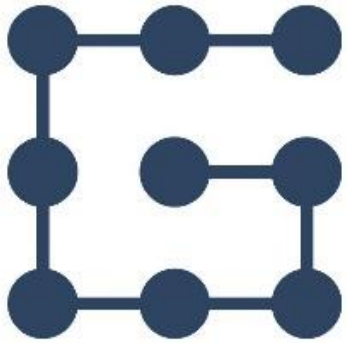


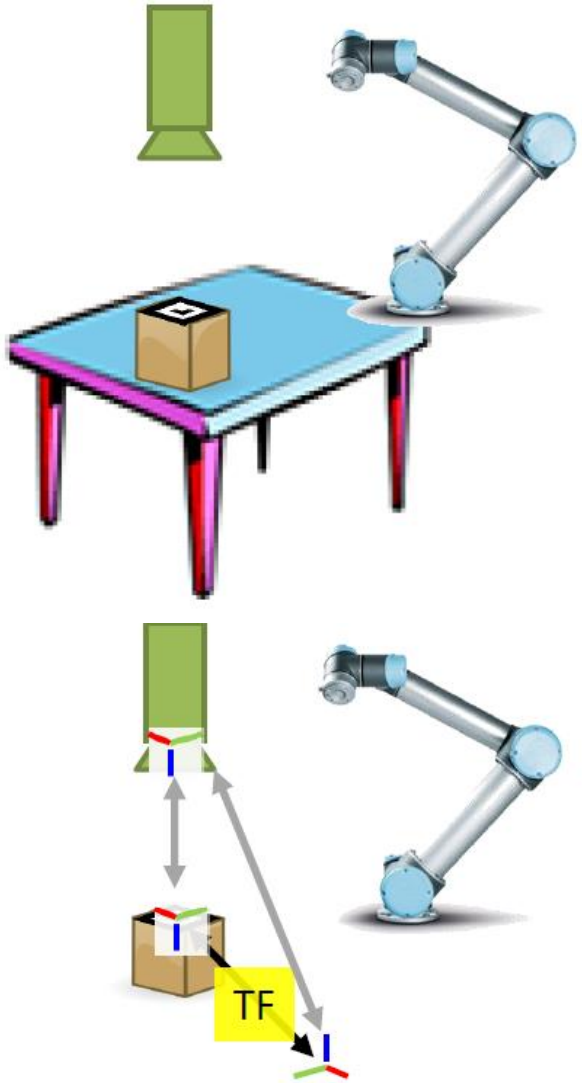
ROS机械臂开发

—— 8.综合应用开发

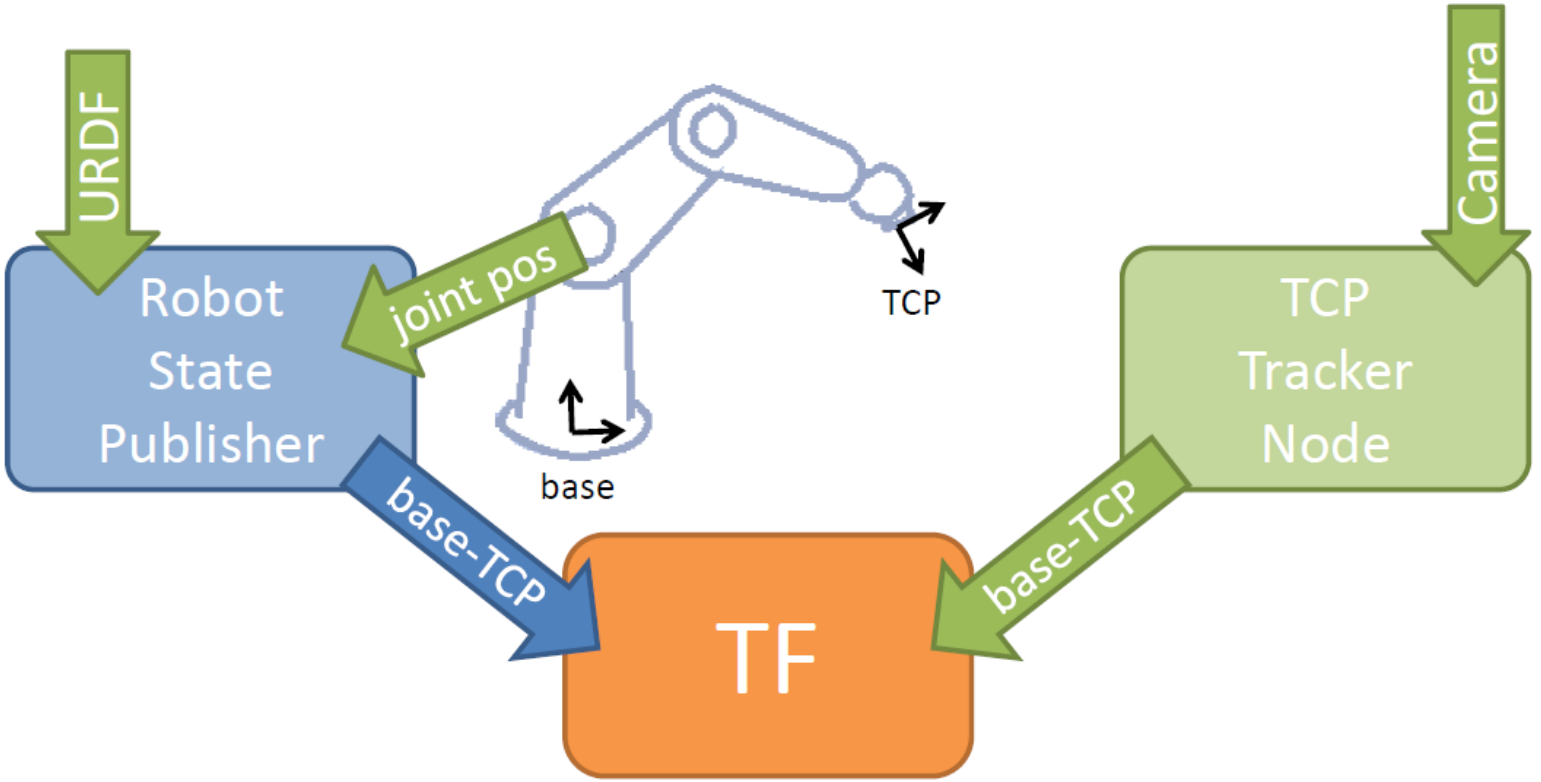
目录

目录

1.机器视觉与机械臂应用

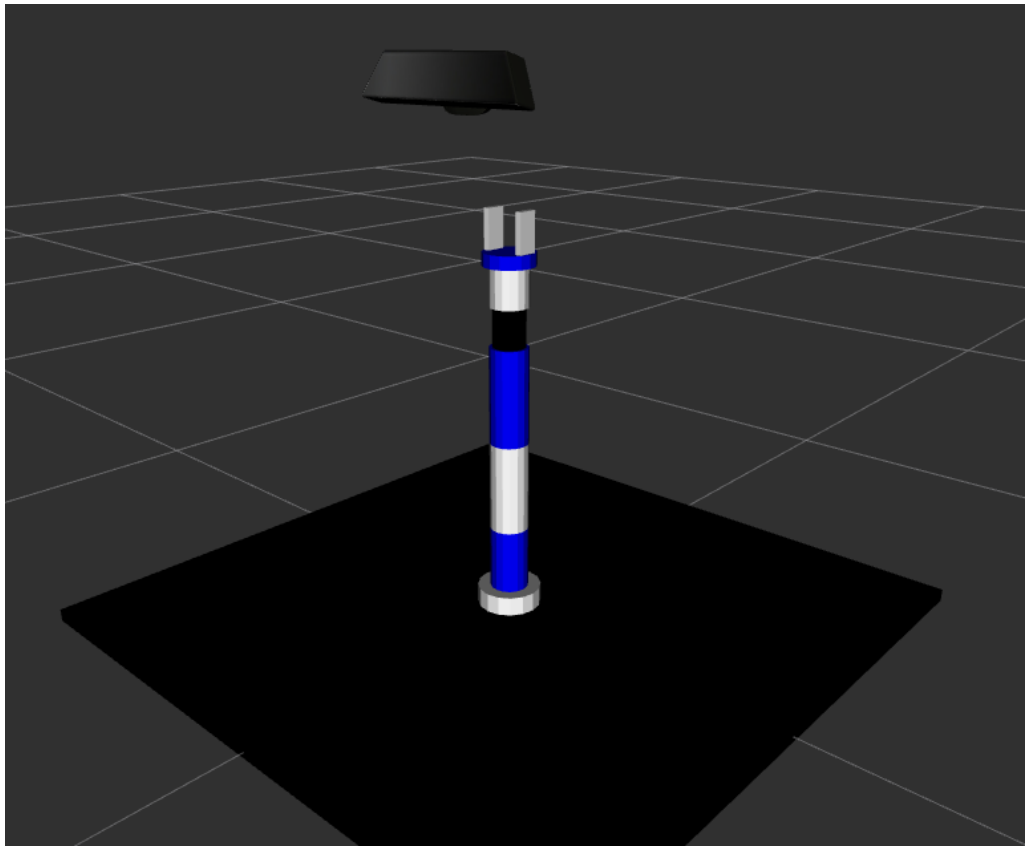


world->target = world->camera
* camera->target



视觉与控制的集成

1.机器视觉与机械臂应用

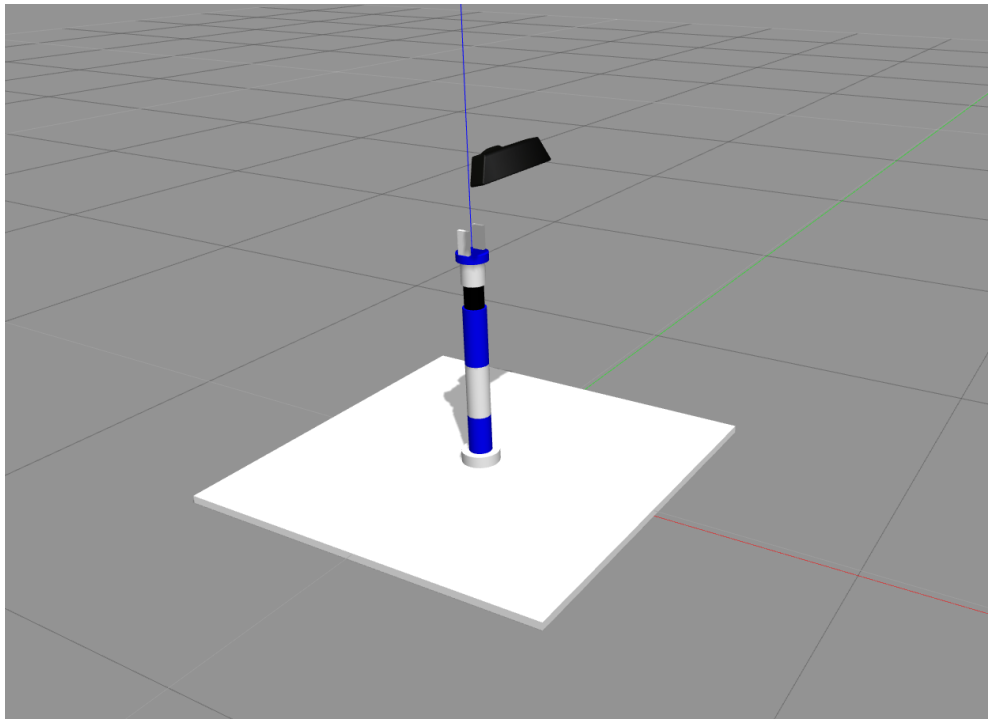


```
<?xml version="1.0"?>
<robot name="marm" xmlns:xacro="http://www.ros.org/wiki/xacro">

    <xacro:include filename="$(find marm_description)/urdf/marm_base.xacro" />
    <xacro:include filename="$(find marm_description)/urdf/kinect_gazebo.xacro" />


    <xacro:property name="deg_to_rad" value="0.01745329251994329577"/>

    <link name="bottom_link">
        <visual>
            <origin xyz=" 0 0 0.02"  rpy="0 0 0"/>
                <geometry>
                    <box size="1 1 0.02" />
                </geometry>
            <material name="Black" />
        </visual>
        <collision>
            <origin xyz=" 0 0 0.02"  rpy="0 0 0"/>
            <geometry>
                <box size="1 1 0.02" />
            </geometry>
        </collision>
        <box_inertial_matrix m="1000" w="1" h="0.02" d="1"/>
    </link>

    <arm_base parent="bottom_link" xyz="0 0 0.02" rpy="0 0 0"/>

    <!-- kinect -->
    <joint name="kinect_joint" type="fixed">
        <origin xyz="0.1 0 0.8" rpy="0 ${75.0 * deg_to_rad} 0" />
        <parent link="base_link"/>
        <child link="kinect_link"/>
    </joint>

    <xacro:kinect_camera prefix="kinect"/>

</robot>
```

模型显示　$ roslaunch marm_description view_marm_with_kinect.launch

1.机器视觉与机械臂应用

仿真环境　　$ roslaunch marm_gazebo marm_with_kinect_world.launch

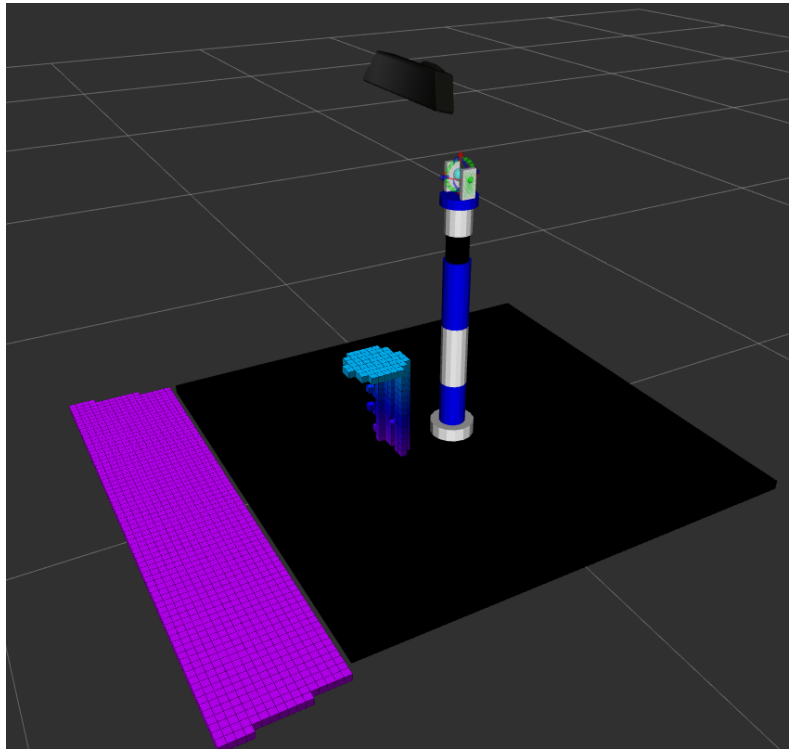


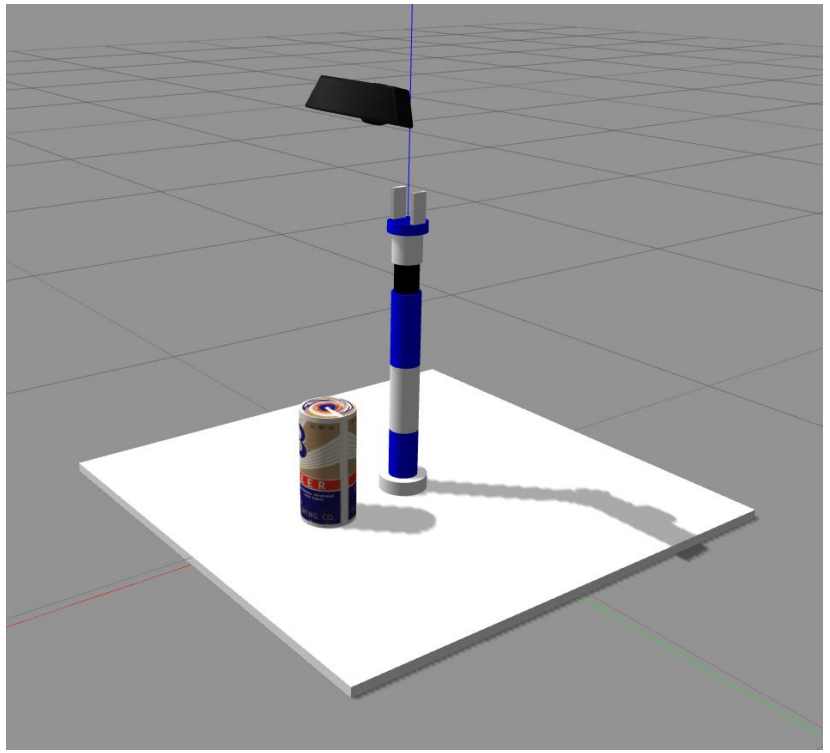
```
<launch>

  <!-- these are the arguments you can pass this launch file, for example paused:=true -->
  <arg name="paused" default="false"/>
  <arg name="use_sim_time" default="true"/>
  <arg name="gui" default="true"/>
  <arg name="headless" default="false"/>
  <arg name="debug" default="false"/>

  <!-- We resume the logic in empty_world.launch -->
  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="debug" value="$(arg debug)" />
    <arg name="gui" value="$(arg gui)" />
    <arg name="paused" value="$(arg paused)"/>
    <arg name="use_sim_time" value="$(arg use_sim_time)"/>
    <arg name="headless" value="$(arg headless)"/>
  </include>

  <!-- Load the URDF into the ROS Parameter Server -->
  <param name="robot_description" command="$(find xacro)/xacro --inorder
                                           '$(find marm_description)/urdf/marm_with_kinect.xacro'" />


  <!-- Run a python script to the send a service call to gazebo_ros to spawn a URDF robot -->
  <node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
    args="-urdf -model marm -param robot_description"/>


</launch>
```

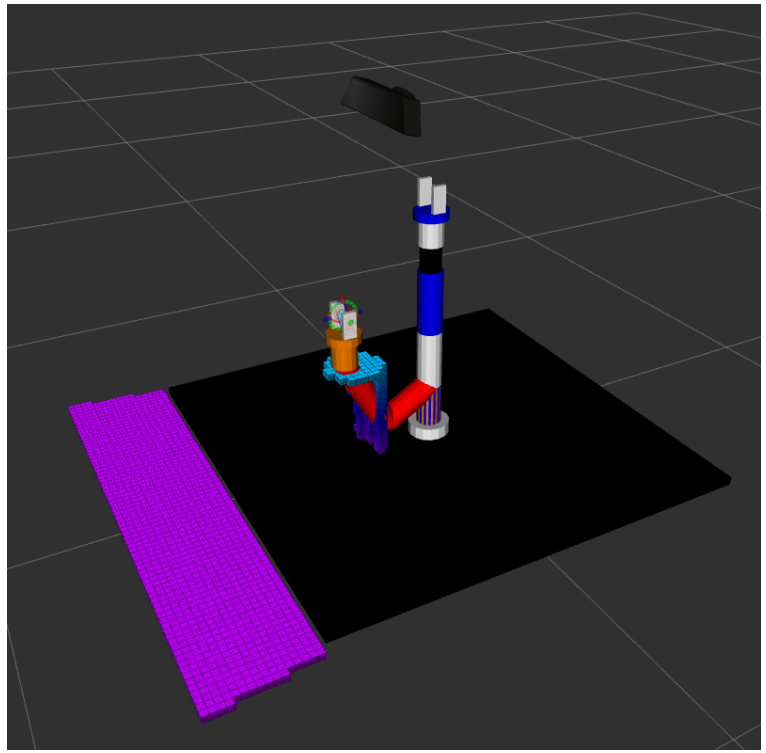
1.机器视觉与机械臂应用

MoveIt！
视觉集成

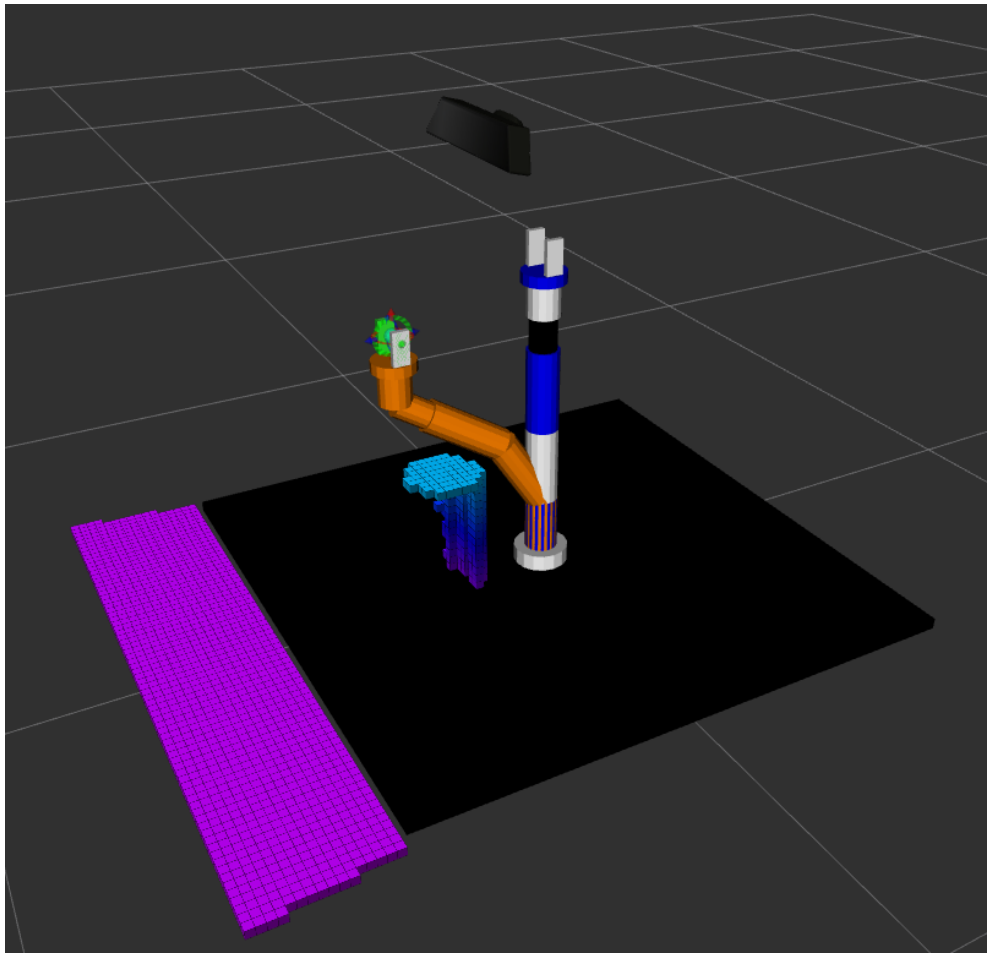
$ roslaunch marm_gazebo marm_with_kinect_bringup_moveit.launch

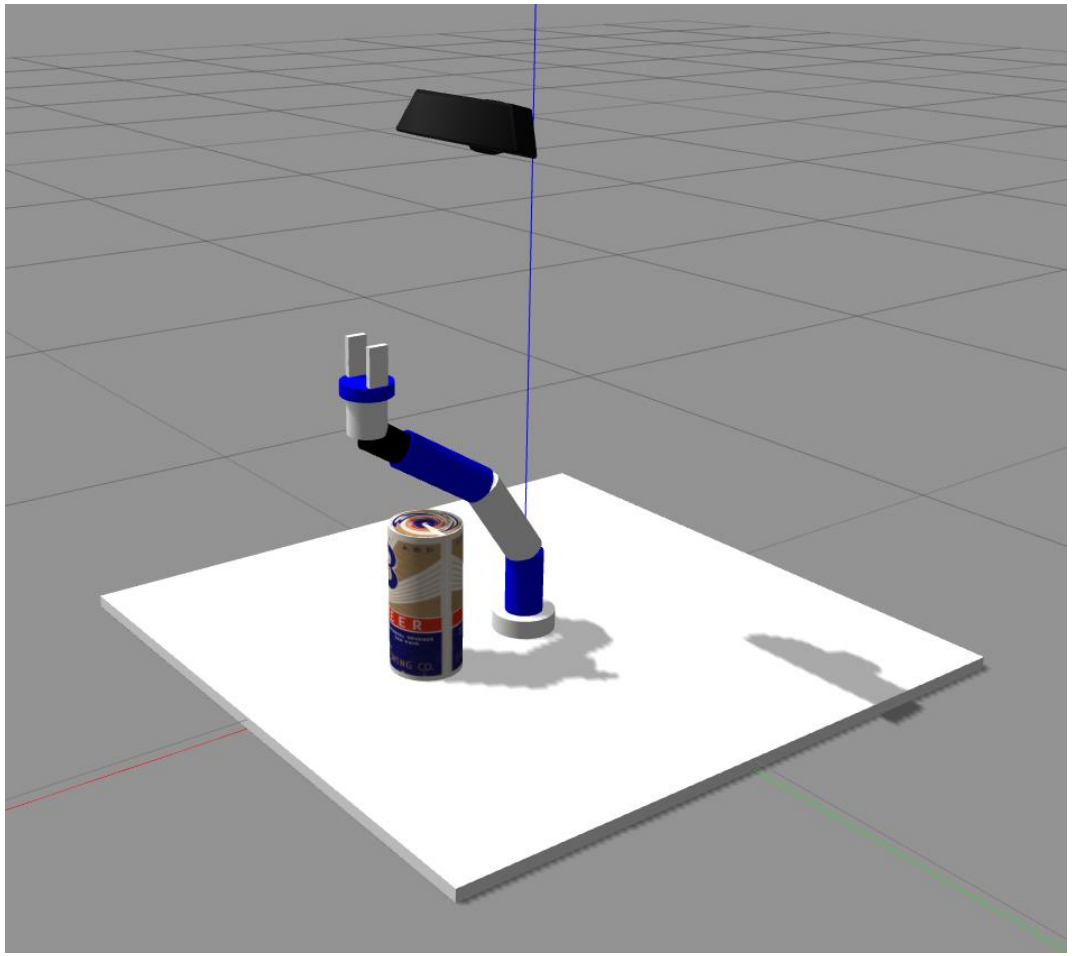






1.机器视觉与机械臂应用





运动控制

1.机器视觉与机械臂应用

```
<launch>

    <!-- Launch Gazebo  -->
    <include file="$(find marm_gazebo)/launch/marm_with_kinect_world.launch" />

    <!-- ros_control arm launch file -->
    <include file="$(find marm_gazebo)/launch/marm_gazebo_states.launch" />

    <!-- ros_control trajectory control dof arm launch file -->
    <include file="$(find marm_gazebo)/launch/marm_trajectory_controller.launch" />

    <!-- moveit launch file -->
    <include file="$(find marm_with_kinect_moveit_config)/launch/moveit_planning_execution.launch" />

</launch>
```

marm_with_kinect_bringup_moveit.launch

1.机器视觉与机械臂应用

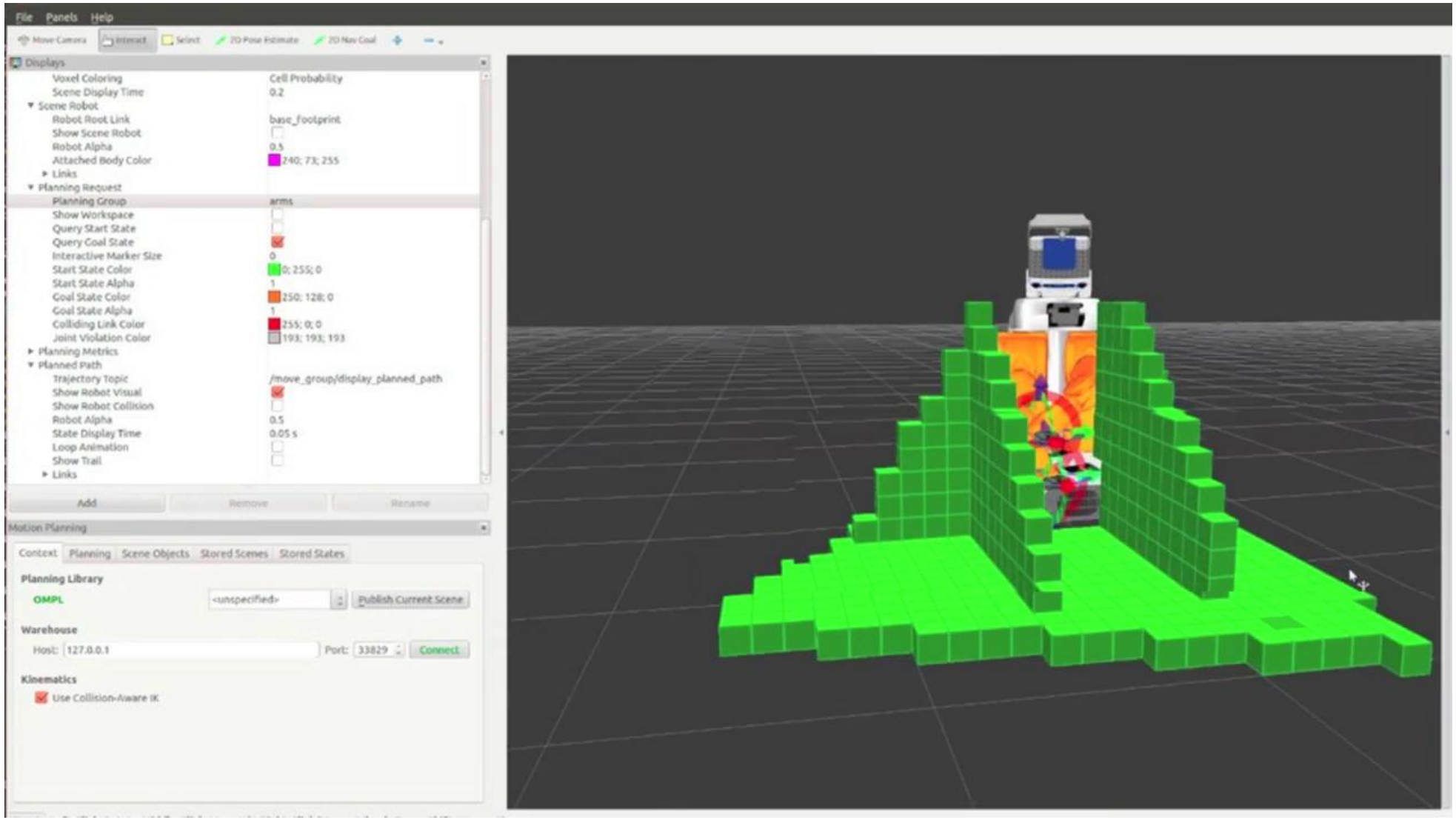
```
<launch>

<rosparam command="load" file="$(find marm_moveit_config)/config/sensors_kinect.yaml" />

<param name="octomap_frame" type="string" value="base_link" />
<param name="octomap_resolution" type="double" value="0.05" />
<param name="max_range" type="double" value="5.0" />

</launch>
```

marm_moveit_config/launch/marm_moveit_sensor_manager.launch.xml

```
sensors:
  - sensor_plugin: occupancy_map_monitor/PointCloudOctomapUpdater
    point_cloud_topic: /kinect/depth/points
    max_range: 10.0
    point_subsample: 1
    padding_offset: 0.1
    padding_scale: 1.0
    max_update_rate: 1.0
    filtered_cloud_topic: filtered_cloud
```

marm_moveit_config/config/sensors_kinect.yaml

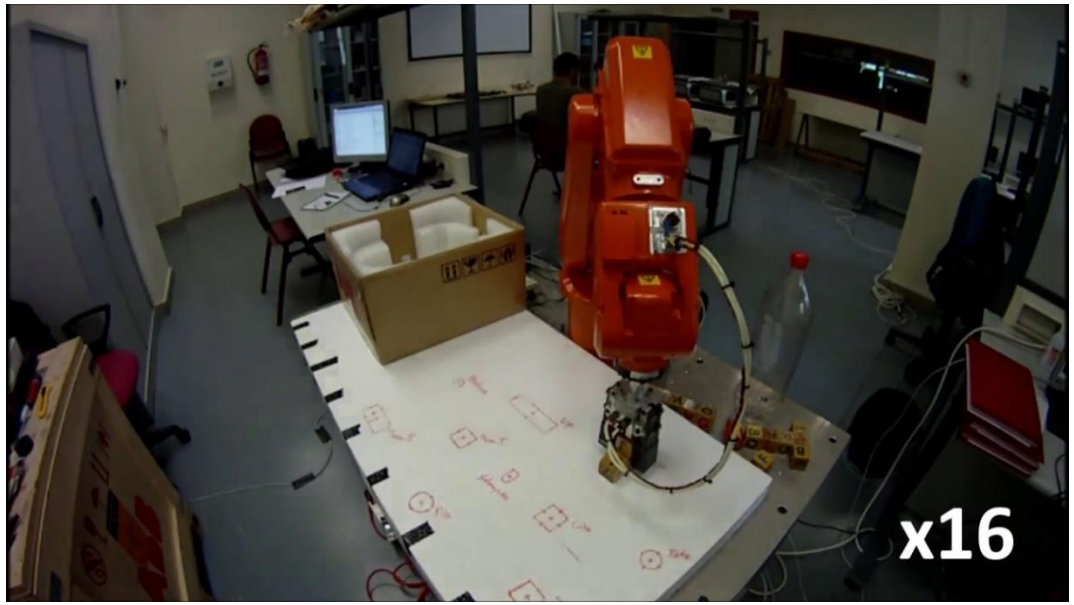
1.机器视觉与机械臂应用

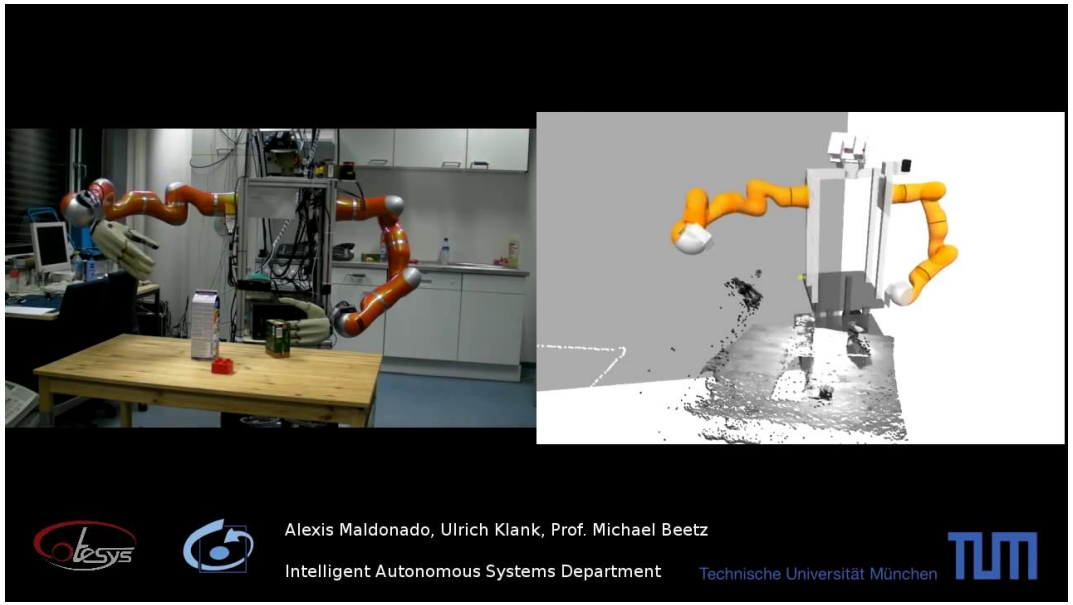


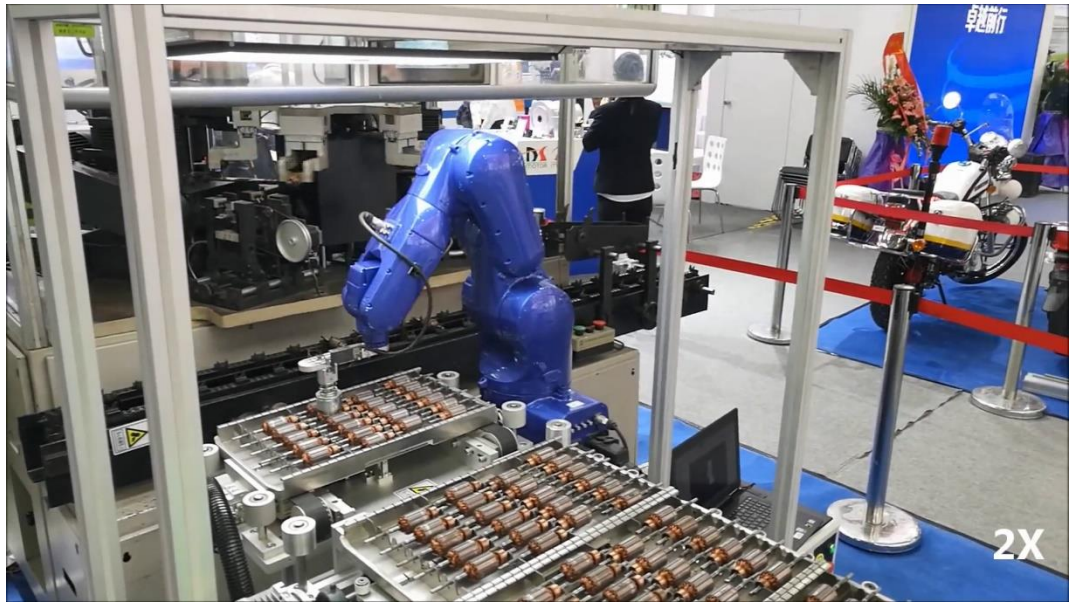
运动控制

目录

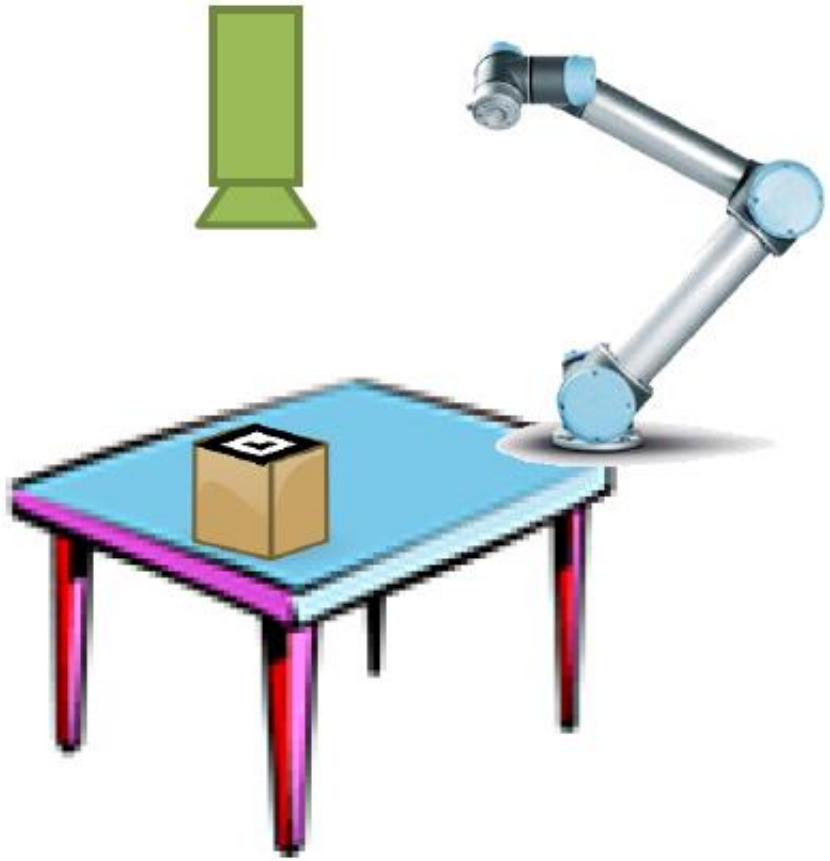
2. Pick&Place中的关键技术





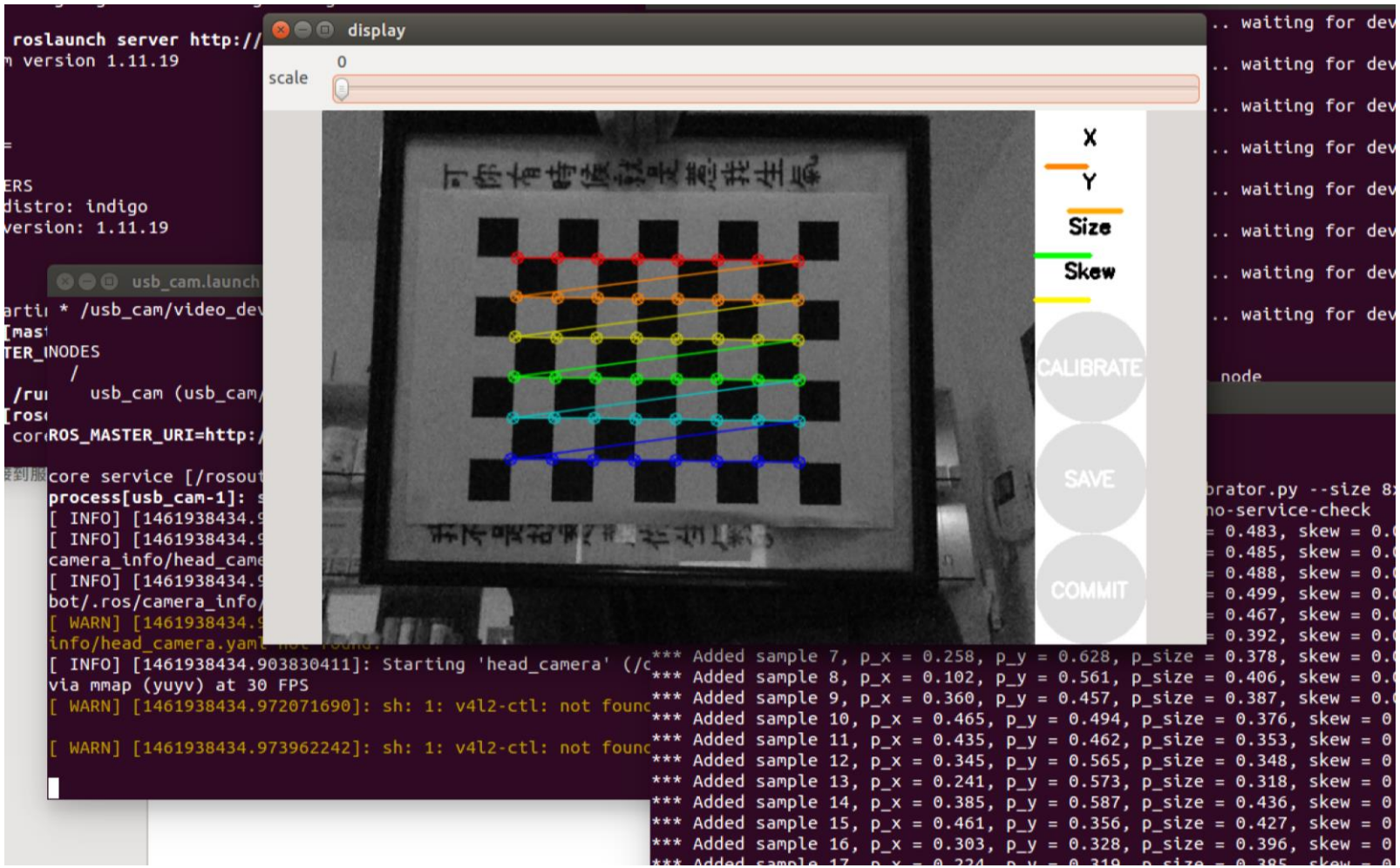


2. Pick&Place中的关键技术



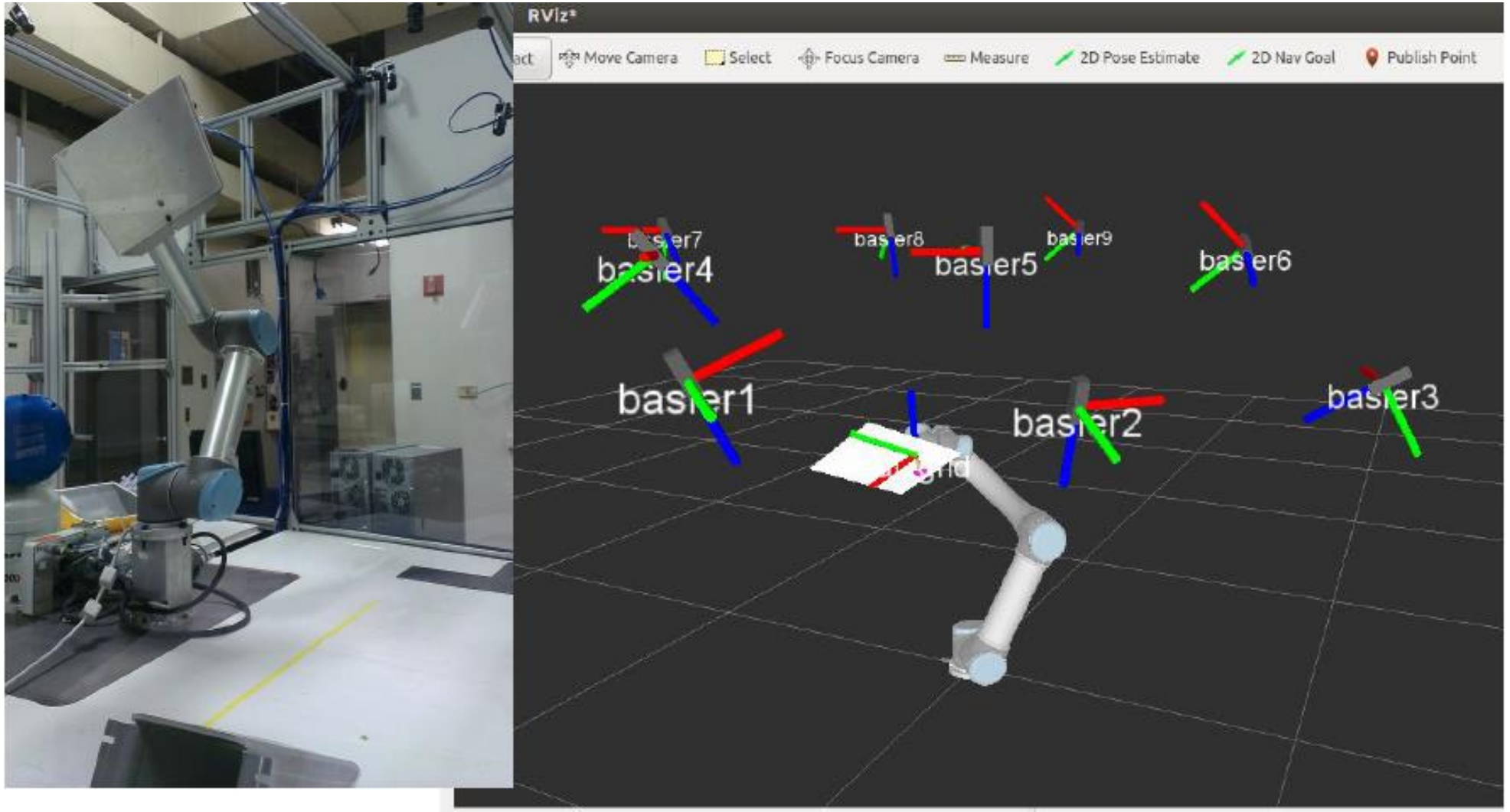
- 摄像头标定（内参、外参）
- 物体识别
- 抓取姿态分析
- 机器人运动规划

2. Pick&Place中的关键技术



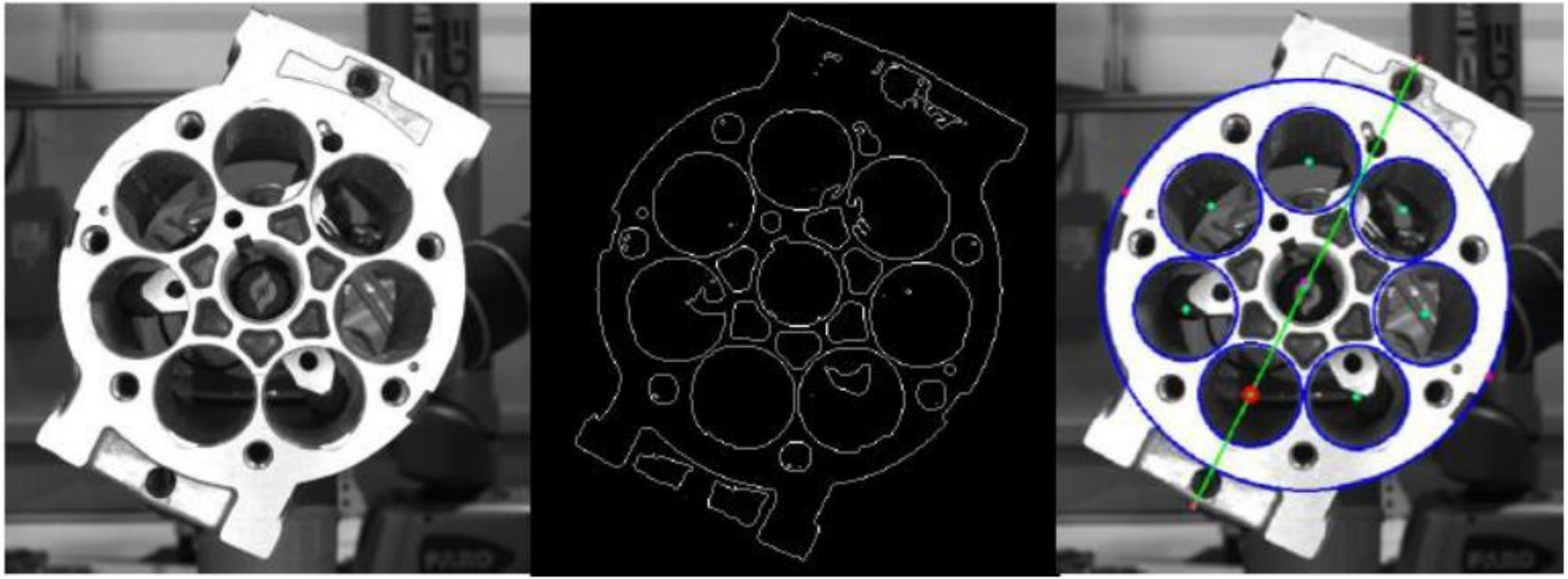
camera_calibration： http://wiki.ros.org/camera_calibration/

2. Pick&Place中的关键技术



visp_hand2eye_calibration: http://wiki.ros.org/visp_hand2eye_calibration
ros easy_handeye: https://github.com/IFL-CAMP/easy_handeye

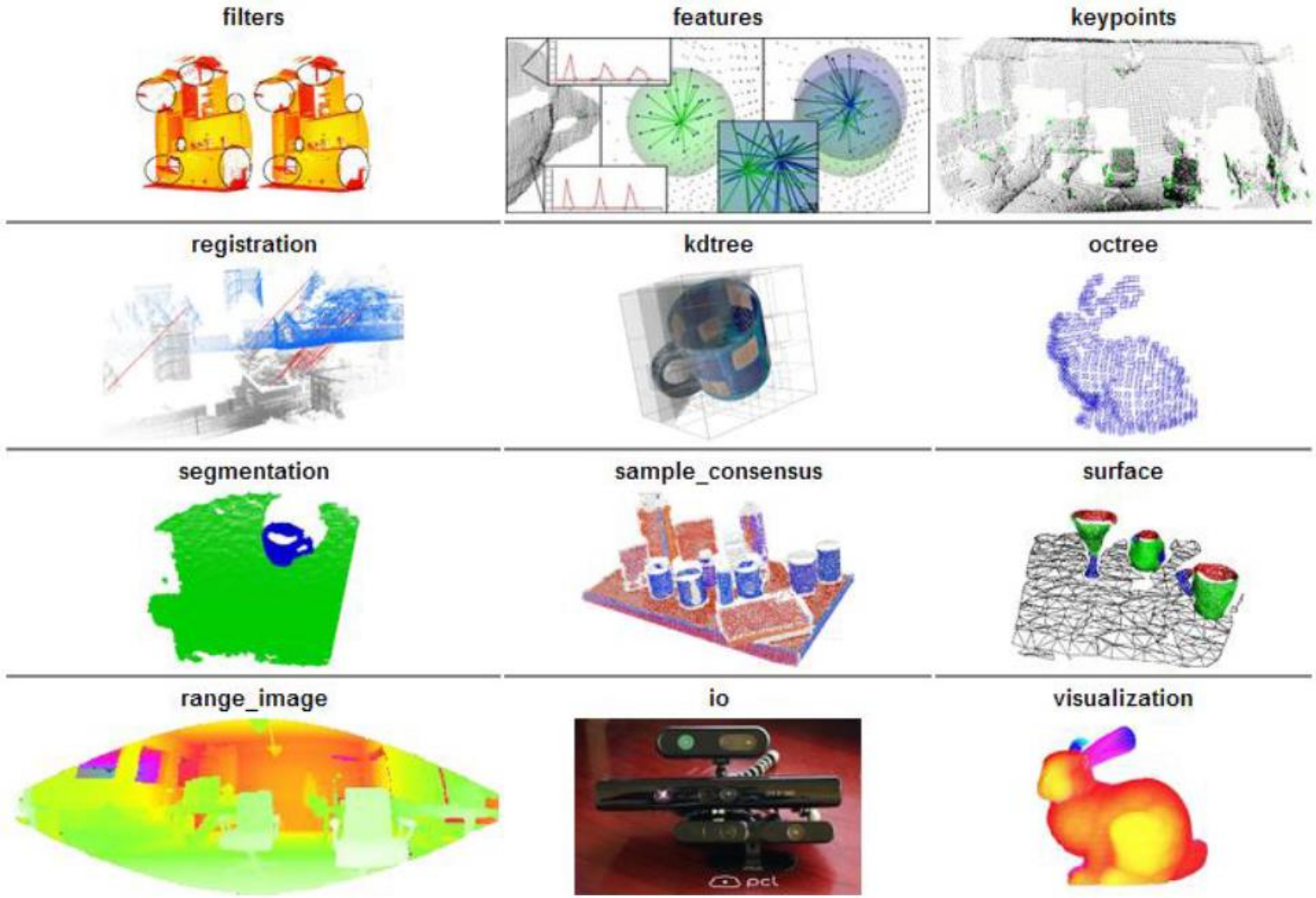
2. Pick&Place中的关键技术



基于OpenCV的图像定位

find_object_2d: http://wiki.ros.org/object_recognition
object_recognition: http://wiki.ros.org/object_recognition

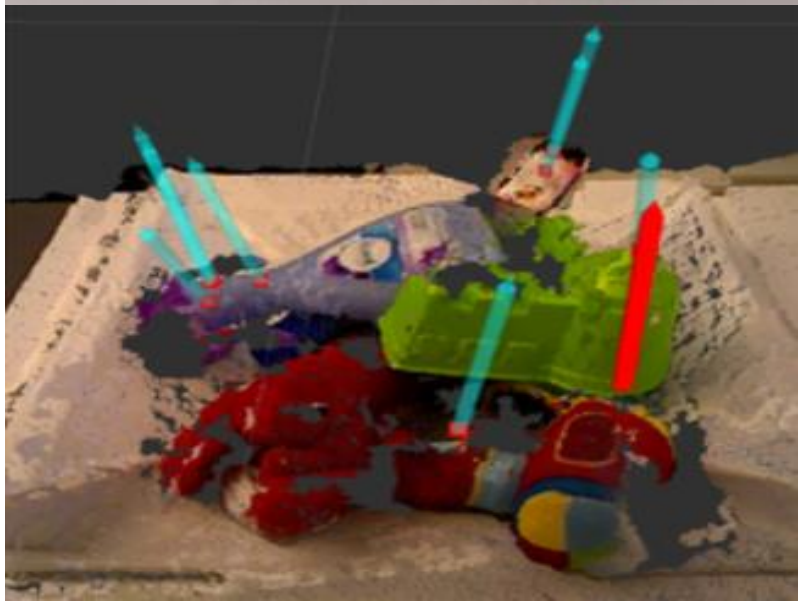
2. Pick&Place中的关键技术

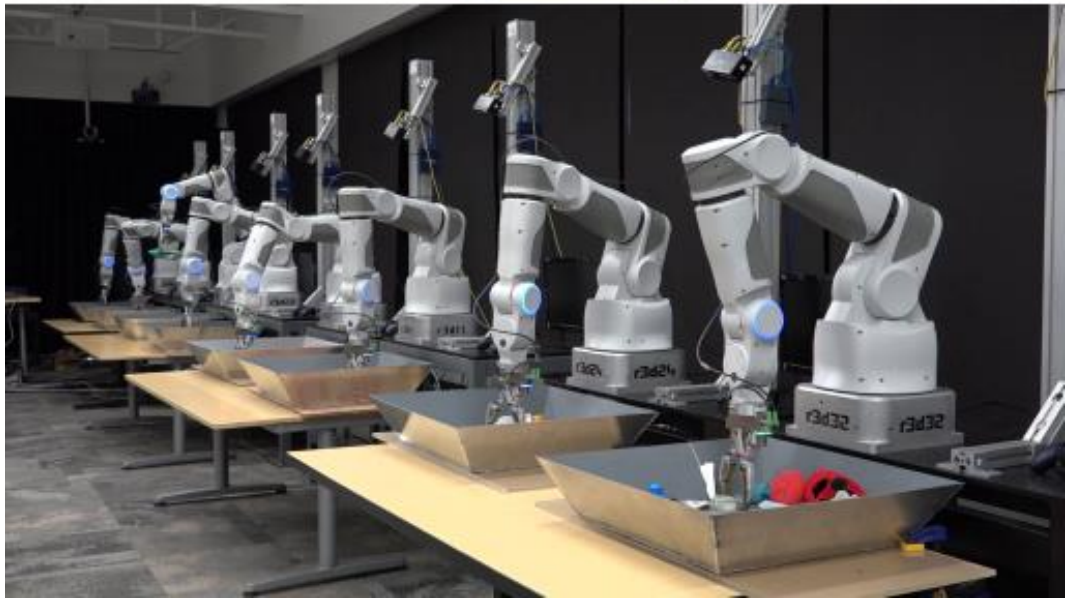


基于PCL的点云信息处理：http://wiki.ros.org/pcl/Tutorials

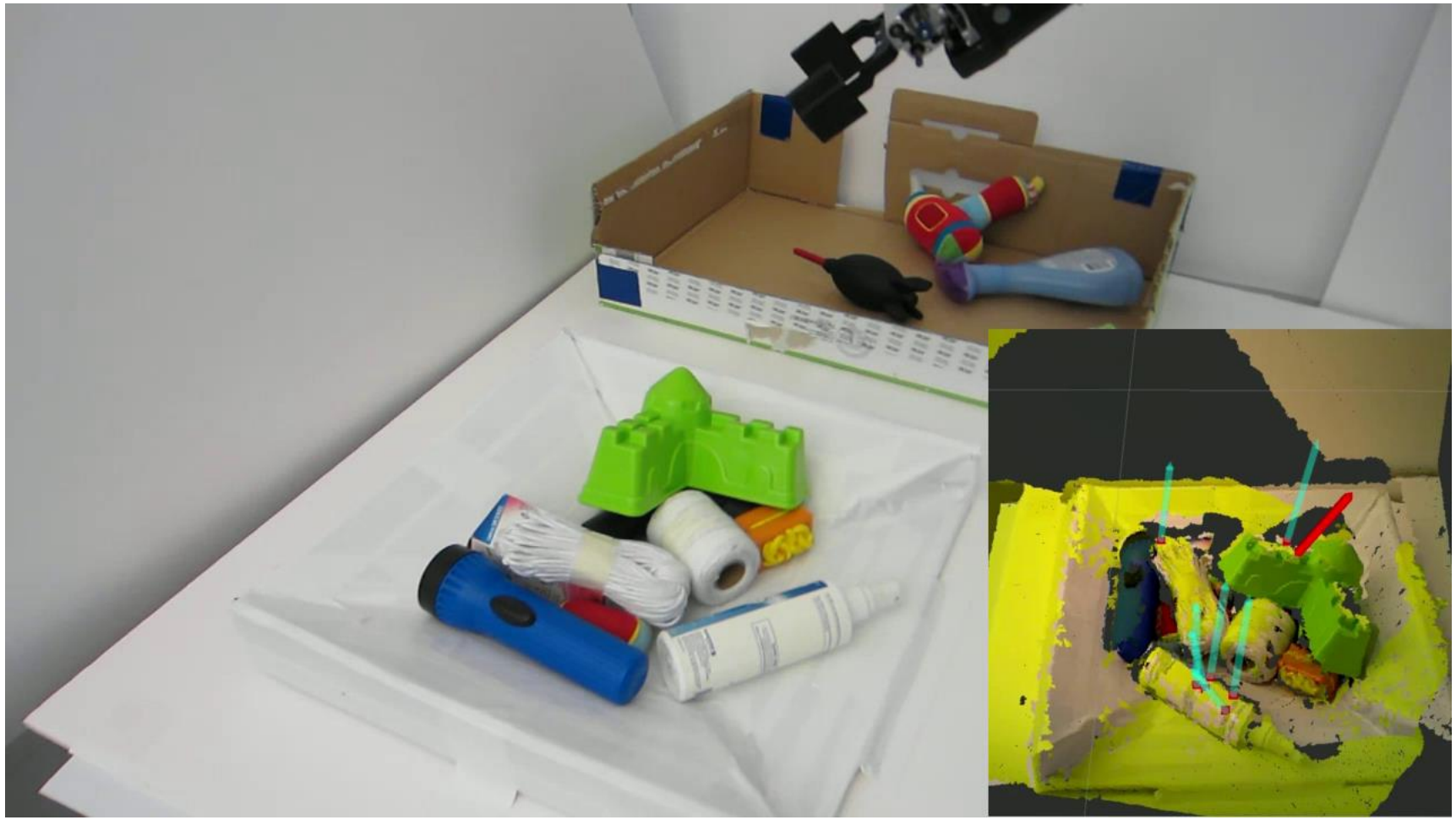
2. Pick&Place中的关键技术

抓取姿态分析





2. Pick&Place中的关键技术



agile_grasp： http://wiki.ros.org/agile_grasp
graspit： http://wiki.ros.org/graspit
moveit_simple_grasps： http://wiki.ros.org/moveit_simple_grasps

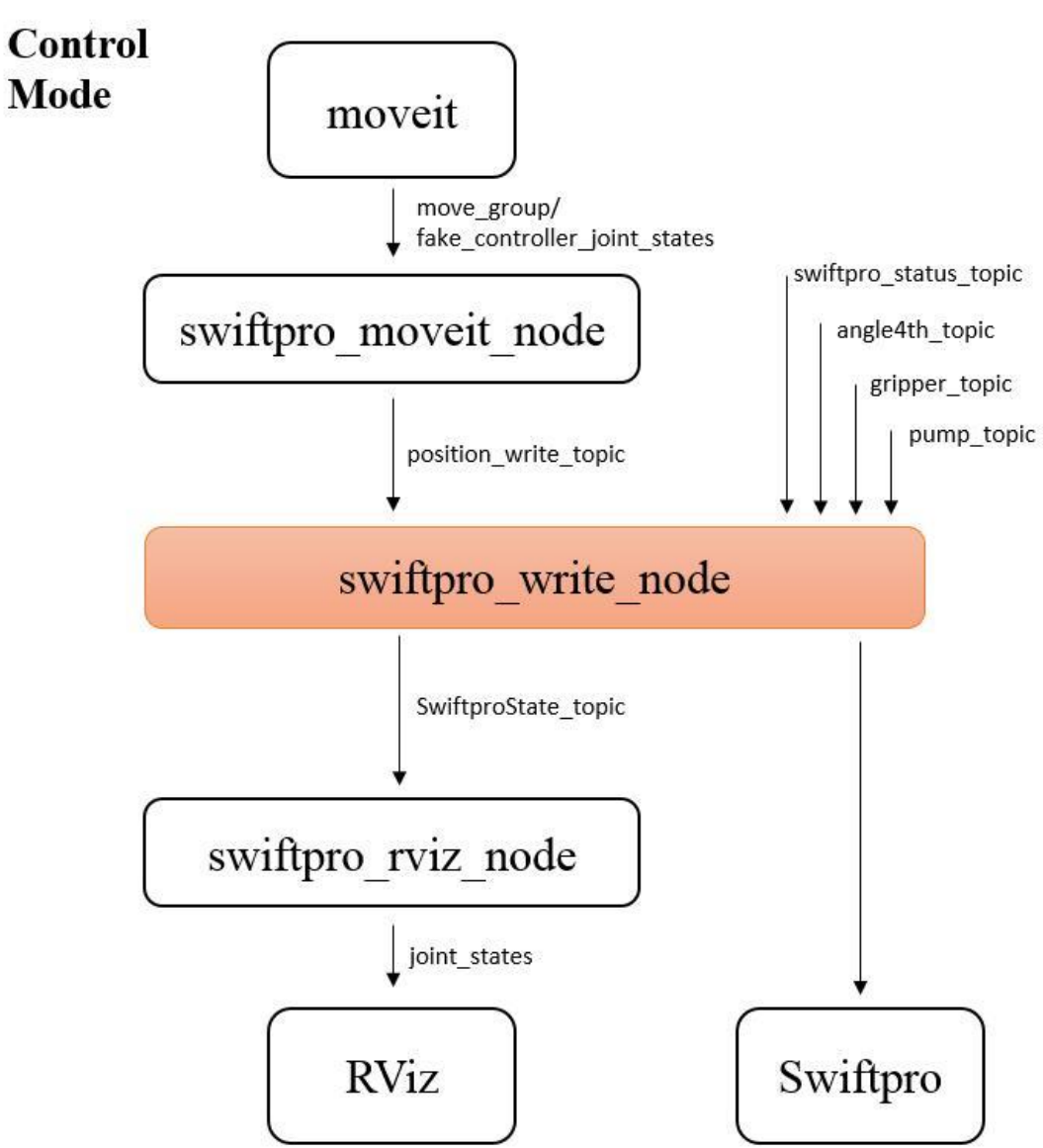
2. Pick&Place中的关键技术

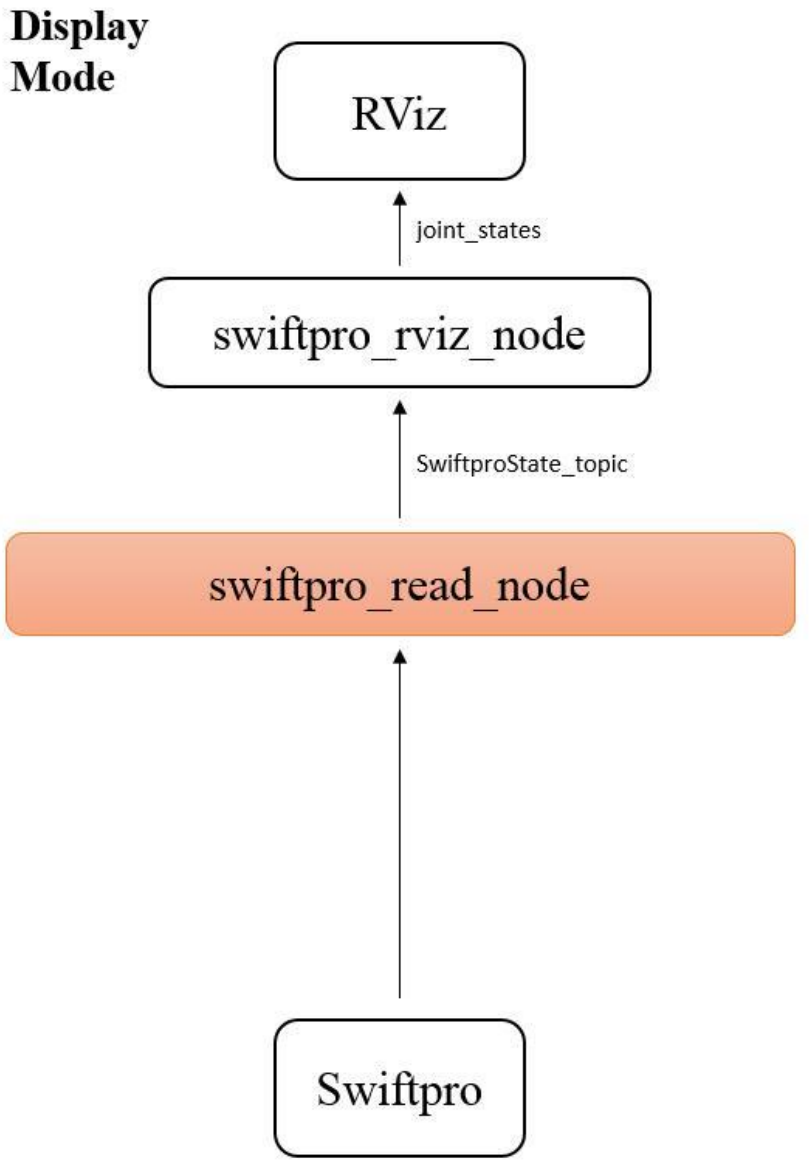
运动规划

目录

➢3. uArm编程控制实验

3. uArm编程控制实验 —— MoveIt!控制





*参见：https://github.com/uArm-Developer/RosForSwiftAndSwiftPro

3. uArm编程控制实验 —— MoveIt!控制

```
# 控制机械臂先回到初始化位置
arm.set_named_target('home')
arm.go()
rospy.sleep(2)

# 设置机械臂的目标位置，使用六轴的位置数据进行描述（单位：弧度）
joint_positions = [3.382213375042324e-05, 0.100806203853102O2, 0.3289288394143105, -0.100806203853102O2]
arm.set_joint_value_target(joint_positions)

# 控制机械臂完成运动
arm.go()
rospy.sleep(1)
```

MoveIt!控制

```
$ roslaunch swiftpro pro_control.launch
$ roslaunch pro_moveit_config demo.launch
$ rosrun uarm_demo moveit_fk_demo.py
```

3. uArm编程控制实验 —— SDK接口

```
pose_pub = rospy.Publisher('position_write_topic', position, queue_size=10)
pump_pub = rospy.Publisher('pump_topic', status, queue_size=1)

# 运动1
pos = position()
pos.x = 200
pos.y = 0
pos.z = 115

pose_pub.publish(pos)
rospy.sleep(5)

# 运动2
pos.z = -40
pose_pub.publish(pos)
pump_pub.publish(1)
rospy.sleep(5)
```

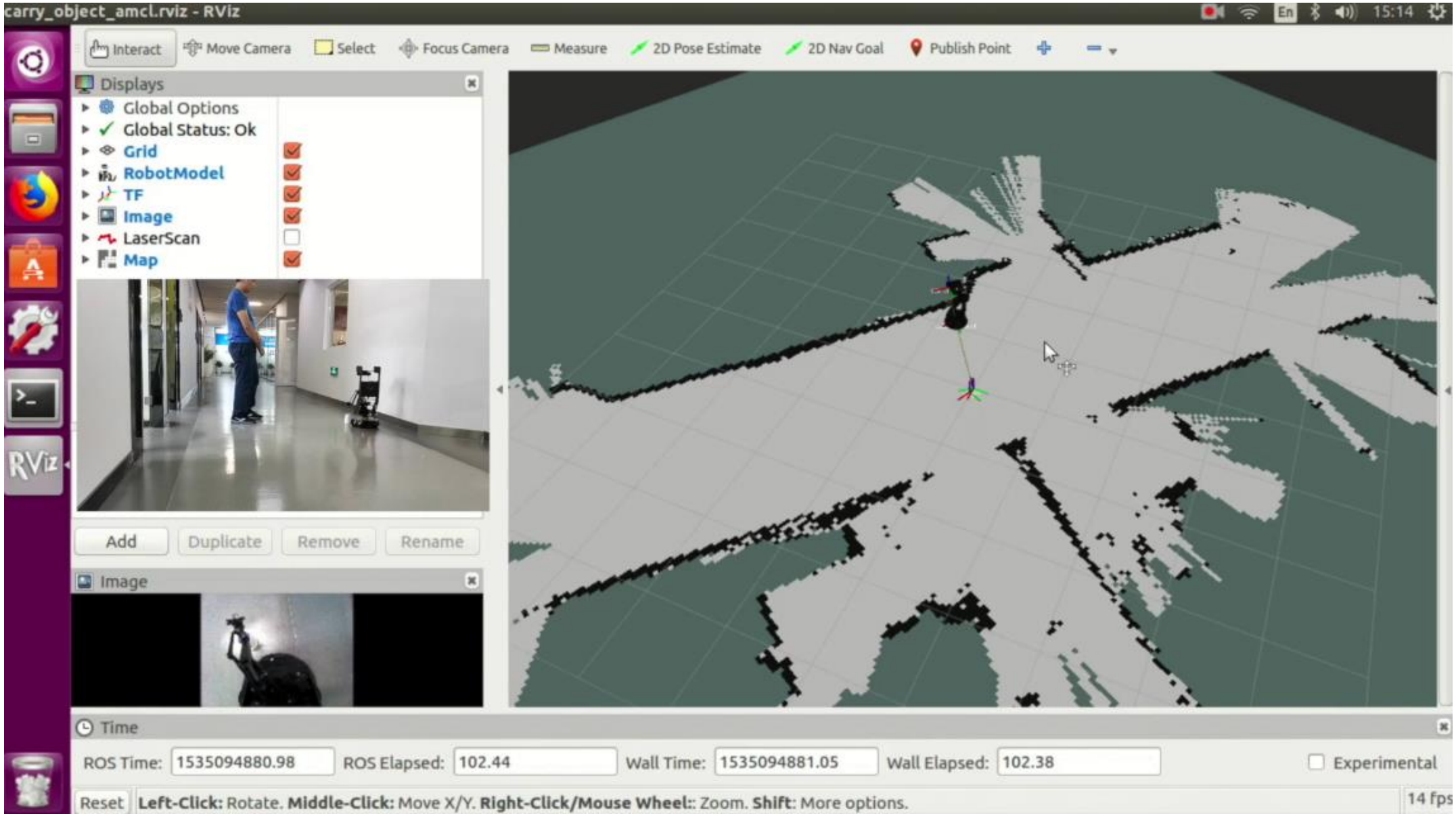
SDK接口

$ roslaunch swiftpro pro_control.launch

$ rosrun uarm_demo uarm_ik_demo.py

目录

4. Spark+uArm抓取实验 —— 演示效果



4. Spark+uArm抓取实验 —— 操作步骤

```
→  spark git:(master) ✗ ./onekey.sh
  SPARK 一键安装管理脚本 [v1.1.0]
  ---- J.xiao | www.nxrobo.com ----

  请根据右侧的功能说明选择相应的序号。
  注意：101～103为相关环境的安装与设置，如果已执行过，不要再重复执行。

    0. 单独编译SPARK
_____________
    1. 让机器人动起来
    2. 远程（手机APP）控制SPARK
    3. 让SPARK跟着你走
    4. 让SPARK使用激光雷达绘制地图
    5. 让SPARK使用深度摄像头绘制地图
    6. 让SPARK使用激光雷达进行导航
    7. 让SPARK使用深度摄像头进行导航
    8. 机械臂与摄像头标定
    9. 让SPARK通过机械臂进行视觉抓取
   10. 使用tensorflow进行物品检测
   11. 语音移动控制
   12. 微信移动控制


_____________
  100. 问题反馈
  101. 完整安装
  102. 单独安装ROS环境
  103. 单独安装SPARK依赖

[注意] 当前系统版本 Ubuntu 16.04.5 LTS !
[注意] 当前ROS版本 kinetic !

请输入数字：
```

Spark + uArm抓取

Thank you!