Jaka_Ros Instruction Manual

Version： V2.2

Date：2023.8.8

Note:

The definition of collaborative robots follows international ISO standards and national standards to protect the safety of operators. We do not recommend directly applying the robot arm to circumstances where the object is a human body. However, when robot users or application developers do need to involve thohuman body in the robot operation, they should configure a safe, reliable, fully tested, and certified safety protection system for the robot arm to protect personnel safety on the premise that users or application developers can fully evaluate personnel safety.

The contents contained in JAKA_ROS Instruction Manual are the exclusive property of JAKA Robotics Co., Ltd. (Hereinafter collectively referred to as JAKA), and shall not be used in any form without the written consent of JAKA.

JAKA_ROS Instruction Manual is subject to revision and improvement on a regular basis by JAKA and its contents are subject to change without notice. Please check the actual product information carefully before using this manual.

The information contained in JAKA_ROS Instruction Manual is not a commitment of JAKA, and JAKA is not responsible for any errors that may occur in this Manual and any accidental or consequential damages caused by the use of this Manual and the products described therein. Please read this Manual carefully before installing and using the product.

The pictures in this Manual are for reference only, please refer to the actual product.

If the robot arm is modified or disassembled, JAKA will not be responsible for after-sales service.

JAKA also reminds the user that safety equipment must be used and safety provisions must beobserved when using and maintaining JAKA robots.

Programmers of the JAKA robot and designers and debuggers of the robot system shall be familiar withthe programming mode and system application installation of JAKA robots.

Manual Instructions

This manual mainly contains an overview, usage, and development method of JAKA_ROS.

This manual is intended for users with certain basic development skills who have received basictraining in robot usage to facilitate the usage and development of JAKA_ROS.

Read more

For more information about our products, please scan the QR codeon the right to visit our official website www.jaka.com.

version information:

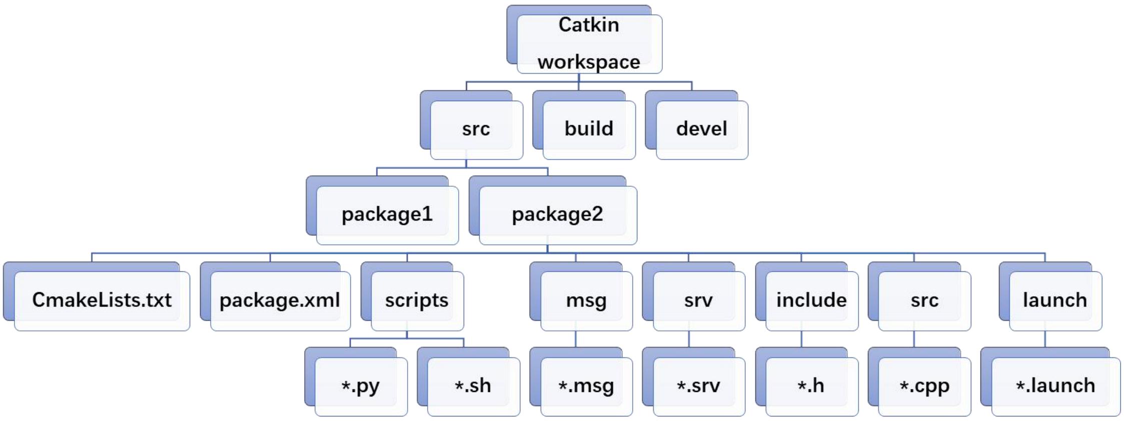
Version	Revision Date	Effective Date	Revision Content	Revision Person
v2.2	2023.8.8	2023.8.15	JAKA SDK version： V2.1.4_8dev-Ming-7_linux	

Content

Chapter 1 Introduction to ROS

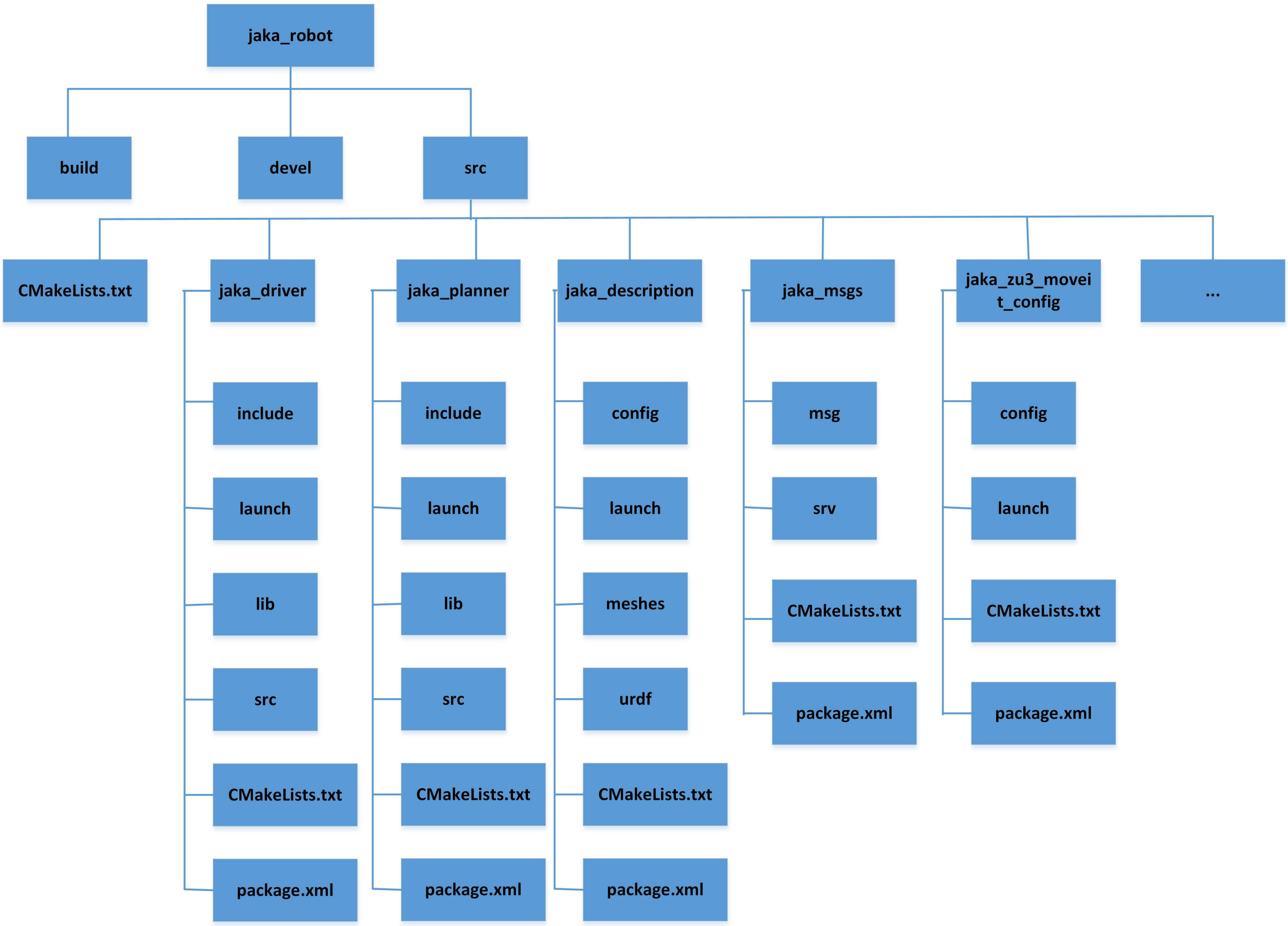
1.1 Structure Introduction

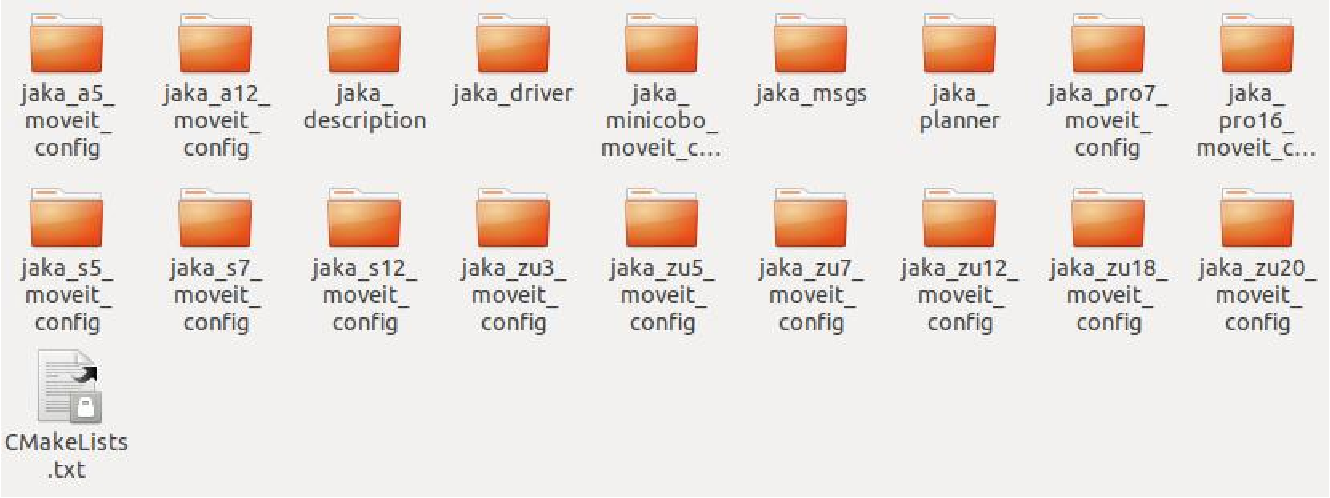
ROS (Robot Operating System) is an open-source meta-operating system for robots. ROS integrates a large number of tools, libraries, and protocols to realize the control of robots. The ROS file system refers to the organizational form of the ROS source code on the hard disk, and its structure is shown in the following figure:



The ROS package structure of the JAKA robot is shown in the figure below:

Note: The build folder and devel folder in the workspace (jaka_robot) need to be recompiled to generate.


jaka_robot
build
devel
src
CMakeLists.txt
jaka_driver
jaka_planner
jaka_description
jaka_msgs
jaka_zu3_moveit_config
...
include
launch
lib
src
CMakeLists.txt
package.xml
include
launch
lib
src
CMakeLists.txt
package.xml
config
launch
meshes
urdf
CMakeLists.txt
package.xml
msg
srv
CMakeLists.txt
package.xml
config
launch
CMakeLists.txt
package.xml


jaka_a5_moveit_config
jaka_a12_moveit_config
jaka_description
jaka_driver
jaka_minicobo_moveit_c...
jaka_msgs
jaka_planner
jaka_pro7_moveit_config
jaka_pro16_moveit_c...
jaka_s5_moveit_config
jaka_s7_moveit_config
jaka_s12_moveit_config
jaka_zu3_moveit_config
jaka_zu5_moveit_config
jaka_zu7_moveit_config
jaka_zu12_moveit_config
jaka_zu18_moveit_config
jaka_zu20_moveit_config
CMakeLists.txt

Each function package is described as follows:

Name	Description
jaka_decription	United Robotics Description Format (urdf)
jaka_driver	Startup package for connecting to the real robot
jaka_jog_panel	Jog motion control panel file
jaka_msgs	Custom message type
jaka_planner	MoveIt trajectory planning package
jaka_minicobo_moveit_config **jaka_pro7_moveit_config** **jaka_pro16_moveit_config** **jaka_zu3_moveit_config** **jaka_zu5_moveit_config** **jaka_zu7_moveit_config** **jaka_zu12_moveit_config** **jaka_zu18_moveit_config** **jaka_a5_moveit_config** **jaka_a12_moveit_config** **jaka_s5_moveit_config** **jaka_s7_moveit_config** **jaka_s12_moveit_config**	Moveit configuration file for each robot model (minicobo, pro7, pro16, zu3, zu5, zu7, zu12, zu18, a5, a12, s5, s7, s12)

1.2 ROS Common Commands

Common commands	Meaning
roscore	Starting the node manager
cd ~/catkin_ws **catkin_make**	Compiling ROS programs
source ./devel/setup.bash	Add the package to the environment variable
rospack list	View package lists and locate packages
rospack find package-name	Finding a package directory
rosrun package-name executable-name	Launching node
rosnode list	Viewing the node list
rosnode info node-name	Viewing node-specific information
rosnode kill node-name	Stopping node
rostopic -h	See all operations of rostopic
rostopic list	View all topic lists
rosrun rqt_graph rqt_graph **rosrun rqt_plot rqt_plot**	Graphical display of topic
rostopic echo [topic]	View a topic information
rostopic type [topic]	View topic message format
rosmsg show [msg_type]	
rosservice -h	View all service operations
rosservice list	View service list
roslaunch package_name file.launch	Run the launch file

Chapter 2 Control the JAKA Robot

2.1 Preparation

2.1.1 Install ROS

Taking Ubuntu 18.04 with x86_64 architecture as an example, the steps to install ROS on Ubuntu 18.04 are as follows:

1. Set up the Ubuntu software repository:

Open a terminal and run the following command to add the ROS repository to the system:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu bionic main" > /etc/apt/sources.list.d/ros-latest.list'
```

2. Add ROS key:

Go on runing the following command in the terminal to add the ROS key:

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

3. Update the package index:

Execute the following command to update the package index:

```
sudo apt update
```

4. Install ROS:

ROS can now be installed. Different installation configurations can be selected, depending on user needs. The following are the steps to fully install the full version of the ROS desktop (including ROS, rqt, rviz, etc.):

```
sudo apt install ros-melodic-desktop-full
```

Note that the ROS Melodic version used here is for Ubuntu 18.04. If you want to install other versions of ROS, just replace "melodic" in the command with the code name of other versions (such as "noetic", "kinetic", etc.).

5. Initialize the ROS environment:

After the installation is complete, the ROS environment needs to be initialized. Run the following command in the terminal:

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

6. Install rosdep:

rosdep can help install dependencies of ROS packages. Run the following command in the terminal:

```
sudo apt install python-rosdep
sudo rosdep init
rosdep update
```

7. Create a workspace:

A ROS workspace needs to be created to organize and build your own ROS packages. Run the following command in a terminal to create a workspace (assuming the workspace is named "catkin_ws"):

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

8. Start using ROS:

ROS has now been successfully installed and a workspace has been created. You can start writing, building, and running ROS packages.

Please note that the above steps are only the basic steps to install and initialize ROS. The specific ROS development and using process may involve more operations and settings. You can refer to the official ROS documentation (http://wiki.ros.org/) for more details and guidance.

2.1.2 Download JAKA_ROS package

The function package (source code) download address of the JAKA robot:

```
https://github.com/JakaCobot/jaka_robot
```

2.1.3 Important Notes

1. Currently only supports ROS1 version.
2. Currently only applicable to x86_64 architecture.
3. The supported ROS versions are melodic and noetic. Manual cases are tested with the melodic version. If some errors occur during use, it is recommended to delete the build folder and devel folder in the workspace (jaka_robot) and the CMakeLists.txt file under the src folder and then recompile.
4. JAKA_ROS controls the robot through the functional interfaces provided by the JAKA SDK. For a detailed description of the interfaces, please refer to the *JAKA C++ User Manual*.
5. Because the DH parameters of the same series of robots are the same, the ROS configuration package of the C5/Pro5 robot can be replaced by the Zu5 configuration package, and the same is true for C7/Zu7, Pro12/Zu12.
6. The jaka_driver server and MoveIt server cannot be started at the same time.

2.2 Control JAKA Robot

2.2.1 Setting up the terminal environment

In ROS, every time you open a new terminal window, you need to run the command "source devel/setup.bash" to set up the correct environment variables. When you open more than one terminal window, you may forget to add the environment variables, which may lead to some errors. To solve this problem, you can set up the ROS workspace as follows so that it will be loaded automatically when you open a terminal.

1. Add the following information at the end of the bash configuration file to automatically load the ROS workspace, and enter the following content in the terminal:

```
echo "source ~/xxx/devel/setup.sh" >> ~/.bashrc
```

Note: xxx is the name of the ros workspace, here my workspace is stored in the home directory, the name is jaka_robot_v2.2, so enter echo "source ~/jaka_robot_v2.2/devel/setup.sh" >> ~/.bashrc, If it is stored in another location, pay attention to modify the path.

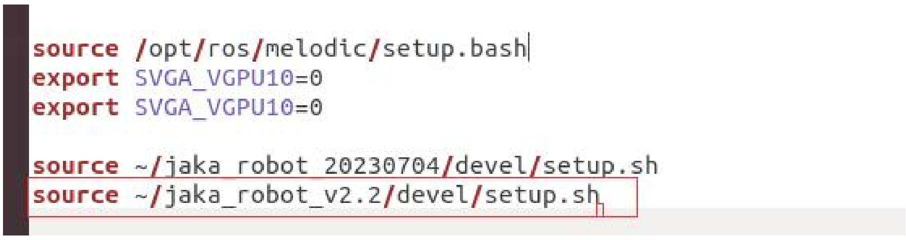
2. To make the configuration take effect, input in the terminal:

```
source ~/.bashrc
```

3. Confirm whether the writing is successful, input in the terminal:

```
gedit ~/.bashrc
```

If you see the following last line in the opened file, it means that the writing is successful:

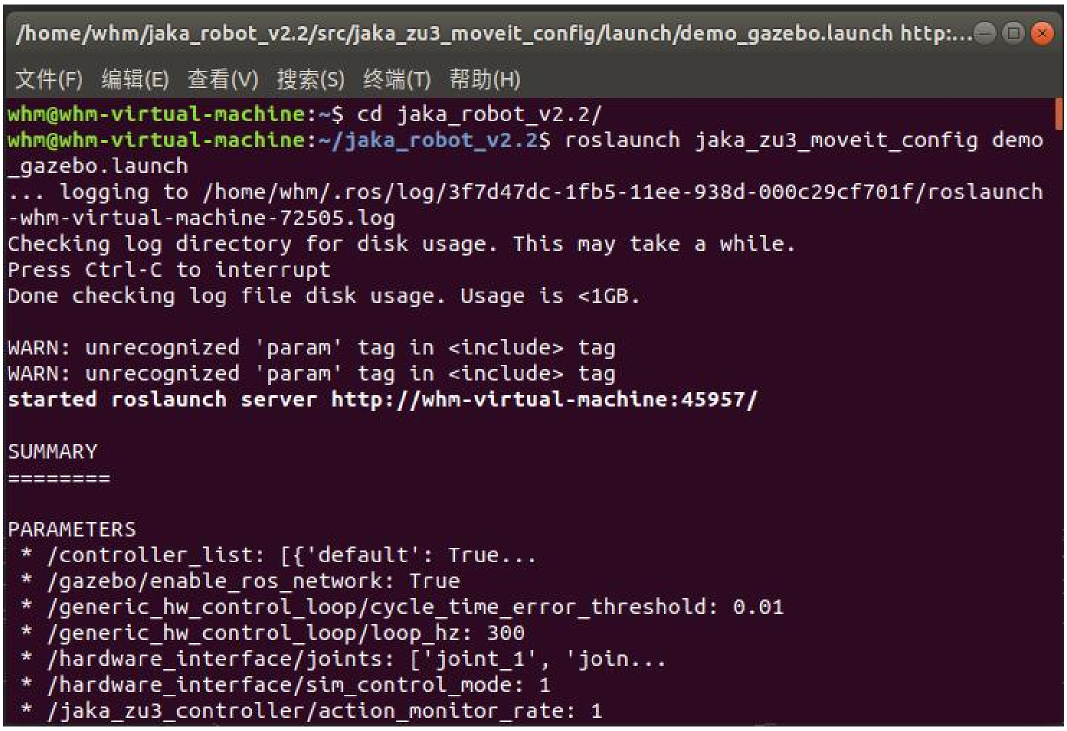


```
source /opt/ros/melodic/setup.bash
export SVGA_VGPU10=0
export SVGA_VGPU10=0

source ~/jaka_robot_20230704/devel/setup.sh
source ~/jaka_robot_v2.2/devel/setup.sh
```

4. Recompile, input in the terminal:

```
catkin_make
```

After that, when you open a new terminal, you no longer need to add environment variables through “source ./devel/setup.bash”.


/home/whm/jaka_robot_v2.2/src/jaka_zu3_moveit_config/launch/demo_gazebo.launch http:...
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
whm@whm-virtual-machine:~$ cd jaka_robot_v2.2/
whm@whm-virtual-machine:~/jaka_robot_v2.2$ roslaunch jaka_zu3_moveit_config demo
_gazebo.launch
... logging to /home/whm/.ros/log/3f7d47dc-1fb5-11ee-938d-000c29cf701f/roslaunch
-whm-virtual-machine-72505.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

WARN: unrecognized 'param' tag in <include> tag
WARN: unrecognized 'param' tag in <include> tag
started roslaunch server http://whm-virtual-machine:45957/

SUMMARY
========

PARAMETERS
 * /controller_list: [{'default': True...
 * /gazebo/enable_ros_network: True
 * /generic_hw_control_loop/cycle_time_error_threshold: 0.01
 * /generic_hw_control_loop/loop_hz: 300
 * /hardware_interface/joints: ['joint_1', 'join...
 * /hardware_interface/sim_control_mode: 1
 * /jaka_zu3_controller/action_monitor_rate: 1
```

Note: If this setting is not performed, the following operations need to add environment variables through the "source ./devel/setup.bash" command when you open a new terminal.

2.2.2 Jaka_Ros driver interface

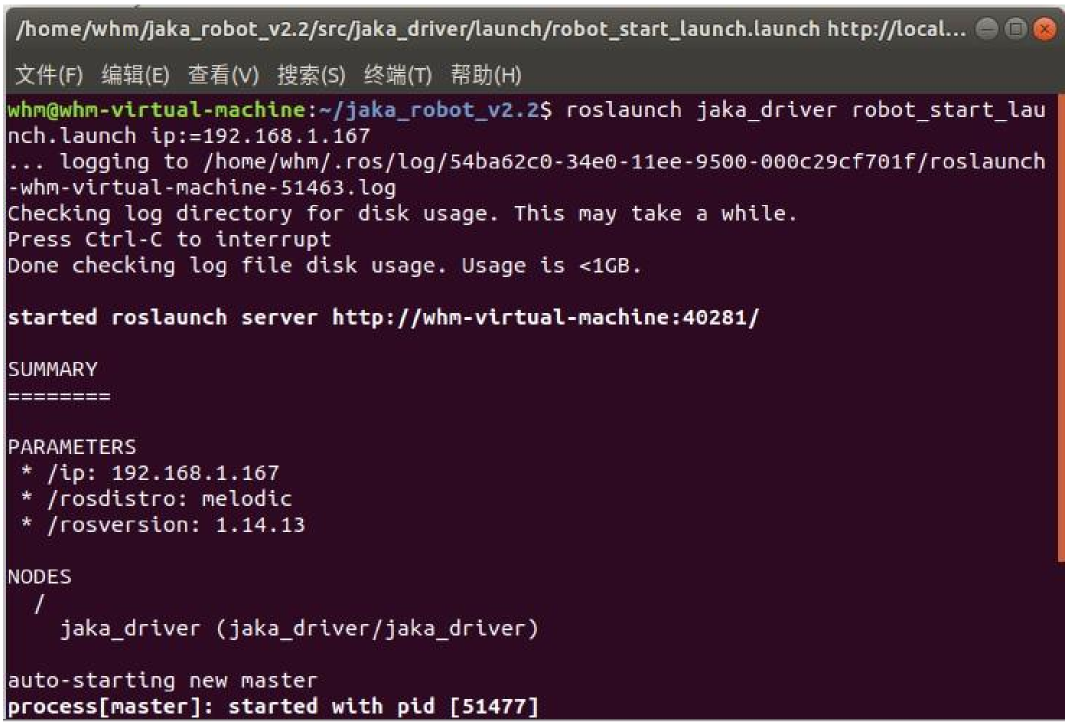
Performed functions: start the Jaka_Ros driver server, and call various functional services with parameters through rosservice call. For details on the communication protocol of the ROS driver interface, see *jaka_driver_interface*.

The operation steps to test the various functional services of the robot are as follows:

1. Launch the robot_start_launch.launch file. Right click in the workspace (jaka_robot_v2.2) and open a terminal to launch the jaka_driver server, passing the parameter to inform the server of the robot IP via the ros command:

```
roslaunch jaka_driver robot_start_launch.launch ip:=192.168.1.167
```

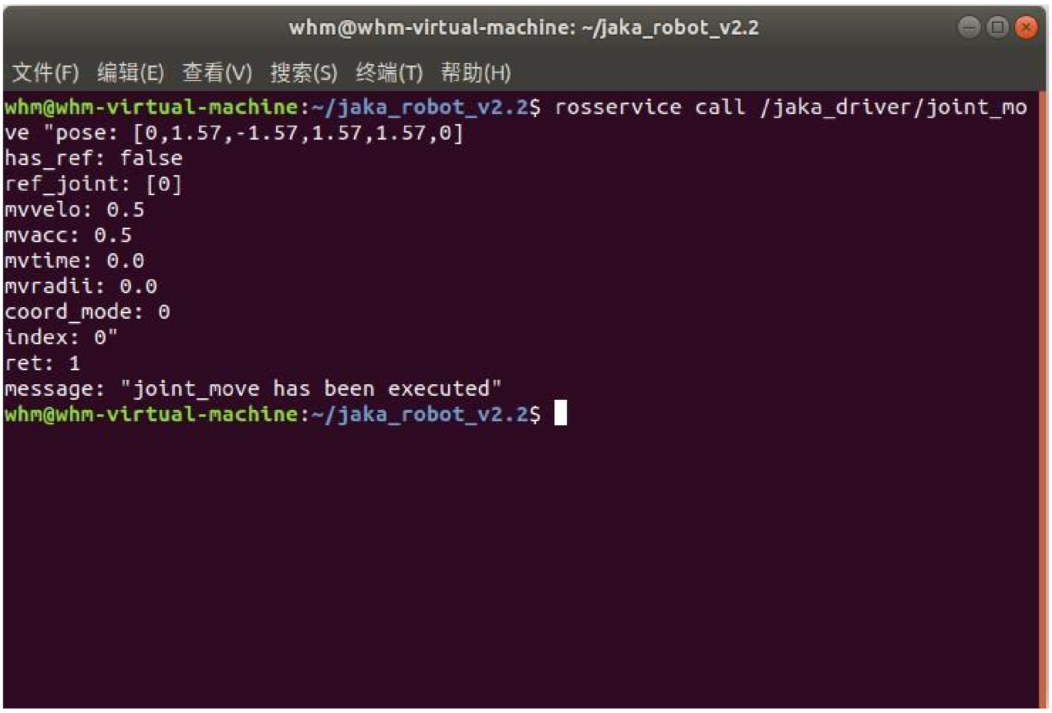
Note: ip:=xxx, xxx is the IP address of the robot, which is a required parameter, otherwise it will report an error if it can't connect to the robot.


/home/whm/jaka_robot_v2.2/src/jaka_driver/launch/robot_start_launch.launch http://local...
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

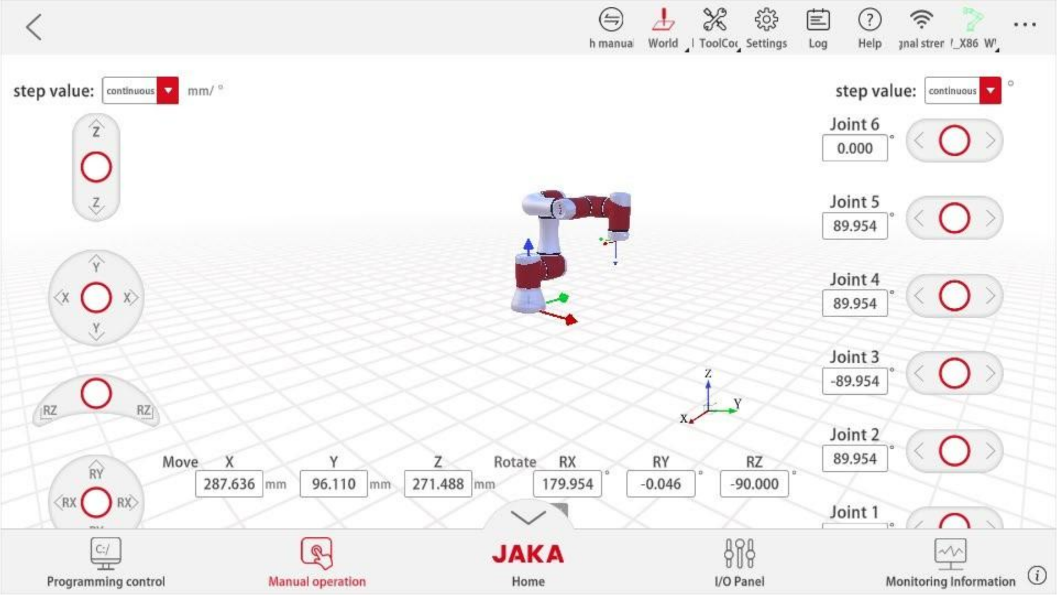
```
whm@whm-virtual-machine:~/jaka_robot_v2.2$ roslaunch jaka_driver robot_start_lau
nch.launch ip:=192.168.1.167
... logging to /home/whm/.ros/log/54ba62c0-34e0-11ee-9500-000c29cf701f/roslaunch
-whm-virtual-machine-51463.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://whm-virtual-machine:40281/

SUMMARY
========

PARAMETERS
 * /ip: 192.168.1.167
 * /rosdistro: melodic
 * /rosversion: 1.14.13

NODES
  /
    jaka_driver (jaka_driver/jaka_driver)

auto-starting new master
process[master]: started with pid [51477]
```

2. **Joint motion**: Use the rosservice call command and input parameters as required to control the robot to do joint motion. **Note:** In the example, the motion interface is set as a blocking interface. If it needs to be changed to a non-blocking interface, it can be realized by modifying the corresponding interface parameters in the jaka_driver.cpp file.

```
rosservice call /jaka_driver/joint_move "pose: [0,1.57,-1.57,1.57,1.57,0]
has_ref: false
ref_joint: [0]
mvvelo: 0.5
mvacc: 0.5
mvtime: 0.0
mvradii: 0.0
coord_mode: 0
index: 0"
```


whm@whm-virtual-machine: ~/jaka_robot_v2.2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
whm@whm-virtual-machine:~/jaka_robot_v2.2$ rosservice call /jaka_driver/joint_mo
ve "pose: [0,1.57,-1.57,1.57,1.57,0]
has_ref: false
ref_joint: [0]
mvvelo: 0.5
mvacc: 0.5
mvtime: 0.0
mvradii: 0.0
coord_mode: 0
index: 0"
ret: 1
message: "joint_move has been executed"
whm@whm-virtual-machine:~/jaka_robot_v2.2$
```



step value: continuous mm/°

step value: continuous °

Joint 6 0.000°

Joint 5 89.954°

Joint 4 89.954°

Joint 3 -89.954°

Joint 2 89.954°

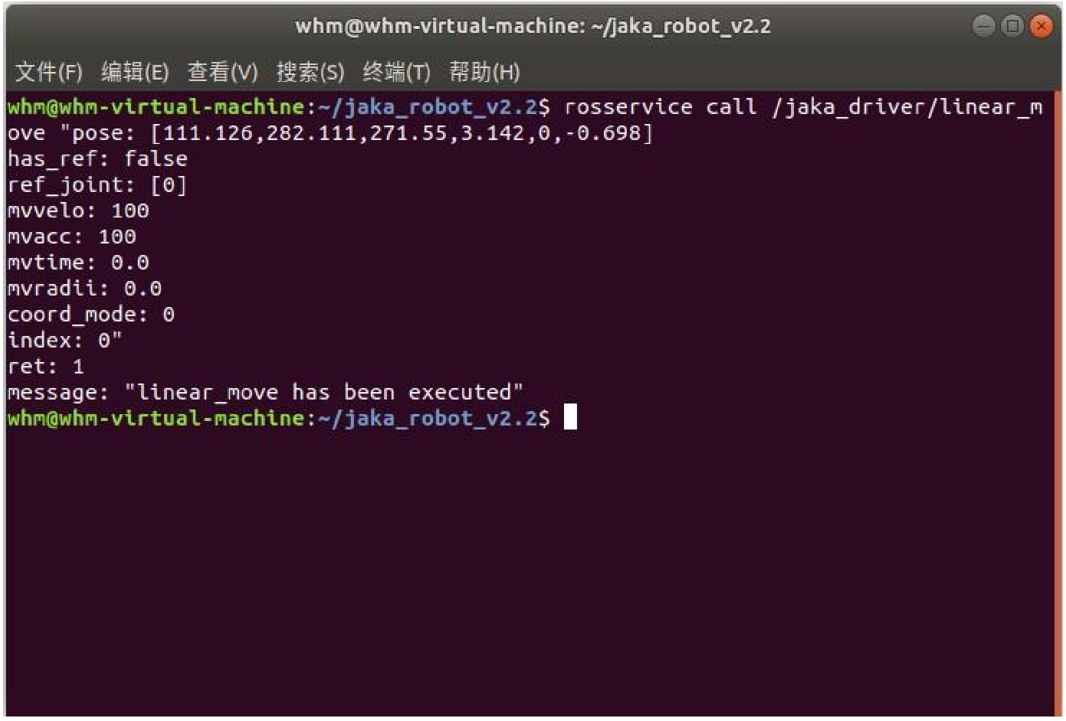
Joint 1

Move X 287.636 mm Y 96.110 mm Z 271.488 mm

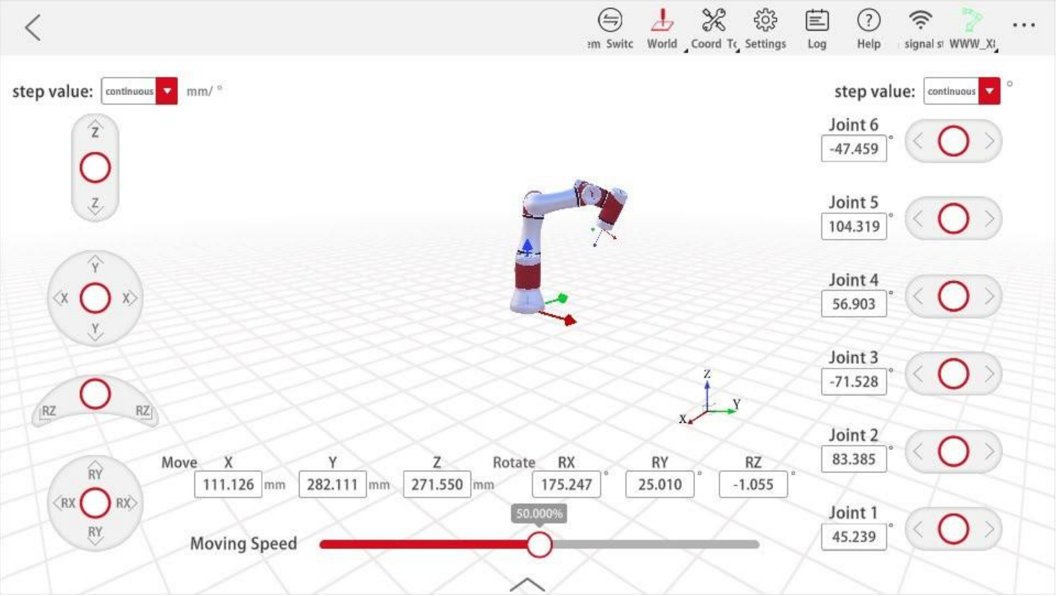
Rotate RX 179.954° RY -0.046° RZ -90.000°

3. **Linear motion**: Use the rosservice call command and input parameters as required to control the robot's motion. **Note:** The pose parameters in the following example are for reference only, and the specific parameters need to be set according to the actual model of the robot(this demo is Zu3), otherwise the execution may fail, and the APP will pop up an error message. The failure reason: (1) exceeded the movement range; (2) encountered a singularity.

```
rosservice call /jaka_driver/linear_move "pose: [111.126,282.111,271.55,3.142,0,-0.698]
has_ref: false
ref_joint: [0]
mvvelo: 100
mvacc: 100
mvtime: 0.0
mvradii: 0.0
coord_mode: 0
index: 0"
```


whm@whm-virtual-machine: ~/jaka_robot_v2.2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
whm@whm-virtual-machine:~/jaka_robot_v2.2$ rosservice call /jaka_driver/linear_m
ove "pose: [111.126,282.111,271.55,3.142,0,-0.698]
has_ref: false
ref_joint: [0]
mvvelo: 100
mvacc: 100
mvtime: 0.0
mvradii: 0.0
coord_mode: 0
index: 0"
ret: 1
message: "linear_move has been executed"
whm@whm-virtual-machine:~/jaka_robot_v2.2$ 
```



step value: continuous

Joint 6 -47.459 °

Joint 5 104.319 °

Joint 4 56.903 °

Joint 3 -71.528 °

Joint 2 83.385 °

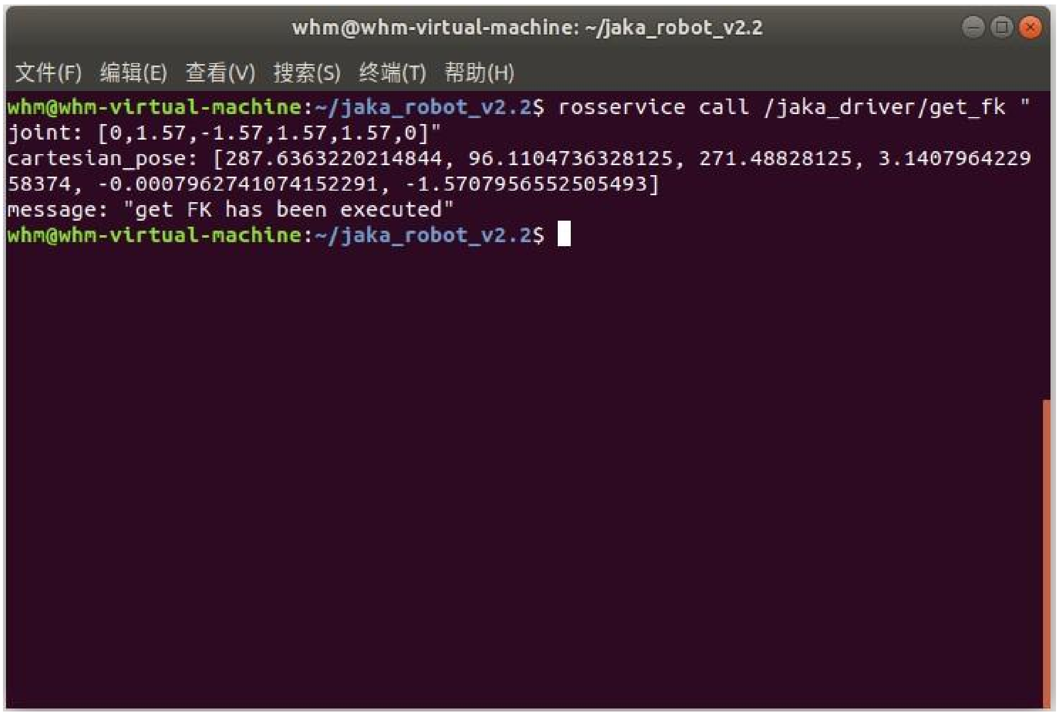
Joint 1 45.239 °

Move X 111.126 mm Y 282.111 mm Z 271.550 mm

Rotate RX 175.247 ° RY 25.010 ° RZ -1.055 °

4. **Robot forward kinematic solution**: Use the rosservice call /jaka_driver/get_fk command and input the parameters as required to find the forward kinematic solution.

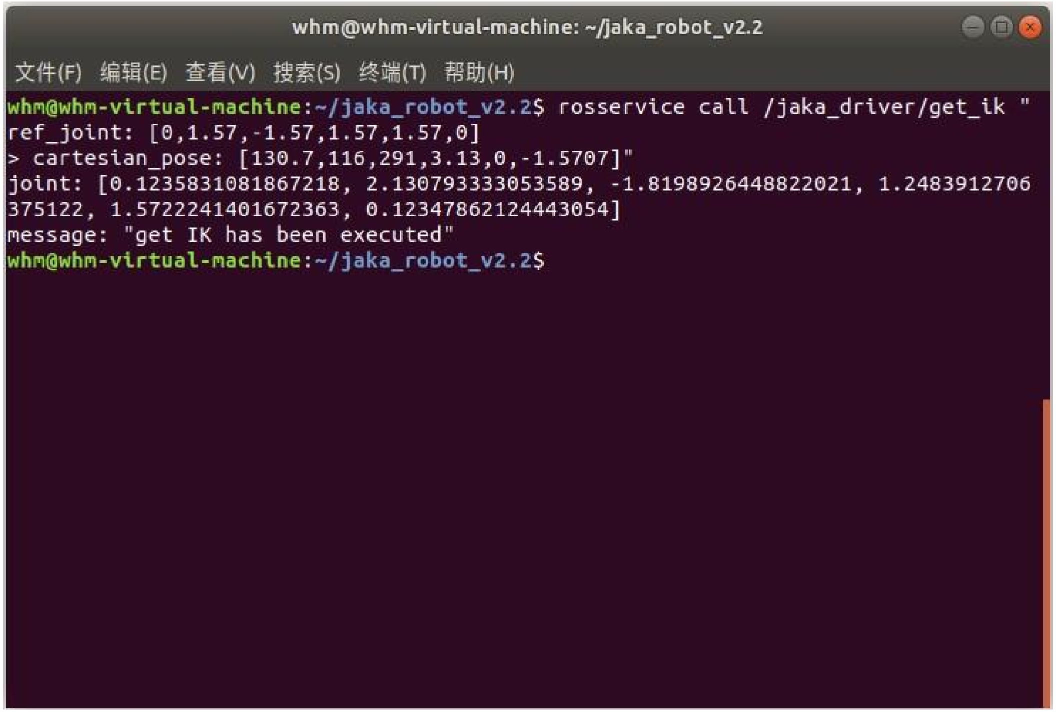
```
rosservice call /jaka_driver/get_fk "joint: [0,1.57,-1.57,1.57,1.57,0]"
```


whm@whm-virtual-machine: ~/jaka_robot_v2.2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
whm@whm-virtual-machine:~/jaka_robot_v2.2$ rosservice call /jaka_driver/get_fk "
joint: [0,1.57,-1.57,1.57,1.57,0]"
cartesian_pose: [287.6363220214844, 96.1104736328125, 271.48828125, 3.1407964229
58374, -0.0007962741074152291, -1.5707956552505493]
message: "get FK has been executed"
whm@whm-virtual-machine:~/jaka_robot_v2.2$
```

5. **Robot inverse kinematics solution:** Use the rosservice call /jaka_driver/get_ik command and input the parameters as required to find the inverse kinematic solution.

```
rosservice call /jaka_driver/get_ik "ref_joint: [0,1.57,-1.57,1.57,1.57,0]
cartesian_pose: [130.7,116,291,3.13,0,-1.5707]"
```


whm@whm-virtual-machine: ~/jaka_robot_v2.2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

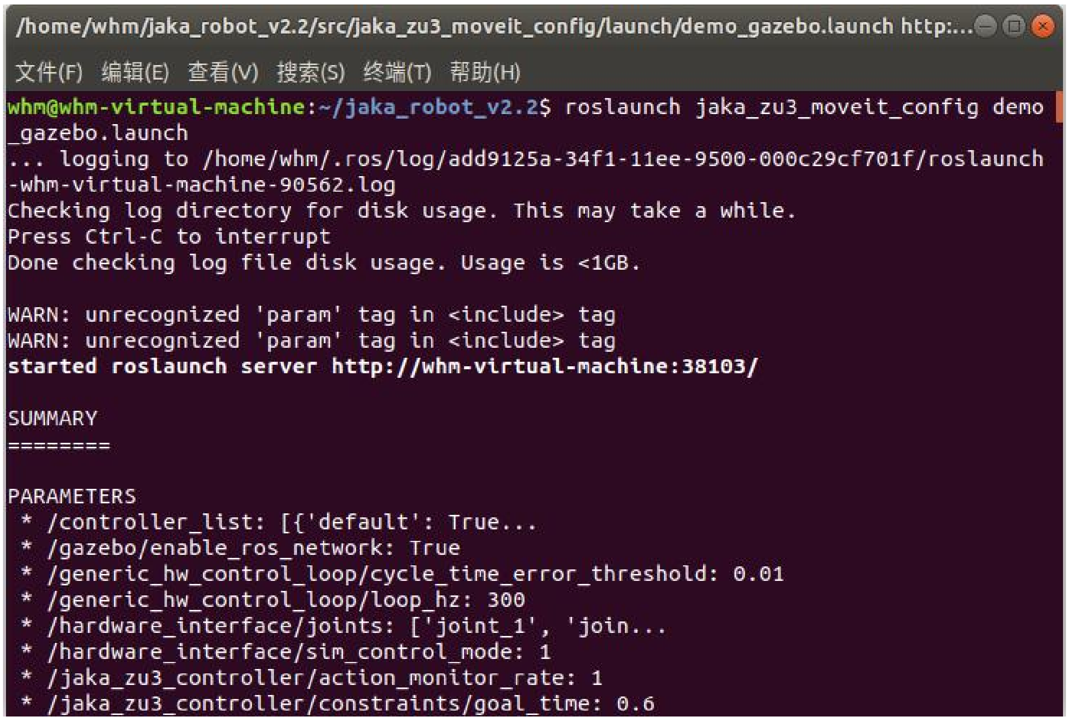
```
whm@whm-virtual-machine:~/jaka_robot_v2.2$ rosservice call /jaka_driver/get_ik "
ref_joint: [0,1.57,-1.57,1.57,1.57,0]
> cartesian_pose: [130.7,116,291,3.13,0,-1.5707]"
joint: [0.1235831081867218, 2.130793333053589, -1.8198926448822021, 1.2483912706
375122, 1.5722241401672363, 0.12347862124443054]
message: "get IK has been executed"
whm@whm-virtual-machine:~/jaka_robot_v2.2$
```

2.2.3 Combined use of Moveit and Gazebo

Performed function: Start Rviz and Gazebo, plan and execute the trajectory path in Rviz, and the robot simulation model in Gazebo will move to the corresponding position. The specific operation steps are:

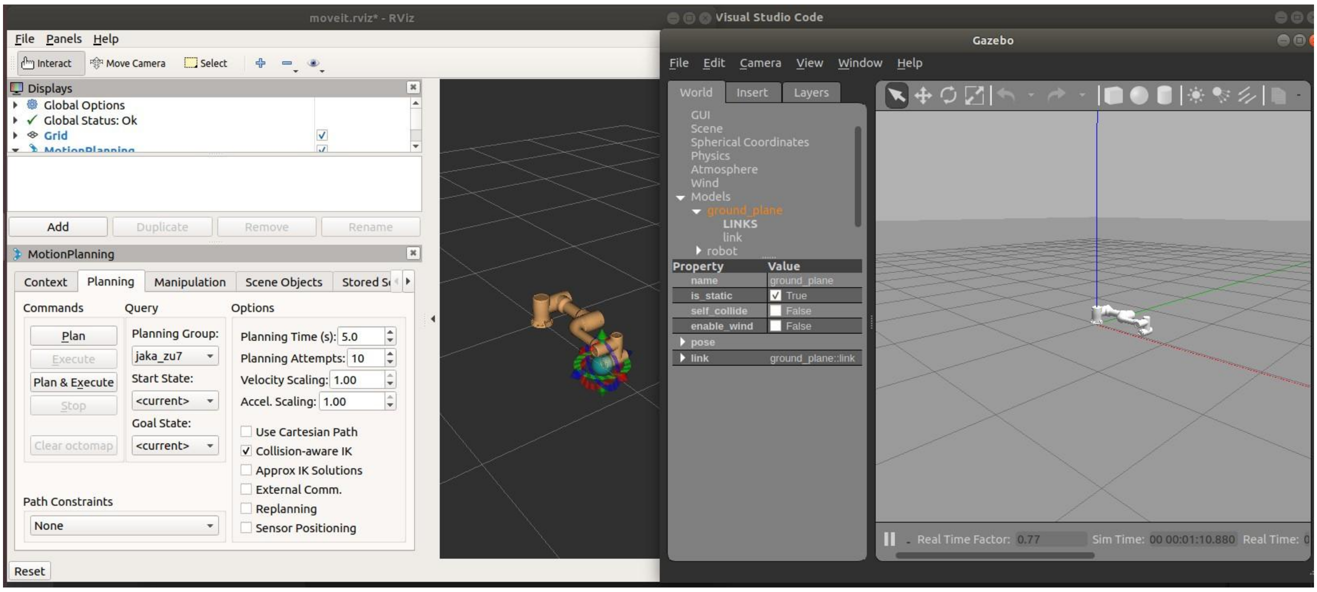
1. Launch the demo_gazebo.launch file of Zu3, and launch Rviz and Gazebo at the same time. If you want to launch other models of robots, modify the input command to the corresponding moveit configuration package (jaka_xx_moveit_config).

```
roslaunch jaka_zu3_moveit_config demo_gazebo.launch
```


/home/whm/jaka_robot_v2.2/src/jaka_zu3_moveit_config/launch/demo_gazebo.launch http:...
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
whm@whm-virtual-machine:~/jaka_robot_v2.2$ roslaunch jaka_zu3_moveit_config demo
_gazebo.launch
... logging to /home/whm/.ros/log/add9125a-34f1-11ee-9500-000c29cf701f/roslaunch
-whm-virtual-machine-90562.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

WARN: unrecognized 'param' tag in <include> tag
WARN: unrecognized 'param' tag in <include> tag
started roslaunch server http://whm-virtual-machine:38103/

SUMMARY
========

PARAMETERS
 * /controller_list: [{'default': True...
 * /gazebo/enable_ros_network: True
 * /generic_hw_control_loop/cycle_time_error_threshold: 0.01
 * /generic_hw_control_loop/loop_hz: 300
 * /hardware_interface/joints: ['joint_1', 'join...
 * /hardware_interface/sim_control_mode: 1
 * /jaka_zu3_controller/action_monitor_rate: 1
 * /jaka_zu3_controller/constraints/goal_time: 0.6
```

Generate the following interface:



Global Status: Ok

Planning Group: jaka_zu7

Start State: <current>

Goal State: <current>

Path Constraints: None

Planning Time (s): 5.0

Planning Attempts: 10

Velocity Scaling: 1.00

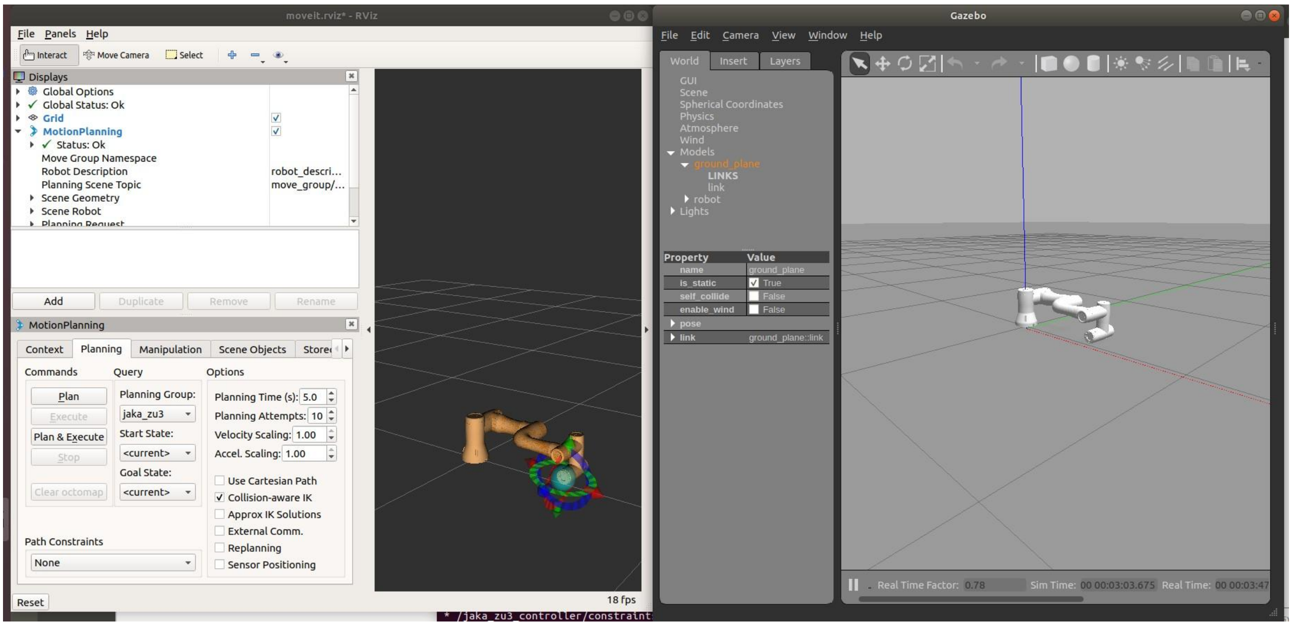
Accel. Scaling: 1.00

☐ Use Cartesian Path
☑ Collision-aware IK
☐ Approx IK Solutions
☐ External Comm.
☐ Replanning
☐ Sensor Positioning

Property	Value
name	ground_plane
is_static	☑ True
self_collide	☐ False
enable_wind	☐ False
pose	
link	ground_plane::link

Real Time Factor: 0.77 Sim Time: 00 00:01:10.880 Real Time: 0

2. Select a target position from the "Goal state" of the RVIZ interface, and click "Plan & Execute", the RVIZ interface will display the trajectory of the robot, and drive the robot simulation model in Gazebo to the set target position.



Global Status: Ok

Status: Ok

Robot Description: robot_descri...

Planning Scene Topic: move_group/...

Planning Group: jaka_zu3

Start State: <current>

Goal State: <current>

Path Constraints: None

Planning Time (s): 5.0

Planning Attempts: 10

Velocity Scaling: 1.00

Accel. Scaling: 1.00

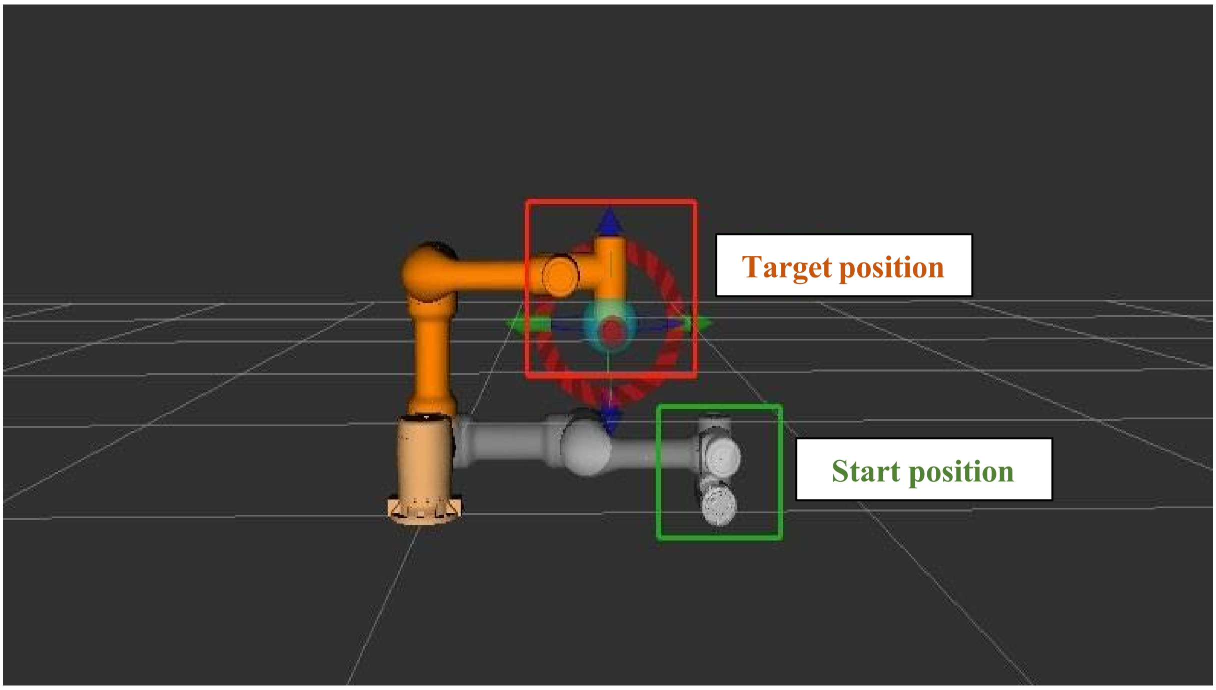
☐ Use Cartesian Path
☑ Collision-aware IK
☐ Approx IK Solutions
☐ External Comm.
☐ Replanning
☐ Sensor Positioning

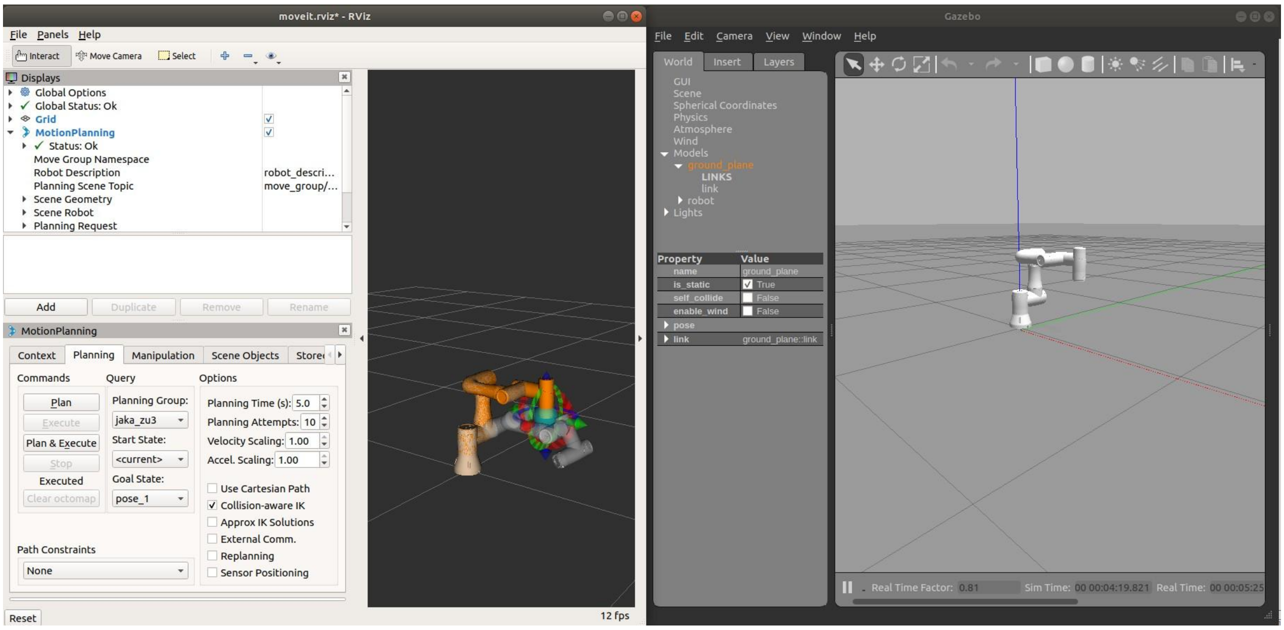
18 fps

* /jaka_zu3_controller/constraint

Property	Value
name	ground_plane
is_static	☑ True
self_collide	☐ False
enable_wind	☐ False
pose	
link	ground_plane::link

Real Time Factor: 0.78 Sim Time: 00 00:03:03.675 Real Time: 00 00:03:47





robot_descri…
move_group/…

Planning Group: jaka_zu3
Start State: <current>
Goal State: pose_1

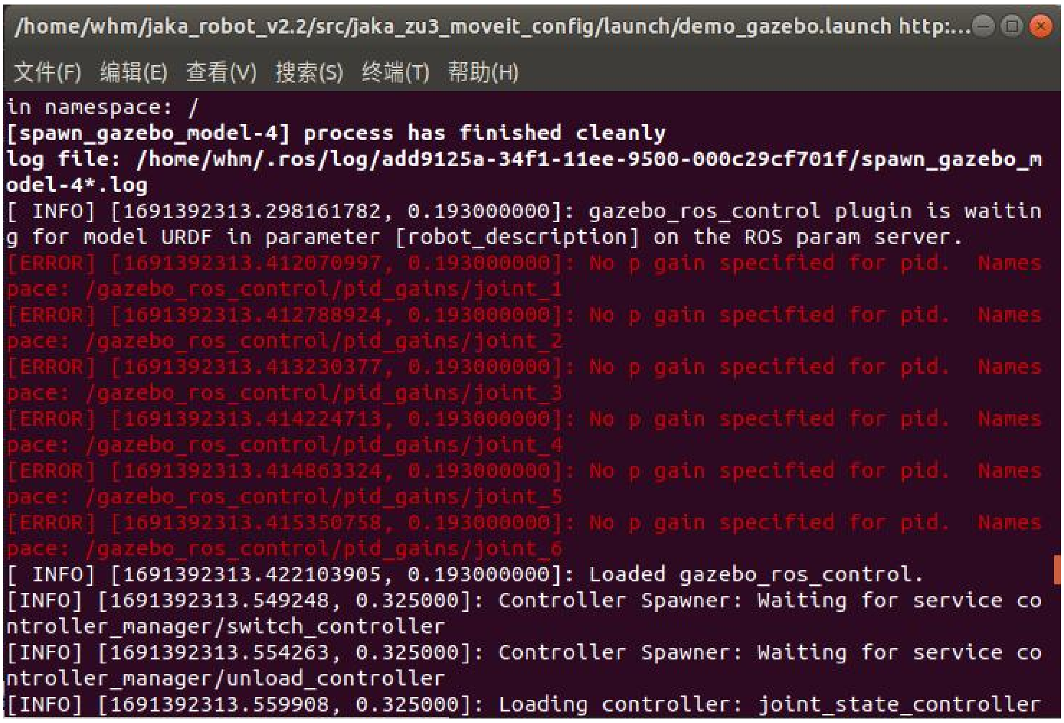
Planning Time (s): 5.0
Planning Attempts: 10
Velocity Scaling: 1.00
Accel. Scaling: 1.00

- □ Use Cartesian Path
- ☑ Collision-aware IK
- □ Approx IK Solutions
- □ External Comm.
- □ Replanning
- □ Sensor Positioning

Path Constraints: None

Property	Value
name	ground_plane
is_static	☑ True
self_collide	□ False
enable_wind	□ False
pose	
link	ground_plane::link

Note: When starting the demo_gazebo.launch file, an error message "No p gain specified for pid." will appear. It is because the controller used in Moveit settings is a position controller. The pid does not need to be set. This error message can be ignored and will not affect the use.


/home/whm/jaka_robot_v2.2/src/jaka_zu3_moveit_config/launch/demo_gazebo.launch http:...
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
in namespace: /
[spawn_gazebo_model-4] process has finished cleanly
log file: /home/whm/.ros/log/add9125a-34f1-11ee-9500-000c29cf701f/spawn_gazebo_m
odel-4*.log
[ INFO] [1691392313.298161782, 0.193000000]: gazebo_ros_control plugin is waitin
g for model URDF in parameter [robot_description] on the ROS param server.
[ERROR] [1691392313.412070997, 0.193000000]: No p gain specified for pid.  Names
pace: /gazebo_ros_control/pid_gains/joint_1
[ERROR] [1691392313.412788924, 0.193000000]: No p gain specified for pid.  Names
pace: /gazebo_ros_control/pid_gains/joint_2
[ERROR] [1691392313.413230377, 0.193000000]: No p gain specified for pid.  Names
pace: /gazebo_ros_control/pid_gains/joint_3
[ERROR] [1691392313.414224713, 0.193000000]: No p gain specified for pid.  Names
pace: /gazebo_ros_control/pid_gains/joint_4
[ERROR] [1691392313.414863324, 0.193000000]: No p gain specified for pid.  Names
pace: /gazebo_ros_control/pid_gains/joint_5
[ERROR] [1691392313.415350758, 0.193000000]: No p gain specified for pid.  Names
pace: /gazebo_ros_control/pid_gains/joint_6
[ INFO] [1691392313.422103905, 0.193000000]: Loaded gazebo_ros_control.
[INFO] [1691392313.549248, 0.325000]: Controller Spawner: Waiting for service co
ntroller_manager/switch_controller
[INFO] [1691392313.554263, 0.325000]: Controller Spawner: Waiting for service co
ntroller_manager/unload_controller
[INFO] [1691392313.559908, 0.325000]: Loading controller: joint_state_controller
```

2.2.4 Combined use of Moveit and real robot

Performed function: start the moveit server and RVIZ, RVIZ plans the trajectory path, and the real robot arm will move to the corresponding position.

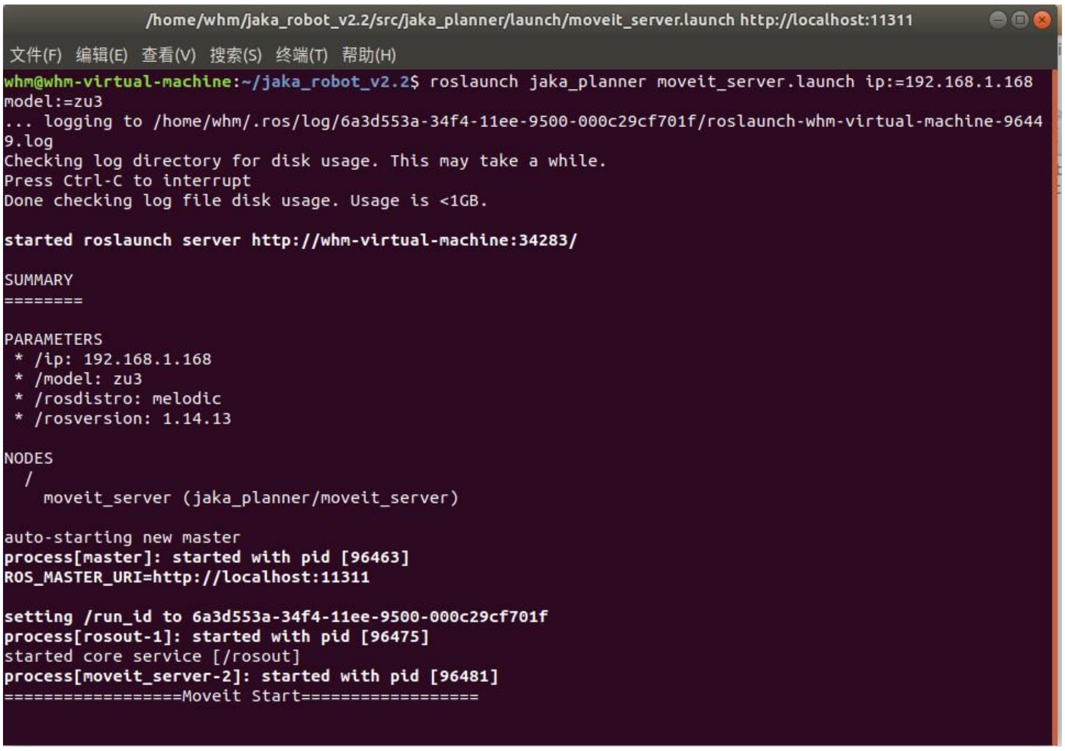
Precondition: The jaka_driver server and the Moveit server cannot be started at the same time, so make sure that the jaka_driver server is turned off before starting the moveit server (that is, robot_start_launch.launch is not started).

The specific operation steps are:

1. Start the moveit_server.launch file (Moveit server). When starting the moveit server, pass the parameters through the ros command to inform the server of the robot model and IP. Open a new terminal and enter:

```
roslaunch jaka_planner moveit_server.launch ip:=192.168.1.168 model:=zu3
```

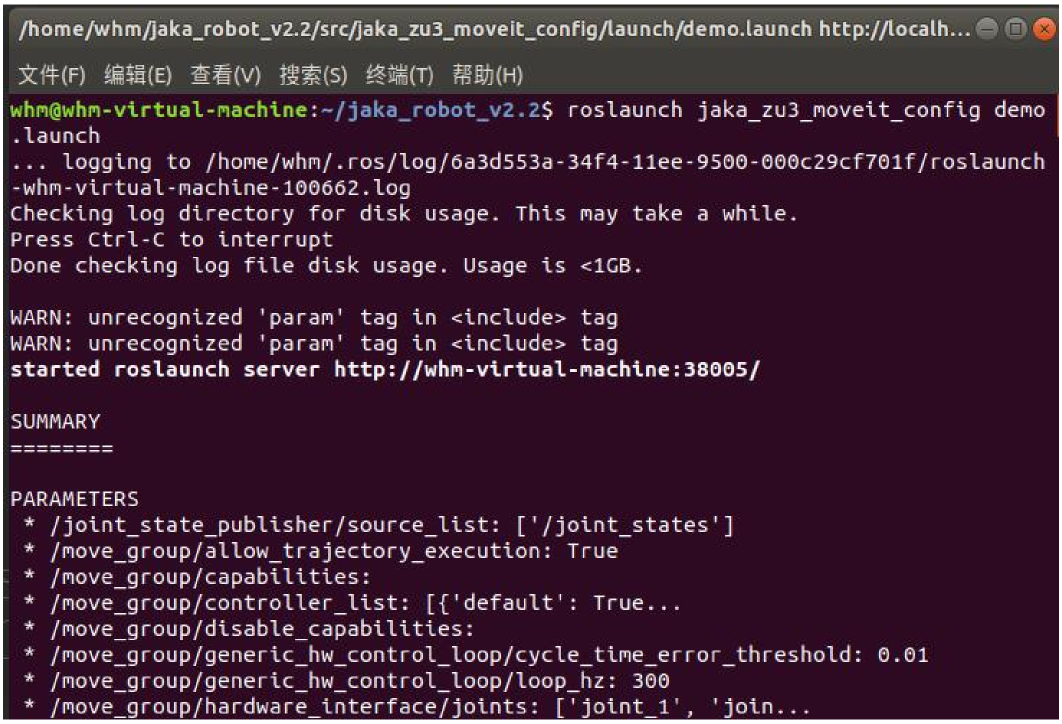
Note: In ip:=xxx, xxx represents the IP address of the robot. In model:=xxx, xxx represents the model of each robot, all in lowercase, for example, zu3, minicobo, or pro16.


/home/whm/jaka_robot_v2.2/src/jaka_planner/launch/moveit_server.launch http://localhost:11311
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
whm@whm-virtual-machine:~/jaka_robot_v2.2$ roslaunch jaka_planner moveit_server.launch ip:=192.168.1.168
model:=zu3
... logging to /home/whm/.ros/log/6a3d553a-34f4-11ee-9500-000c29cf701f/roslaunch-whm-virtual-machine-9644
9.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://whm-virtual-machine:34283/

SUMMARY
========

PARAMETERS
 * /ip: 192.168.1.168
 * /model: zu3
 * /rosdistro: melodic
 * /rosversion: 1.14.13

NODES
  /
    moveit_server (jaka_planner/moveit_server)

auto-starting new master
process[master]: started with pid [96463]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 6a3d553a-34f4-11ee-9500-000c29cf701f
process[rosout-1]: started with pid [96475]
started core service [/rosout]
process[moveit_server-2]: started with pid [96481]
==================Moveit Start==================
```

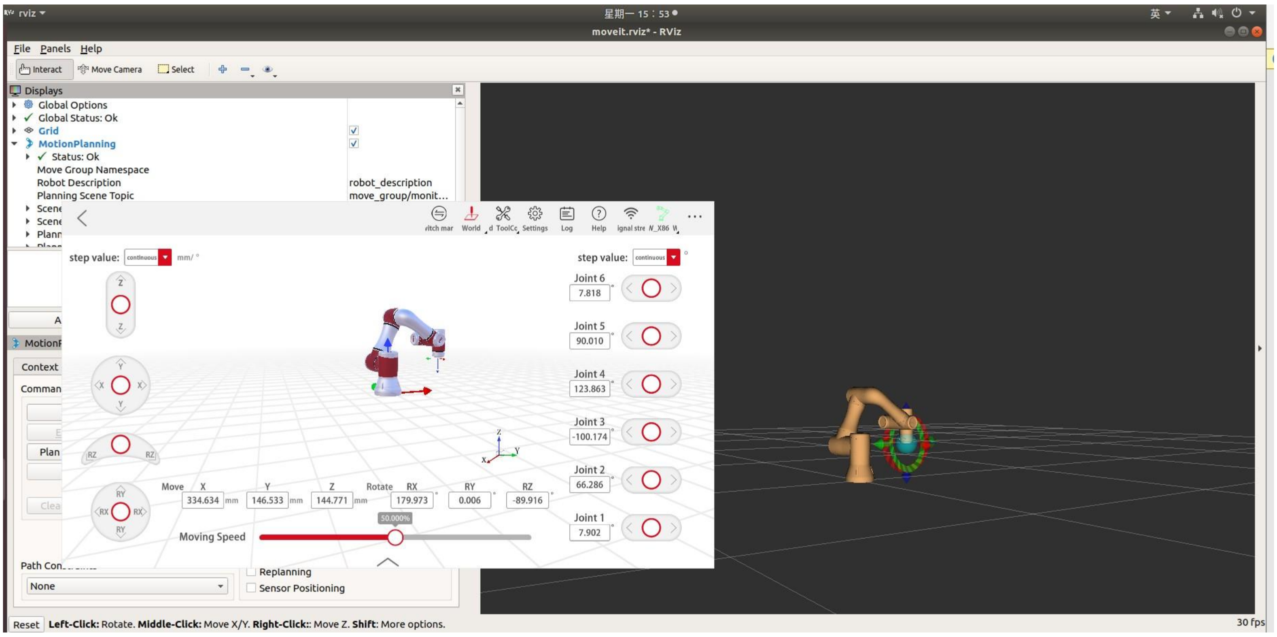
2. Open another terminal and start the demo.launch file (Moveit client).

```
roslaunch jaka_zu3_moveit_config demo.launch
```


/home/whm/jaka_robot_v2.2/src/jaka_zu3_moveit_config/launch/demo.launch http://localh...
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
whm@whm-virtual-machine:~/jaka_robot_v2.2$ roslaunch jaka_zu3_moveit_config demo
.launch
... logging to /home/whm/.ros/log/6a3d553a-34f4-11ee-9500-000c29cf701f/roslaunch
-whm-virtual-machine-100662.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

WARN: unrecognized 'param' tag in <include> tag
WARN: unrecognized 'param' tag in <include> tag
started roslaunch server http://whm-virtual-machine:38005/

SUMMARY
========

PARAMETERS
 * /joint_state_publisher/source_list: ['/joint_states']
 * /move_group/allow_trajectory_execution: True
 * /move_group/capabilities:
 * /move_group/controller_list: [{'default': True...
 * /move_group/disable_capabilities:
 * /move_group/generic_hw_control_loop/cycle_time_error_threshold: 0.01
 * /move_group/generic_hw_control_loop/loop_hz: 300
 * /move_group/hardware_interface/joints: ['joint_1', 'join...
```

Generate the following interface: the orientation displayed by Rviz is consistent with the current orientation of the physical robot.



Robot Description: robot_description

Planning Scene Topic: move_group/monit...

step value: continuous mm/°

step value: continuous °

Joint 6: 7.818

Joint 5: 90.010

Joint 4: 123.863

Joint 3: -100.174

Joint 2: 66.286

Joint 1: 7.902

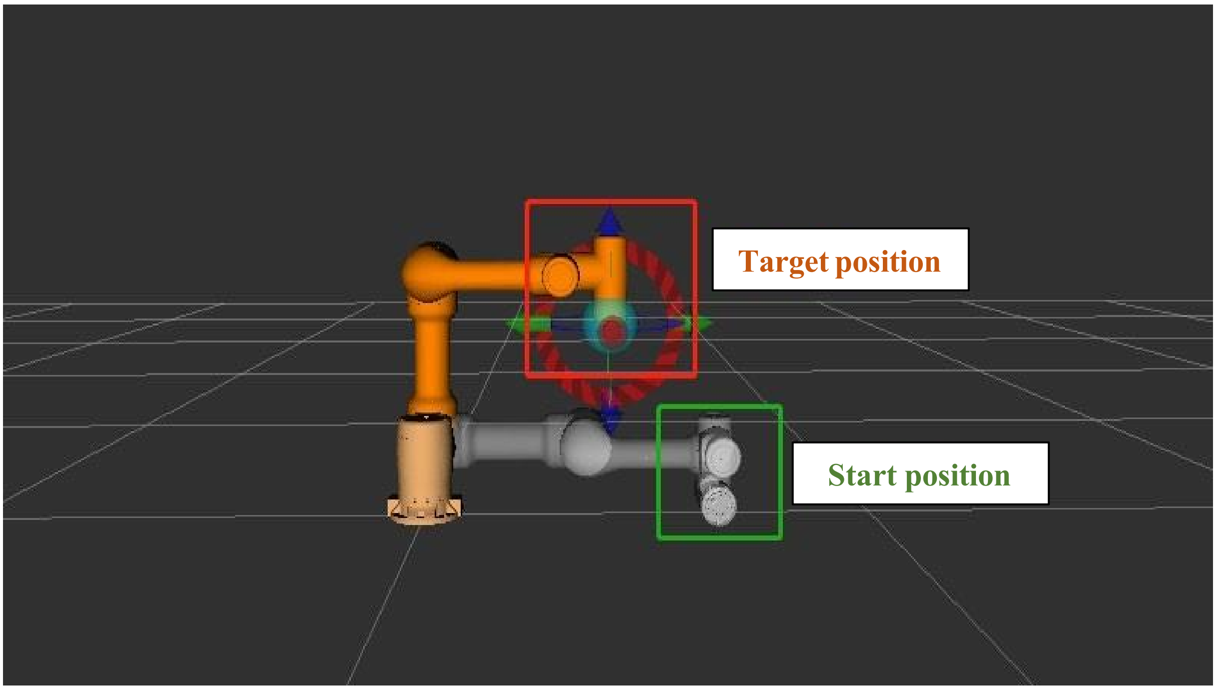
Move X: 334.634 mm; Y: 146.533 mm; Z: 144.771 mm

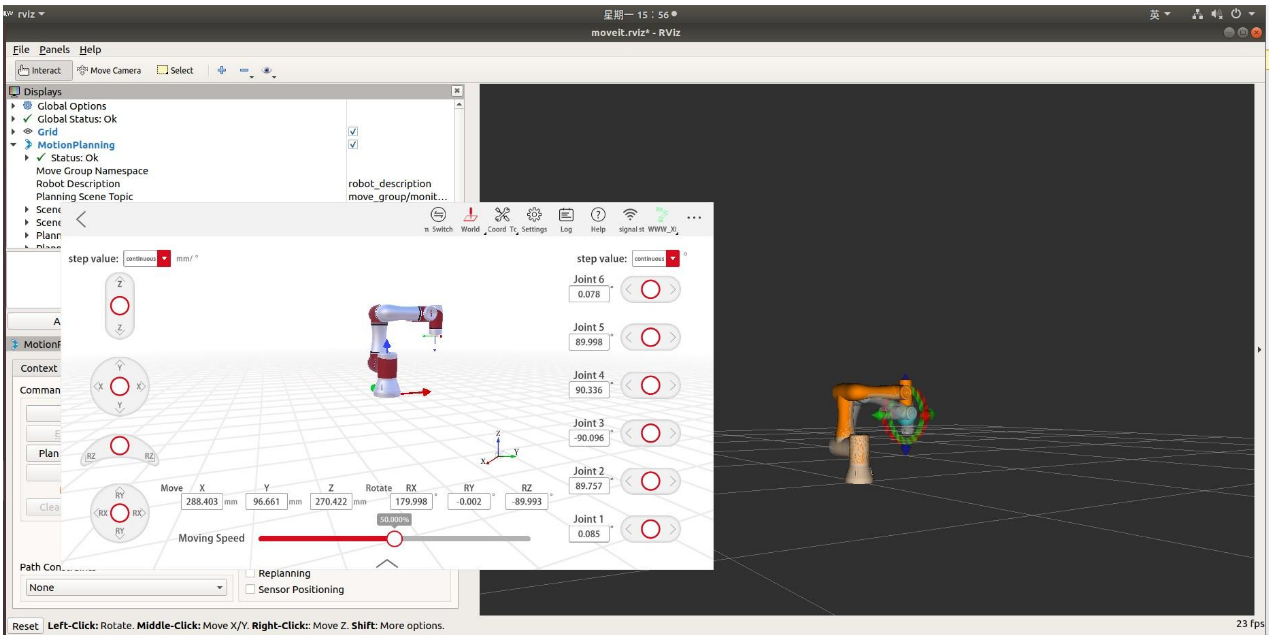
Rotate RX: 179.973; RY: 0.006; RZ: -89.916

Moving Speed: 50.000%

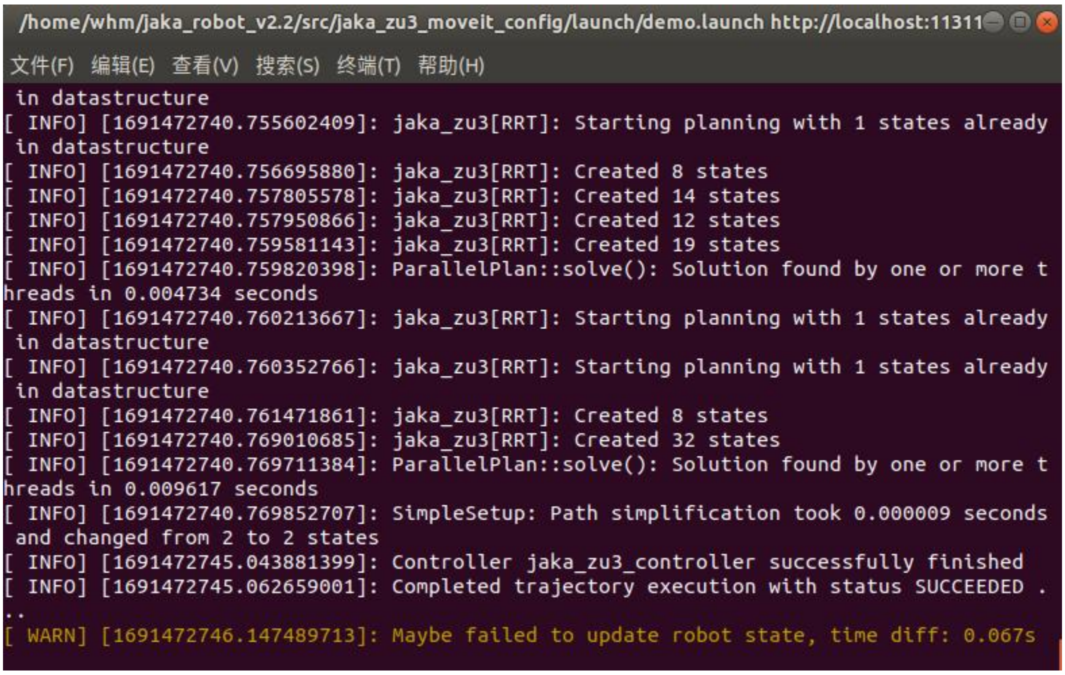
Path Con...: None

3. Select a target position from the "Goal state" of the RVIZ interface, and click "Plan & Execute", the RVIZ interface will display the trajectory of the robot, and drive the physical robot to the set target position.





Note: Click "Plan & Execute", the following warnings may appear after the robot motion stops. This is caused by the asynchronous update time of the ros controller and joint_states, which will not affect the robot motion and can be ignored.


/home/whm/jaka_robot_v2.2/src/jaka_zu3_moveit_config/launch/demo.launch http://localhost:11311
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
 in datastructure
[ INFO] [1691472740.755602409]: jaka_zu3[RRT]: Starting planning with 1 states already
 in datastructure
[ INFO] [1691472740.756695880]: jaka_zu3[RRT]: Created 8 states
[ INFO] [1691472740.757805578]: jaka_zu3[RRT]: Created 14 states
[ INFO] [1691472740.757950866]: jaka_zu3[RRT]: Created 12 states
[ INFO] [1691472740.759581143]: jaka_zu3[RRT]: Created 19 states
[ INFO] [1691472740.759820398]: ParallelPlan::solve(): Solution found by one or more t
hreads in 0.004734 seconds
[ INFO] [1691472740.760213667]: jaka_zu3[RRT]: Starting planning with 1 states already
 in datastructure
[ INFO] [1691472740.760352766]: jaka_zu3[RRT]: Starting planning with 1 states already
 in datastructure
[ INFO] [1691472740.761471861]: jaka_zu3[RRT]: Created 8 states
[ INFO] [1691472740.769010685]: jaka_zu3[RRT]: Created 32 states
[ INFO] [1691472740.769711384]: ParallelPlan::solve(): Solution found by one or more t
hreads in 0.009617 seconds
[ INFO] [1691472740.769852707]: SimpleSetup: Path simplification took 0.000009 seconds
 and changed from 2 to 2 states
[ INFO] [1691472745.043881399]: Controller jaka_zu3_controller successfully finished
[ INFO] [1691472745.062659001]: Completed trajectory execution with status SUCCEEDED .
..
[ WARN] [1691472746.147489713]: Maybe failed to update robot state, time diff: 0.067s
```

4. Code control: Please refer to the following document:

/jaka_robot_v2.2/src/jaka_planner/src/moveit_test.cpp

Note: (1) The sample code model is Zu3. If the robot is another model, you need to change the code and confirm whether the location in the code is reachable and in a safe area.

(2) Startup method: Enter the following commands in the three terminals respectively

```
roslaunch jaka_planner moveit_server.launch ip:=192.168.1.168 model:=zu3
roslaunch jaka_zu3_moveit_config demo.launch
rosrun jaka_planner moveit_test
```

Chapter 3 JAKA ROS Application Demo

The download address of demo and instruction document:

https://github.com/JakaCobot/jaka_robot