PocketGo User Guide

Issue: V1.0
Date: 2025-04-29

SHENZHEN DOBOT CORP 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 DOBOT CORP 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 introduces the method for collecting task data using the handheld acquisition device, as well as the demonstration code and operational workflow for algorithm training and inference.

Intended audience

This document is intended for:

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

Revision history

Date	Version	Revised content
2025-04-29	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 high potential hazard that, 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 potential risk that, if ignored, could result in robot arm damage, data loss, or unpredictable result.
NOTE	Provides additional information to emphasize or supplement important points in the main text.

Contents

1. Product Introduction

1.1 Overview

Currently, both domestic and international research on Artificial General Intelligence (AGI) is experiencing a new surge, with Embodied AI widely regarded as a critical step toward AGI. As a highly promising field, Embodied AI empowers robots to acquire knowledge and skills through imitation learning, derived from observing human demonstrations. This technique, often referred to as behavior cloning, enables robots to master a wide range of essential abilities, from simple pick-and-place tasks to intricate manipulation operations.

This kit provides a single-arm data acquisition device, enabling users to collect task data through direct handheld operation. It also includes demonstration code for basic imitation learning data collection, algorithm training, and inference, along with a complete API interface for the platform.

With this hardware platform and customizable development interfaces, flexible and diverse 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

1.2.1 Handheld Data Acquisition Unit

Figure 1.1 Handheld gripper

- Standard gripping width: 73.5 ± 1.5 mm
- Effective gripping depth: 100 mm
- Maximum gripping weight: ≤ 2 kg
- Net weight: 0.93 kg

1.2.2 Robot's End-Effector

Figure 1.2 End-of-arm gripper

- Standard gripping width: 73.5 ± 1.5 mm
- Effective gripping depth: 100 mm
- Maximum gripping weight: ≤ 2 kg
- Net weight: 1.58 kg

End-of-arm camera

a. Camera media module:

Port 1 connects to the video capture card.

Port 2 is a Type-C charging port.

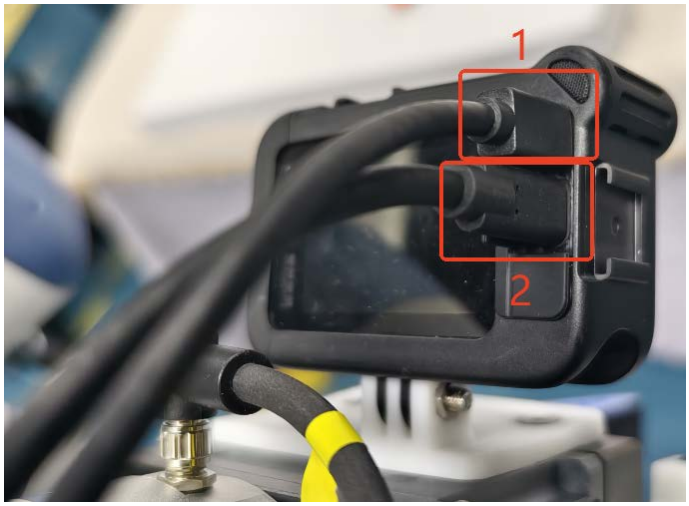


Figure 1.3 Camera media module

b. Video capture card:

Port 1 connects to Port 1 of the camera media module.

Port 2 connects to the host computer.

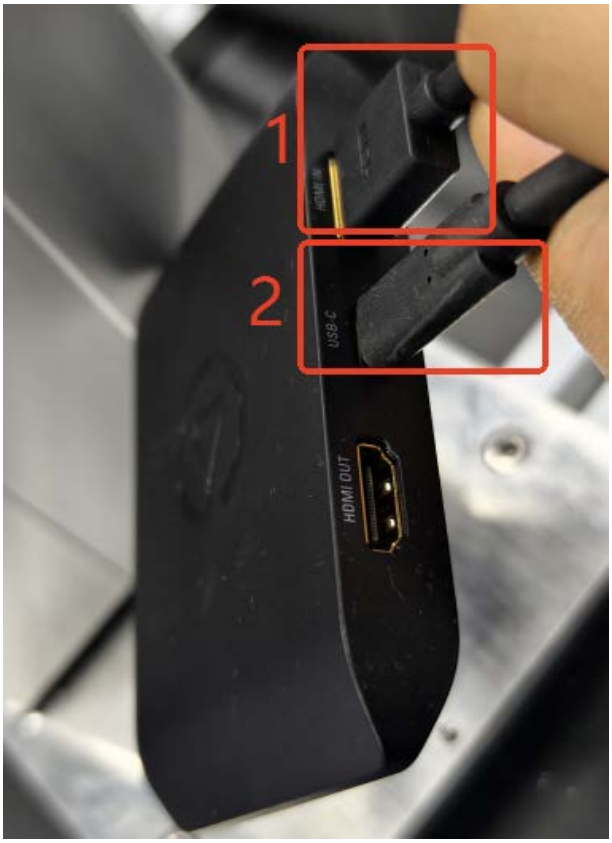


Figure 1.4 Video capture card

c. The USB port of the host computer (recommended USB 3.0) connects to Port 2 of the video capture card.

Figure 1.5 Host computer

1.3 Product Installation

Please scan the QR code below to access product unboxing and usage videos.

PocketGo Quick Start

2. Operating Environment Configuration

2.1 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.
- Avoid using Chinese characters when naming folders. It is recommended to use English letters and numbers.

2.2 Installing CUDA and cuDNN

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 appropriate version for 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.
5. Use vim to open the user configuration file in the terminal (if vim is not installed, please install and learn how to use vim).

```
vim ~/.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 appropriate version for your operating system, and obtain the cuDNN download and installation commands.

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

2.3 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 vim to open the user configuration file in the terminal.

```
vim ~/.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
```

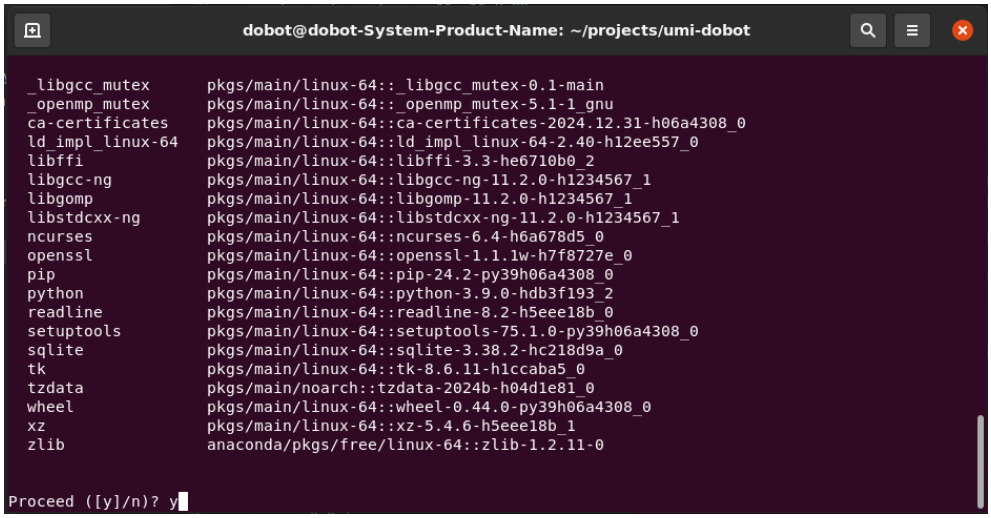

2.4 Configuring the Virtual Environment

1. In the terminal or Anaconda command window, create a new virtual environment and install Python 3.9.0.

   For example, create a virtual environment named "HDC":

   Create a virtual environment

   conda create -n HDC python==3.9.0

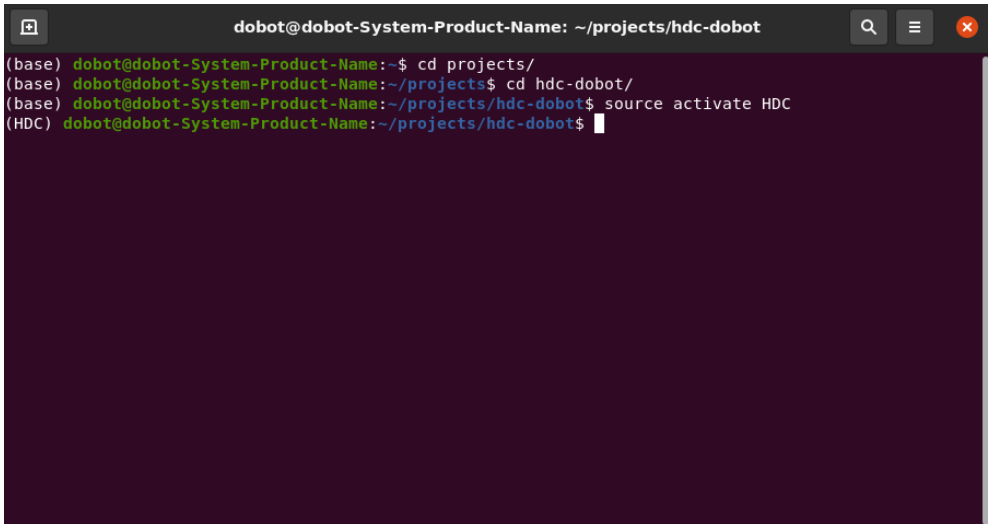


   Enter "y" to confirm the download when prompted.

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

   Activate the virtual environment

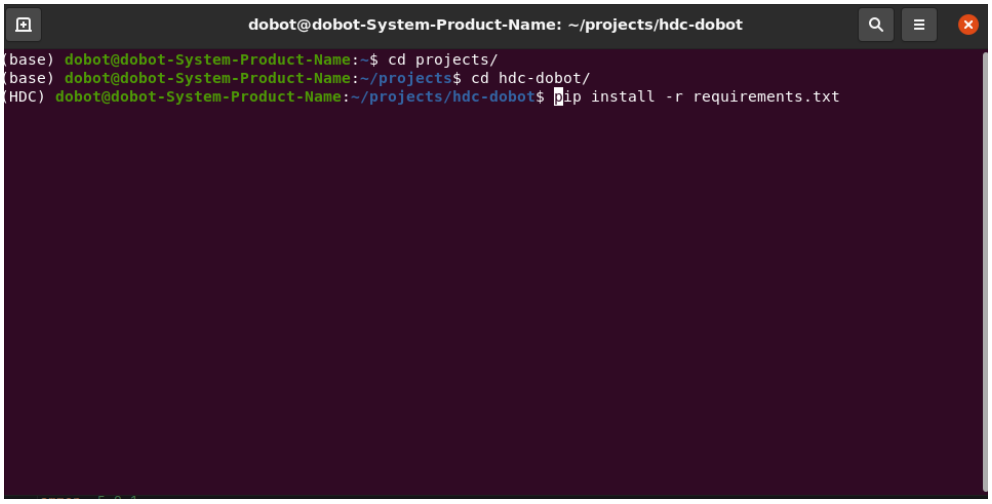
   conda activate HDC



3. Install dependencies.

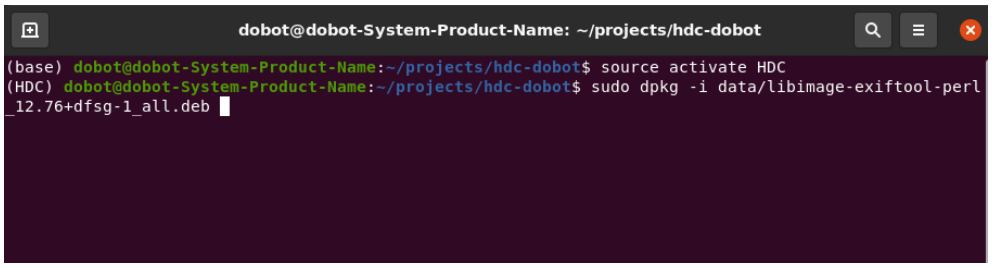
```
sudo apt-get install cmake gcc g++
```

```
Install required Python package

pip install -r requirements.txt
```



4. Install exiftool.

```
sudo dpkg -i data/libimage-exiftool-perl_12.76+dfsg-1_all.deb
```



2.5 Activating GoPro

1. Activate the GoPro according to the instructions in the GoPro product manual.
2. Update the camera firmware following the official guide: GoPro Labs (https://gopro.github.io/labs/).
3. Point the GoPro camera at the QR code below to update the parameters on the GoPro.

2.6 Installing Docker and Its Image

1. Install Docker.

Install the essential software for Docker

sudo apt-get update

sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common

Add the official Docker repository

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

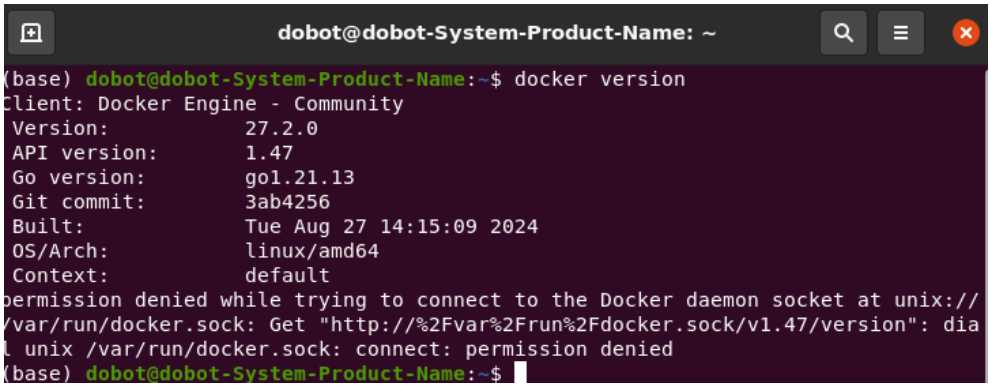
Install Docker

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io

Test the installation

docker version



2. Install the Docker image.

   Method 1: Online Installation

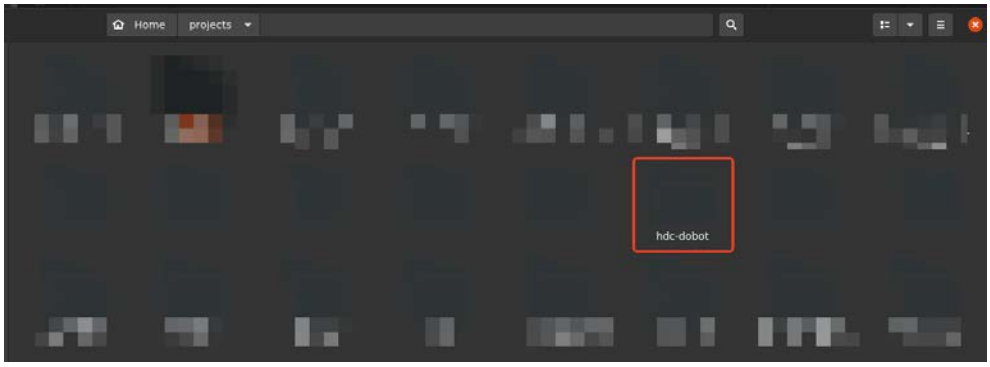
```
sudo docker pull chicheng/openicc:latest

sudo docker pull chicheng/orb_slam3:latest
```

   Method 2: Offline Installation

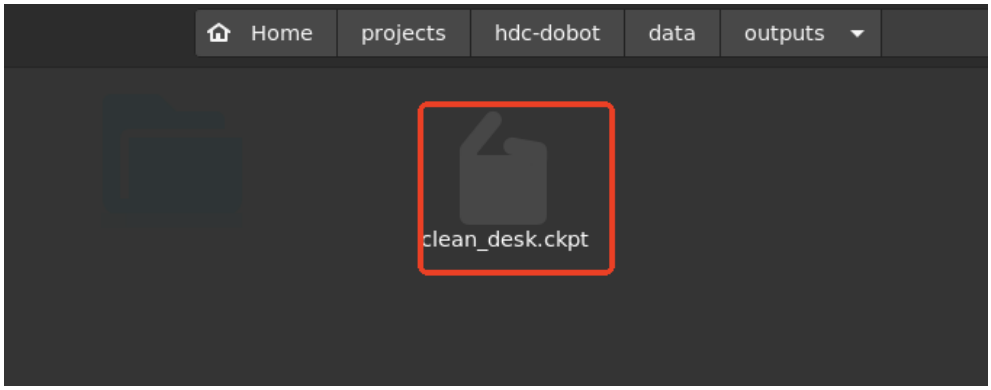
```
sudo docker load -i openicc.tar

sudo docker load -i orb_slam3.tar
```

2.7 Project Code

1. Copy the project code to a known directory.



2. Copy the provided two model files into the following directories accordingly.



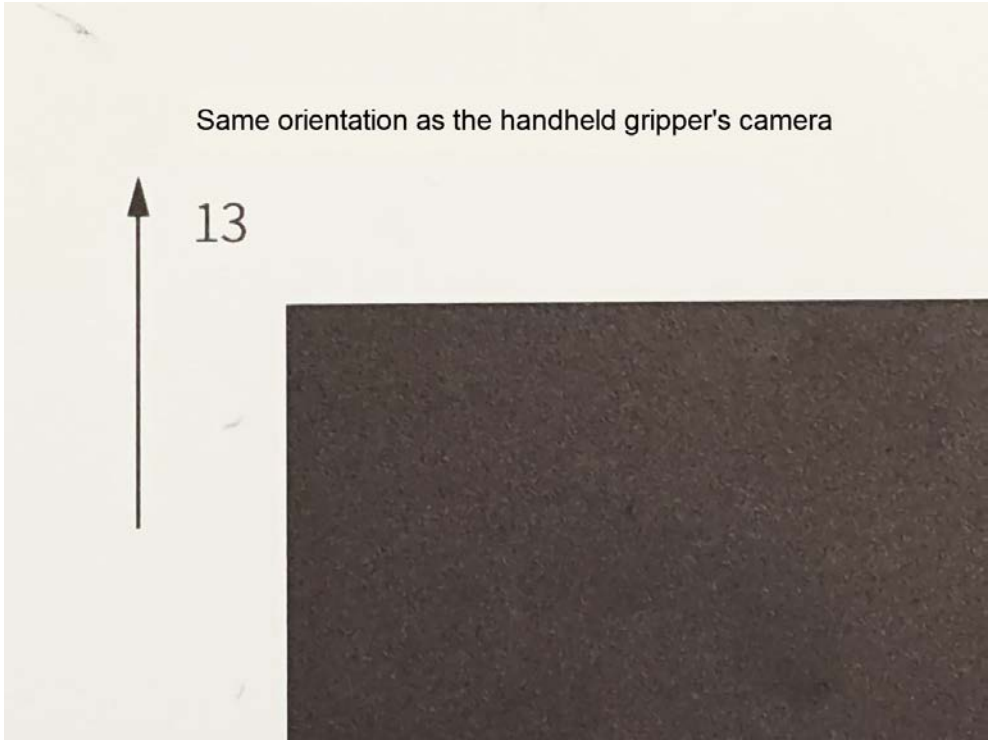
3. Data Collection

3.1 Procedure

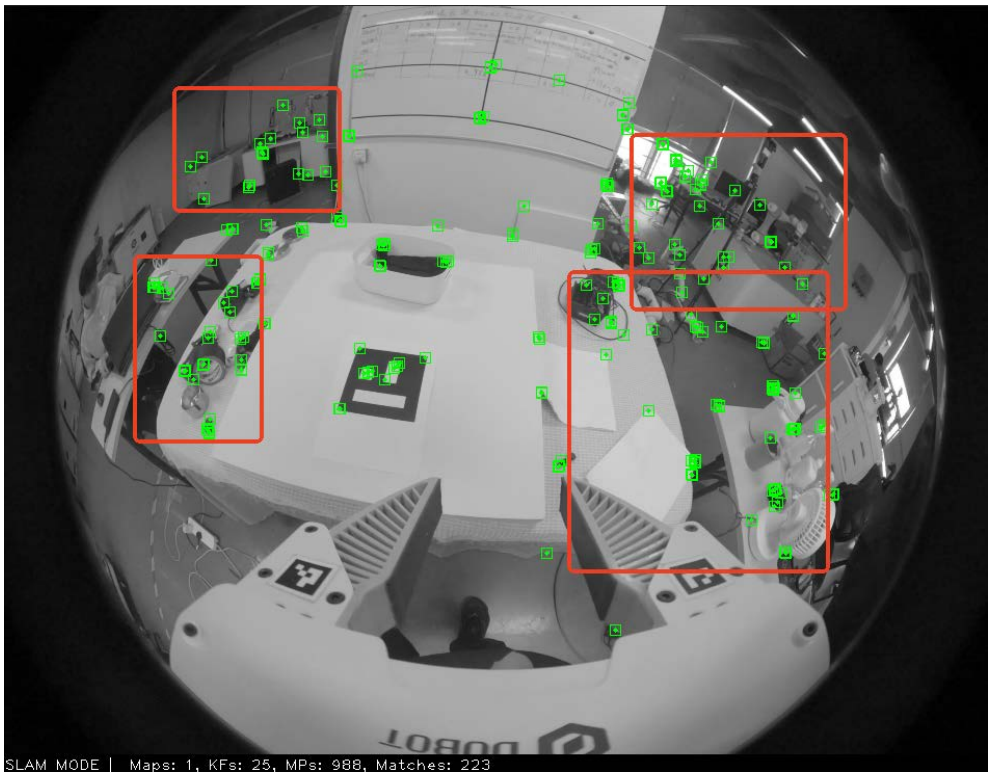
1. Scan the QR code provided in the Product Installation section and follow the map construction video tutorial to place the desktop vision calibration board on the table and record the **map construction video**.
2. Scan the QR code provided in the Product Installation section and follow the gripper calibration video tutorial to record the **gripper calibration video**. The gripper should perform five consecutive open-close cycles for calibration.
3. Scan the QR code provided in the Product Installation section and follow the data collection video tutorial to record the **data collection video**.

3.2 Precautions

1. Before starting **map construction video** recording, ensure that the upper-left arrow direction of the desktop vision calibration board is aligned with the camera orientation of the handheld gripper. Do not rotate beyond ±90° in the RZ (Z-axis rotation) direction, as this may cause SLAM failure.



2. Background requirements for the camera field of view:



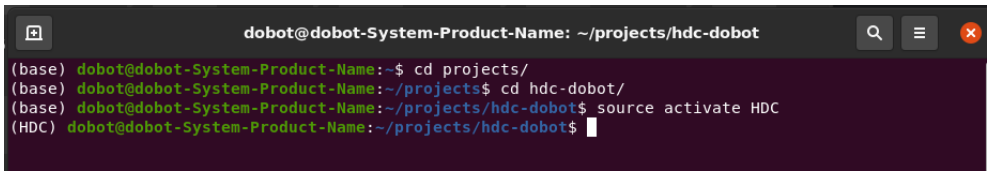
a. Since this solution relies on ORB-SLAM3 for trajectory generation, the algorithm depends on feature points in the environment.

b. The environment within the camera's field of view should have clear textures and significant depth variation. Avoid areas with large monotone textures such as solid-colored walls or floors.

c. Due to lighting and environmental variations, a new map construction video and gripper calibration video must be recorded before each batch of data collection.

3.3 Data Standardization

1. Open the HDC virtual environment via the command line and navigate to the project code directory.



2. Copy the data.

   a. Data directory structure and naming conventions (example)

      i. ① in the figure above represents the path (user-definable name) where all the task data is stored.

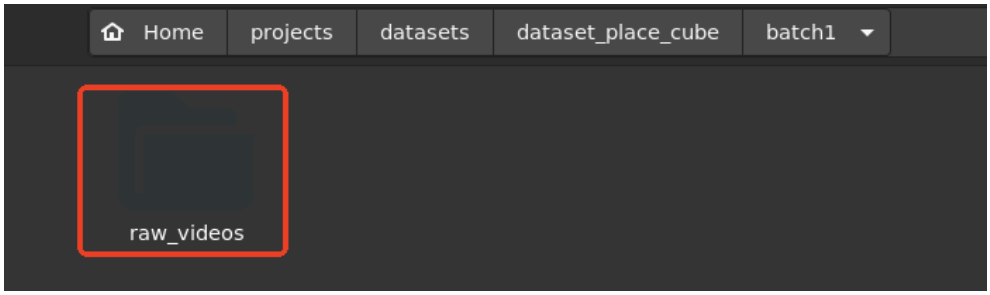
      ii. ② in the figure above represents the task name (user-defined).

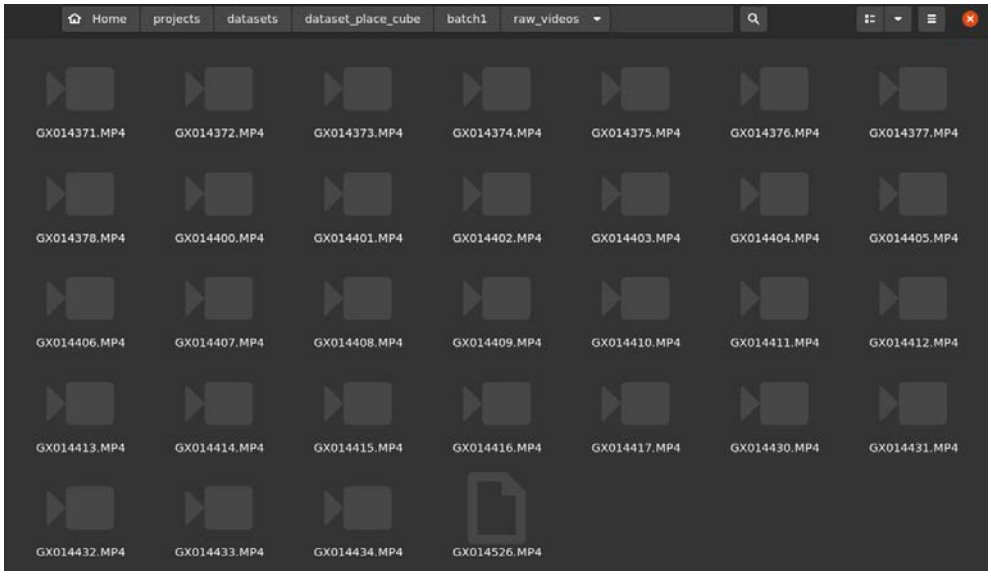
      iii. ③ in the figure above represents a specific batch of data under this task (user-definable name).

      iv. ④ in the figure above represents intermediate data generated during processing (no user input required).

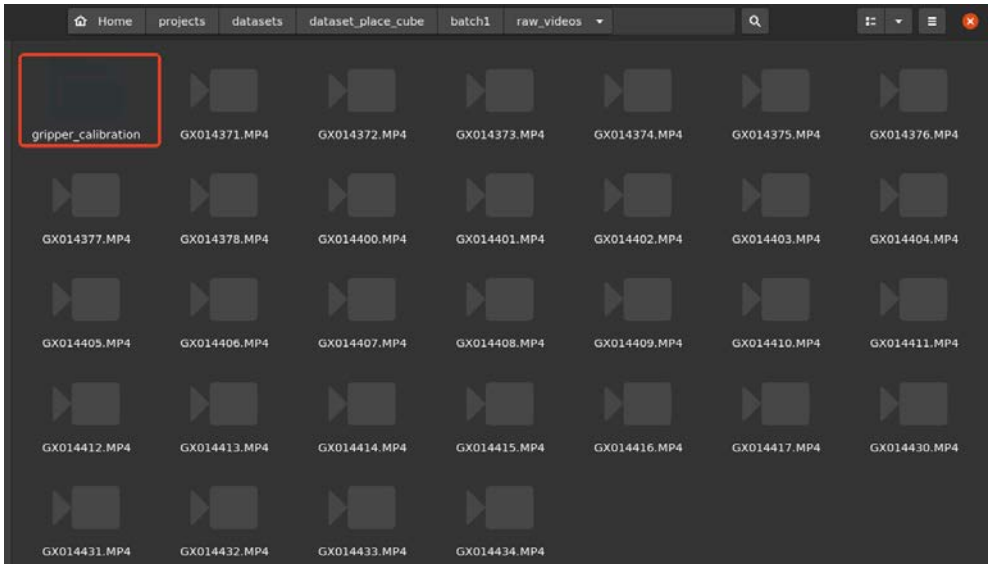
      v. ⑤ in the figure above represents the storage location for GoPro camera video data (fixed name, cannot be changed).

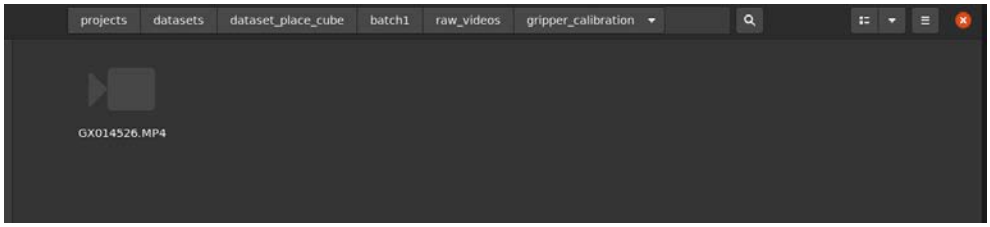
   b. Refer to convention a, after customizing the path, create a "raw_videos" folder under the path, and copy the video data (including **map construction videos, gripper calibration videos, and task collection videos**) to the "raw_videos" folder.



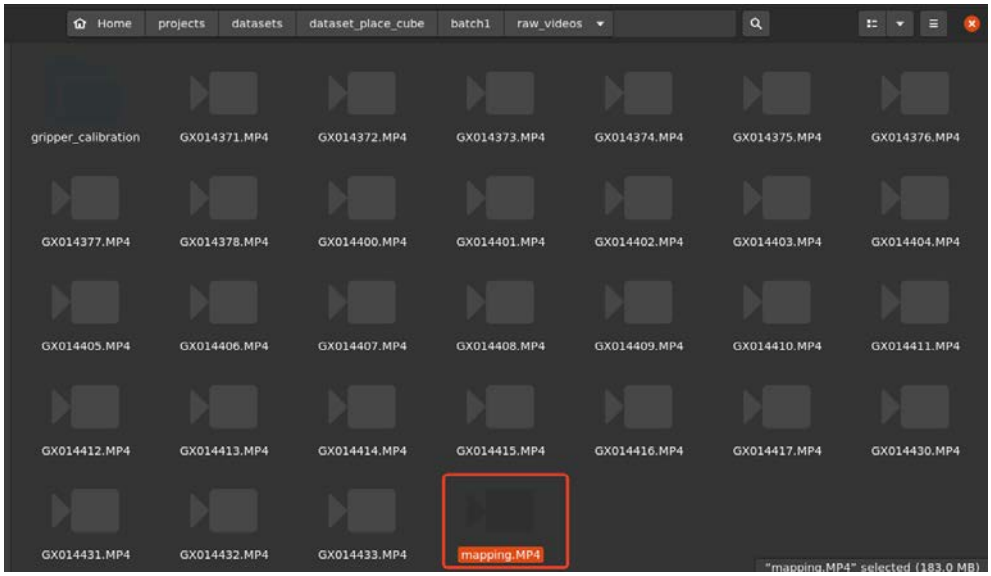


c. Create a "gripper_calibration" folder and move the gripper calibration video into it (no need to rename).





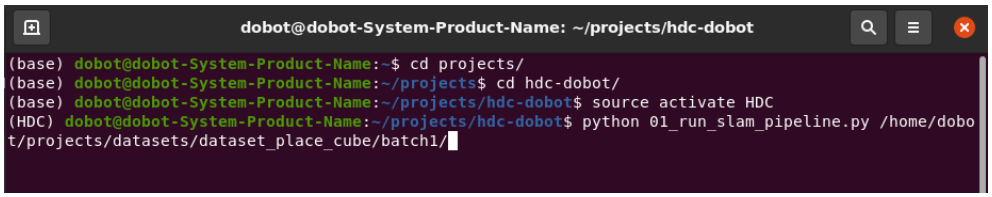
d. Rename the map construction video to "mapping.mp4".

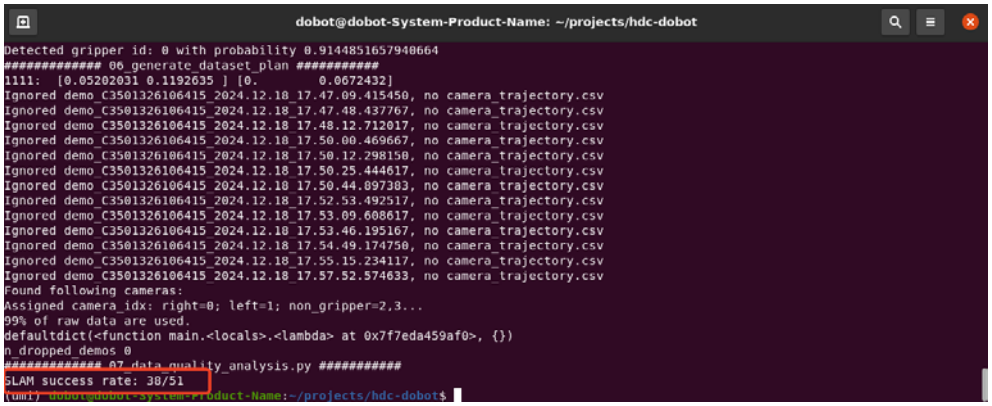


3. Calculate the trajectory.

   a. Run the following command in the terminal.

```
python 01_run_slam_pipeline.py
/home/dobot/projects/datasets/dataset_place_cube/batch1/
```





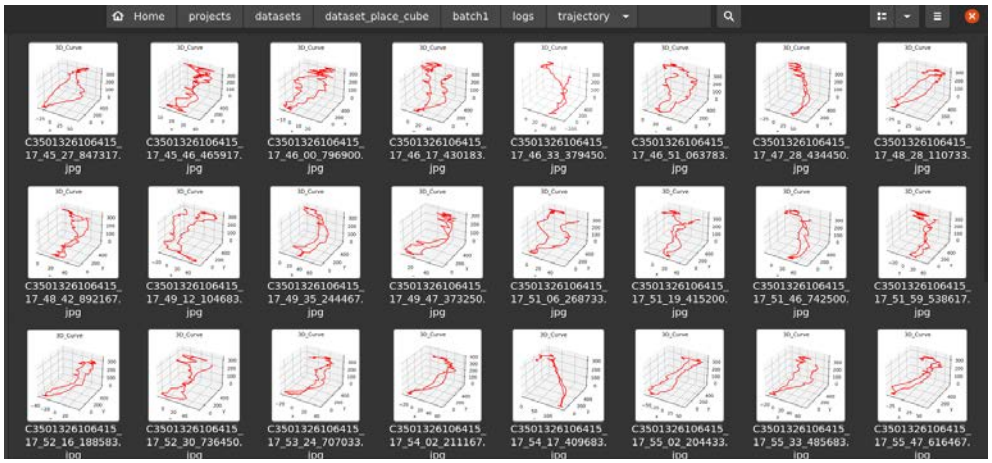
   b. Evaluate the data quality:

      i. As shown in the last figure of step a, a total of 51 task collection videos were processed, with 38 successfully mapped using SLAM.

      ii. The generated trajectory files can be found in the "batch1/logs/trajectory" directory.

      iii. If the SLAM trajectory quality is poor, possible causes include:

         ① Low scene complexity in the map construction video, leading to fewer feature points, which affects SLAM accuracy.

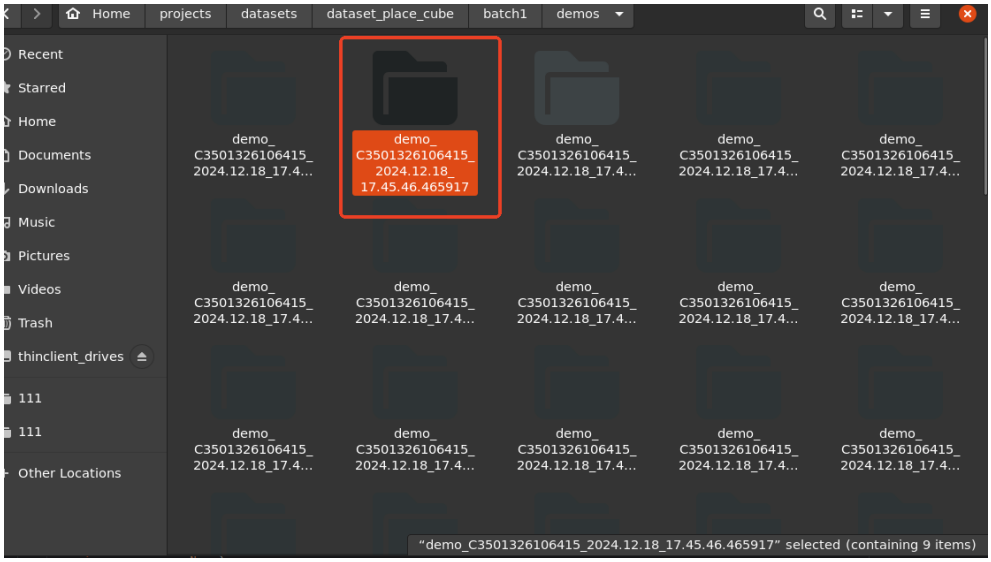


         ② Limited data volume. In such cases, it is recommended that

users manually delete the low-quality trajectory data, as shown in the figure below:

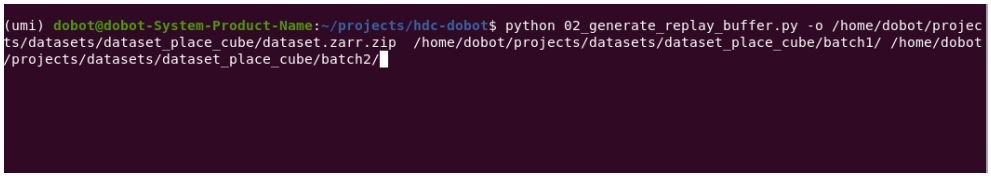
demo_C3501326106415_2024.12.18_17.45.46.465917.jpg

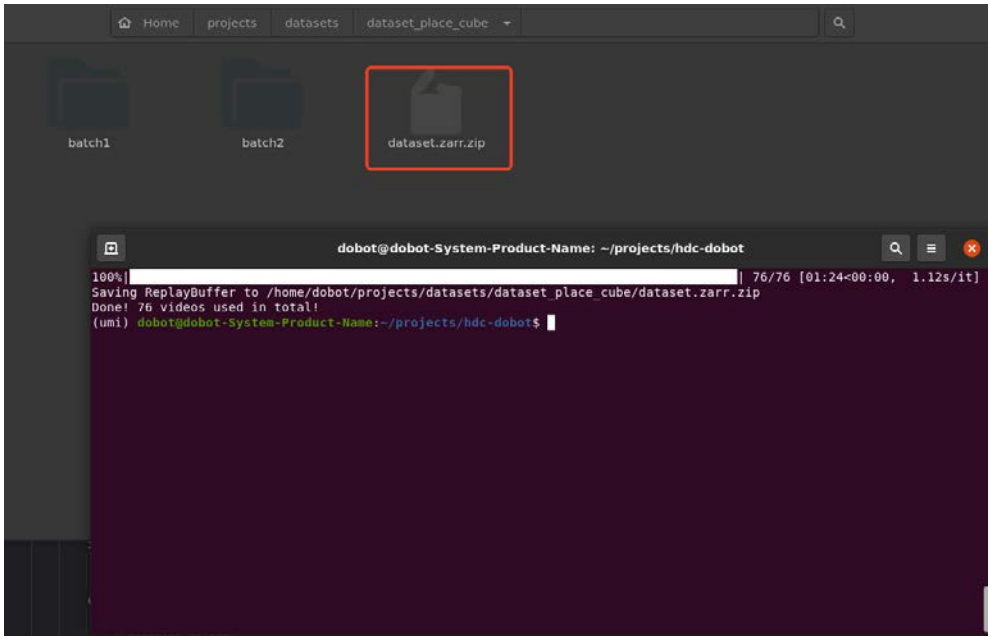
Locate the folder, find the trajectory file with the same name and delete it.



4. Compress and package data: Enter the following command to multiple batches of collected data into one compressed archive.

```
python 02_generate_replay_buffer.py -o
/home/dobot/projects/datasets/dataset_place_cube/dataset.zarr.zip
/home/dobot/projects/datasets/dataset_place_cube/batch1/
/home/dobot/projects/datasets/dataset_place_cube/batch2/
```

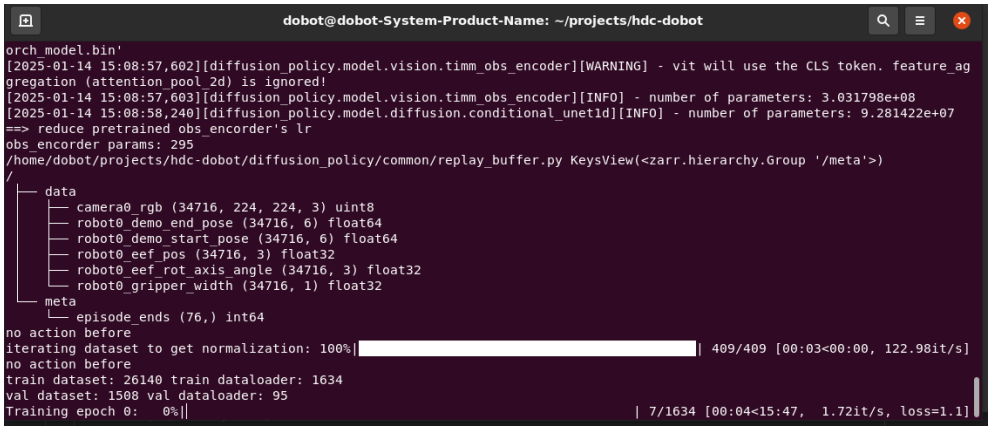




4. Model Training

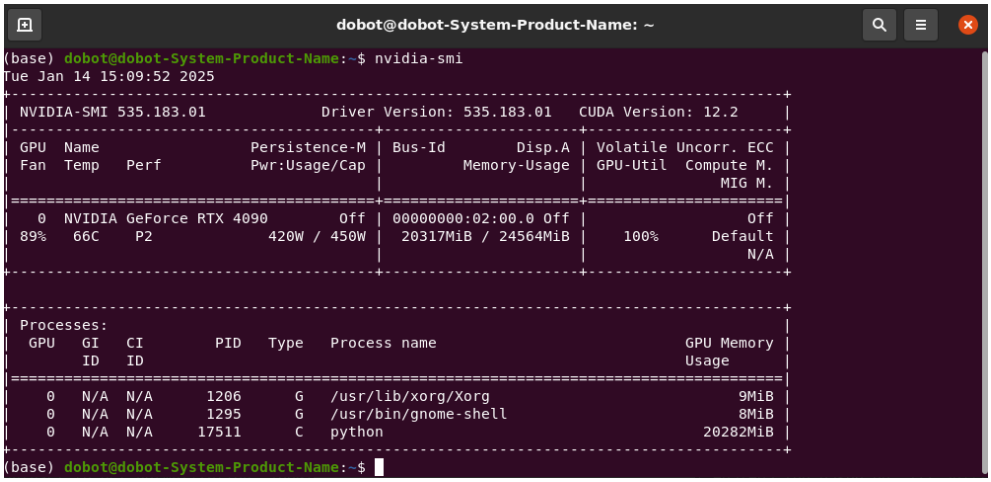
1. Referencing the data collected in Chapter 3, run the following command in the terminal.

```
python 03_train.py --config-name=train_diffusion_unet_timm_umi_workspace
task.dataset_path=/home/dobot/projects/datasets/dataset_place_cube/dataset.zarr.zip
```

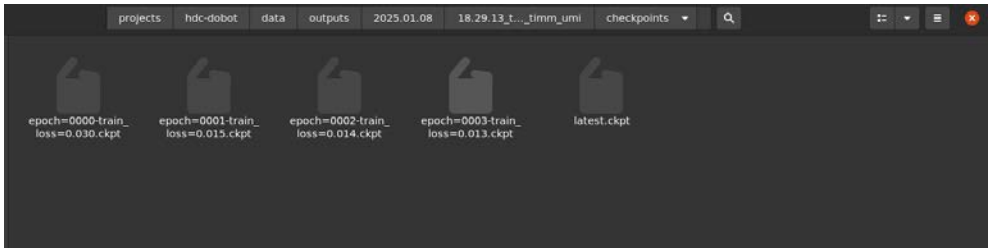


NOTICE

It is recommended to use a graphics card with at least 11GB of VRAM for training.

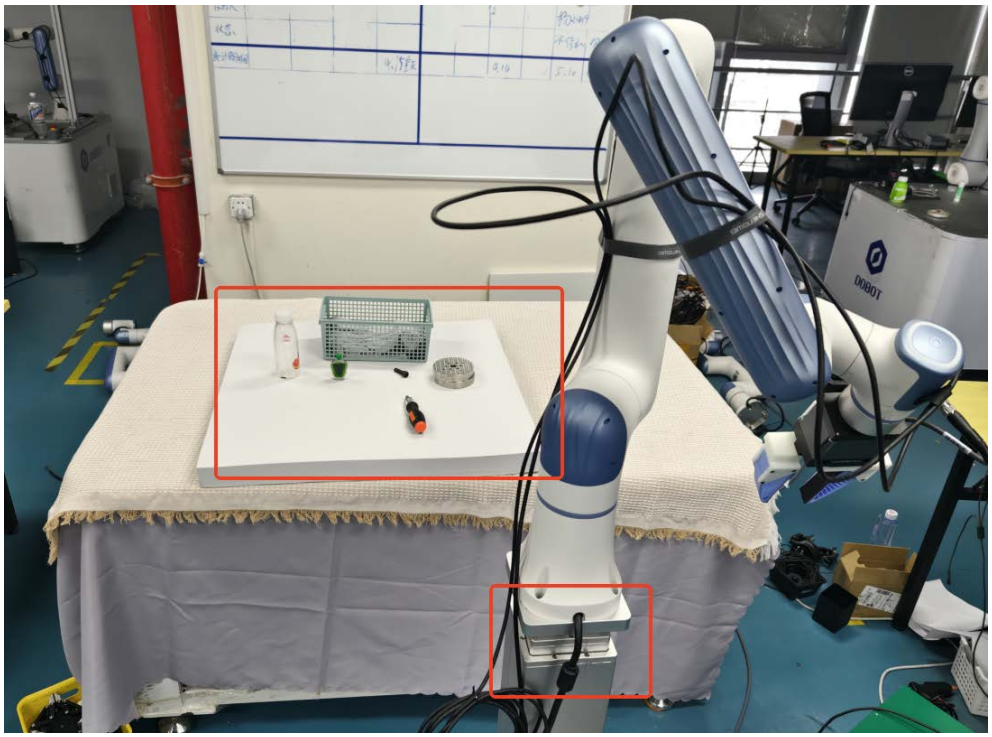


2. After training is complete, the trained model will be saved in the "checkpoints" folder within the following directory.

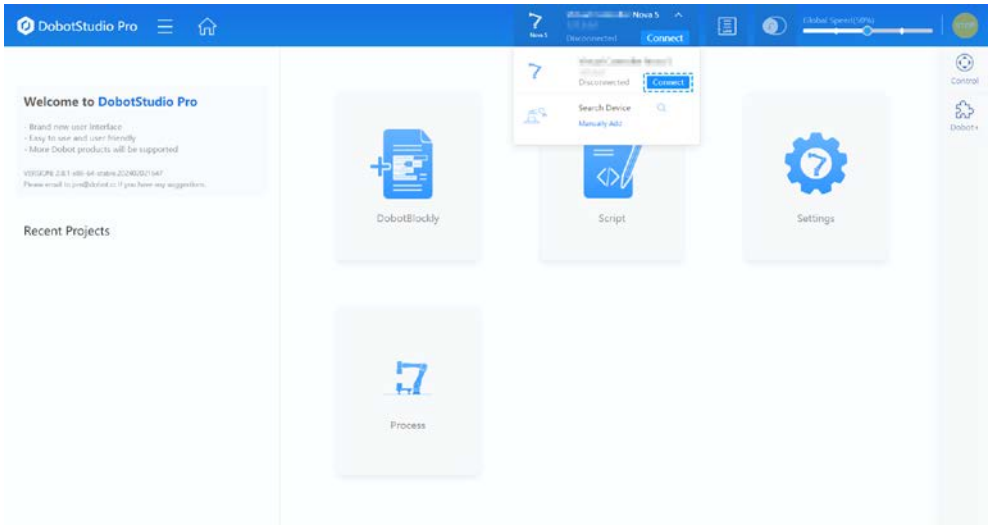


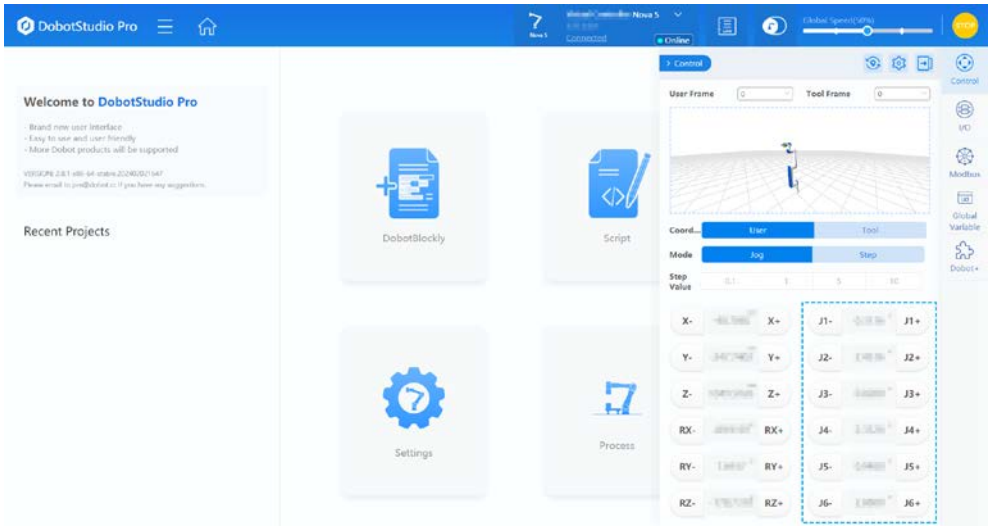
5. Autonomous Inference

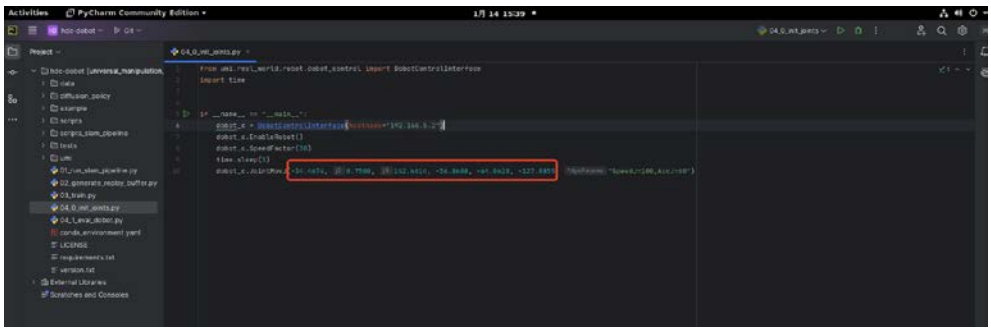
1. Ensure the robot arm is correctly installed, the tail wiring direction of the robot arm should be back towards the target worktable.
2. Manually move the gripper to approximately the same position and posture as at the start of data collection (precise alignment is not required).



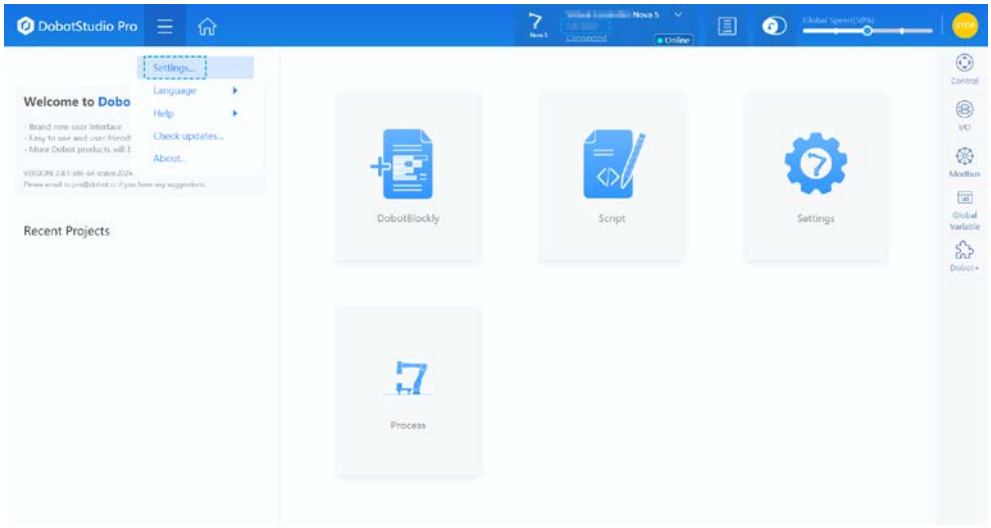
3. Open DobotStudio Pro, connect to the robot arm, and record the current joint angles. Use these values as the initial position and enter them into the script.

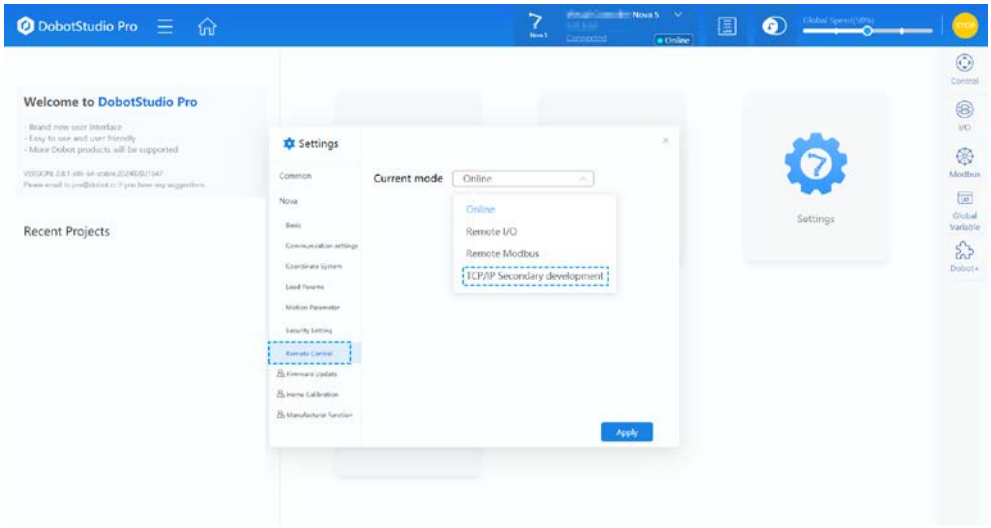


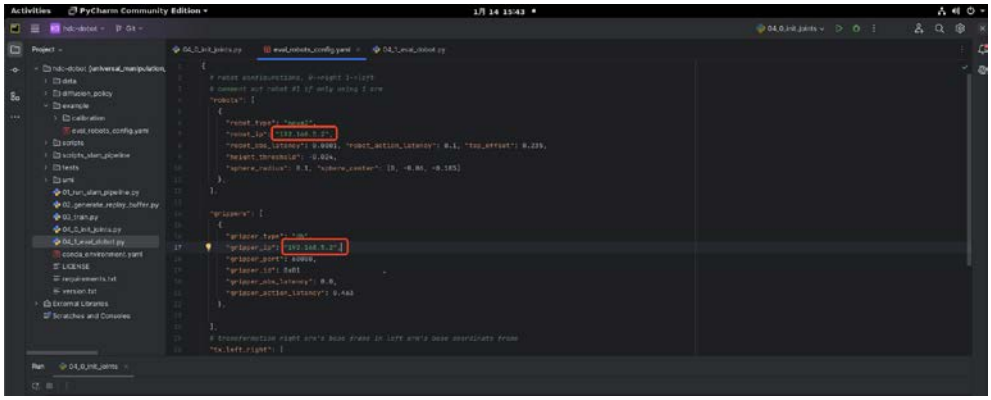




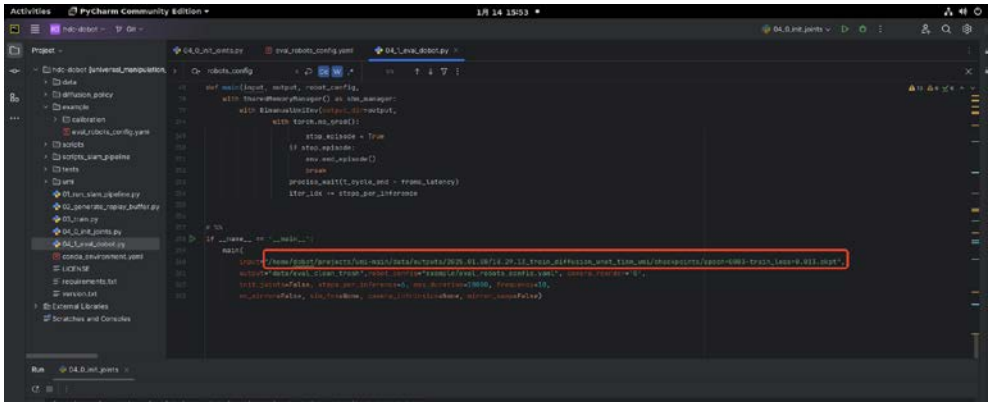
4. Set the robot arm to remote mode and enter its IP address in the script (the robot arm and gripper share the same IP).







5. Modify the path to the trained model in the script.



6. Run the following commands in the terminal (ensure the robot's operational safety).

Initialize the robot arm

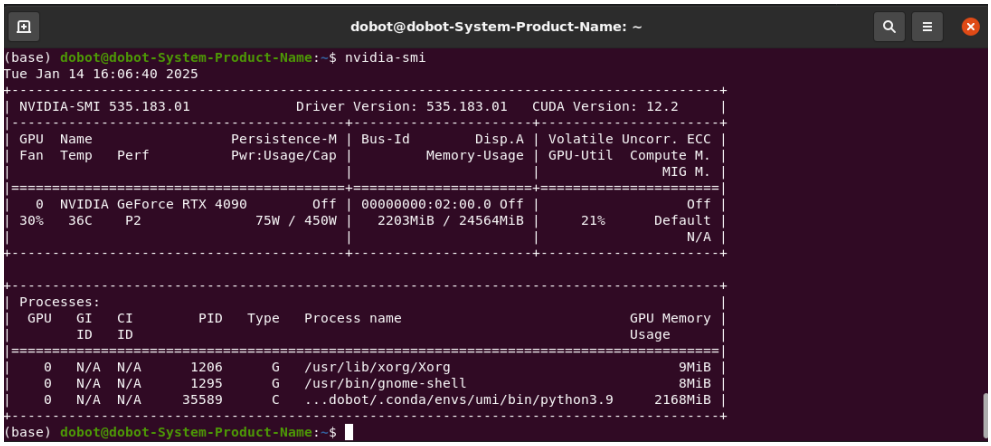
python 04_0_init_joints.py

Autonomous inference

python 04_1_eval_dobot.py

NOTICE

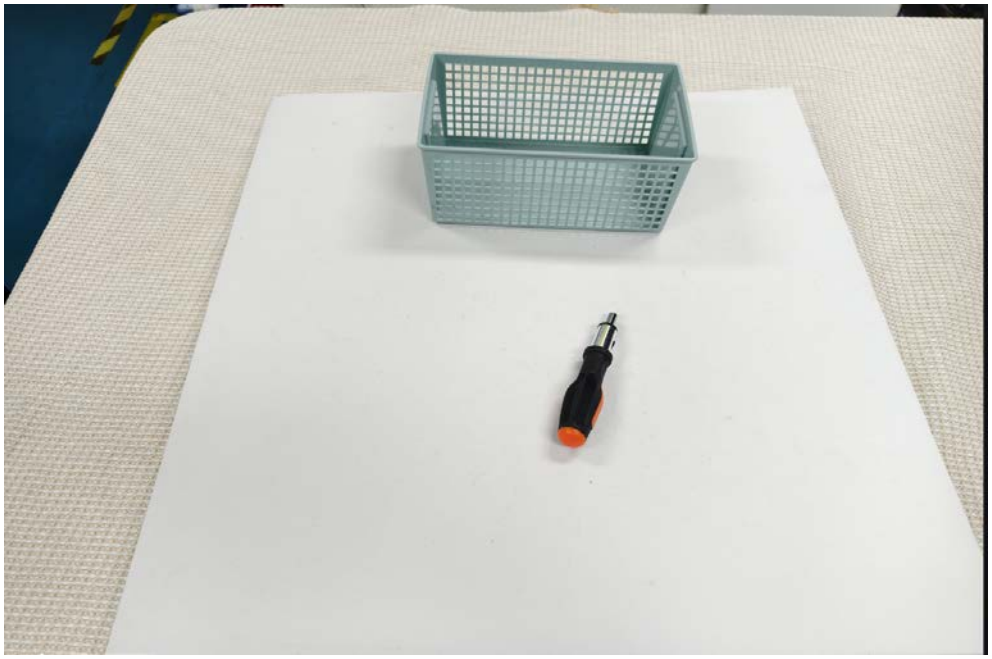
At least 2GB of VRAM is required for inference.



6. Task Example

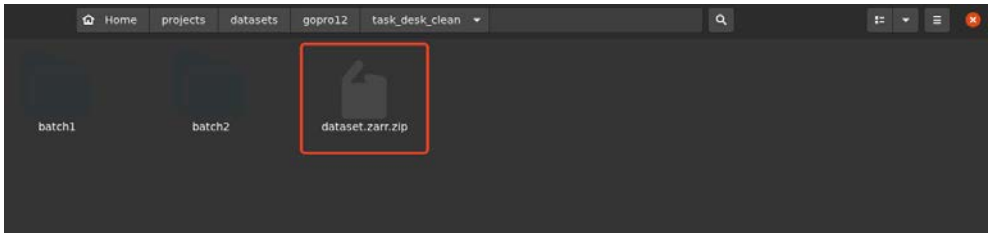
The PocketGo product provides an example that includes:

- Sample dataset. Users can use this dataset to train new models.
- Pre-trained task model. A ready-to-use model that allows for autonomous inference of a pick-and-place task.
- Example task: The robot picks up objects from the tabletop and places them into a storage box. The scenario is illustrated below:



6.1 Training

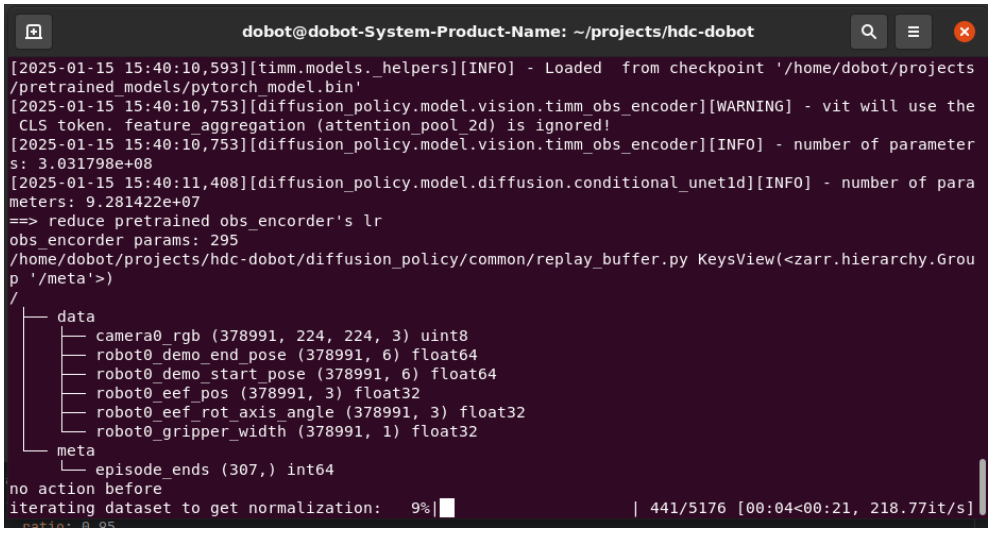
1. Copy the provided dataset to a known directory.



2. Enter the training command.

```
python 03_train.py --config-name=train_diffusion_unet_timm_umi_workspace task.dataset_path=/home/dobot/projects/datasets/gopro12/task_desk_clean/dataset.zarr.zip
```

3. Wait for the training process to complete.



6.2 Running the Sample Model

1. Copy the provided sample model to a known directory.

2. Follow the steps outlined in the Autonomous Inference section to execute the task.

7. Q&A

Q1: Why is data standardization processing slow?

A1: This is due to the computational complexity of the SLAM algorithm. Please be patient while the process completes.

Q2: Why is the performance unsatisfactory?

A2:

(1) Check data quality, including SLAM success rate and trajectory maps.

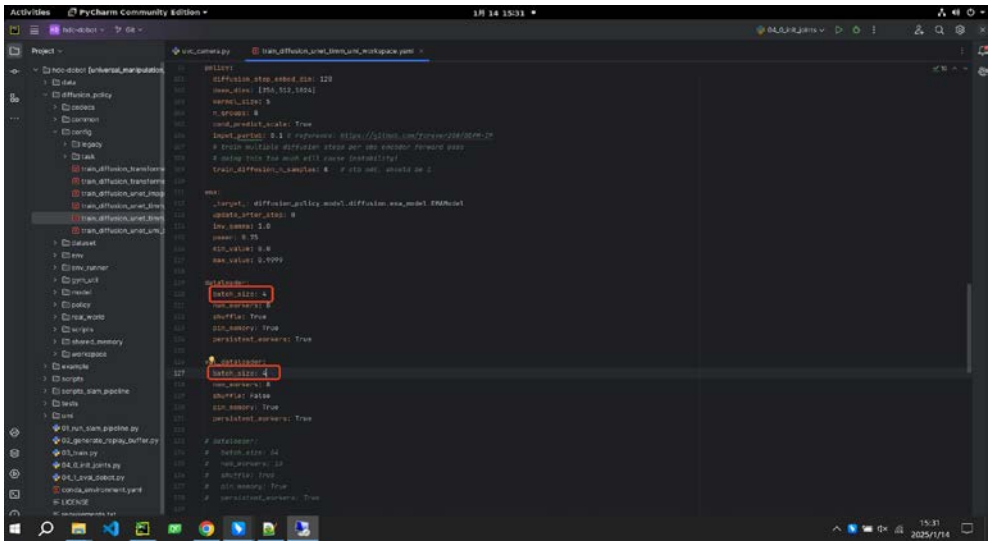
(2) Increase the data volume, at least 200 task samples are recommended.

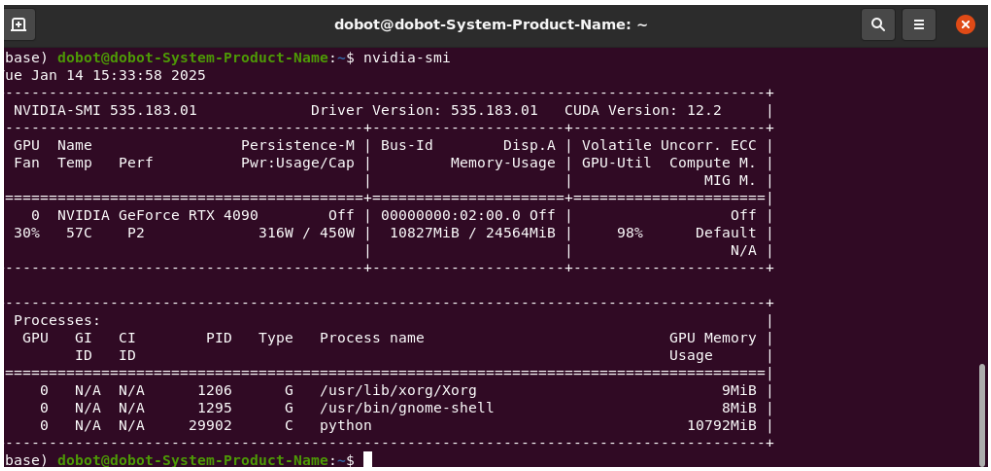
Q3: What if there is insufficient VRAM during training?

A3:

(1) Try modifying the batch_size, a minimum of around 10GB of VRAM is required.

(2) It is highly recommended to use a high-performance graphics card (e.g., we use a 24GB RTX 4090).





Q4: What if the sample model's performance unsatisfactory?

A4: Consider expanding the dataset with on-site data.

