Linker Hand L20

Product Manual

1. Product Overview

Product Overview

The Linker Hand L20 is a high-performance dexterous hand with 20 degrees of freedom. It can simulate the natural grasping method similar to that of a human hand, enabling more diverse and precise operations. Adopting the linkage transmission method and being driven by self-developed motors, it can achieve precise control and excellent motion performance.

Equipped with a variety of sensors, including force sensors, vision sensors, and tactile sensors, it can realize multi-modal perception, endowing it with stronger environmental adaptability and intelligent interaction capabilities. It is compatible with ROS and QT applications, provides ROS

plugins, and supports users' secondary development, thus expanding more application possibilities.

With its high degree of freedom, multi-modal perception ability, and force-position hybrid algorithm, the Linker Hand L20 can be widely applied in scenarios such as education and scientific research, industrial automation, household assistance, and health care and nursing.

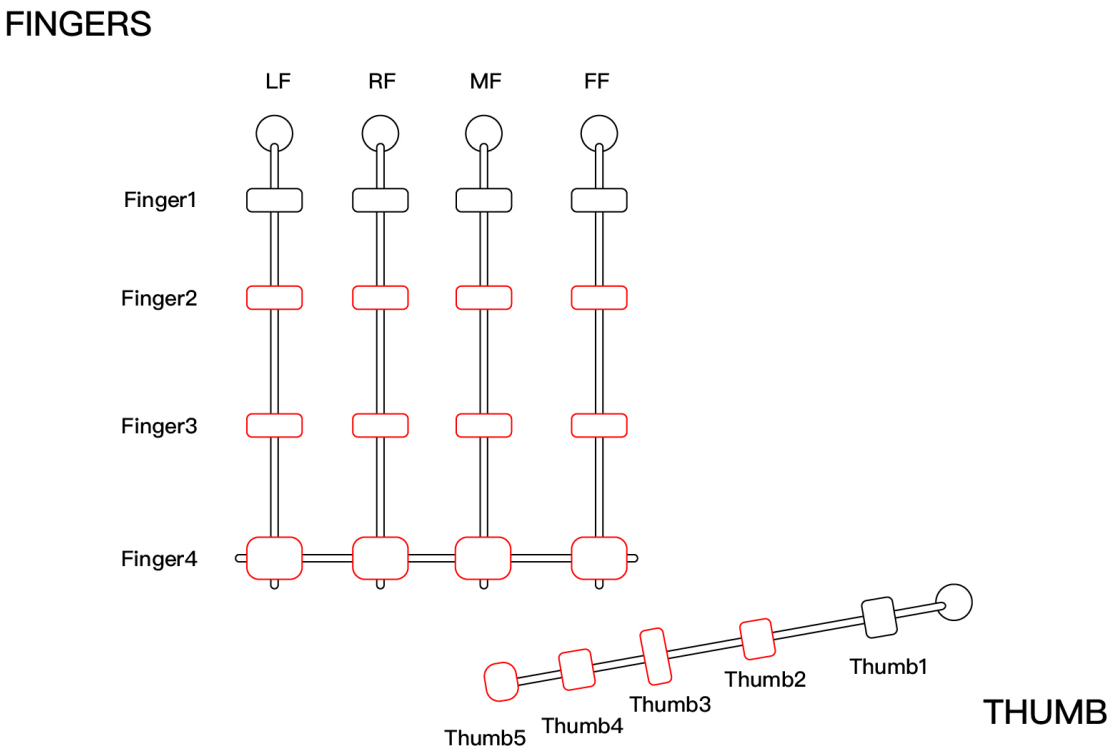
Main Features

1. **High Degree of Freedom**
   - Each finger independently has 4 degrees of freedom, meeting the needs of complex and precise operations.
2. **Multi-sensors**
   - Equipped with an advanced multi-sensor system, including cameras, electronic skin, etc., it constructs an all-round visual perception mode, realizes precise environmental perception and interaction ability, and adapts to a variety of application scenarios.
3. **Cloud-based intelligent**
   - With innovative cloud-based intelligent technology, users do not need to write code. They can quickly deploy through the skill library cloud service, achieve efficient customized operations, and reduce the difficulty of use.
4. **Data Collection**
   - With the self-developed high-efficiency data collection capability, a data farm can be established.
5. **Sturdy and Durable**
   - It is impact-resistant and can withstand drops, making it suitable for various training scenarios of embodied intelligence.

2. Product Display

3. Example of Degrees of Freedom

L20（16 ctive degrees of freedom+5 passive degrees of freedom）

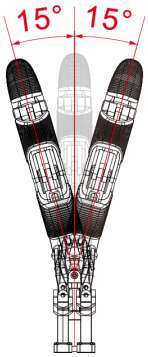


Joints	Min（°）	Max（°）	Note
LF2、RF2、MF2、FF2、THUMB2（Bending）	0	LF3、RF3、MF3、FF3—>80 THUMB3—>74	
LF3、RF3、MF3、FF3、THUMB3（Bending）	0	LF3、RF3、MF3、FF3—>87 THUMB3—>50	
LF4、RF4、MF4、FF4、THUMB4（Side swing）	0	LF3、RF3、MF3、FF3—>±30 THUMB3—>95	
THUMB5(Rotation)	0	82	

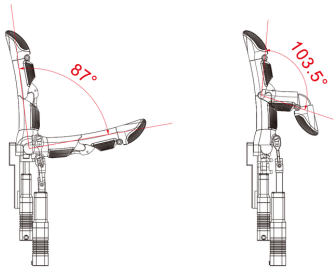
4. Motion Space Range

Four-finger Structure

For the four-finger structure of the Linker Hand L20 dexterous hand, the second joint and the first joint of the four fingers can be manually adjusted at an angle according to the shape of the object to be grasped. The second joint and the third joint of the four fingers achieve coupled linkage through the connection of linkages.



Four fingers side swing angle schematic diagram



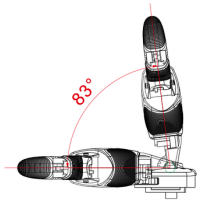
the first joint of the four fingers

the second joint joint of the four fingers

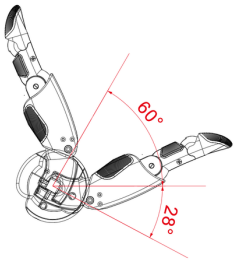
the third joint of the four fingers

Thumb Structure

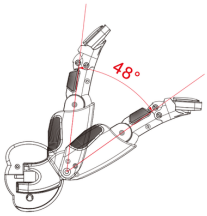
For the thumb structure of the Linker Hand L20 dexterous hand, the third joint and the second joint of the thumb achieve coupled linkage through the connection of linkages, and the first joint of the thumb is independently driven by a rotating motor.



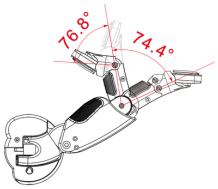
Side swing thumb



Rotating thumb



First joint of thumb



Second and third joints of thumb

5. Scene Demostration



6. Sensor System

Tactile Sensing (Standard Configuration)

The Linker Hand L20 is equipped with fingertip sensors that utilize high-sensitivity sensing technology. It can predict and sense the presence and distance of objects. When in contact with an object, it can accurately capture three - dimensional forces and identify changes in surface texture and temperature.

Fingertip Tactile Sensors

Technical parameters		
sampling frequency		⩾50Hz
		0-20N
force application	measurement accuracy	0.1N
	measurement resolution	0.5%FS
	measurement precision	2%FS
	measurement accuracy	0.25N
	directional resolution	45°

Miniature Three-dimensional Force Sensors Inside the Fingers

Technical parameters	
Measurement range	0-50N
maximum force endured	200N
measurement resolution	0 . 2N
fastest measurement rate	1KHz

Vision (Optional Configuration)

The Linker Hand dexterous hand adopts the design method of high-sensitivity fingertip cameras + palm cameras + wrist cameras, enabling multi-visual fusion perception. The minimum remote operation system is equipped with a depth camera on the arm.

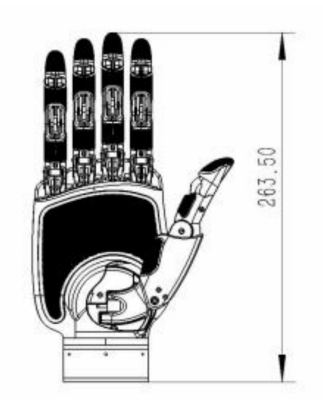
Visual-tactile Perception (Optional Configuration)

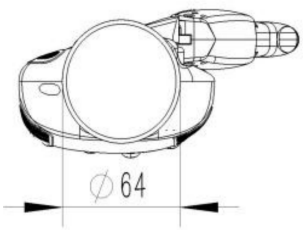
The Linker Hand dexterous hand has a visual-tactile perception mode, which essentially combines vision with a large deep learning model. The principle of this technical solution is to use a high-precision camera to capture the deformation of variable flexible materials. When subjected to force, the shape of the grid deforms, and our miniature binocular cameras record this deformation. Then, based on the well-trained large deep learning model, the depth information and movement trend of the object are mapped out.

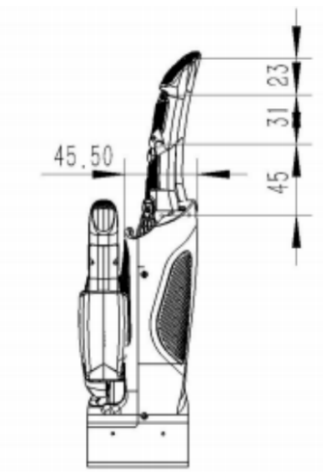
7. Product Parameters

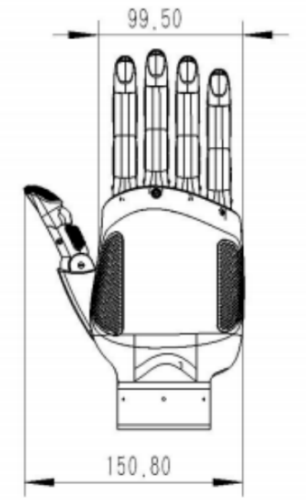
Product	Linker Hand L20
Degrees of Freedom	18
Number of Joints	16+2active. 5passive
Transmission Mode	Linkage drive
Drive Mode	Self-developed joint module
Control Interface	CAN/RS485
Weight	1000g
Maximum Load	5kg
Operating Voltage	DC24V±10%
Static Current	0.2A
Average Current during No-load Movement	1A
Maximum Current	3A
Repeat Positioning Accuracy	±0.20mm
Maximum Grip Force of the Thumb	15N
Maximum Grip Force of the Four Fingers	10N
Maximum Lateral Rotation Range of the Thumb	1.54rad （88.3°）
Bending Angle of the Four Fingers	1.57rad （90°）
Side Swing Speed of the Thumb	3.28rad/s(188.17°/s)
Bending Speed of the Four Fingers	1.99rad/s （114.29°/s）
Bending Speed of the Thumb	1.07rad/s （62.26°/s）

8. Appearance Dimensions









9. Communication Methods

All versions of the dexterous hand support the use of the CAN bus debugging port or EtherCAT.

EtherCAT (Ethernet for Control Automation Technology) is a fieldbus based on 100Mbps Ethernet. It is currently used in many systems, and the latest version of the Linker Hand can be well compatible with both the EtherCAT and ROS systems. Using EtherCAT or CAN in combination with ROS requires a multi-core PC with good performance or our AI-Box device, as well as a standard Ethernet port. The EtherCAT protocol used by the Linker Hand mainly relies on the upper host computer or AI-Box to complete this work.

Supported Functions

- Enable and disable position control
- Change the PID values for torque control
- Perform restrictive operations, such as cutting off force, current, temperature, etc.
- Reset the motor
- Adjust the data transmission rate of the motor and tactile sensors
- Track errors and status indicator lights in the components
- Download the latest firmware to the motor module
- Download the latest Skill function module to the hand controller
- Obtain data from the vision sensor

Control Strategy

Under the default configuration, EtherCAT can rely on the host computer or AI-Box to implement the position control strategy. More complex control algorithms can be used, which can integrate the information of joint and tactile sensors, and even achieve the integration of visual signals through ROS. The torque loop inside the motor unit closes at a frequency of 5kHz. The PID

settings of this loop can be changed in real time. If different control strategies are required, you can purchase existing control strategies from the Skill Store cloud service and use them directly without programming. It also supports downloading new firmware to the motor.

Microcontroller

The Linker Hand uses a self-developed microcontroller for the embedding of the entire robot system. All microcontrollers are connected to the internal CAN bus and can be accessed through the EtherCAT interface.

10. Code Examples

Examples of the Linker Hand ROS SDK program

examples

- 0000-linker_hand_pybullet (PyBullet Simulation example)
- 0001-get_linker_hand_state (Obtain the current status of the Linker hand)
- 0002-gui_control(control it through the graphical interface)
- 0003-get_linker_hand_force (obtain the data of the Linker hand force sensor)
- 0004-get_linker_hand_speed (Obtain the current speed of the Linker hand force)
- 0005-get_linker_hand_current (Obtain the current current of the Linker hand force)

- 0101-lipcontroller (The tactile sensor cooperates with the Linker hand to perform pinching actions)
- 0102-gesture-Show-OK (Use Python to control the hand to make an "OK" gesture)
- 0103-gesture-Show-Surround-Index-Finger (Use Python to control the hand to rotate the index finger)
- 0104-gesture-Show-Wave (Use Python to control the hand to make a wave motion)
- 0105-gesture-Show-Ye (Use Python to control the hand to perform a set of complex demonstration actions)

- 1001-human-dex (Use the Linker Hand for imitation learning training and achieve autonomous object grasping)
- 1002-linker_unidexgrasp (The Unidexgrasp dexterous hand grasping algorithm based on the Linker Hand)

Instructions for the Linker Hand Configuration File

Whether it is the real LinkerHand or the simulated one, the parameter file needs to be configured first. Modify the corresponding configuration parameters according to actual needs.

(1) Modify the configuration file and configure it for the physical LinkerHand or the simulation environment.

```
$ cd Linker_Hand_SDK_ROS/src/linker_hand_sdk/linker_hand_sdk_ros/config
$ sudo vim setting.yaml
```

Since the graphical interface can only control one LinkerHand individually, corresponding configurations need to be made in the configuration file to match the physical LinkerHand.

Linker Hand Example 100

The LinkerHand Example 100 provides rich example cases and source code, fully demonstrating the functions of the LinkerHand.

- Preparation
  Start the SDK

```
#Open a new terminal  Start ros
$ roscore
```

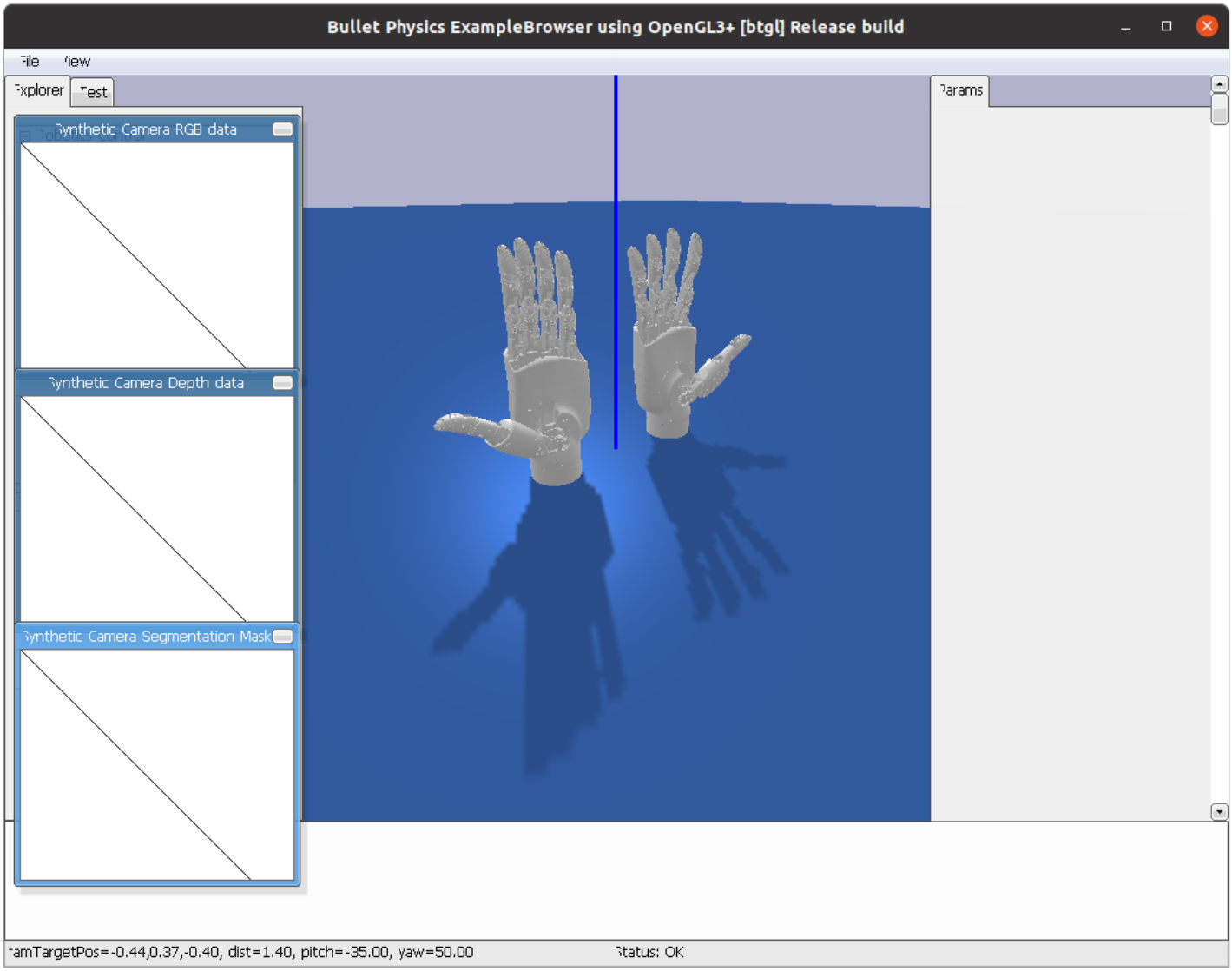
Open a new terminal and start ROS SDK

```
$ cd Linker_Hand_SDK_ROS/
$ source ./devel/setup.bash
$ roslaunch linker_hand_sdk_ros linker_hand.launch
```

0000-PyBullet Simulation Example

Open a new terminal. The SDK can control the LinkerHand in the simulator.

```
$ cd Linker_Hand_SDK_ROS/
$ source ./devel/setup.bash
$ rosrun linker_hand_pybullet linker_hand_pybullet.py
```



0001-Obtain the Current Status of the Linker Hand . The status values include range values and radian values

Open a new terminal

```
$ cd Linker_Hand_SDK_ROS/
$ source ./devel/setup.bash
_If the loop parameter is True, the terminal will print the current status values of the LinkerHand dexterous hand in a loop. If it is False, the terminal will only print the current status values of the LinkerHand dexterous hand once
$ rosrun L20_get_linker_hand_state L20_get_linker_hand_state.py _loop:=True
```

0002-Graphical Interface Control

The graphical interface control allows you to control the independent movement of each joint of the LinkerHand L10 and L20 through sliders. You can also add buttons to record the values of all current sliders and save the current movement status of each joint of the LinkerHand. Replay actions through functional buttons.

Controlling LinkerHand using gui_control:

To control the dexterous hand through the gui_control interface, you need to start linker_hand_sdk_ros and operate the LinkerHand in the form of a topic.

```
Open a new terminal  Start ros
$ roscore
```

Open a new terminal start ROS SDK

```
$ cd Linker_Hand_SDK_ROS/
$ source ./devel/setup.bash
$ roslaunch linker_hand_sdk_ros linker_hand.launch
```

After a successful startup, there will be prompt information such as the SDK version, CAN interface status, dexterous hand configuration information, and the current joint speed of the Linker hand

Open a new terminal to start the gui control.

Copy the bash code

```
$ cd Linker_Hand_SDK_ROS/
$ source ./devel/setup.bash
$ rosrun gui_control gui_control.py
```

After starting, the UI interface will pop up. You can control the movement of the corresponding LinkerHand joints through the sliders. And you can save the current slider data by adding buttons on the right side for reuse

0003-Obtain the data of the force sensor of the Linker Hand

Open a new terminal

```
$ cd Linker_Hand_SDK_ROS/
$ source ./devel/setup.bash
_If the oop parameter is True, the terminal will print the current status values of the LinkerHand dexterous hand in a loop. If it is False, the terminal will only print the current status values of the LinkerHand dexterous hand once
$ rosrun get_linker_hand_force get_linker_hand_force.py _loop:=False
```

```
#2025-01-15 15:43:16  no data for the left hand#2025-01-15 15:43:16  The normal force of the five fingers of the right hand: [0.0, 0.0, 0.0, 0.0, 0.0]#2025-01-15 15:43:16  The tangential force of the five fingers of the right hand: [0.0, 0.0, 0.0, 0.0, 0.0]#2025-01-15 15:43:16  The direction of the tangential force of the five fingers of the right hand: [255.0, 255.0, 255.0, 255.0, 255.0]#2025-01-15 15:43:16  The proximity sensing of the five fingers of the right hand: [0.0, 0.0, 0.0, 0.0, 0.0]
```

0004-Obtain the current speed of the force of the Linker Hand

Open a new terminal

```
$ cd Linker_Hand_SDK_ROS/
$ source ./devel/setup.bash
_If the oop parameter is True, the terminal will print the current status values of the LinkerHand dexterous hand in a loop. If it is False, the terminal will only print the current status values of the LinkerHand dexterous hand once
$ rosrun get_linker_hand_speed get_linker_hand_speed.py _loop:=False
#2025-01-15 15:57:17  no data for the left hand#2025-01-15 15:57:17  The current of the five fingers of the right hand is: [180, 250, 250, 250, 250]
```

0005 - Obtain the current current of the force of the Linker Hand

Open a new terminal

```
$ cd Linker_Hand_SDK_ROS/
$ source ./devel/setup.bash
_If the oop parameter is True, the terminal will print the current status values of the LinkerHand dexterous hand in a loop. If it is False, the terminal will only print the current status values of the LinkerHand dexterous hand once
$ rosrun get_linker_hand_current get_linker_hand_current.py _loop:=False
#2025-01-15 16:25:29  no data for the left hand#2025-01-15 16:25:29  The current of the five fingers of the right hand is: [42, 42, 42, 42, 42]
```

0101-Pinch operation with the tactile sensor in cooperation with the Linker Hand

To use this example, you need to start linker_hand_sdk_ros

```
Open a new terminal  Start ros
$ roscore
```

Open a new terminal start ROS SDK

```
$ cd Linker_Hand_SDK_ROS/
$ source ./devel/setup.bash
$ roslaunch linker_hand_sdk_ros linker_hand.launch
```

After a successful startup, there will be prompt information such as the SDK version, the status of the CAN interface, the configuration information of the dexterous hand, and the current joint speed of the dexterous hand.

Open a new terminal to use the demonstration example

```
python ./<Your file path>/lipcontroller.py
```

- If the terminal prints out "**Start the demonstration**", it means the normal operation. At this time, if the hand settings are correct, the index finger and middle finger should start to perform the pinching action. It will stop when pinching an object, and will continue to try to pinch after the object is taken away until it pinches an object or reaches the limit. The limit state is shown in the following figure

- lipcontroller.py This is a demonstration demo developed based on version 7. When applying it to the demonstrations of other versions, you need to adjust the opposing posture of the thumb and index finger. Otherwise, the action of "**pinching the index finger and thumb together**" cannot be achieved.

0102- Use Python to control the hand to make an "OK" gesture

To use this example, you need to start linker_hand_sdk_ros

Copy the bash code

```
Open a new terminal  Start ros
$ roscore
```

Open a new terminal start ROS SDK

```
$ cd Linker_Hand_SDK_ROS/
$ source ./devel/setup.bash
$ roslaunch linker_hand_sdk_ros linker_hand.launch
```

After a successful startup, there will be prompt information such as the SDK version, the status of the CAN interface, the configuration information of the Linker hand, and the current joint speed of the Linker hand.

```
python ./<Your file path>/gesture-Show-OK.py
#After starting, the terminal will print "Testing", and at this time, the hand will start to make an "OK" gesture, and the middle finger, ring finger, and little finger will bend and straighten
```

0103- Use Python to control the hand to make the index finger rotate

To use this example, you need to start linker_hand_sdk_ros

```
Open a new terminal  Start ros
$ roscore
```

Open a new terminal start ROS SDK

```
$ cd Linker_Hand_SDK_ROS/
```

```
$ source ./devel/setup.bash
$ roslaunch linker_hand_sdk_ros linker_hand.launch
```

After a successful startup, there will be prompt information such as the SDK version, the status of the CAN interface, the configuration information of the Linker hand, and the current joint speed of the Linker hand.

Open a new terminal to use the demonstration example

```
python ./<Your file path>/gesture-Show-Surround-Index-Finger.py
#After starting, the terminal will print "Testing", and at this time, the hand will start to clench into a fist and extend the index finger, and the index finger will keep repeating the rotation
```

0104- Use Python to control the hand to make wave motion

To use this example, you need to start linker_hand_sdk_ros

Copy the bash code

```
Open a new terminal  Start ros
$ roscore
```

Open a new terminal start ROS SDK

```
$ cd Linker_Hand_SDK_ROS/
$ source ./devel/setup.bash
$ roslaunch linker_hand_sdk_ros linker_hand.launch
```

After a successful startup, there will be prompt information such as the SDK version, the status of the CAN interface, the configuration information of the Linker hand, and the current joint speed of the Linker hand.

Open a new terminal to use the demonstration example

Copy the bash code

```
python ./<Your file path>/gesture-Show-Wave.py
#After starting, the terminal will print "Testing", and at this time, the thumb of the hand will stretch outward and remain still, while the other four fingers will start to make wave motions
```

0105- Use Python to control the hand to perform a set of complex demonstration actions

To use this example, you need to start linker_hand_sdk_ros

```
Open a new terminal  Start ros
$ roscore
```

Open a new terminal start ROS SDK

```
$ cd Linker_Hand_SDK_ROS/
$ source ./devel/setup.bash
$ roslaunch linker_hand_sdk_ros linker_hand.launch
```

After a successful startup, there will be prompt information such as the SDK version, the status of the CAN interface, the configuration information of the Linker hand, and the current joint speed of the Linker hand.

Open a new terminal to use the demonstration example

Copy the bash code

```
python ./<Your file path>/gesture-Show-Ye.py
#After starting, the terminal will print "Testing", and at this time, the hand will start to perform a set of complex movements to demonstrate the flexibility of the hand
```

-This example is a demonstration demo developed based on version 7. When applying it to the demonstrations of other versions, you need to adjust the opposing posture of the thumb and index finger. Otherwise, the action of "**pinching or opposing the index finger and thumb together**" cannot be achieved.

1001- Use LinkerHand for imitation learning training

To use this example, you need to use the ROS Noetic system on Ubuntu 20.04. The hardware is the LinkerRobot humanoid robot. You can also use other robotic arms or robots for imitation learning training, as long as you modify the corresponding data topics.

For detailed usage instructions, please refer to thehuman-dex README.md

1、Configuration environment

```
cd human-dex
conda create -n human-dex python=3.8.10
conda activate human-dex
pip install torchvision
pip install torch
pip install -r requirements.txt
```

2、install

```
mkdir -p your_ws/src
cd your_ws/src
git clone https://github.com/linkerbotai/human-dex.git
cd ..
catkin_make
source ./devel/setup.bash
```

3、run

```
collect data
 roslaunch record_hdf5 record_hdf5.launch
open a new terminal to send the data collection command
rostopic pub /record_hdf5 std_msgs/String "data:
'{\"method\":\"start\",\"type\":\"humanplus\"}'"
```

4、Training

Copy the bash code

```
cd humanplus/scripts/utils/HIT
python3 imitate_episodes_h1_train.py --task_name data_cb_grasp --ckpt_dir
cb_grasp/ --policy_class HIT --chunk_size 50 --hidden_dim 512 --batch_size 48
--dim_feedforward 512 --lr 1e-5 --seed 0 --num_steps 100000 --eval_every 1000
--validate_every 1000 --save_every 1000 --no_encoder --backbone resnet18 --
same_backbones --use_pos_embd_image 1 --use_pos_embd_action 1 --dec_layers 6 -
-gpu_id 0 --feature_loss_weight 0.005 --use_mask --data_aug
```

5、Reproduce/Evaluate

Copy the bash code

```
cd humanplus/scripts
python3 cb.py
```

1002- The Unidexgrasp dexterous hand grasping algorithm based on LinkerHand

The original Unidexgrasp algorithm uses the ShadowHand. The following provides the relevant code for developing the Unidexgrasp algorithm on LinkerHand.
For detailed usage instructions, please refer to thelinker_unidexgrasp

Part of Generating the Grasping Posture

For the part of the grasping posture, a mapping scheme is adopted. The hand posture of the shadowhand output by the model is mapped to the hand posture of the LinkerHand L20 for subsequent development.

1. Configure the environment

Copy the code in the command line

```
conda create -n unidexgrasp python=3.8
conda activate unidexgrasp
conda install -y pytorch==1.10.0 torchvision==0.11.0 torchaudio==0.10.0 cudatoolkit=11.3 -c pytorch -c conda-forge
conda install -y https://mirrors.bfsu.edu.cn/anaconda/cloud/pytorch3d/linux-64/pytorch3d-0.6.2-py38_cu113_pyt1100.tar.bz2
pip install -r requirements.txt
cd thirdparty/pytorch_kinematics
pip install -e .
cd ../nflows
pip install -e .
cd ../
git clone https://github.com/wrc042/CSDF.git
cd CSDF
pip install -e .
cd ../../
```

2. Training
   GraspIPDF

Copy the code in the command line

```
python ./network/train.py --config-name ipdf_config \
                          --exp-dir ./ipdf_train
```

GraspGlow

Copy the code in the command line

```
python ./network/train.py --config-name glow_config \
                          --exp-dir ./glow_train
python ./network/train.py --config-name glow_joint_config \
                          --exp-dir ./glow_train
```

ContactNet

Copy the code in the command line

```
python ./network/train.py --config-name cm_net_config \
                          --exp-dir ./cm_net_train
```

3. Verification

Copy the code in the command line

```
python ./network/eval.py  --config-name eval_config \
                          --exp-dir=./eval
```

1. Mapping
   Visualize the results

Copy the code in the command line

```
python ./tests/visualize_result_l20_shadow.py --exp_dir 'eval' --num 3
```

Save the results for the subsequent development of the reinforcement learning algorithm.

Copy the code in the command line

```
python ./tests/data_for_RL.py
```