1、Robotic arm control

Website：http://docs.ros.org/en/melodic/api/moveit_tutorials/html/index.html

HD camera MoveIt：~/software/transbot_library/src/transbot_config_camera

Astra_MoveIt：~/software/transbot_library/src/transbot_config_astra

Robotic arm control function package：~/software/transbot_library/src/transbot_description

Before usingg this function, we need to close the APP remote control process and all the functions that have been turned on. MoveIT recommends running in a virtual machine.

(The computer must have a discrete GPU!!!)

1.1、Start up

This lesson is mainly to realize the synchronous operation of the real machine and MoveIT simulation. Take jetson nano and monocular camera configuration as an example, the Astra configuration is similar. It is recommended to set the virtual machine as the host. MoveIT will start slowly. After starting in the virtual machine, set jetson nano as the slave.

Virtual machine side (Host)

```
roslaunch transbot_config_camera demo.launch  # HD camera
roslaunch transbot_config_astra demo.launch   # astra
```

jetson nano side（Slave）

```
rosrun transbot_description 03_machine_move.py
```

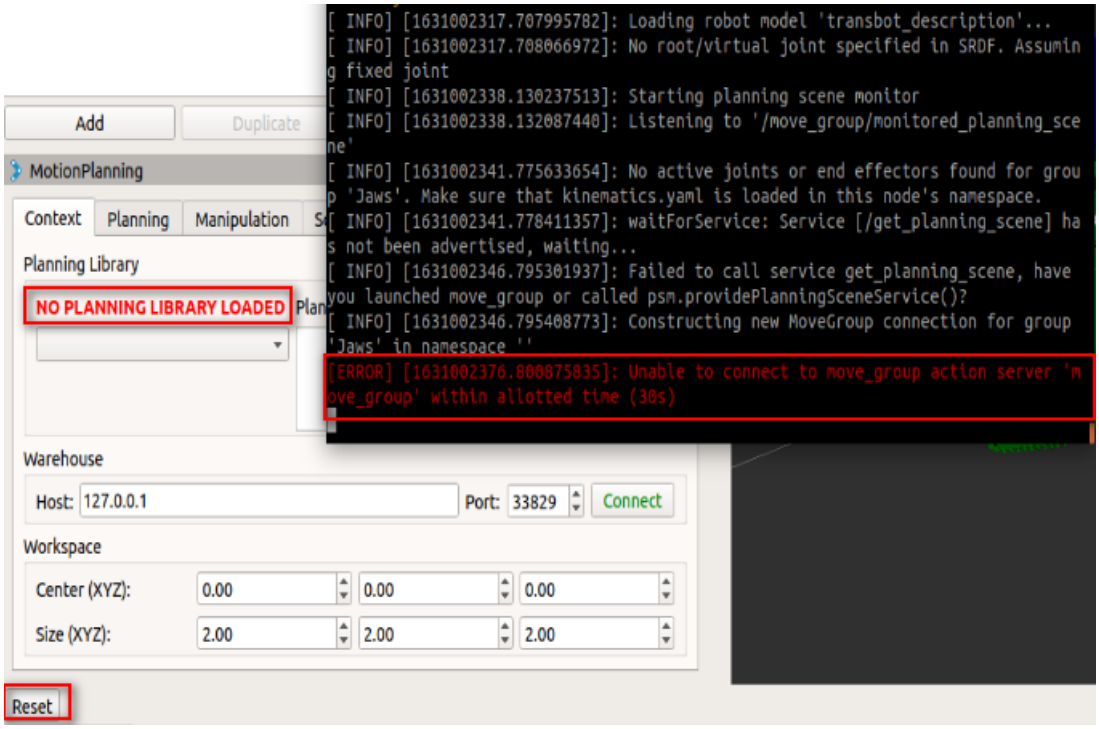
The MoveIt simulation environment start up need some time, wait patiently, observe the terminal.

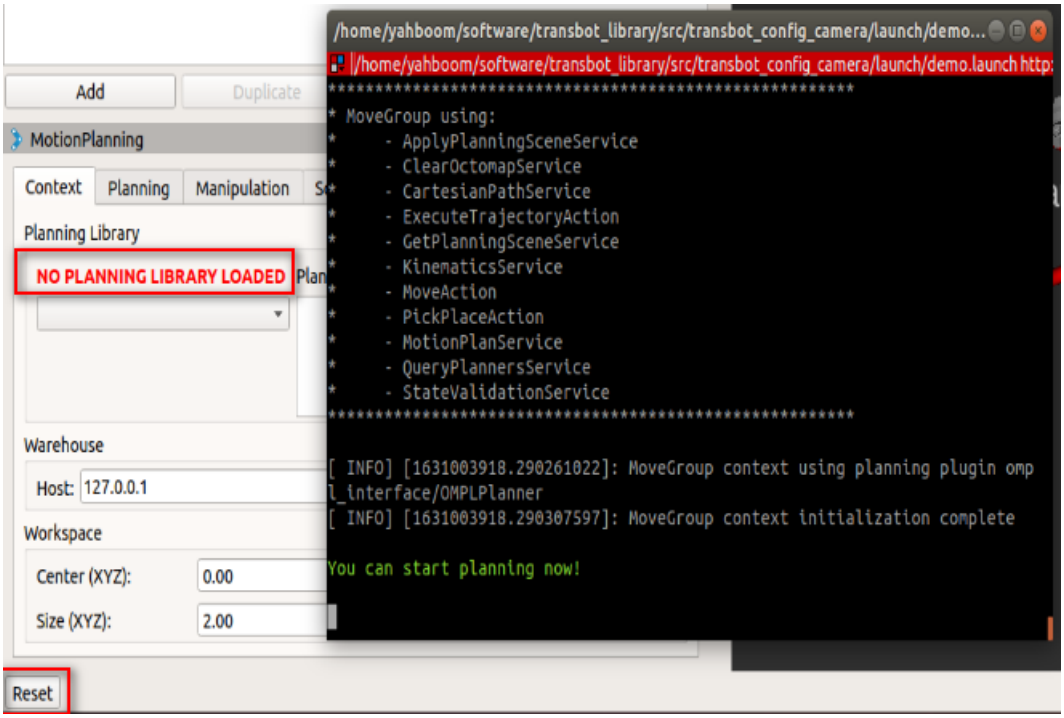
If an error as shown in the figure below appears.

Solution: If the terminal reports an error and the motion planning library is not loaded, click [Reset] in the lower left corner to reload.

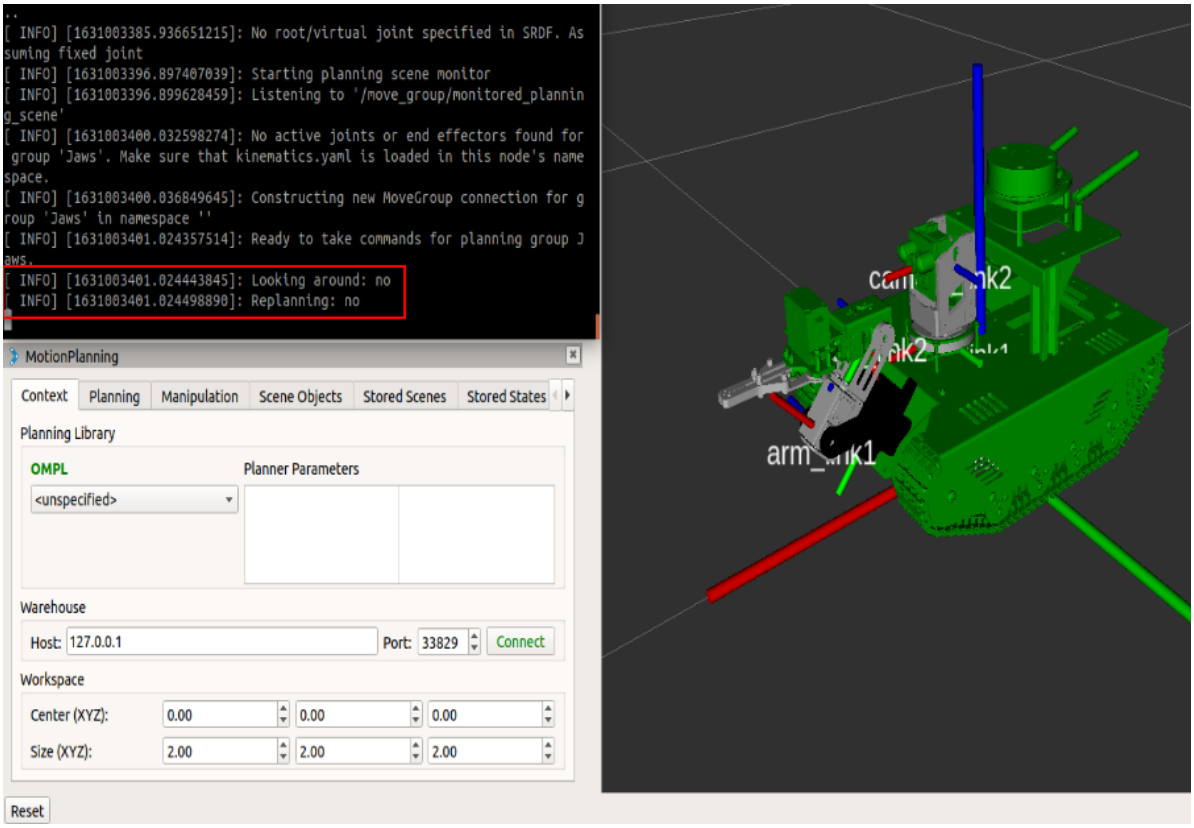
During the loading process, don't rush to click [Reset].

If you click before loading, the system will reload, which will cause it to fail to start.

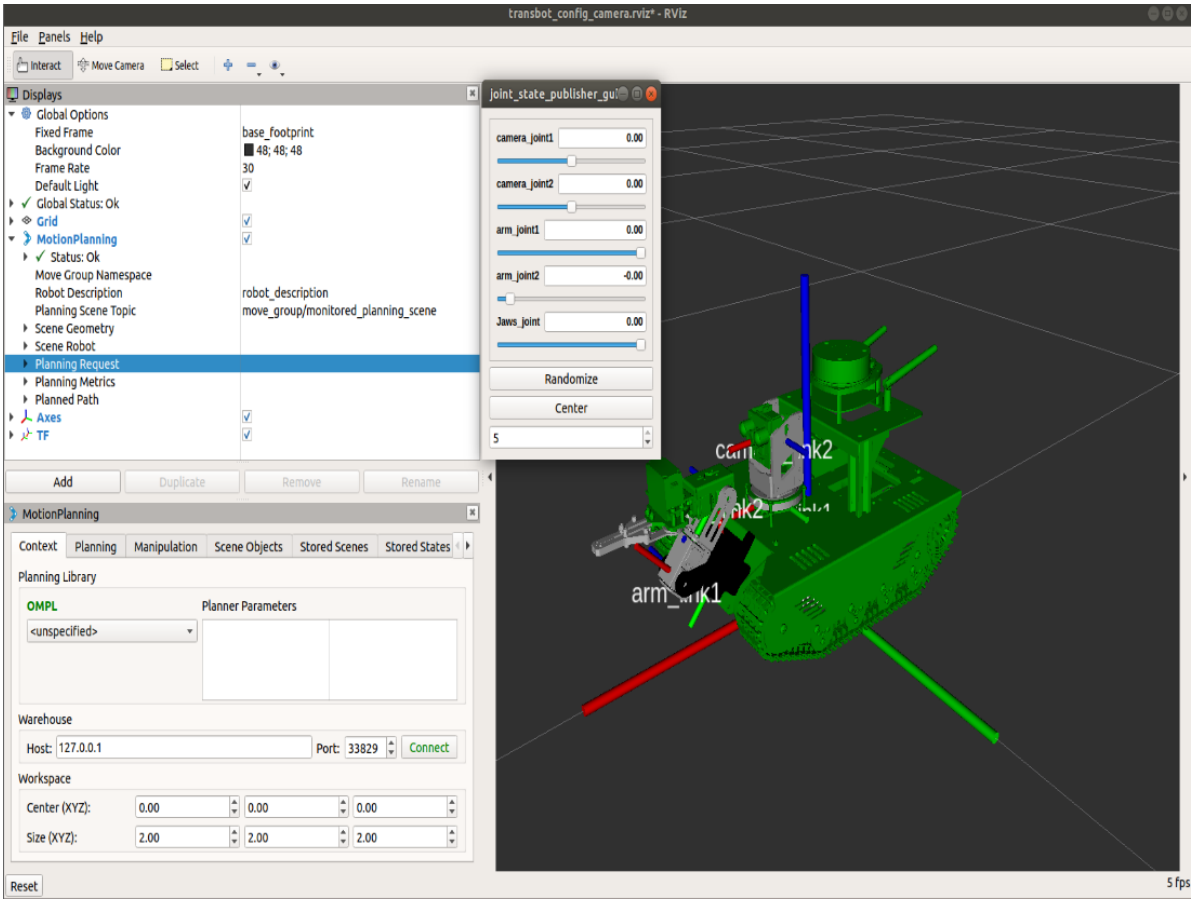




As shown in the figure below, if [Replanning: yes] appears on the terminal and a green [OMPL] appears in Planning Library, which means system be started up successfully.



As shown below.



After the startup is successful, as shown in the figure above, slide the sliding bar of [joint_state_publisher_gui], and the robot arm and camera servo pan/tilt will make corresponding actions.

During the operation, do not slide too fast to avoid accidents.

【camera_joint1】: The camera servo pan/tilt rotates left and right, the left is positive and the right is negative.

【camera_joint2】: The camera servo pan/tilt rotates up and down, up and down.

【arm_joint1】: The servo close to the car body, the smaller the data, it is lower.

【arm_joint2】: The second servo of the robotic arm, the smaller the data, it is lower.

【Jaws_joint】: The gripper, the smaller the data, the gripper will open bigger angle.

【Randomize】: Randomly generate actions. Without safety protection measures, clicks are not allowed, which is very dangerous! ! !

【Center】: One-key middle. When it is first opened, it is in the centering position by default. Click at this time, and no response can be seen.

1.2、Source code analysis

demo.launch

```
  <arg name="use_gui" default="true" />
  <arg name="use_rviz" default="true" />
  <!-- If needed, broadcast static tf for robot root -->
  <!-- We do not have a robot connected, so publish fake joint states -->
  <node name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher" unless="$(arg use_gui)">
    <rosparam param="source_list">[move_group/fake_controller_joint_states]
</rosparam>
  </node>
  <node name="joint_state_publisher" pkg="joint_state_publisher_gui"
type="joint_state_publisher_gui" if="$(arg use_gui)">
    <rosparam param="source_list">[move_group/fake_controller_joint_states]
</rosparam>
  </node>
  <!-- Given the published joint states, publish tf for the robot links -->
  <node name="robot_state_publisher" pkg="robot_state_publisher"
type="robot_state_publisher" respawn="true" output="screen" />
... ...
  <!-- Run Rviz and load the default config to see the state of the move_group
node -->
  <include file="$(find transbot_config_camera)/launch/moveit_rviz.launch"
if="$(arg use_rviz)">
    <arg name="rviz_config" value="$(find
transbot_config_camera)/rviz/transbot_config_camera.rviz"/>
    <arg name="debug" value="$(arg debug)"/>
  </include>
```

At startup, the GUI controller and rviz are set up.

py file: 03_machine_move.py

Subscribe to the topic [/joint_states] published by MoveIt to get the current position of the robot in MoveIt.

```
subscriber = rospy.Subscriber("/joint_states", JointState, JointTopic)
```

Callback【JointTopic】

```
def JointTopic(msg):
    # If it is not the data of the topic, return directly
    if not isinstance(msg, JointState): return
    if len(msg.position) == 10:
        joints1 = (msg.position[0] * RA2DE) + 90
        joints2 = -(msg.position[1] * RA2DE) + 90
        joints3 = (msg.position[2] * RA2DE) + 225
        joints4 = (msg.position[3] * RA2DE) + 45
        joints5 = (msg.position[4] * RA2DE) * 5 / 3 + 180
        bot.set_pwm_servo(1, int(joints1))
        if int(joints2) > 140: joints2 = 140
        bot.set_pwm_servo(2, int(joints2))
        bot.set_uart_servo_angle_array(joints3, joints4, joints5)
    elif len(msg.position) == 9:
        joints1 = (msg.position[0] * RA2DE) + 90
        joints2 = (msg.position[1] * RA2DE) + 225
        joints3 = (msg.position[2] * RA2DE) + 45
        joints4 = (msg.position[3] * RA2DE) * 5 / 3 + 180
        bot.set_pwm_servo(1, int(joints1))
        bot.set_uart_servo_angle_array(joints2, joints3, joints4)
```

Take a HD camera as an example, print [msg.name] and [msg.position] messages as follows:

```
msg.name:  ['camera_joint1', 'camera_joint2', 'arm_joint1', 'arm_joint2',
'Jaws_joint',
            'right_joint2', 'right_joint3', 'left_joint1', 'left_joint2',
'left_joint3']
msg.position:  (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
```

The content of [msg.name] corresponds to [joint_state_publisher_gui], ['camera_joint1','camera_joint2','arm_joint1','arm_joint2','Jaws_joint'] is what we need to pay attention to.