深蓝学院 shenlanxueyuan.com

古月居 GYH.AI

ROS机械臂开发：从入门到实战

—— 第5讲：搭建仿真环境一样玩转ROS机械臂

主讲人 胡春旭

机器人博客“古月居”博主

《ROS机器人开发实践》作者

武汉精锋微控科技有限公司 联合创始人

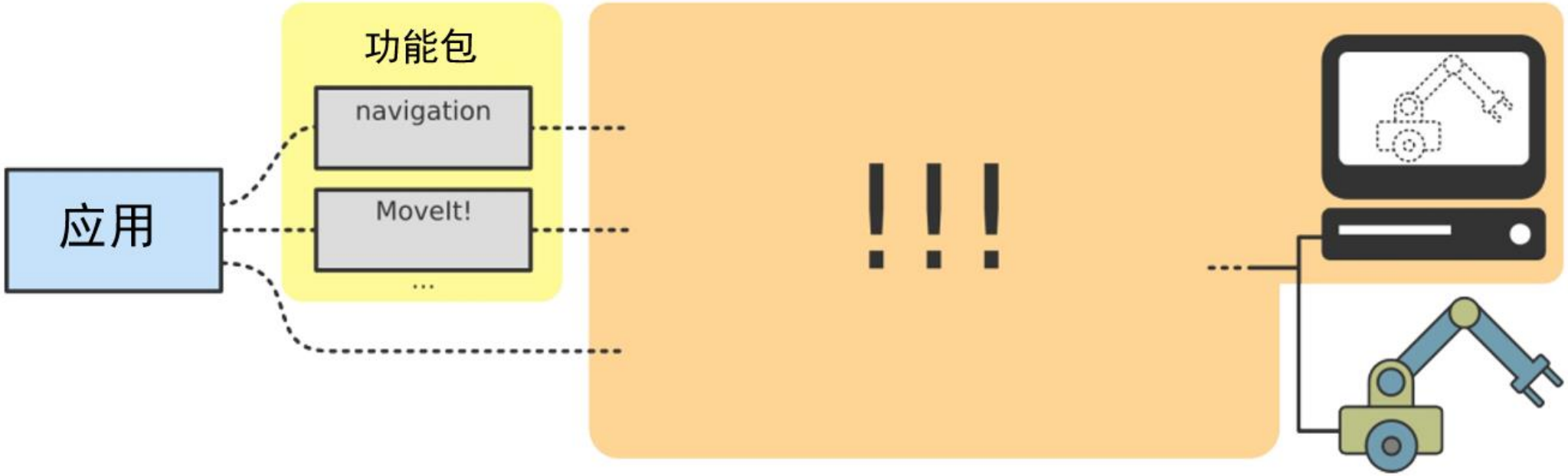
华中科技大学 自动化学院 硕士

目录

古月居 GYH.AI

1. ROS中的控制器插件

1. ROS中的控制器插件



ros_control是什么?

- ROS为开发者提供的机器人控制中间件
- 包含一系列控制器接口、传动装置接口、硬件接口、控制器工具箱等等
- 可以帮助机器人应用功能包快速落地，提高开发效率

1. ROS中的控制器插件

- **控制器管理器**

提供一种通用的接口来管理不同的控制器。

- **控制器**

读取硬件状态，发布控制命令，完成每个joint的控制。

- **硬件资源**

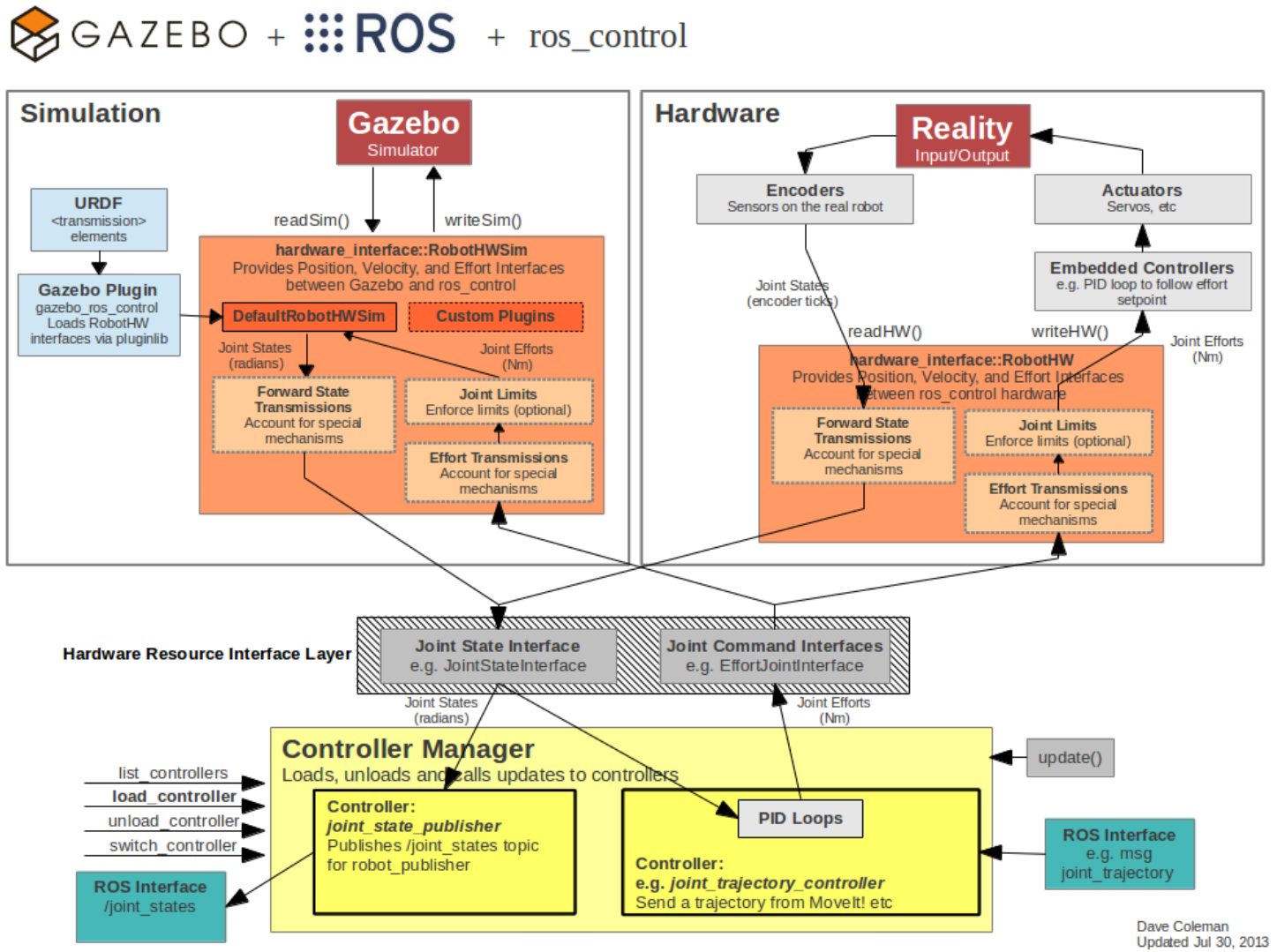
为上下两层提供硬件资源的接口。

- **机器人硬件抽象**

机器人硬件抽象和硬件资源直接打交道，通过write和read方法完成硬件操作。

- **真实机器人**

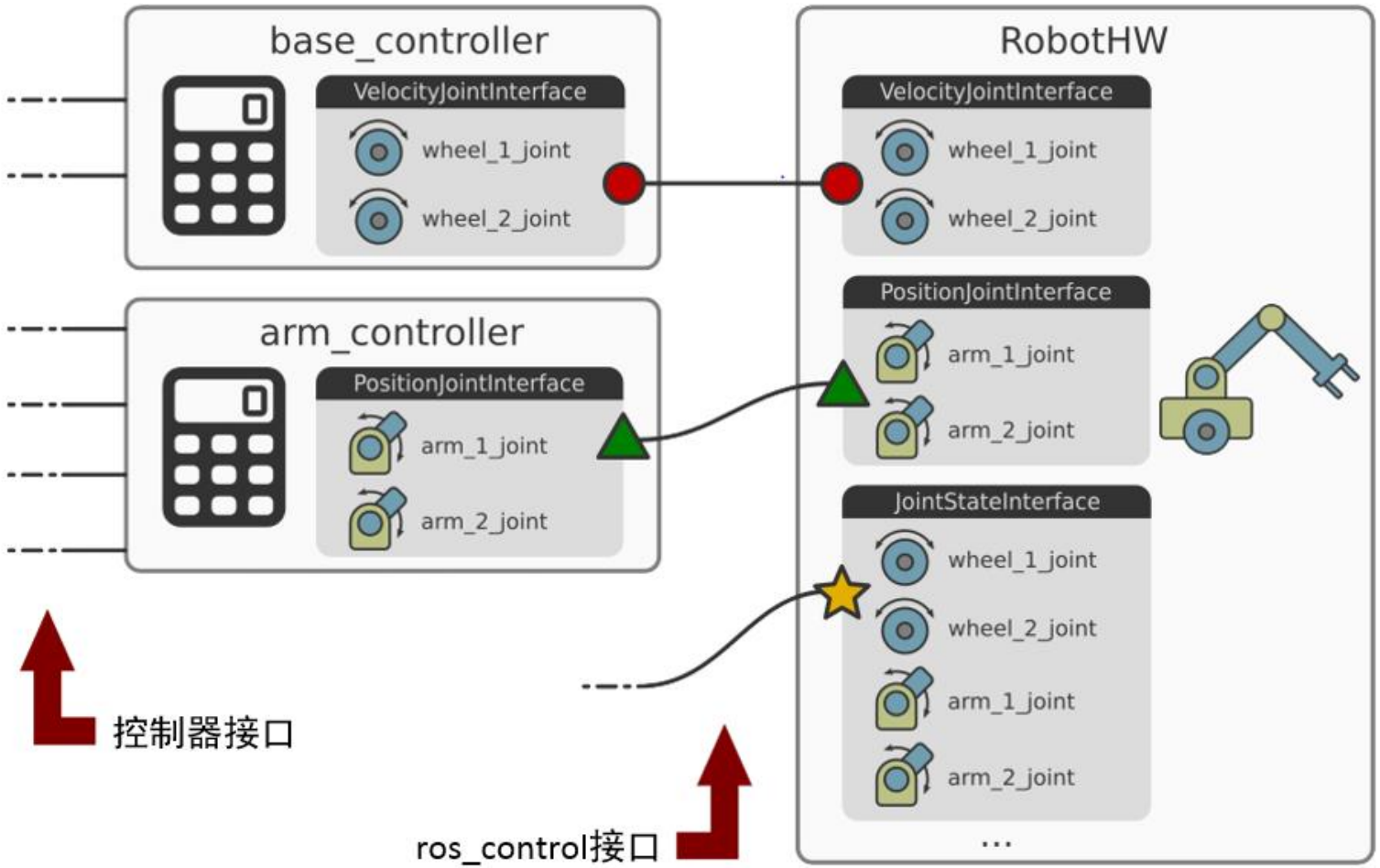
执行接收到的命令。



1. ROS中的控制器插件

控制器（Controllers）：

- joint_state_controller
- joint_effort_controller
- joint_position_controller
- joint_velocity_controller



可参考：https://github.com/ros-controls/ros_control/wiki/controller_interface

2. 完善机器人模型

2. 完善机器人模型

第一步：为link添加惯性参数和碰撞属性

```
<link name="link_1">
  <inertial>
    <origin xyz="-0.010934 0.23134 0.0051509" rpy="0 0 0" />
    <mass value="0.00001" />
    <inertia ixx="10" ixy="0.0" ixz="0.0" iyy="10" iyz="0.0" izz="10" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://probot_description/meshes/link_1.STL" />
    </geometry>
    <material name="">
      <color rgba="0.79216 0.81961 0.93333 1" />
    </material>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://probot_description/meshes/link_1.STL" />
    </geometry>
  </collision>
</link>
```

第二步：为joint添加传动装置

```
<!-- Transmissions for ROS Control -->
<xacro:macro name="transmission_block" params="joint_name">
  <transmission name="tran1">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="${joint_name}">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor1">
      <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
</xacro:macro>

<xacro:transmission_block joint_name="joint_1"/>
<xacro:transmission_block joint_name="joint_2"/>
<xacro:transmission_block joint_name="joint_3"/>
<xacro:transmission_block joint_name="joint_4"/>
<xacro:transmission_block joint_name="joint_5"/>
<xacro:transmission_block joint_name="joint_6"/>
```

第三步：添加gazebo控制器插件

```
<!-- ros_control plugin -->
<gazebo>
  <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
    <robotNamespace>/probot_anno</robotNamespace>
    <robotSimType>gazebo_ros_control/DefaultRobotHWSim</robotSimType>
    <legacyModeNS>true</legacyModeNS>
  </plugin>
</gazebo>
```

2. 完善机器人模型

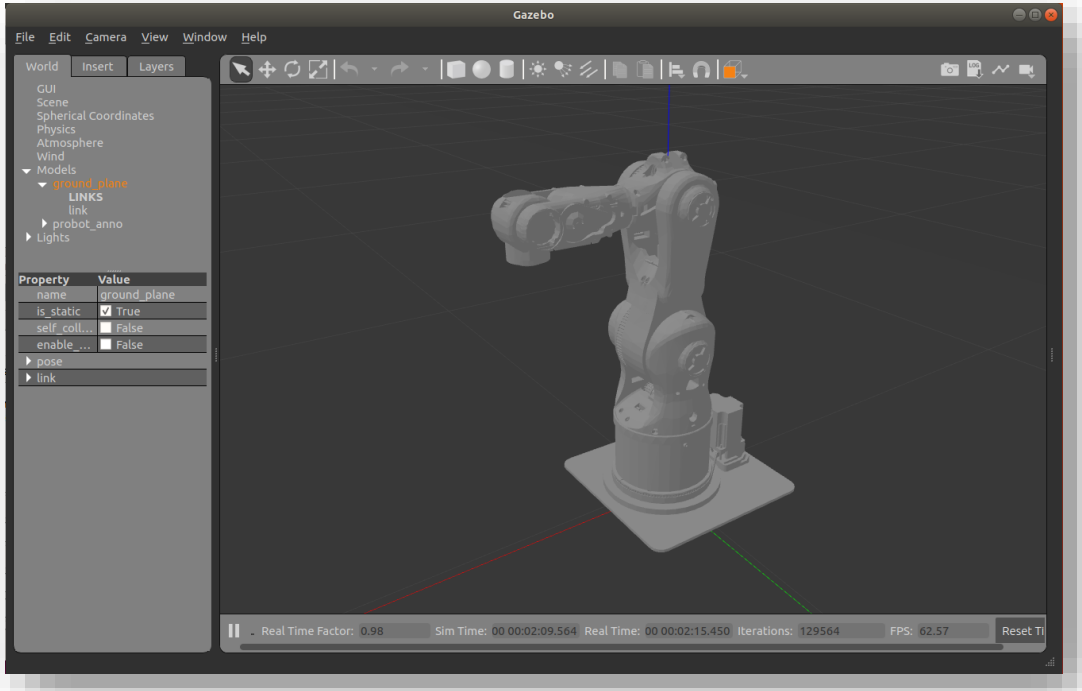
在gazebo中加载机器人模型

```
<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 probot_description)/urdf/probot_anno.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 probot_anno -param robot_description"/>


</launch>
```

probot_gazebo/…/probot_anno_gazebo_world.launch

2. 完善机器人模型

启动仿真环境 $ roslaunch probot_gazebo probot_anno_gazebo_world.launch

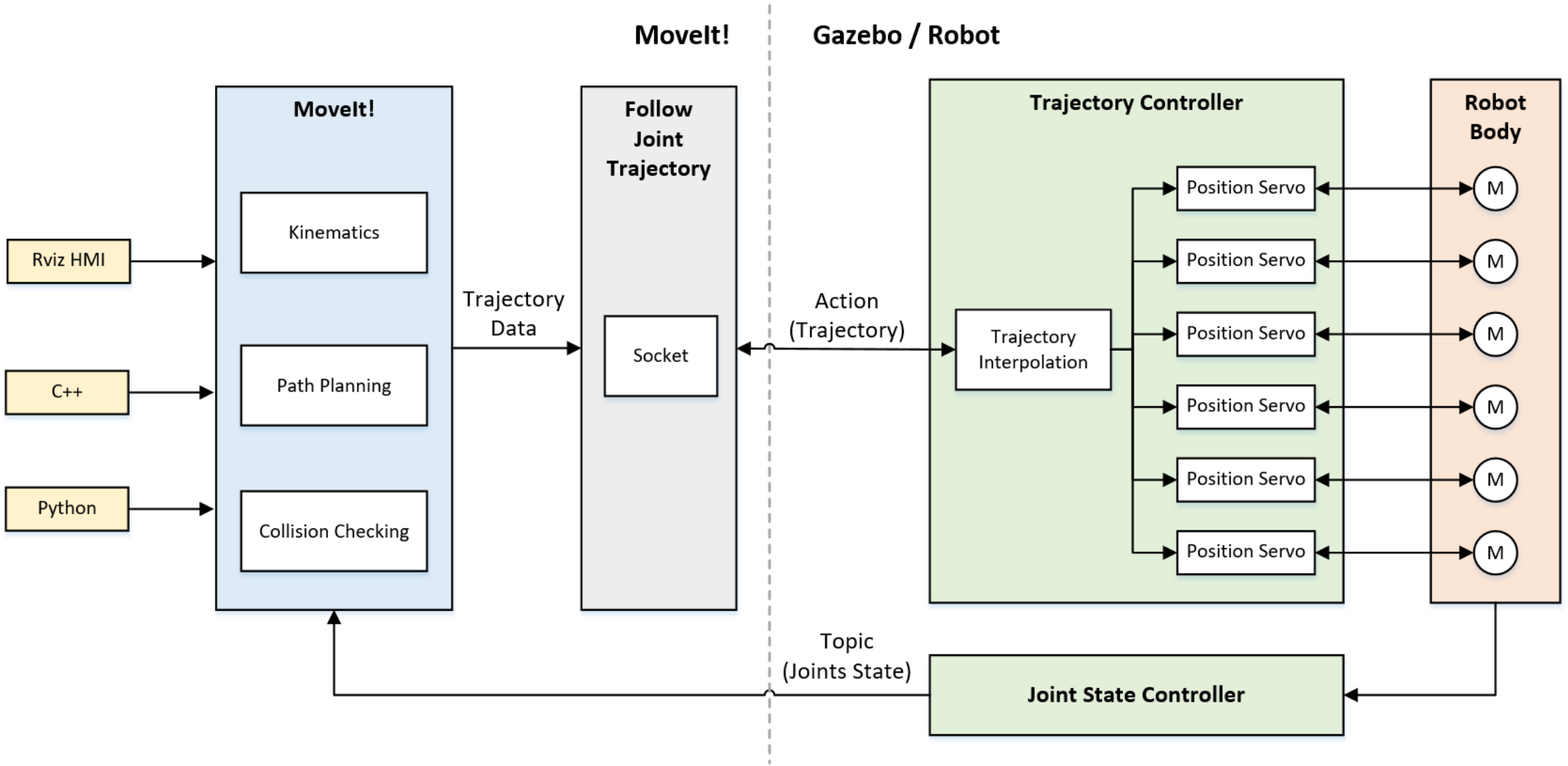


建议：为保证模型顺利加载，请提前将模型文件库下载并放置到~/.gazebo/models下

https://bitbucket.org/osrf/gazebo_models/downloads/

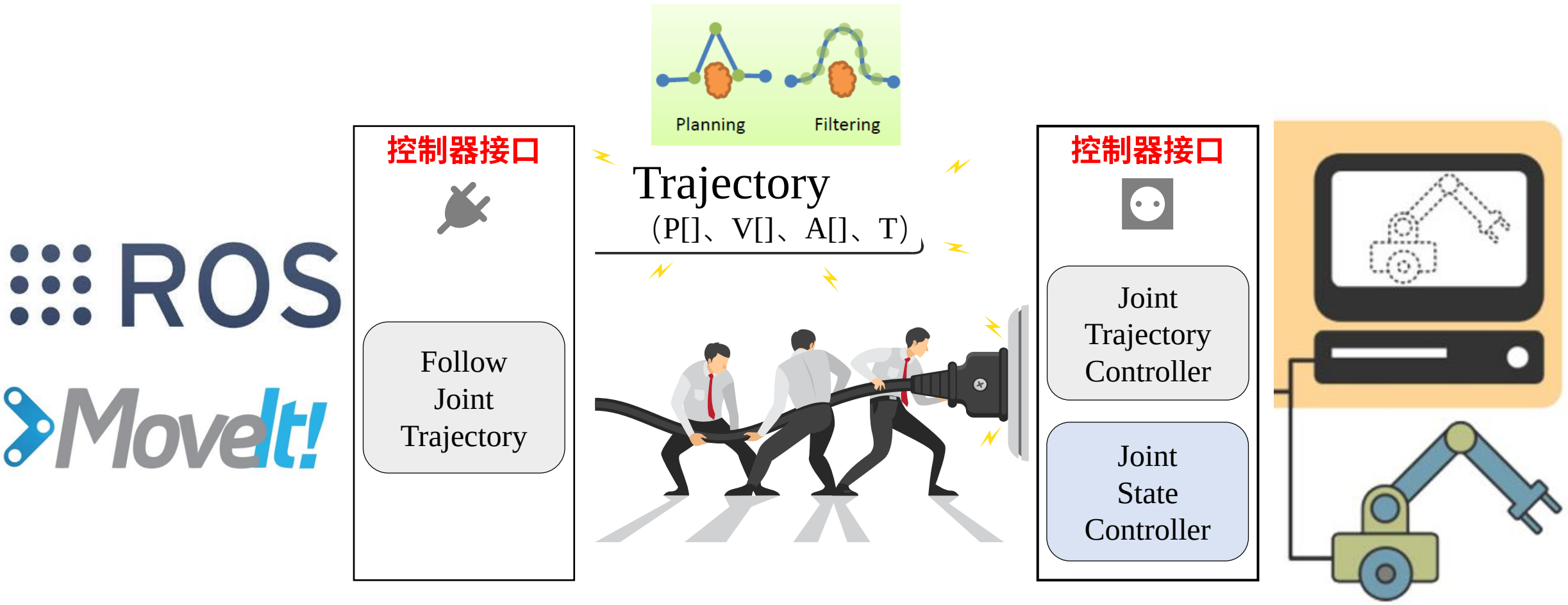
3. 构建MoveIt!+Gazebo仿真

3. 构建MoveIt!+Gazebo仿真



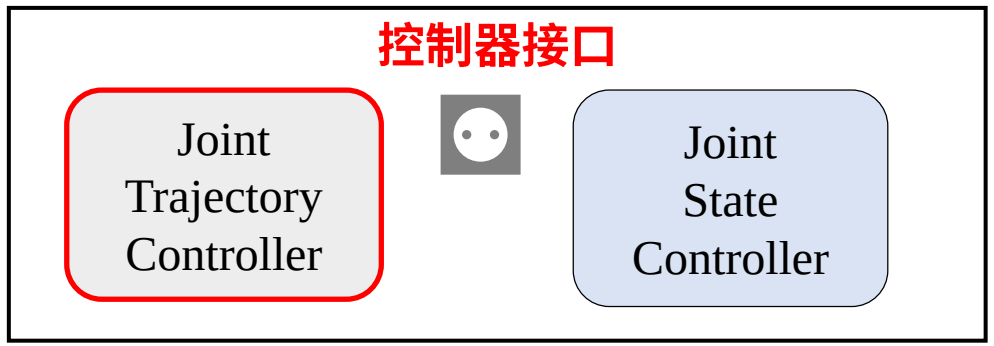
MoveIt!机器人控制框架

3. 构建MoveIt!+Gazebo仿真



MoveIt!+Gazebo仿真框架

3. 构建MoveIt!+Gazebo仿真



关节轨迹控制器

- 线性样条：位置连续，速度、加速度不连续。
- 三次样条：位置和速度连续，加速度不连续。
- 五次样条：位置、速度、加速度都连续。

(1) 参数配置

```yaml
probot_anno:
  arm_joint_controller:
    type: "position_controllers/JointTrajectoryController"
    joints:
      - joint_1
      - joint_2
      - joint_3
      - joint_4
      - joint_5
      - joint_6

    gains:
      joint_1:   {p: 1000.0, i: 0.0, d: 0.1, i_clamp: 0.0}
      joint_2:   {p: 1000.0, i: 0.0, d: 0.1, i_clamp: 0.0}
      joint_3:   {p: 1000.0, i: 0.0, d: 0.1, i_clamp: 0.0}
      joint_4:   {p: 1000.0, i: 0.0, d: 0.1, i_clamp: 0.0}
      joint_5:   {p: 1000.0, i: 0.0, d: 0.1, i_clamp: 0.0}
      joint_6:   {p: 1000.0, i: 0.0, d: 0.1, i_clamp: 0.0}
```

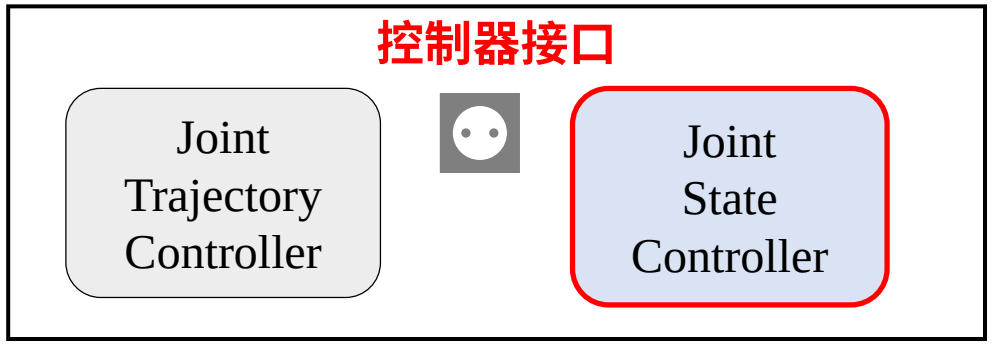
probot_gazebo/config/probot_anno_trajectory_control.yaml

(2) 控制器启动

```xml
<launch>

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

    <node name="arm_controller_spawner" pkg="controller_manager" type="spawner" respawn="false"
          output="screen" ns="/probot_anno" args="arm_joint_controller"/>

</launch>
```

probot _gazebo/…/probot_anno_trajectory_controller.launch

3. 构建MoveIt!+Gazebo仿真



关节状态控制器

(1) 参数配置

```yaml
probot_anno:
  # Publish all joint states -----------------------
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50
```

probot_gazebo/config/probot_anno_gazebo_joint_states.yaml

```xml
<launch>
    <!-- 将关节控制器的配置参数加载到参数服务器中 -->
    <rosparam file="$(find probot_gazebo)/config/probot_anno_gazebo_joint_states.yaml" command="load"/>

    <node name="joint_controller_spawner" pkg="controller_manager" type="spawner" respawn="false"
          output="screen" ns="/probot_anno" args="joint_state_controller" />

    <!-- 运行robot_state_publisher节点，发布tf  -->
    <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher"
        respawn="false" output="screen">
        <remap from="/joint_states" to="/probot_anno/joint_states" />
    </node>

</launch>
```

(2) 控制器启动

probot _gazebo/…/probot_anno_gazebo_states.launch

3. 构建MoveIt!+Gazebo仿真

古月居 GYH.AI

MoveIt!控制器

控制器接口

Follow Joint Trajectory

```yaml
controller_manager_ns: controller_manager
controller_list:
  - name: probot_anno/arm_joint_controller
    action_ns: follow_joint_trajectory
    type: FollowJointTrajectory
    default: true
    joints:
      - joint_1
      - joint_2
      - joint_3
      - joint_4
      - joint_5
      - joint_6
```

(1) 参数配置

probot_anno_moveit_config/config/controllers_gazebo.yaml

```xml
<launch>
    <arg name="moveit_controller_manager" default="moveit_simple_controller_manager/MoveItSimpleControllerManager"/>
    <param name="moveit_controller_manager" value="$(arg moveit_controller_manager)"/>

    <!-- gazebo Controller -->
    <rosparam file="$(find probot_anno_moveit_config)/config/controllers_gazebo.yaml"/>

</launch>
```

(2) 控制器启动

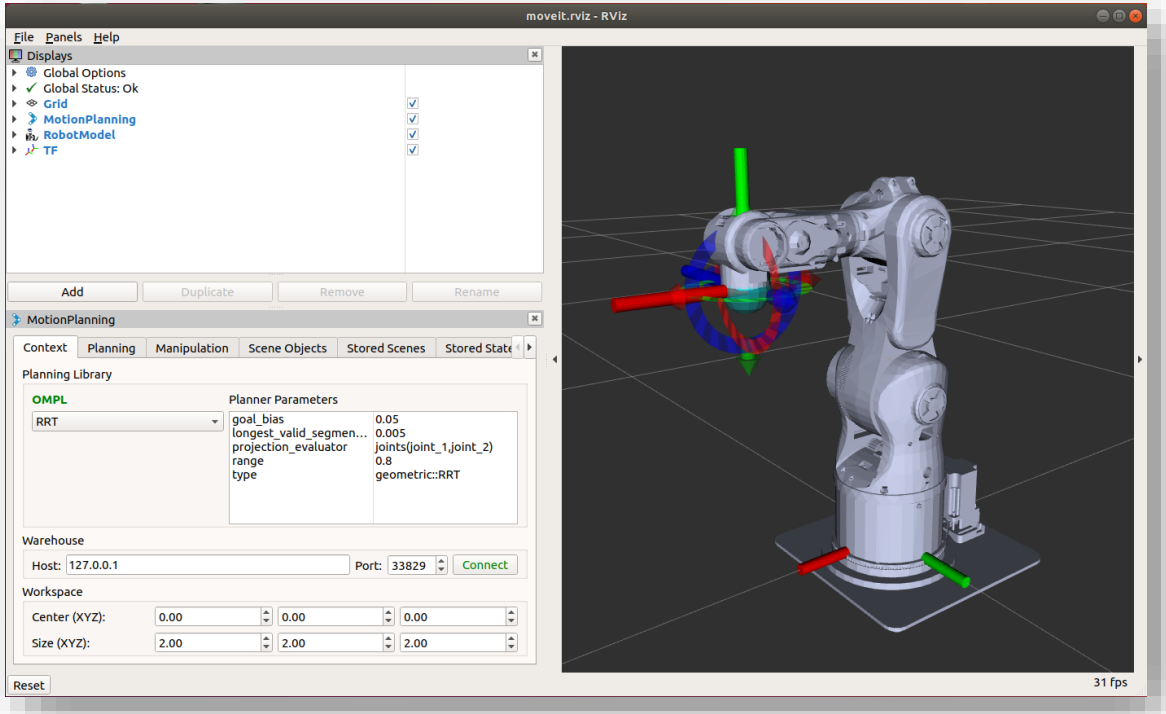
probot_anno_moveit_config /launch/ probot_anno_moveit_controller_manager.launch

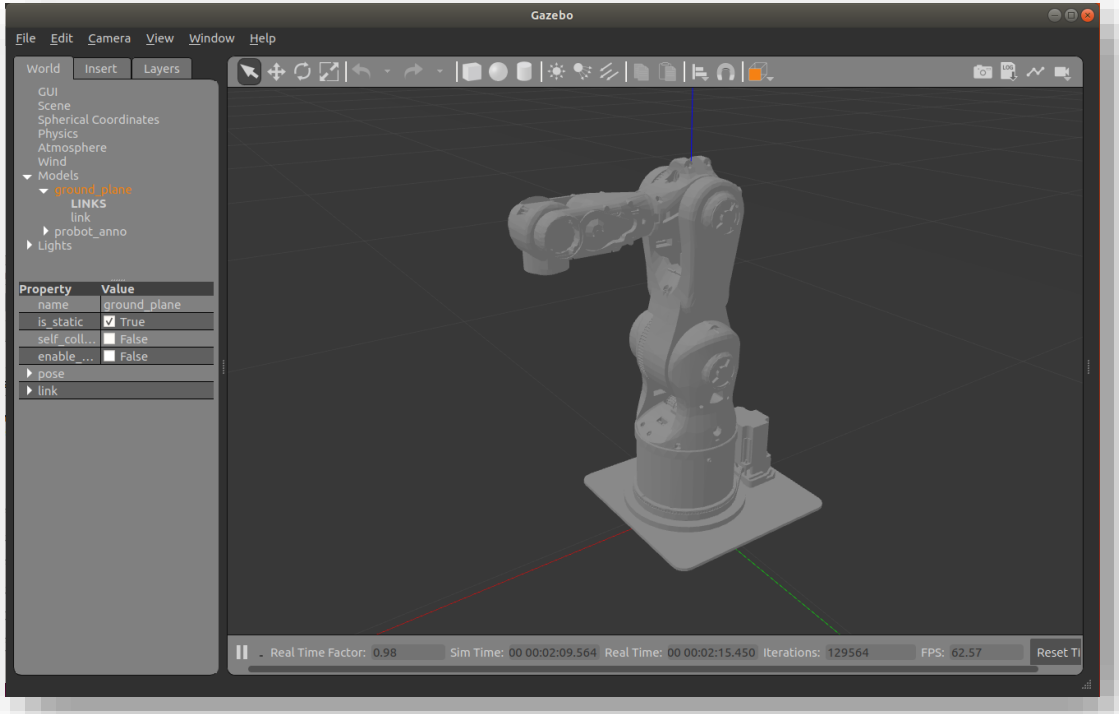
3. 构建MoveIt!+Gazebo仿真

```
<launch>

    <!-- Launch Gazebo  -->
    <include file="$(find probot_gazebo)/launch/probot_anno/probot_anno_gazebo_world.launch" />

    <!-- ros_control arm launch file -->
    <include file="$(find probot_gazebo)/launch/probot_anno/probot_anno_gazebo_states.launch" />

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

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

</launch>
```

probot_gazebo/…/probot_anno_bringup_moveit.launch

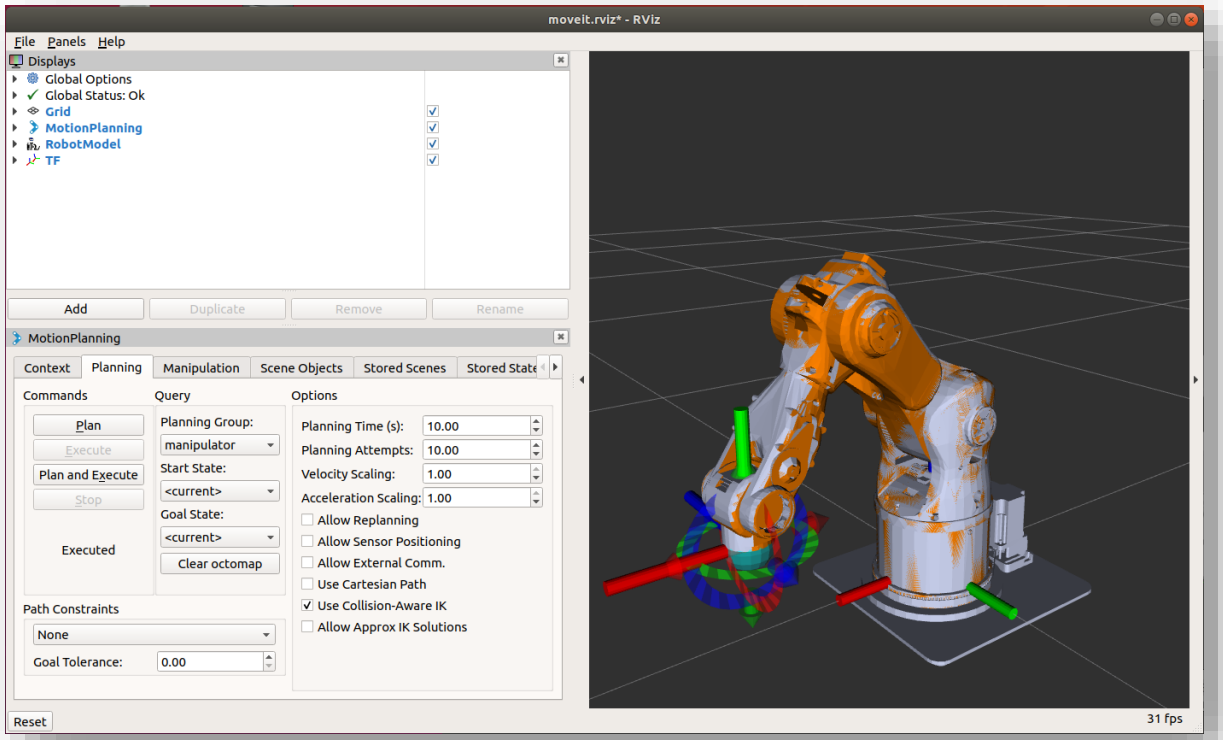
3. 构建MoveIt!+Gazebo仿真

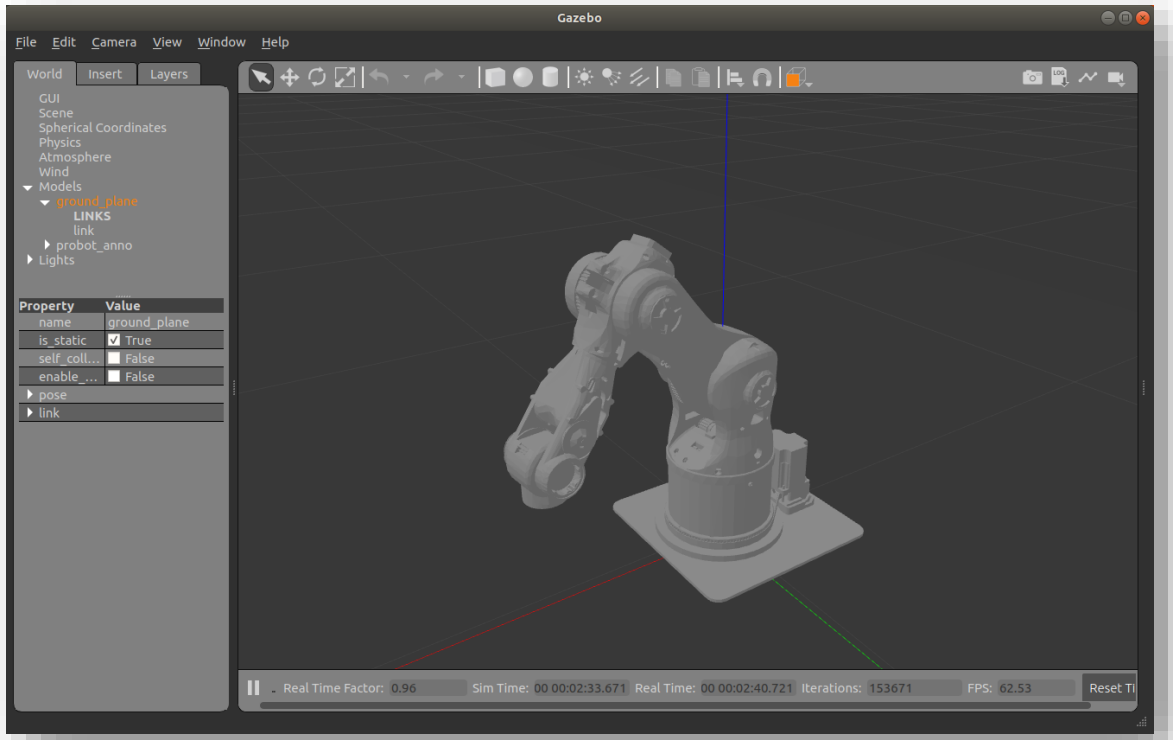




启动仿真系统 $ roslaunch probot_gazebo probot_anno_bringup_moveit.launch

3. 构建MoveIt!+Gazebo仿真





通过MoveIt!控制机械臂运动，gazebo仿真环境和rviz中的机器人状态保持一致

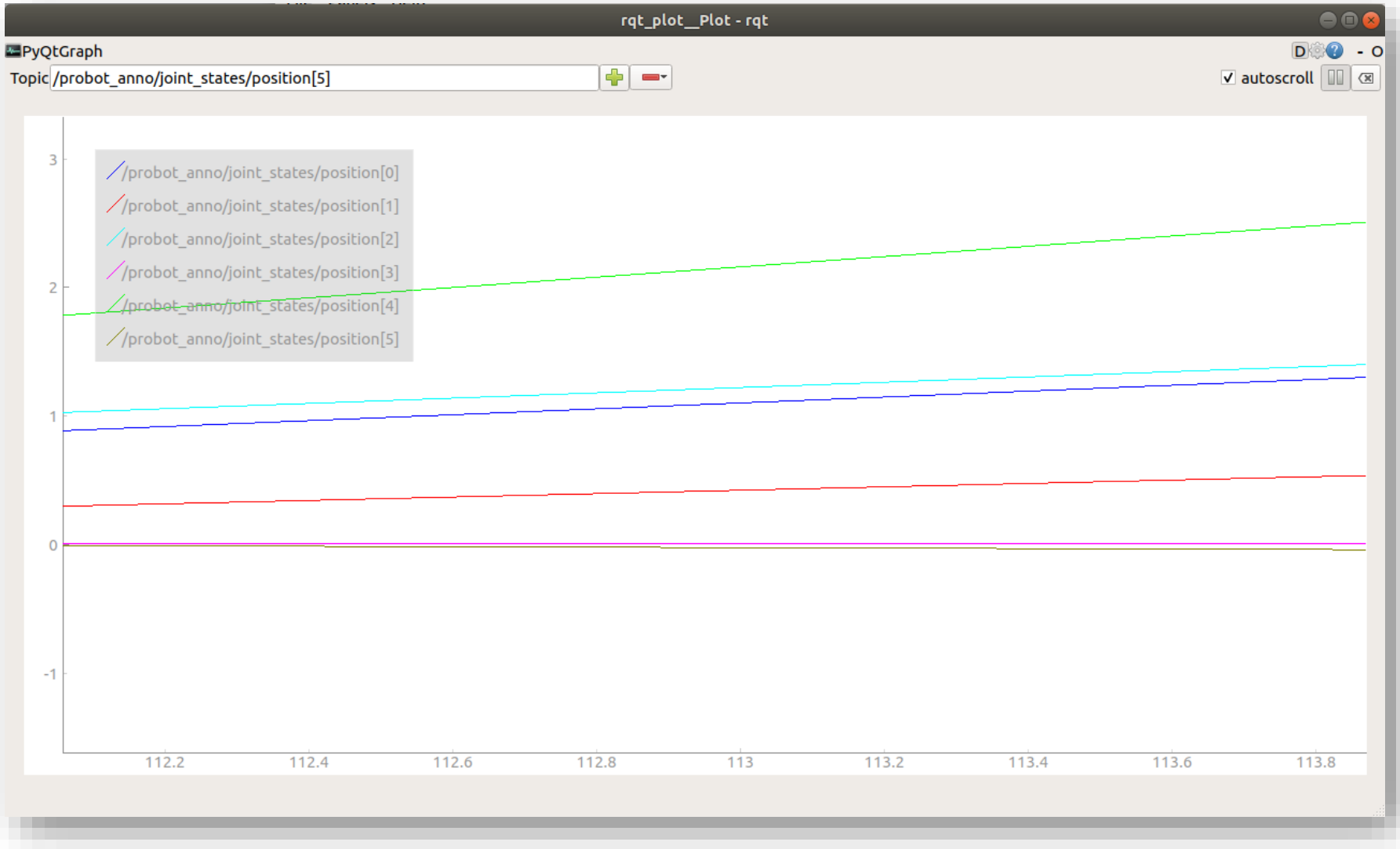
3. 构建MoveIt!+Gazebo仿真

古月居 GYH.AI

Trajectory 数据示例

```
➜ ~  rostopic echo /probot_anno/arm_joint_controller/follow_joint_trajectory/goal
WARNING: no messages received and simulated time is active.
Is /clock being published?
header:
  seq: 2
  stamp:
    secs: 98
    nsecs: 401000000
  frame_id: ''
goal_id:
  stamp:
    secs: 98
    nsecs: 401000000
  id: "/move_group-3-98.401000000"
goal:
  trajectory:
    header:
      seq: 0
      stamp:
        secs: 0
        nsecs:         0
      frame_id: "base_footprint"
    joint_names: [joint_1, joint_2, joint_3, joint_4, joint_5, joint_6]
    points:
      -
        positions: [0.02073119376275212, -0.6294906079865967, -0.058876006393081326, 1.1515114310967078e-06, 0.6884366361530443, 0.020726704731801604]
        velocities: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
        accelerations: [1.0014907029399591, 0.0, 0.0, 0.0, 0.0, 0.0]
        effort: []
        time_from_start:
          secs: 0
          nsecs:         0
      -
        positions: [0.08690572796471953, -0.6289783023588708, -0.07106899129458027, 1.735993976618141e-07, 0.7001102586859531, 0.08690469668490647]
        velocities: [0.31139514179373595, 0.002410738292475822, -0.0573760935090687, -4.601725724035484e-06, 0.05493214856317976, 0.3114114128097502]
        accelerations: [1.0073513873703999, 0.00779864627791953, -0.18560947054382337, -1.4886406915993996e-05, 0.17770340201071552, 1.0074040234855988]
        effort: []
        time_from_start:
          secs: 0
          nsecs: 363527260
      -
        positions: [0.15308026216668694, -0.6284659967311449, -0.08326197619607922, -8.043126357730796e-07, 0.7117838812188619, 0.15308268863801133]
        velocities: [0.5022862030508732, 0.003888566072360782, -0.09254871475692918, -7.422669885255769e-06, 0.08860658572991609, 0.5023124485048213]
        accelerations: [0.9200558928206964, 0.007122827797718587, -0.16952484433754286, -1.3596374192465343e-05, 0.16230390332914352, 0.9201039675725379]
        effort: []
        time_from_start:
          secs: 0
          nsecs: 513666066
```

3. 构建MoveIt!+Gazebo仿真



可视化显示各轴的运动状态

本讲小结

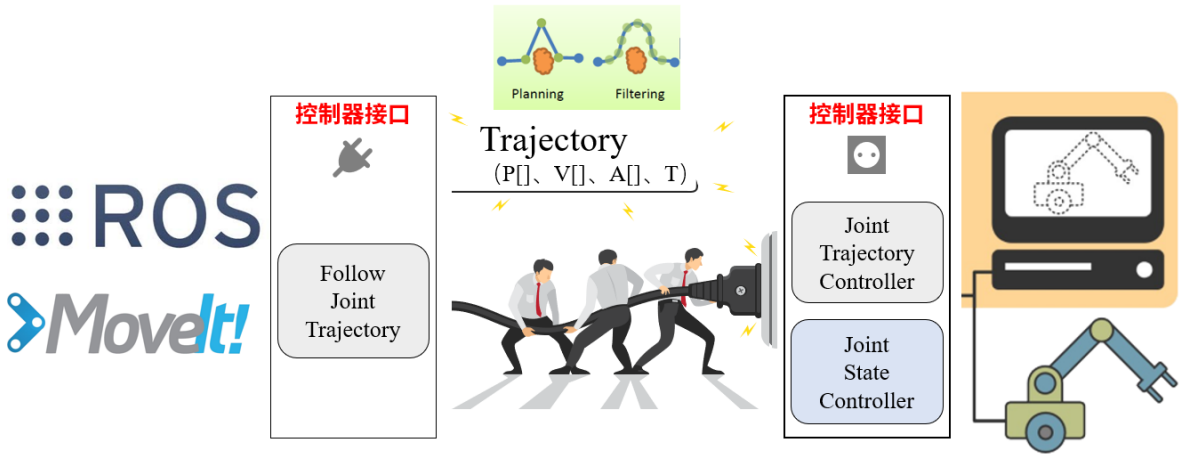
ROS中的控制器插件

- ROS为开发者提供的机器人控制中间件
- 包含一系列控制器接口、传动装置接口、硬件接口、控制器工具箱等等
- 可以帮助机器人应用功能包快速落地，提高开发效率

完善机器人模型

- 为link添加惯性参数和碰撞属性
- 为joint添加传动装置
- 添加gazebo控制器插件

构建MoveIt!+Gazebo仿真



本讲作业

1. 使用任意机器人模型(可用之前完成的URDF模型)，完成MoveIt!+Gazebo仿真中涉及的所有参数配置文件及启动文件；
2. 测试配置完成的仿真系统，可通过MoveIt!控制Gazebo中的机械臂运动，机器人状态保持一致。

- Gazebo Tutorials
http://www.gazebosim.org/tutorials

- gazebo和rviz有具体的区别吗？哪个更好用？
https://mp.weixin.qq.com/s/i3zoCz0PxzJgHcBGoJEBjg

- ROS技术点滴 —— ros_control
https://mp.weixin.qq.com/s/VqUCgNSrj5d-tEQgB44Y8Q

- ROS探索总结（二十四）—— 使用gazebo中的插件
http://www.guyuehome.com/388

Thank You

怕什么真理无穷，进一寸有一寸的欢喜

更多精彩，欢迎关注

古月居

古月春旭