IP to 192.168.5.2.



5. Click **Apply**.

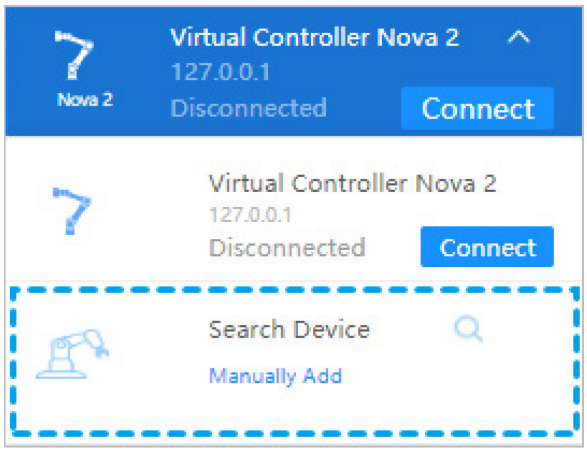
If the LAN1 interface of the CCBOX is used for connection, reconnect the LAN1 interface to the electric control cabinet via the network cable after the configuration is completed. Then, connect the **slave-hand operation terminal** to the electric control cabinet, as shown in the figure below.

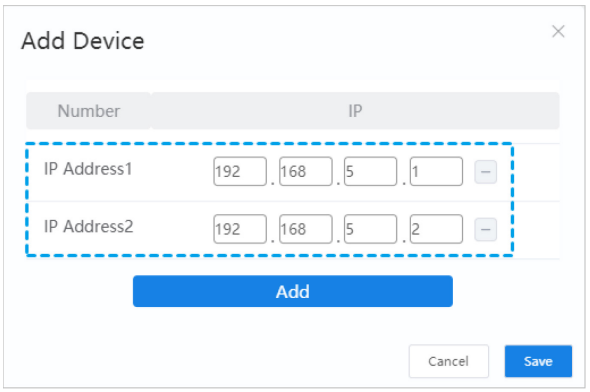


3.5.2 Configuring slave hand 1 and 2

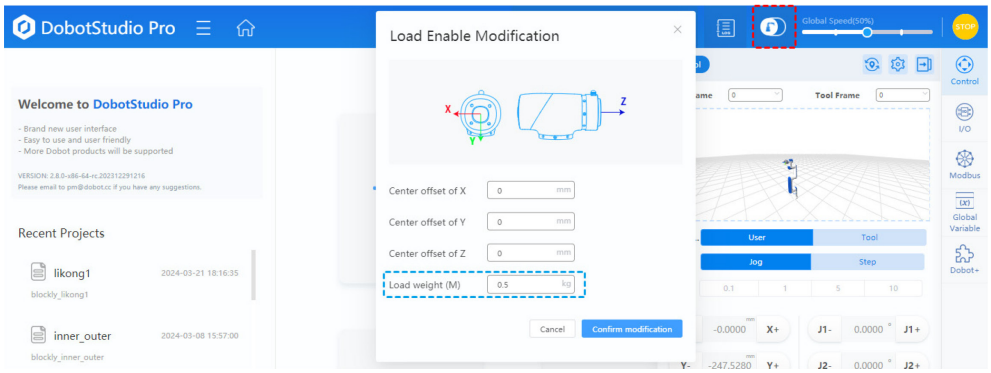
Perform the following operations for slave hand 1 and slave hand 2 respectively.

1. Manually add the IP addresses of the two slave hands in DobotStudio Pro.





2. Use DobotStudio Pro to connect to the slave hand.
   - Slave hand 1 (left hand): Nova 2 (192.168.5.1).
   - Slave hand 2 (right hand): Nova 2 (192.168.5.2).
3. Click the **Enable** switch at the top right of the interface and set the payload to 0.5kg.

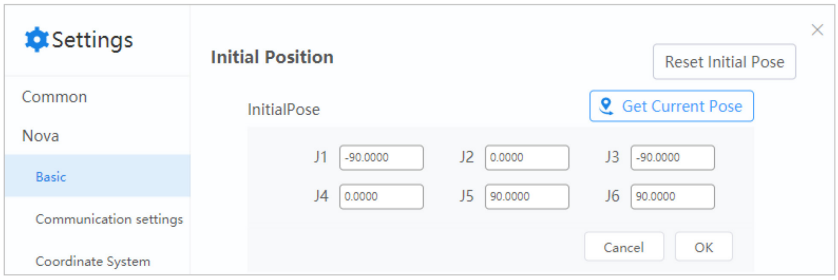


4. Set the initial posture for slave hand.
   a) Go to **Settings > Basic settings** page, click **Reset Initial Pose** on the right of **Initial Posture**.

b) Set the Initial Pose to the following values (which represent the joint angles J1 to J6 respectively), and click **OK**.

Slave hand 1: {-90, 0,-90, 0, 90, 90}

Slave hand 2: 90, 0, 90, 0, -90, -90}

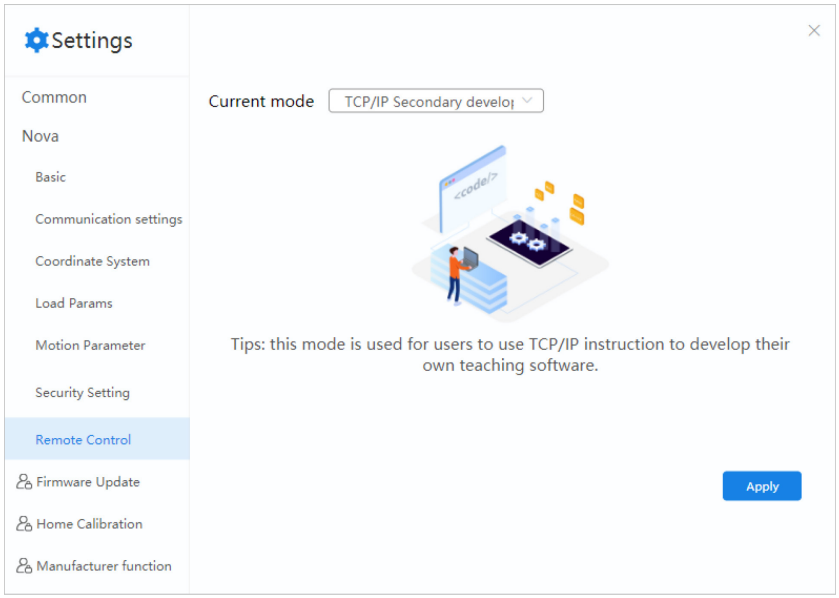


c) Long-press **Move to Default Pose** until the slave hand reaches the point.

NOTICE

During moving, observe carefully. If a collision is likely to occur, release the button and jog to avoid obstacles before continuing to move to the initial point.

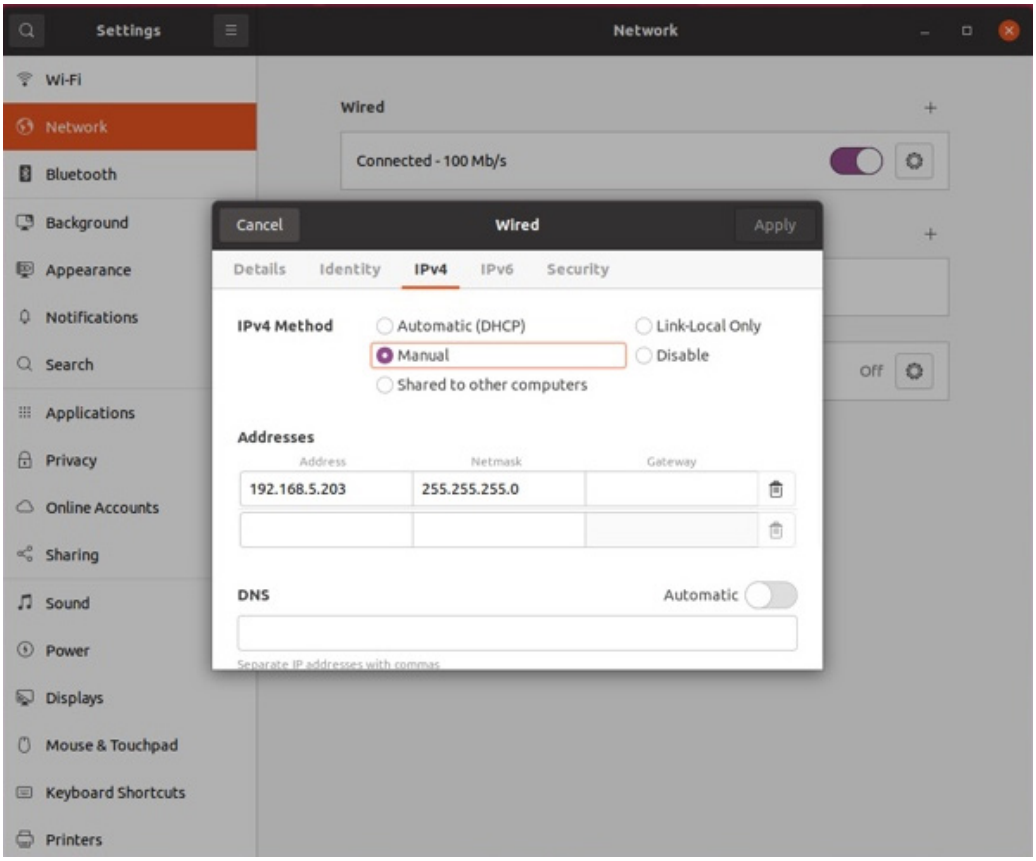
5. Go to **Settings > Remote control** page, set the current mode to **TCP/IP secondary development**, and click **Apply**.



After completing the above configurations, all subsequent operations will be performed on the **training host**. If the slave hand reports errors or other abnormal statuses, use the slave-hand operation terminal to open DobotStudio Pro for troubleshooting.

3.6 Configuring the training host IP

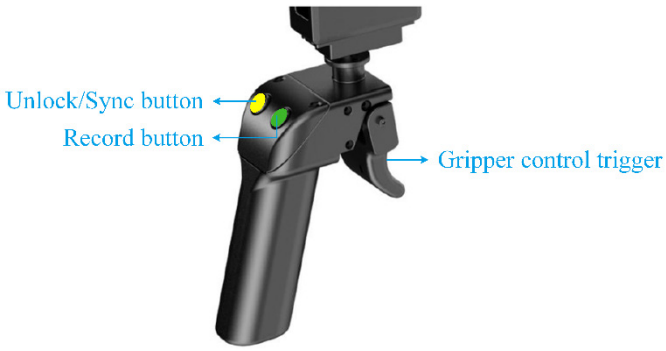
Modify the IP address of the training host used for connecting to the electric control cabinet to the 192.168.5.xxx subnet (subnet mask 255.255.255.0), without conflicting with the IPs of the two slave hands and the slave-hand operation terminal. For example, set it to 192.168.5.102 (the figure below takes Ubutu's settings interface as an example). After configuration, turn on the switch of the corresponding wired network.



4. Teleoperation and Data Collection

4.1 How to use master hand

This section introduces how to use the handle at the end of the master hand. Refer to this section to operate the robot after completing the configuration as described below.



- Unlock/Sync button (left yellow button):
  - Short-press (0.5s) to lock or unlock the master hand. When unlocked, the master hand can be freely dragged.
  - Long-press (2s) to sync or disconnect sync. When synced, the slave hand will slowly sync to the current posture and normal operation speed of the master hand. Once the sync is completed, teleoperation control of the slave hand can begin.

  The left and right master hands need to use this button separately for unlocking and synchronization.
- Record button (right green button):

  Short-press (0.5s) to start recording and collecting data, short-press again to stop recording.

  Once the master-slave sync is successful, pressing the button on either side will start or stop the overall data recording of the X-Trainer. Recording data for only one side is not supported.
- Gripper control trigger:

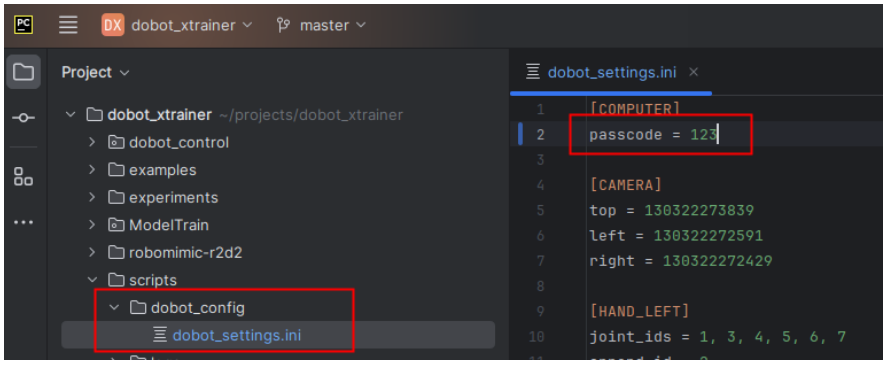
  Press the trigger to close the slave-hand gripper. Release the trigger to open the slave-hand gripper. The gripper stroke can be controlled by the distance the trigger is pressed.

NOTICE

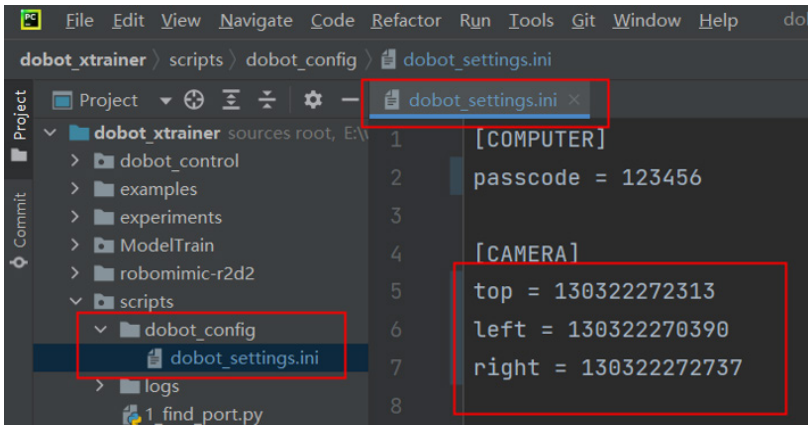
- Hold the handle with both hands before unlocking to prevent accidental falling of the master or slave hand.
- Ensure an appropriate safety distance between the master hands and lock them in a safe position before synchronizing the master and slave hands to prevent accidental falling, collision, or other anomalies.
- Do not collide with the master hand after it is locked to prevent servo damage.
- Ensure the slave hands do not interfere with each other during operation, which could damage the robot arms or cameras.

4.2 Configuring and collecting data

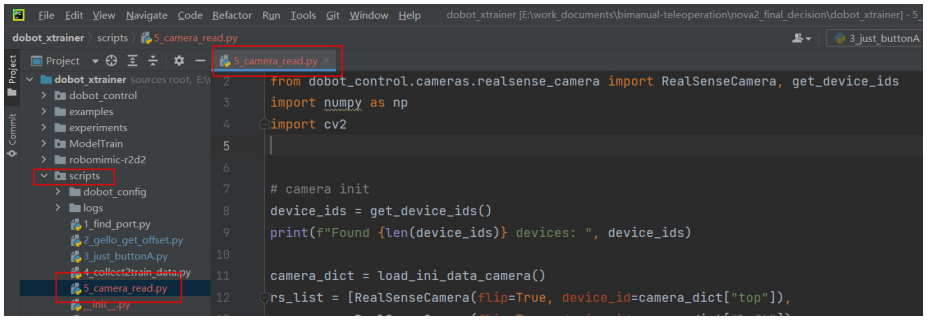
1. Open the Dobot_Xtrainer folder in IDE (here takes PyCharm as an example).
2. Modify the **scripts > dobot_config > dobot_settings.ini** file.
   a) Change the password to your local computer's password for executing the sudo commands.



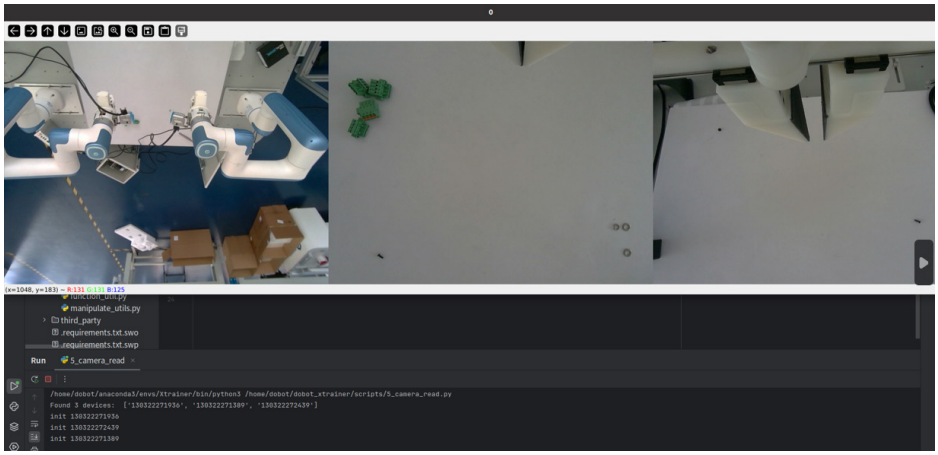
b) Fill in the IDs for each camera.
top: Global camera.
left: Slave hand 1 camera.
right: Slave hand 2 camera.



After saving the configuration, it is recommended to run **scripts > 5_camera_read.py** to check if the program can correctly read the camera images.



The expected output is shown in the figure below.

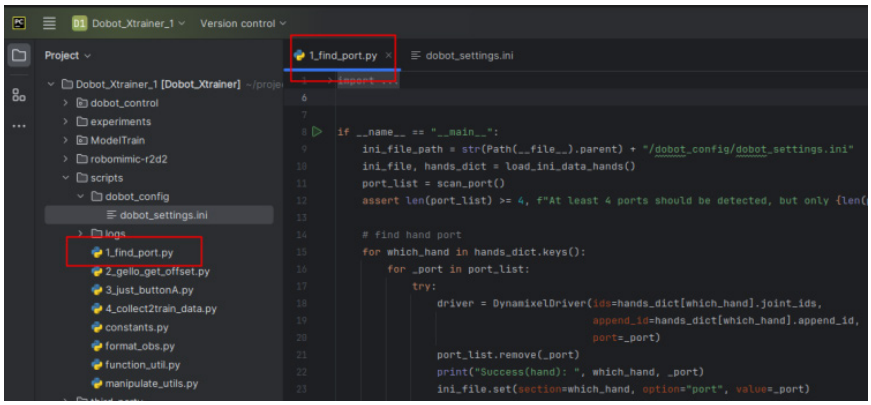


If the camera images are abnormal or the script reports an error, please reconnect the camera data cables and verify that each camera ID is correctly filled in, then rerun the script.

After completing the camera test, please manually stop the script.

3. Run **scripts > 1_find_port.py** to configure the USB interfaces for the two master hands and two slave-hand grippers.

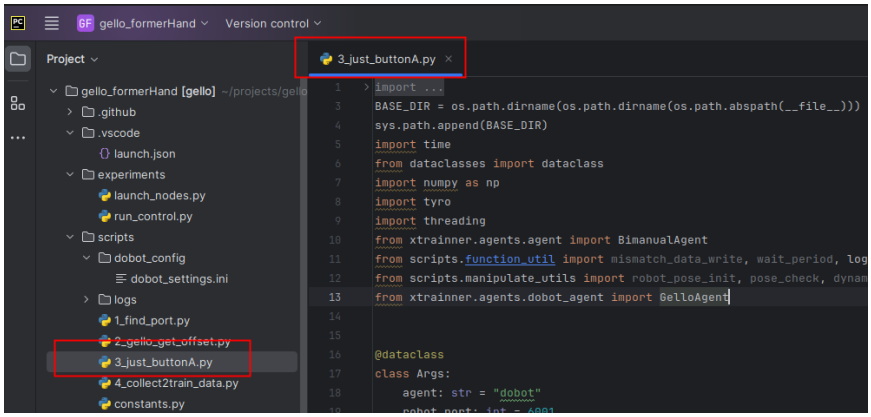
   The script needs to be re-run every time the training host is powered off or the USB interfaces are reconnected.



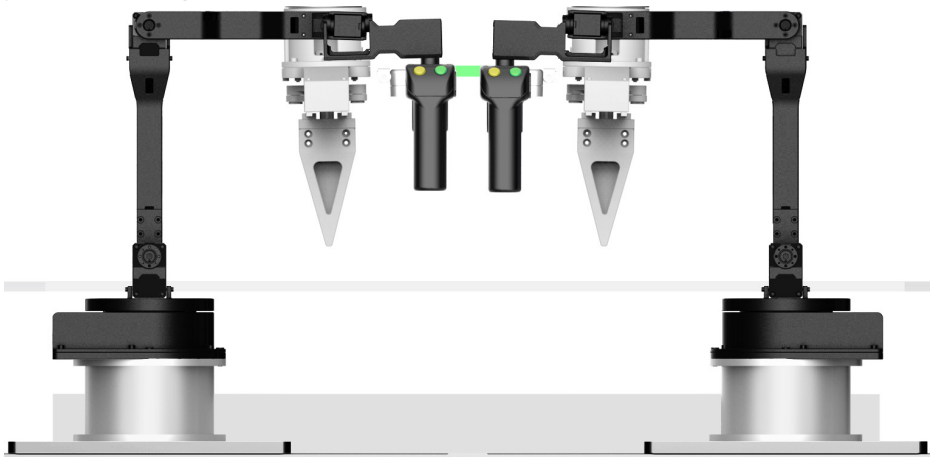
4. Set the sync posture for master and slave hands.

   After completing this step, the synchronized posture information of the master and slave hands will be saved in dobot_settings.ini. Unless there are issues such as desynchronization of the master and slave hands or loss of the configuration file, there is no need to repeat the settings.

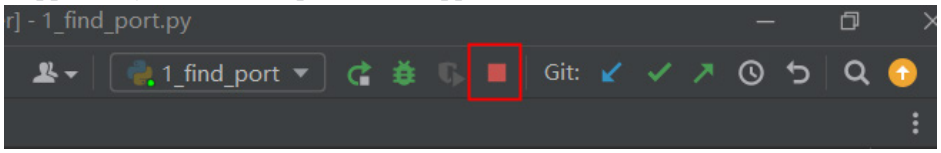
   a) Run **scripts > 3_just_buttonA.py**. After running, you can unlock and drag the master hand.



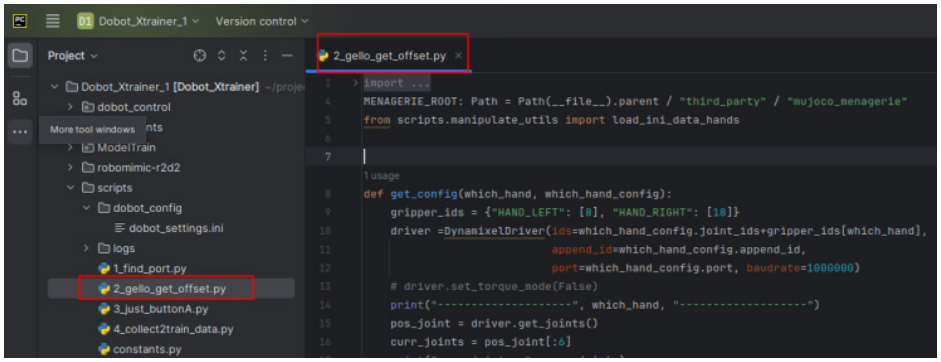
b) Hold the handles at the end of both master hands and short-press the yellow button to unlock them. Drag both master hands until their posture matches the initial posture of the slave hands (refer to 3.5.2 Configuring slave hand 1 and 2). Short-press the yellow button again to lock the master hands.



c) Stop running **3_just_buttonA.py**. Before proceeding to the next step, ensure that the script has completely stopped. For example, in PyCharm, if the stop button does not disappear after the first click, click the stop button again. The script is considered stopped only when the stop button disappears.

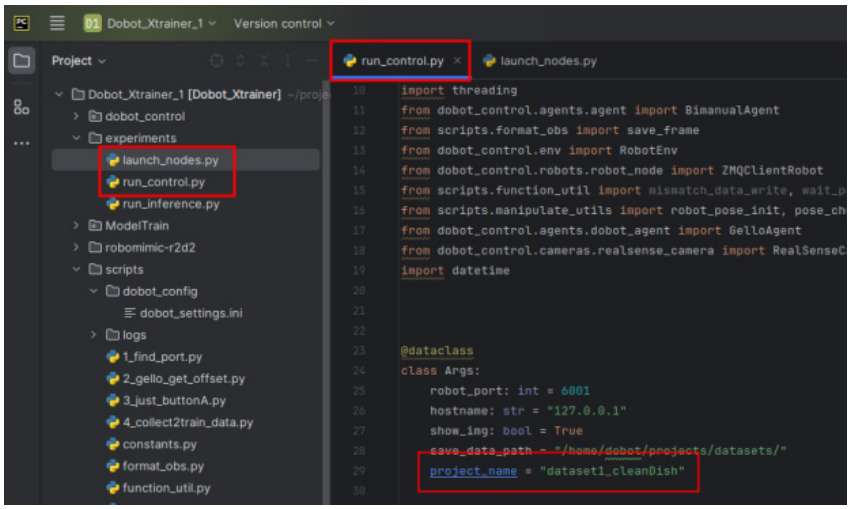


d) Run **scripts > 2_gello_get_offset.py** to set the current posture of the master and slave hands as the base value for synchronization.



5. Run **experiments > launch_nodes.py** to start the server process for the slave hands.
6. Open **experiments > run_control.py**, define the dataset name (project_name) according to the specific task you are training, then save and run the script.

   After running the above scripts, you can start the teleoperation and recording of training data.



For safety, Dobot has set limits for teleoperation related to speed and operation position. Triggering these limits will cause teleoperation to stop, and the indicator on the table will turn red to signal an alarm. In such cases, you need to stop and re-run the **run_control.py** script to clear the alarm.

NOTICE

After re-running the script, it is recommended to:

1. Short-press the yellow button to unlock the master hands, and drag them to the initial position and lock them.
2. Long-press the yellow button to sync the slave hand posture.
3. Unlock the master hands for teleoperation.

By default, the slave-hand gripper can only operate above the EVA foam area on the workbench. If necessary, you can modify the activity area limits for the slave hand in the **run_control.py** script (as shown in the figure below). **Ensure you have thoroughly evaluated the potential safety risks before making any changes.**

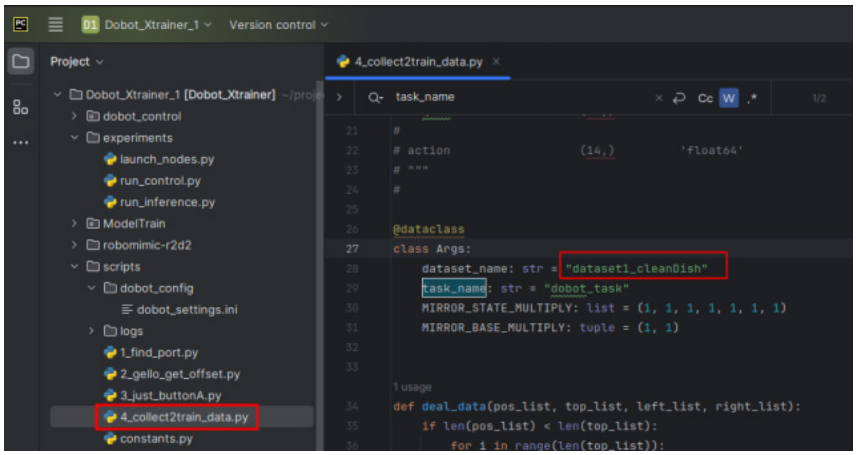
5. Algorithm Training for Imitation Learning

5.1 Processing training dataset

After data collection is complete, you can find a folder with the same name as the dataset (i.e., the **project_name** in **experiments > run_control.py**) in the datasets folder, which is at the same level as Dobot_XTrainer (hereinafter referred to as the “training data folder”).

This folder should initially contain only a **collect_data** folder, which holds the raw collected data. If you need to use the collected data for algorithm training, you'll first need to compress and consolidate it. The steps are as follows:

1. Open **scripts > 4_collect2train_data.py**. Modify the dataset_name to the name of the actual recorded dataset.



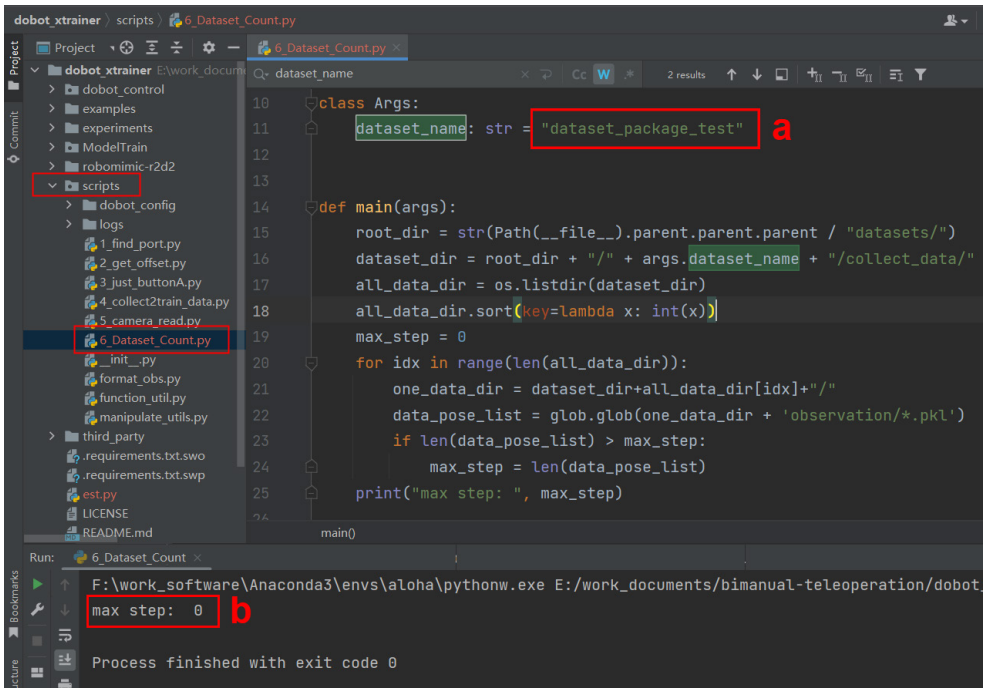
2. Run the script to compress and integrate the training data.
3. Once the script has finished running, two new folders will be generated in the training data folder: **output_videos** and **train_data**.
   - **output_videos** contains the videos captured by the camera during training, which can be used to check for any issues in the data collection process.
   - **train_data** contains the processed training data, which will be used in subsequent training steps.

5.2 Setting dataset parameters

Open **ModelTrain > contants.py**, and modify the following parameters:

```
DATA_DIR = '~/projects/datasets/dataset1_cleanDish' ①
TASK_CONFIGS = {
    # dobot move cube new
    ② 'move_cube_new': {
        'dataset_dir': DATA_DIR + '/train_data', ③
        'episode_len': 900, ④
        'train_ratio': 0.9, ⑤
        'camera_names': ['top', 'left_wrist', 'right_wrist'] ⑥
    },
```

- ①: The path of the dataset folder, which is the parent directory of the aforementioned **train_data** folder.
- ②: Dataset name.
- ③: Set to **/train_data**, so that ① + ③ equals the full path of the **train_data** folder from the previous section.
- ④: The number of collection steps. Run **scripts > 6_Dataset_Count.py** to get the maximum number of steps in the dataset, then set this parameter to a slightly higher value. For example, if the maximum number of steps in the dataset is 410, you can set this parameter to 500.



  - a: Run the script after modifying it with the actual dataset name.
  - b: The maximum number of steps obtained for this dataset after running the script.
- ⑤: The proportion of the training set; the remaining data will be used as the validation set.
- ⑥: Camera name. If camera-related code has not been modified, keep the defaults.

After modification, save the file.

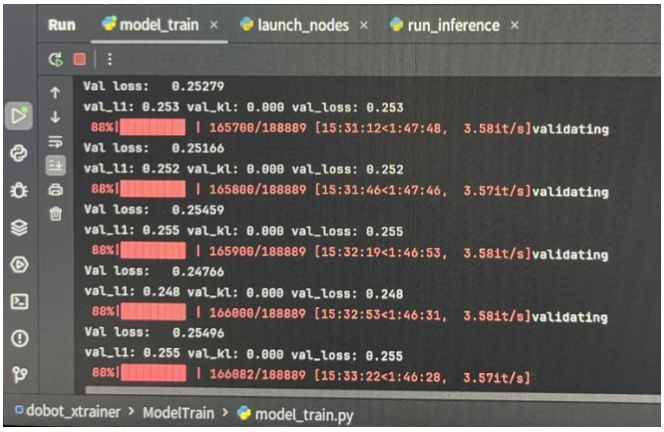
5.3 Setting training-related parameters

Open **ModelTrain > model_train.py**, and modify the following parameters (modify the default value of the corresponding row):

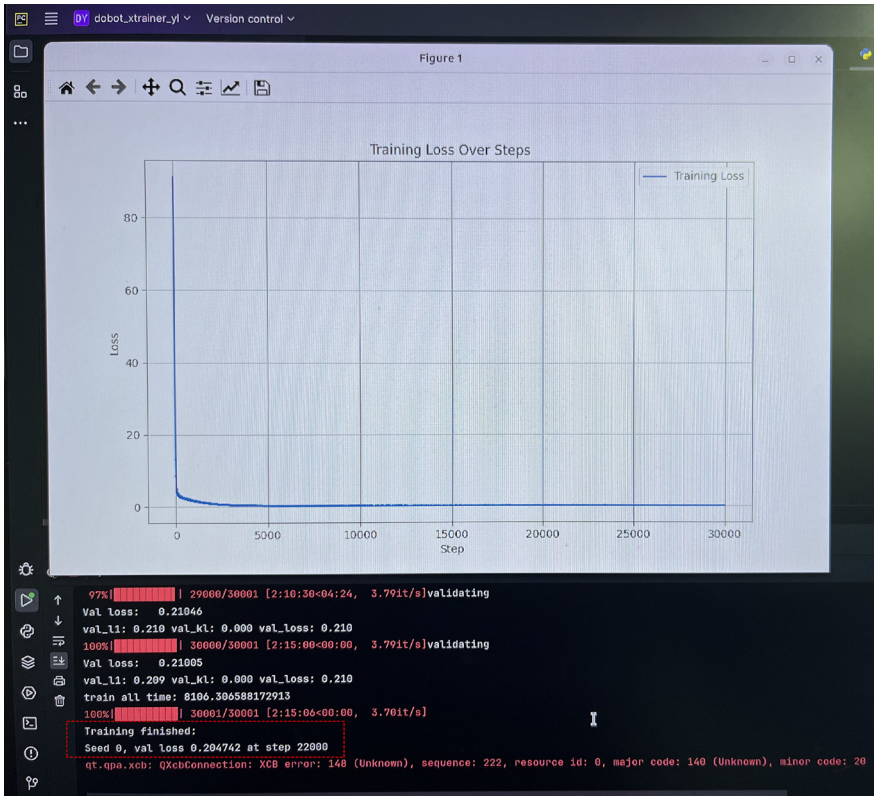
```
def arg_config():
    parser = argparse.ArgumentParser()
    parser.add_argument('--ckpt_dir', action='store', type=str, help='ckpt_dir', default='./ckpt/ckpt_move_cube_new',required=False)
    parser.add_argument('--task_name', action='store', type=str, default='move_cube_new',help='task_name', required=False)
    parser.add_argument('--batch_size', action='store', type=int, help='batch_size', default=16, required=False)
    parser.add_argument('--seed', action='store', type=int, help='seed', default=0,required=False)
    parser.add_argument('--num_steps', action='store', type=int, help='num_steps', default=200, required=False)
    parser.add_argument('--lr', action='store', type=float, help='lr', default=2e-5,required=False)
    parser.add_argument('--load_pretrain', action='store_true', default=False)  # Ignore this parameter and leave the default setting
    parser.add_argument('--eval_every', action='store', type=int, default=100, help='eval_every', required=False)  # Ignore this parameter and leave the defa
    parser.add_argument('--validate_every', action='store', type=int, default=100, help='validate_every',required=False)
    parser.add_argument('--save_every', action='store', type=int, default=1000, help='save_every', required=False)
    # parser.add_argument('--resume_ckpt_path', action='store', type=str, help='resume_ckpt_path', default= "./ckpt/ckpt_closh/ckpt_closh_puping2_new10000/policy_last.ck
    parser.add_argument('--resume_ckpt_path', action='store', type=str, help='resume_ckpt_path', required=False)
    parser.add_argument('--skip_mirrored_data', action='store_true')

    parser.add_argument('--kl_weight', action='store', type=int, help='KL divergence weight,recommended set 10 or 100', default=10,required=False)
    parser.add_argument('--chunk_size', action='store', type=int, help='The model predicts the length of the output action sequence at a time', default=45,
    parser.add_argument('--hidden_dim', action='store', type=int, help='hidden_dim', default=512, required=False)
    parser.add_argument('--dim_feedforward', action='store', type=int, help='dim_feedforward', default=3200,required=False)
    parser.add_argument('--temporal_agg', action='store_true', default=True)
    parser.add_argument('--no_encoder', action='store_true', default=False)

    return vars(parser.parse_args())
```

- ckpt_dir: Set the path where the trained model will be saved, user-defined.
- task_name: Dataset name.
- batch_size: Size of the batch during training, set it based on the graphics card's performance (e.g., 16 for a single RTX 4090).
- num_steps: Number of training iterations.
- lr: Learning rate.
- validate_every: Number of iterations between each validation.
- save_every: Number of iterations between each model save.
- resume_ckpt_path: If you need to continue training from a previously saved model, refer to the commented-out code in the previous line and add the default parameter, with the value being the address of the saved model.
- chunk_size: Length of the action sequence predicted by the algorithm at one time.
- temporal_agg: Action time smoothing, recommended to set to True.
- Other parameters: Can be kept as defaults.

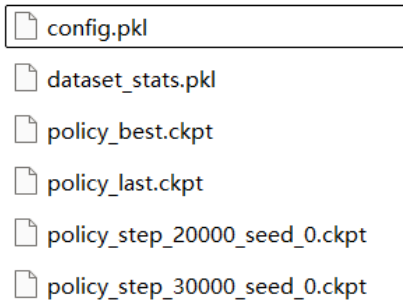
After modification, save the script and run it to start training, as shown in the figure below. Be patient, as training can take some time. For the first training session, an Internet connection is required to download the pre-trained model. Subsequent training sessions will not require an Internet connection.



Once training is complete, the console will display "Train Finished" and a curve graph showing the training results will pop up, as shown in the figure below.



The trained models are saved in the folder specified by the **ckpt_dir** parameter. This folder contains multiple models. For inference, it is common to use either **policy_last.ckpt** (the last trained model) or **policy_best.ckpt** (the model with the best validation accuracy).



You can open **ModelTrain > model_inference_test.py** to test the model's inference. By inputting the collected dataset images and the slave-hand joint values (including gripper stroke) into the trained model, you can compare the predicted action values of the slave-hand joints and the gripper with the true values of the data labels to get an initial assessment of the model's prediction accuracy.

6. Running the Autonomous Inference

1. Open **experiments > run_inference.py**, and modify the following parameters:

```
# Initialize the model
model_name = 'policy_last.ckpt'
# model_name = 'policy_best.ckpt'
model = Imitate_Model(ckpt_dir='./ckpt/ckpt_move_cube_new', ckpt_name=model_name)
```

```
episode_len = 900 # 完成任务总步长
```

   - model_name: Model name. Set it to **policy_last.ckpt** (the latest trained model) or **policy_best.ckpt** (the model with the best validation accuracy).
   - ckpt_dir: Path to the model directory, as specified in **ModelTrain > model_train.py**.
   - episode_len: Length of the task steps, which must be less than or equal to the value specified in **ModelTrain > contants.py**. You can adjust this parameter to trim the length of the action sequence for autonomous inference. For example, if the robot does not complete the target action during autonomous inference, you can increase the step length; if the robot performs some extra actions after completing the target action, you can decrease the step length.

   After modification, save the file.
2. Run **experiments > launch_nodes.py** to start the server process for the slave hands.
3. Run **experiments > run_inference.py** to start the autonomous operation of the slave hand based on inference.

NOTICE

Ensure that the parameters in run_inference.py match those used during the training in model_train.py.

7. Data Interface

Dobot provides a set of DEMOs to demonstrate how to get data from Dobot XTrainer. These DEMOs are located in the example folder of the Dobot XTrainer project.

- example_agent.py: Demonstrate how to get the joint angles and button information of the master hand.
- example_camera.py: Demonstrate how to obtain images from the camera.
- example_dobot_robot.py: Demonstrate how to set the indicator and get joint angles of the slave hand.
- example_gripper.py: Demonstrate how to control the slave-hand gripper.
- example_read_data_from_dataset.py: Demonstrate how to parse data from the dataset.

