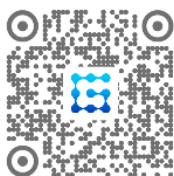


ROS机械臂开发：从入门到实战

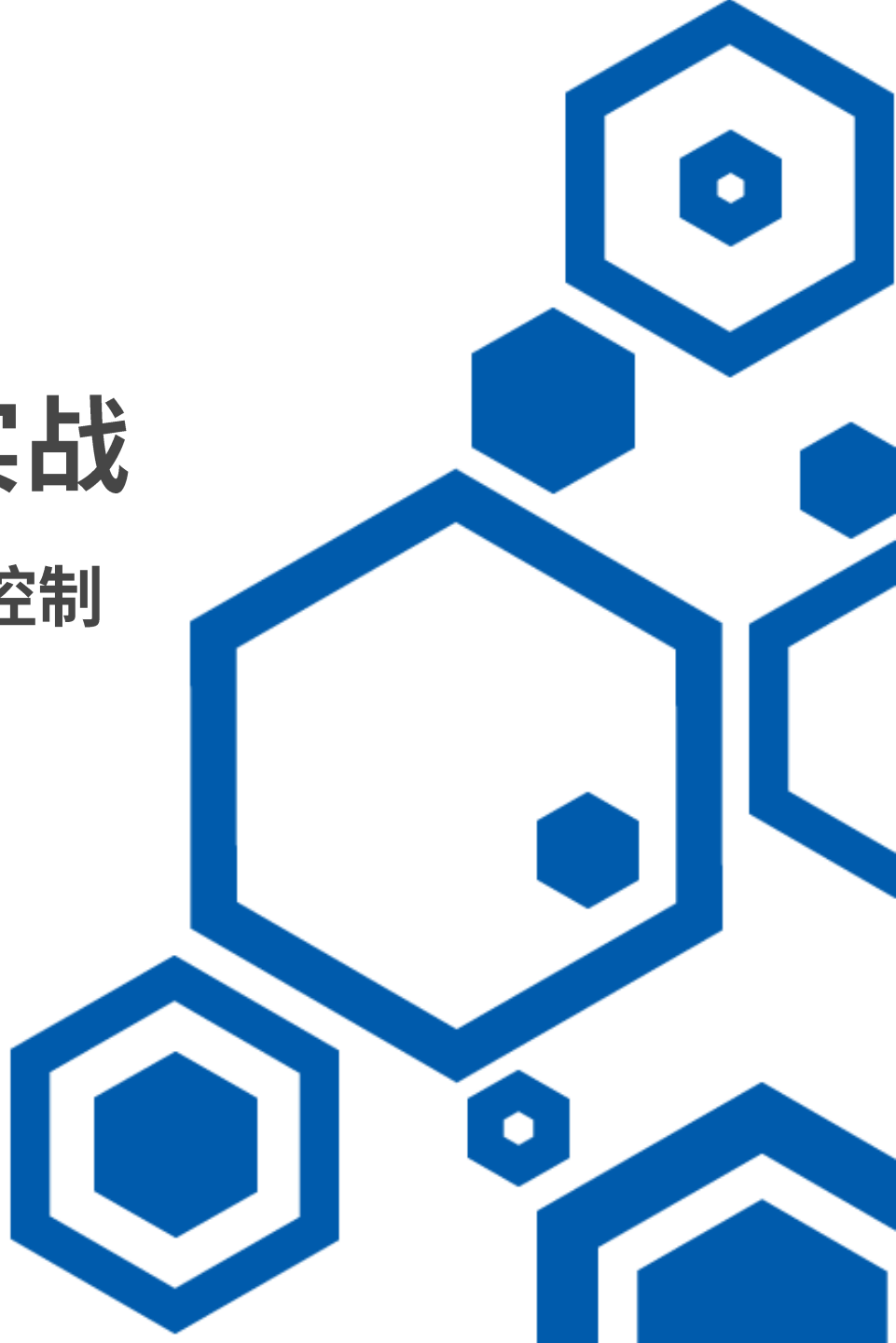
—— 第6讲：MoveIt!编程驾驭机械臂运动控制



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



 1. MoveIt!的编程接口

 2. 关节空间运动

 3. 笛卡尔空间运动

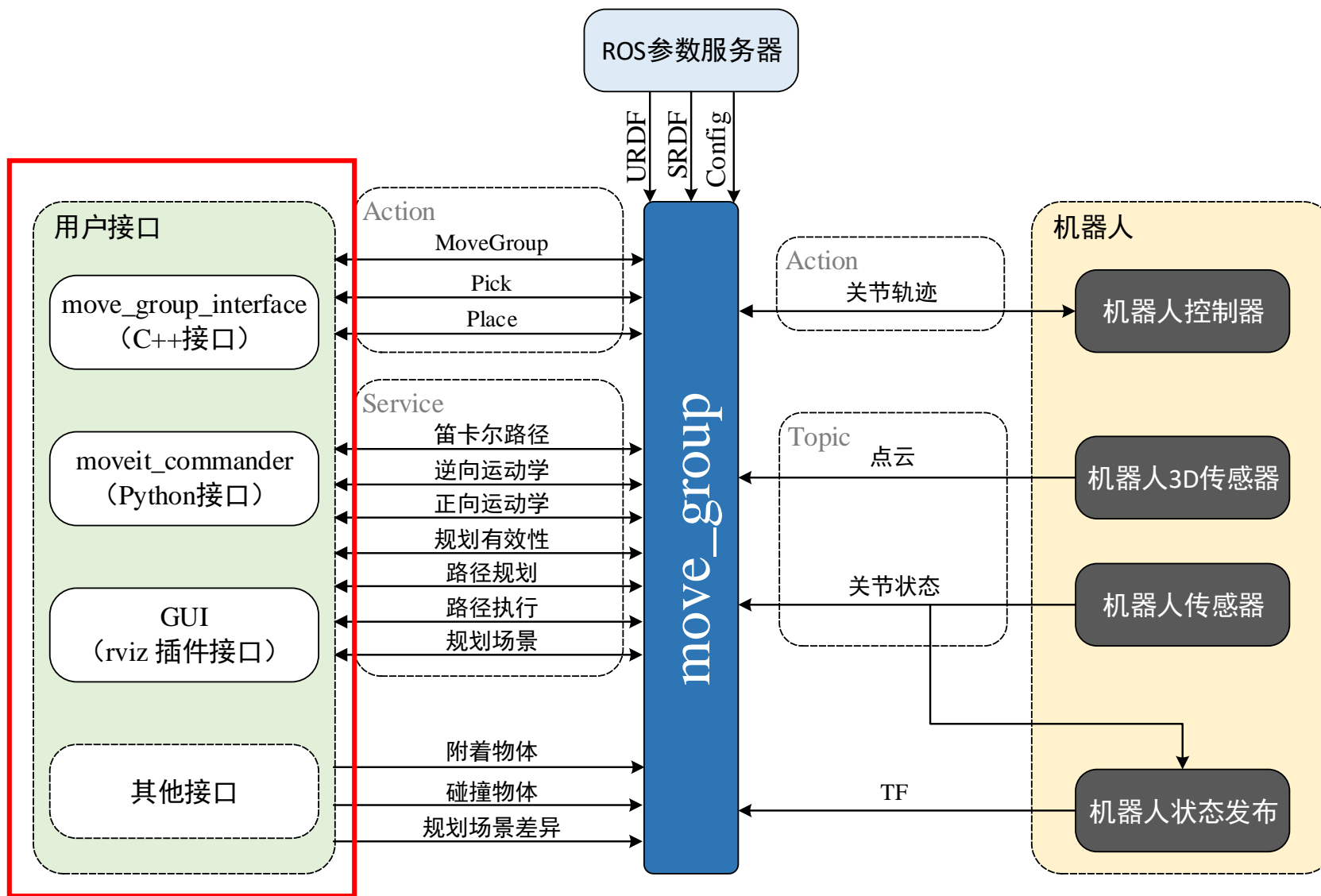
 4. 自主避障运动



1. MoveIt!的编程接口



1. MoveIt!的编程接口



MoveIt!的核心节点——move_group



1. MoveIt!的编程接口

"move_group" Python Interface

```
group = moveit_commander.MoveGroupCommander("left_arm")

pose_target = geometry_msgs.msg.Pose()
pose_target.orientation.w = 1.0
pose_target.position.x = 0.7
pose_target.position.y = -0.05
pose_target.position.z = 1.1
group.set_pose_target(pose_target)

plan1 = group.plan()
```

Python

"move_group" C++ Interface

```
moveit::planning_interface::MoveGroup group("right_arm");

geometry_msgs::Pose target_pose;
target_pose.orientation.w = 1.0;
target_pose.position.x = 0.28;
target_pose.position.y = -0.7;
target_pose.position.z = 1.0;
group.setPoseTarget(target_pose);

moveit::planning_interface::MoveGroup::Plan my_plan;
bool success = group.plan(my_plan);
```

C++

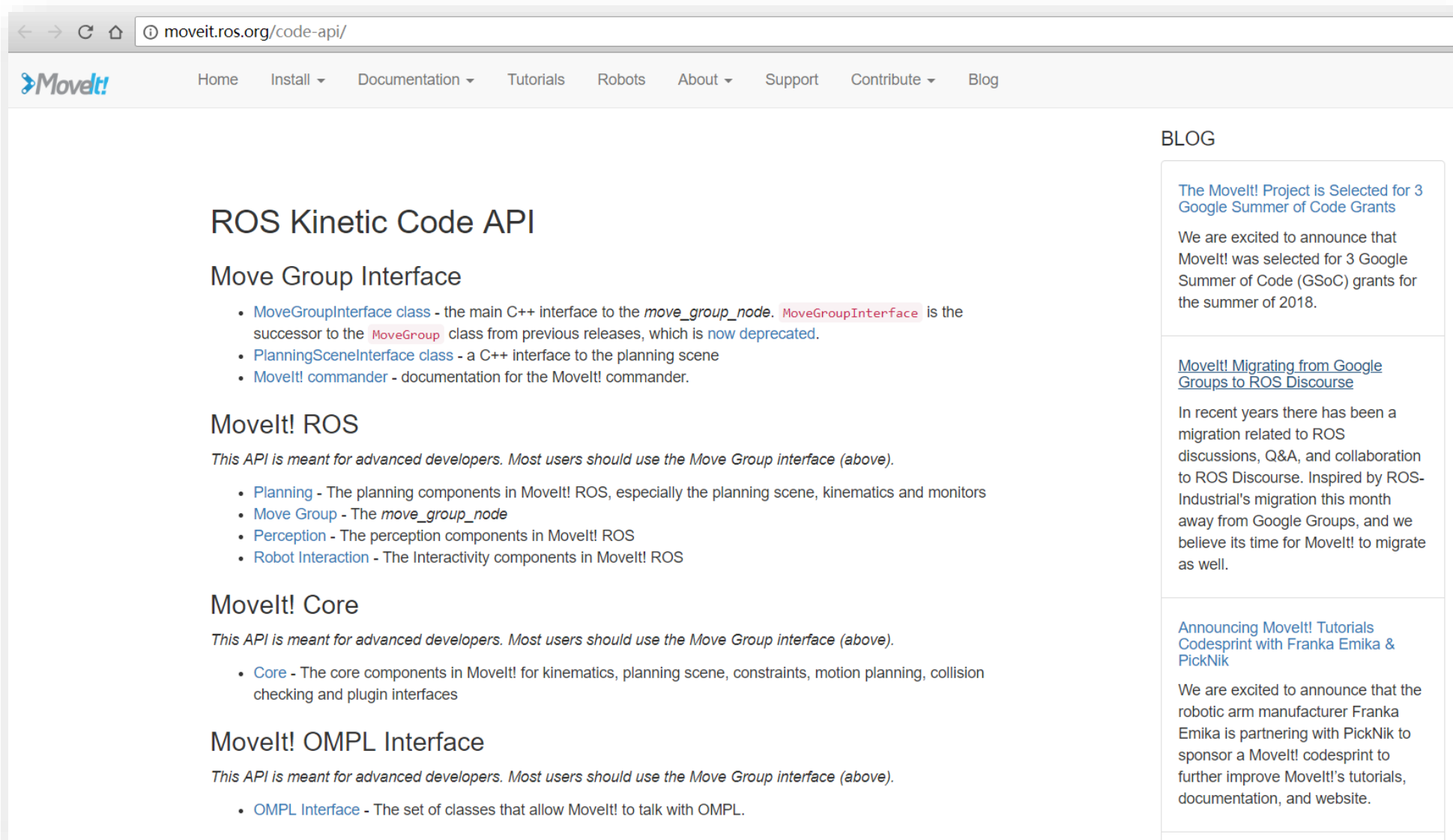


1. MoveIt!的编程接口

- 连接控制需要的规划组
- 设置目标位姿（关节空间或笛卡尔空间）
- 设置运动约束（可选）
- 使用MoveIt!规划一条到达目标的轨迹
- 修改轨迹（如速度等参数）
- 执行规划出的轨迹



1. MoveIt!的编程接口



The screenshot shows the MoveIt! ROS Kinetic Code API website. The browser address bar displays `moveit.ros.org/code-api/`. The website has a navigation bar with links: Home, Install, Documentation, Tutorials, Robots, About, Support, Contribute, and Blog. The main content area is titled "ROS Kinetic Code API" and "Move Group Interface". It lists three main components: `MoveGroupInterface` class, `PlanningSceneInterface` class, and `MoveIt! commander`. Below this, there are sections for "MoveIt! ROS" and "MoveIt! Core", each with a note that the API is for advanced developers and a list of components. The "MoveIt! OMPL Interface" section is also present. On the right side, there is a "BLOG" section with three articles: "The MoveIt! Project is Selected for 3 Google Summer of Code Grants", "MoveIt! Migrating from Google Groups to ROS Discourse", and "Announcing MoveIt! Tutorials Codesprint with Franka Emika & PickNik".

moveit.ros.org/code-api/

Home Install Documentation Tutorials Robots About Support Contribute Blog

ROS Kinetic Code API

Move Group Interface

- [MoveGroupInterface class](#) - the main C++ interface to the `move_group_node`. `MoveGroupInterface` is the successor to the `MoveGroup` class from previous releases, which is [now deprecated](#).
- [PlanningSceneInterface class](#) - a C++ interface to the planning scene
- [MoveIt! commander](#) - documentation for the MoveIt! commander.

MoveIt! ROS

This API is meant for advanced developers. Most users should use the Move Group interface (above).

- [Planning](#) - The planning components in MoveIt! ROS, especially the planning scene, kinematics and monitors
- [Move Group](#) - The `move_group_node`
- [Perception](#) - The perception components in MoveIt! ROS
- [Robot Interaction](#) - The Interactivity components in MoveIt! ROS

MoveIt! Core

This API is meant for advanced developers. Most users should use the Move Group interface (above).

- [Core](#) - The core components in MoveIt! for kinematics, planning scene, constraints, motion planning, collision checking and plugin interfaces

MoveIt! OMPL Interface

This API is meant for advanced developers. Most users should use the Move Group interface (above).

- [OMPL Interface](#) - The set of classes that allow MoveIt! to talk with OMPL.

BLOG

[The MoveIt! Project is Selected for 3 Google Summer of Code Grants](#)

We are excited to announce that MoveIt! was selected for 3 Google Summer of Code (GSoC) grants for the summer of 2018.

[MoveIt! Migrating from Google Groups to ROS Discourse](#)

In recent years there has been a migration related to ROS discussions, Q&A, and collaboration to ROS Discourse. Inspired by ROS-Industrial's migration this month away from Google Groups, and we believe its time for MoveIt! to migrate as well.

[Announcing MoveIt! Tutorials Codesprint with Franka Emika & PickNik](#)

We are excited to announce that the robotic arm manufacturer Franka Emika is partnering with PickNik to sponsor a MoveIt! codesprint to further improve MoveIt!'s tutorials, documentation, and website.

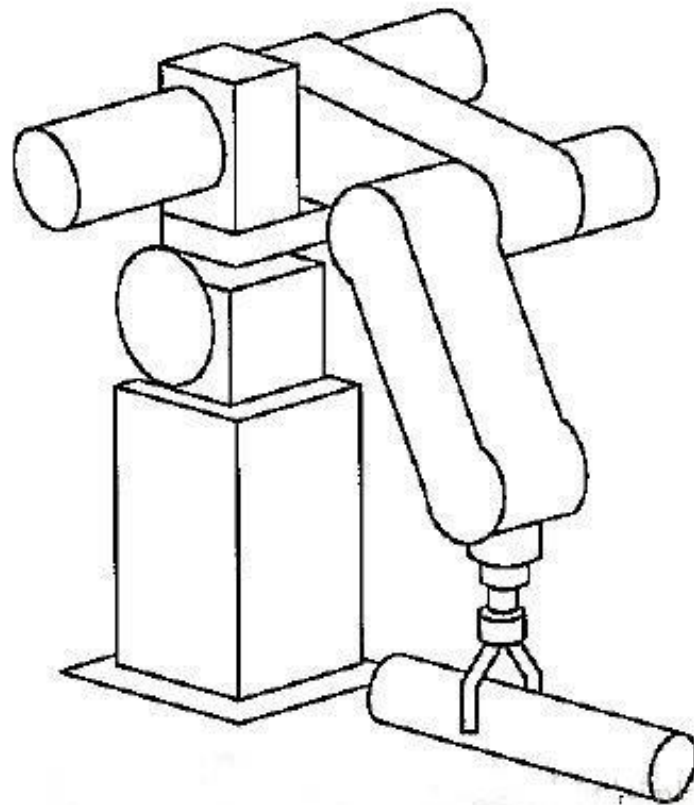
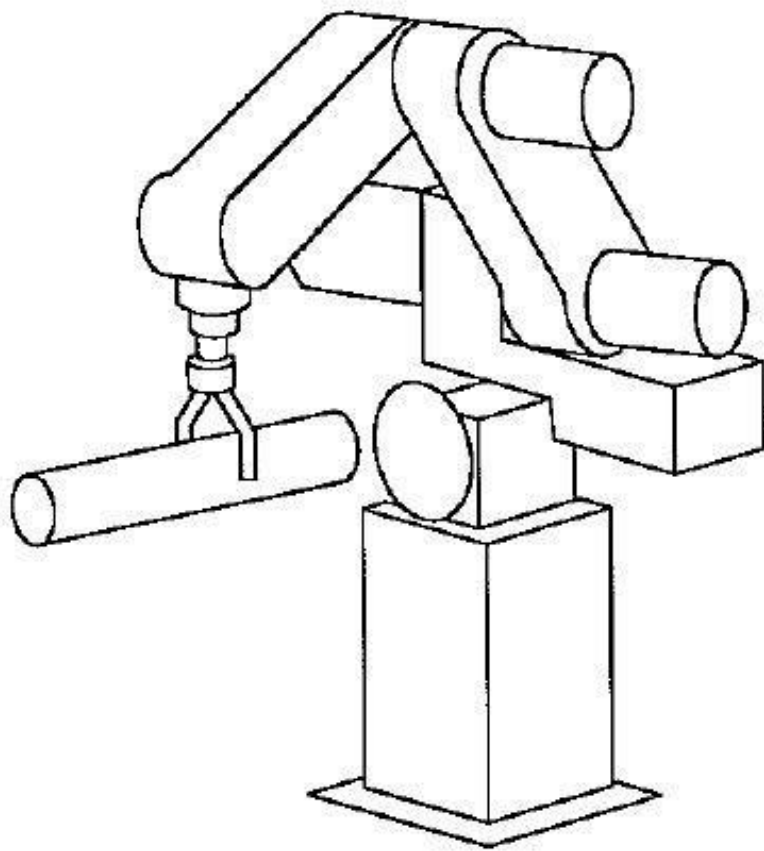
<http://moveit.ros.org/code-api/>



2. 关节空间运动



2. 关节空间运动



点到点运动：不需要在笛卡尔空间规划末端运动轨迹，机器人各个关节运动不需要联动。



2. 关节空间运动

```
# 初始化需要使用move group控制的机械臂中的arm group
arm = moveit_commander.MoveGroupCommander('manipulator')

# 设置机械臂运动的允许误差值
arm.set_goal_joint_tolerance(0.001)

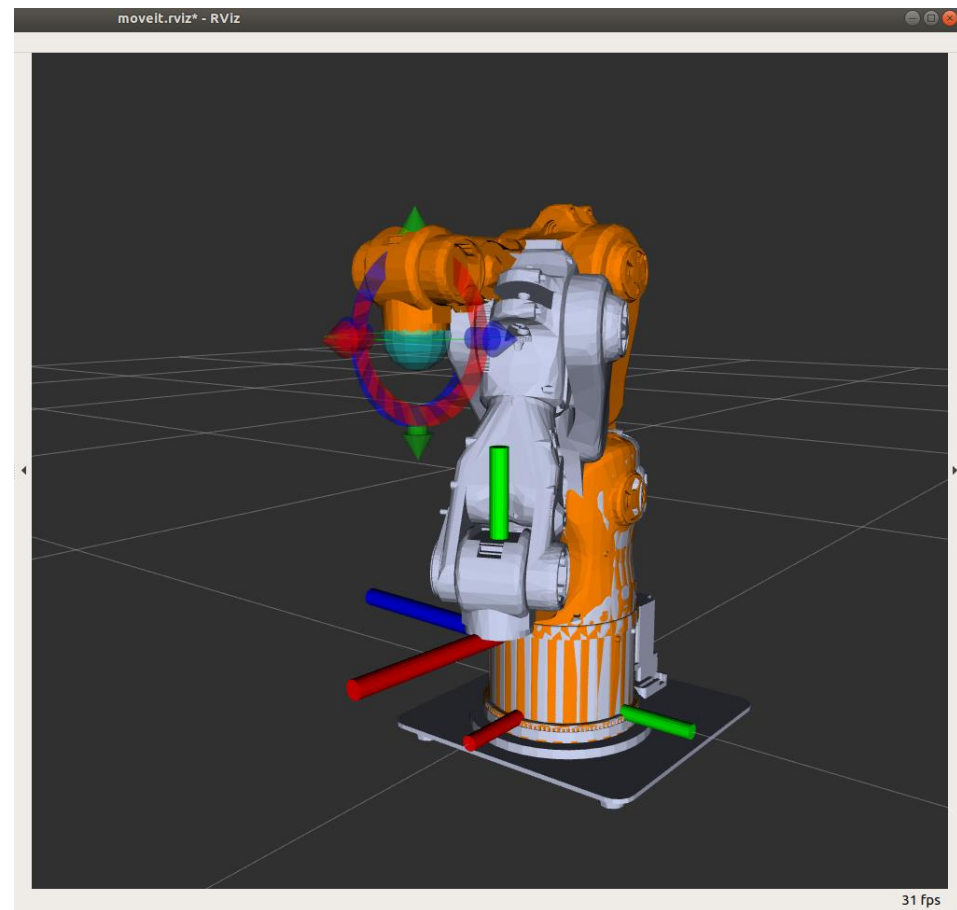
# 设置允许的最大速度和加速度
arm.set_max_acceleration_scaling_factor(0.5)
arm.set_max_velocity_scaling_factor(0.5)

# 控制机械臂先回到初始化位置
arm.set_named_target('home')
arm.go()
rospy.sleep(1)

# 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 = [0.391410, -0.676384, -0.376217, 0.0, 1.052834, 0.454125]
arm.set_joint_value_target(joint_positions)

# 控制机械臂完成运动
arm.go()
rospy.sleep(1)

# 控制机械臂先回到初始化位置
arm.set_named_target('home')
arm.go()
rospy.sleep(1)
```



正向运动学
规划例程

```
$ roslaunch probot_anno_moveit_config demo.launch
$ rosrn probot_demo moveit_fk_demo.py
```



关键API的实现步骤

```
arm = moveit_commander.MoveGroupCommander('manipulator')  
  
joint_positions = [0.391410, -0.676384, -0.376217, 0.0, 1.052834, 0.454125]  
  
arm.set_joint_value_target(joint_positions)  
  
arm.go()
```

- 创建规划组的控制对象；
- 设置关节空间运动的目标位姿；
- 完成规划并控制机械臂完成运动。



2. 关节空间运动

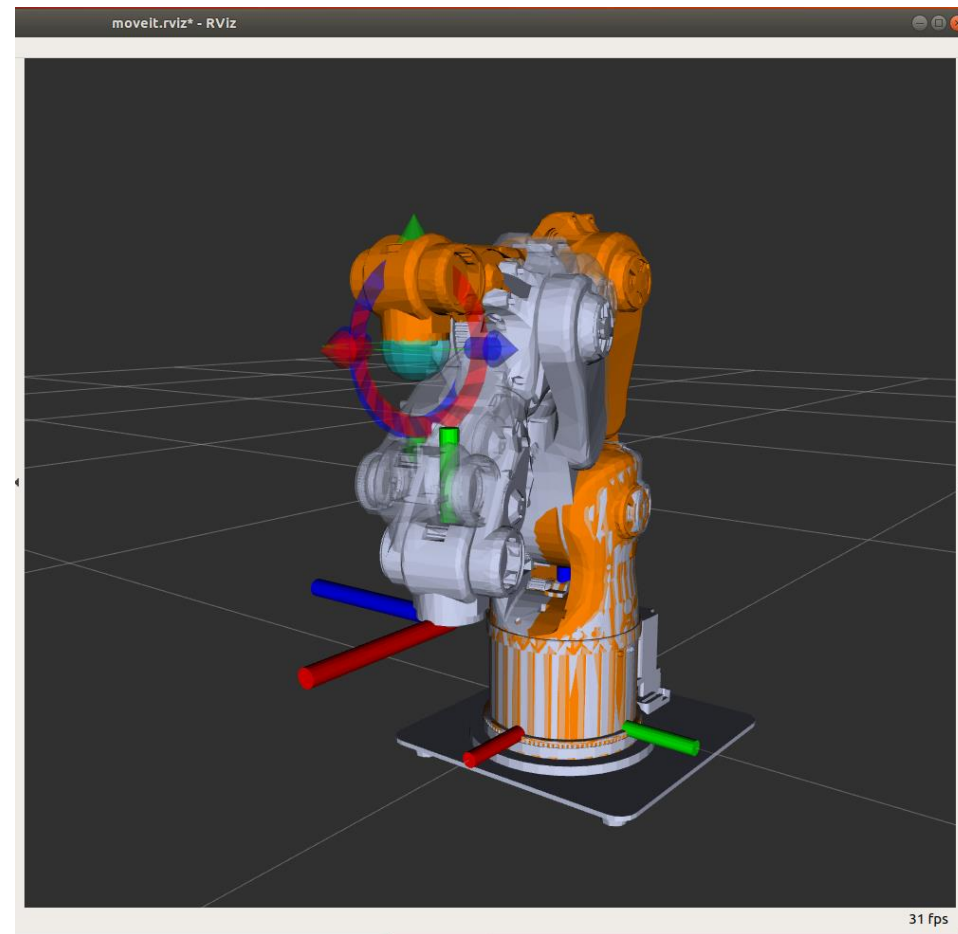
```
# 设置机械臂工作空间中的目标位姿，位置使用x、y、z坐标描述，
# 姿态使用四元数描述，基于base_link坐标系
target_pose = PoseStamped()
target_pose.header.frame_id = reference_frame
target_pose.header.stamp = rospy.Time.now()
target_pose.pose.position.x = 0.2593
target_pose.pose.position.y = 0.0636
target_pose.pose.position.z = 0.1787
target_pose.pose.orientation.x = 0.70692
target_pose.pose.orientation.y = 0.0
target_pose.pose.orientation.z = 0.0
target_pose.pose.orientation.w = 0.70729

# 设置机械臂当前的状态作为运动初始状态
arm.set_start_state_to_current_state()

# 设置机械臂终端运动的目标位姿
arm.set_pose_target(target_pose, end_effector_link)

# 规划运动路径
traj = arm.plan()

# 按照规划的运动路径控制机械臂运动
arm.execute(traj)
rospy.sleep(1)
```



**逆向运动学
规划例程**

```
$ roslaunch probot_anno_moveit_config demo.launch
```

```
$ rosrun probot_demo moveit_ik_demo.py
```



关键API的实现步骤

```
arm = moveit_commander.MoveGroupCommander('manipulator')
end_effector_link = arm.get_end_effector_link()

reference_frame = 'base_link'
arm.set_pose_reference_frame(reference_frame)

arm.set_pose_target(target_pose, end_effector_link)

traj = arm.plan()
arm.execute(traj)
```

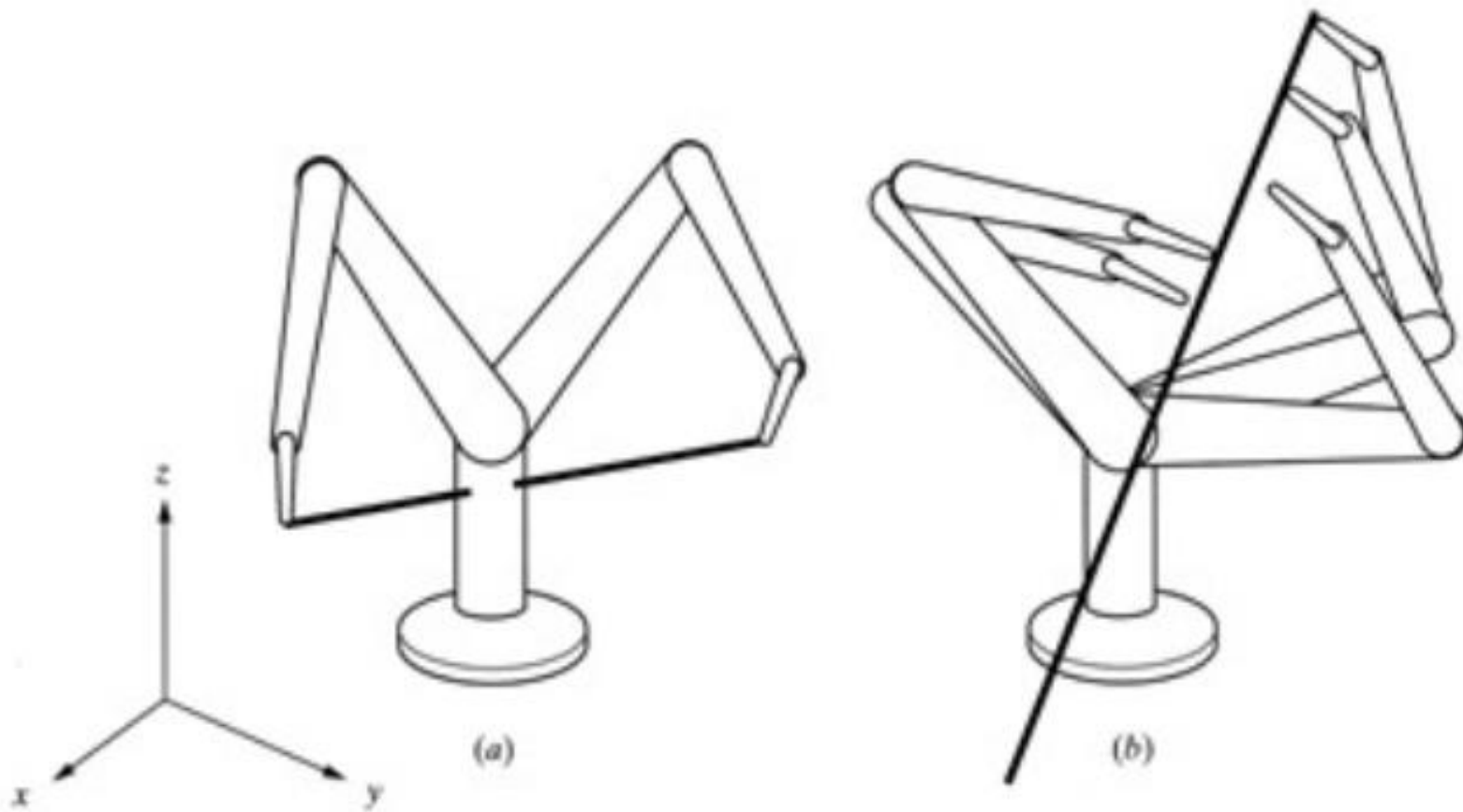
- 创建规划组的控制对象；
- 获取机器人的终端link名称；
- 设置目标位姿对应的参考坐标系和起始、终止位姿；
- 完成规划并控制机械臂完成运动。



3. 笛卡尔空间运动



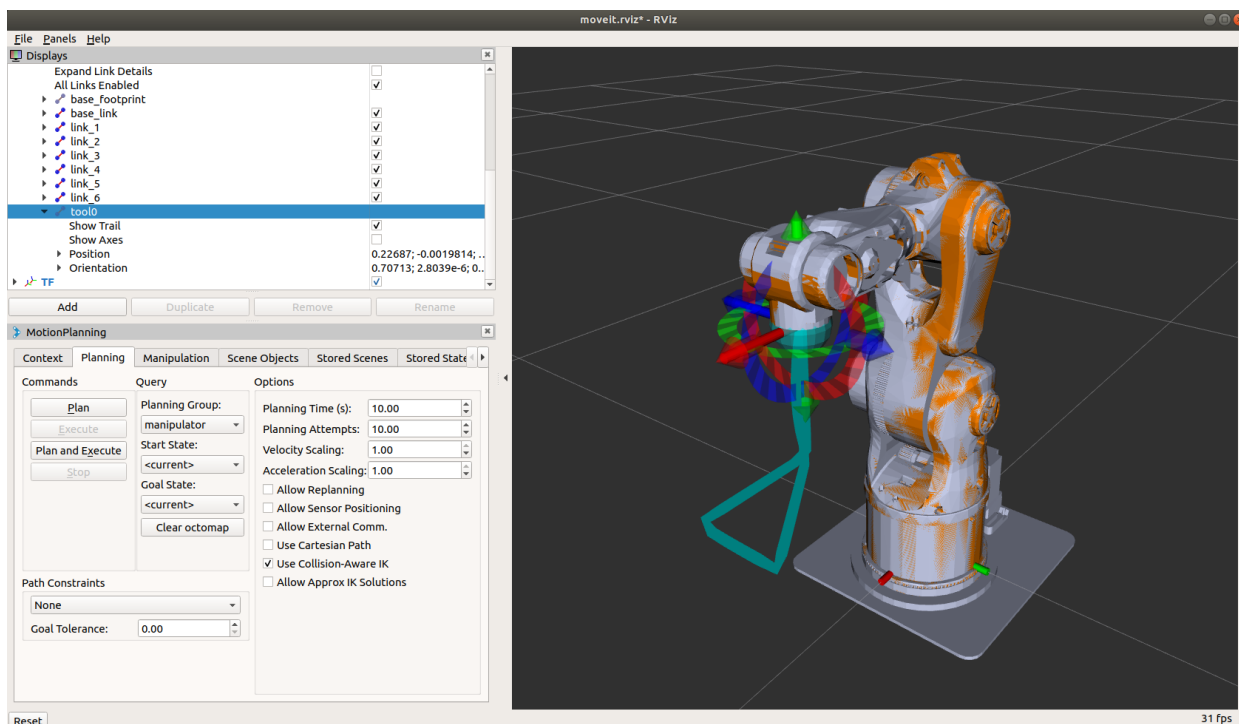
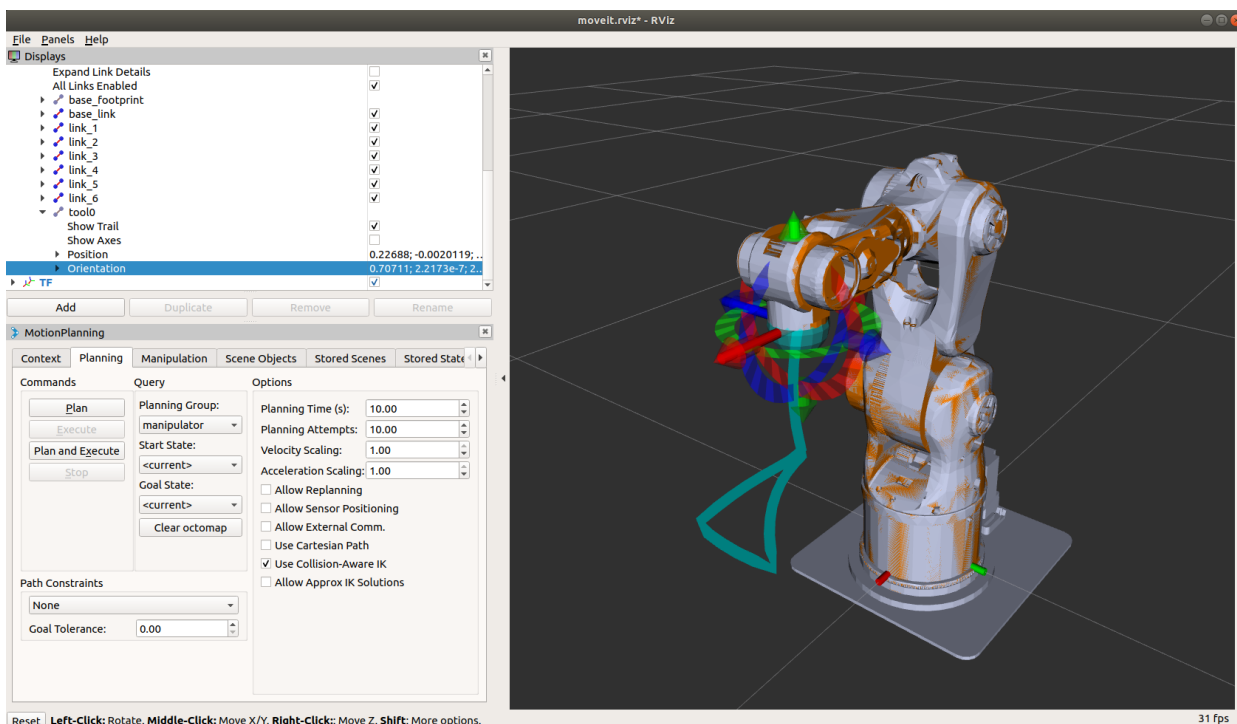
3. 笛卡尔空间运动



笛卡尔路径约束，路径点之间的路径形状是一条直线



3. 笛卡尔空间运动



工作空间 规划例题

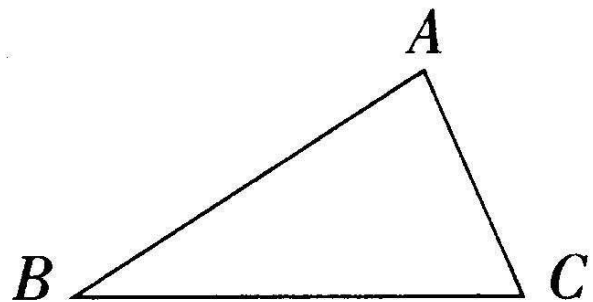
```
$ roslaunch probot_anno_moveit_config demo.launch
```

```
$ rosrun probot_demo moveit_cartesian_demo.py _cartesian:=True (走直线)
```

```
$ rosrun probot_demo moveit_cartesian_demo.py _cartesian:=False (自由曲线)
```




关键API的实现步骤



```
(plan, fraction) = arm.compute_cartesian_path (  
    waypoints,      # waypoint poses, 路点列表  
    0.01,           # eef_step, 终端步进值  
    0.0,            # jump_threshold, 最小移动值  
    True)           # avoid_collisions, 避障规划
```

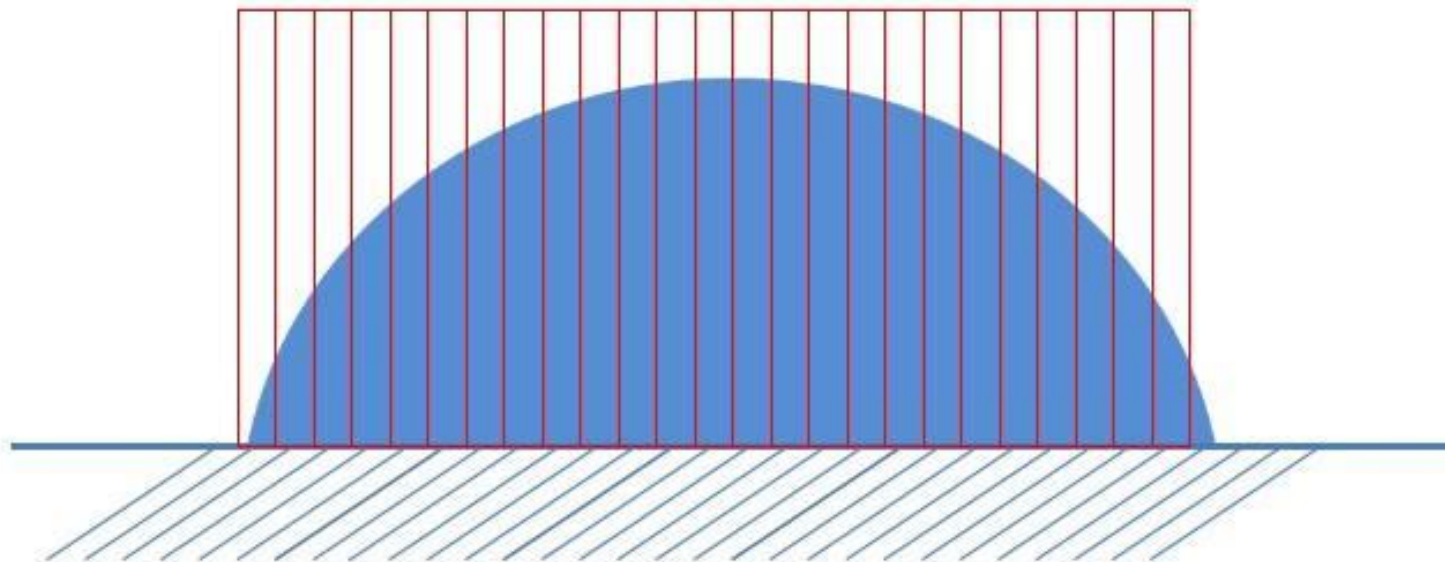
返回值

- **plan**: 规划出来的运动轨迹
- **fraction**: 描述规划成功的轨迹在给定路点列表中的覆盖率[0~1]。如果fraction小于1，说明给定的路点列表没办法完整规划。



3. 笛卡尔空间运动

如何走出笛卡尔空间下的圆弧轨迹？

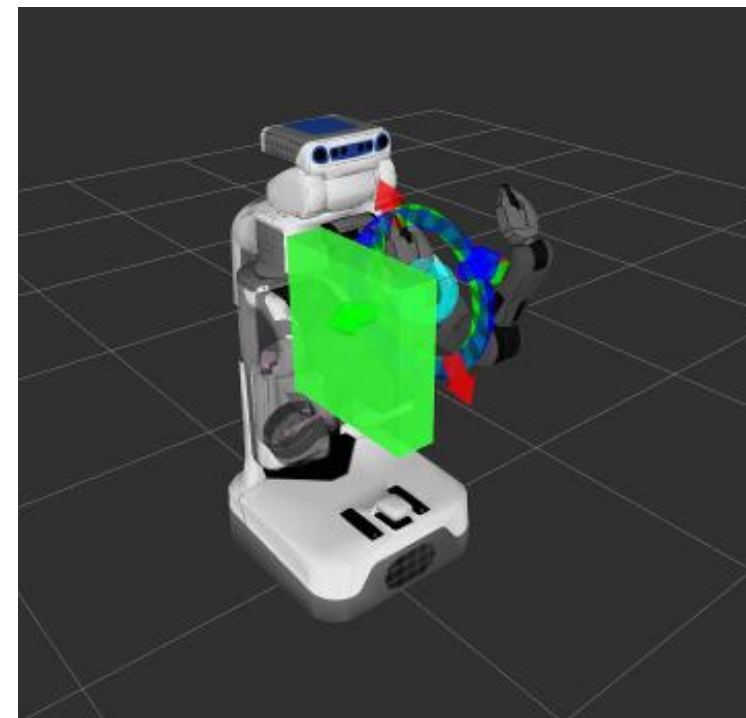
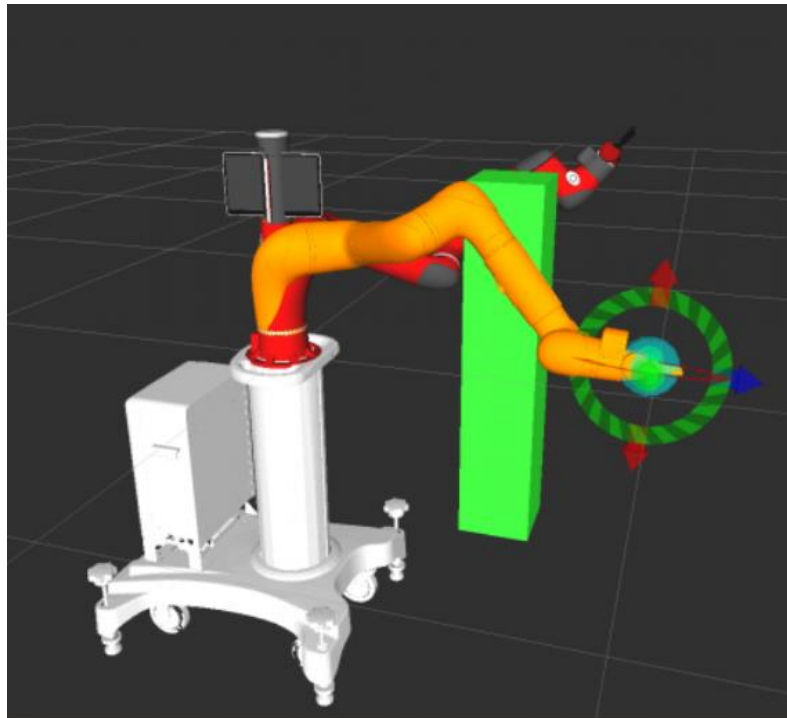
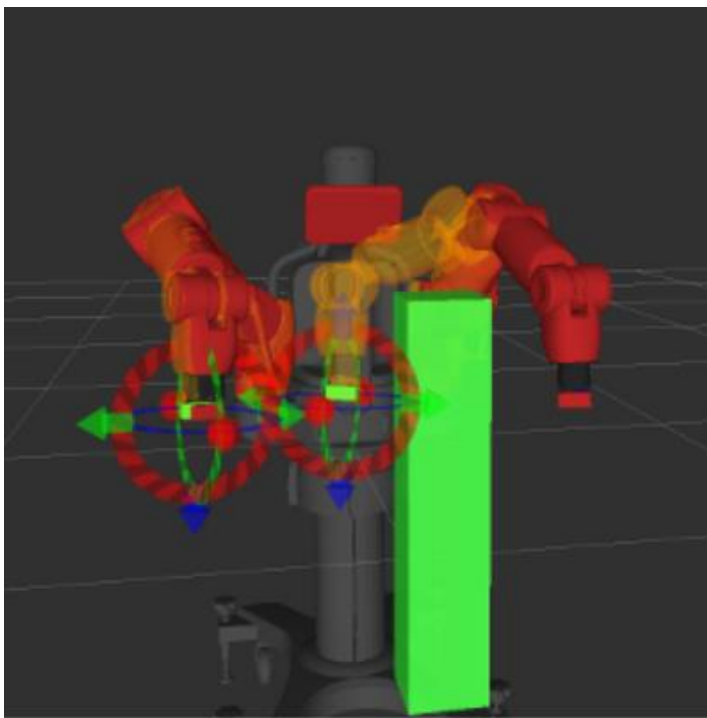




4. 自主避障运动



4. 自主避障运动



MoveIt!可以在运动规划时检测碰撞，并规划轨迹绕过障碍



4. 自主避障运动

➤ 监听信息

- 状态信息

(State Information)

机器人的关节话题joint_states;

- 传感器信息

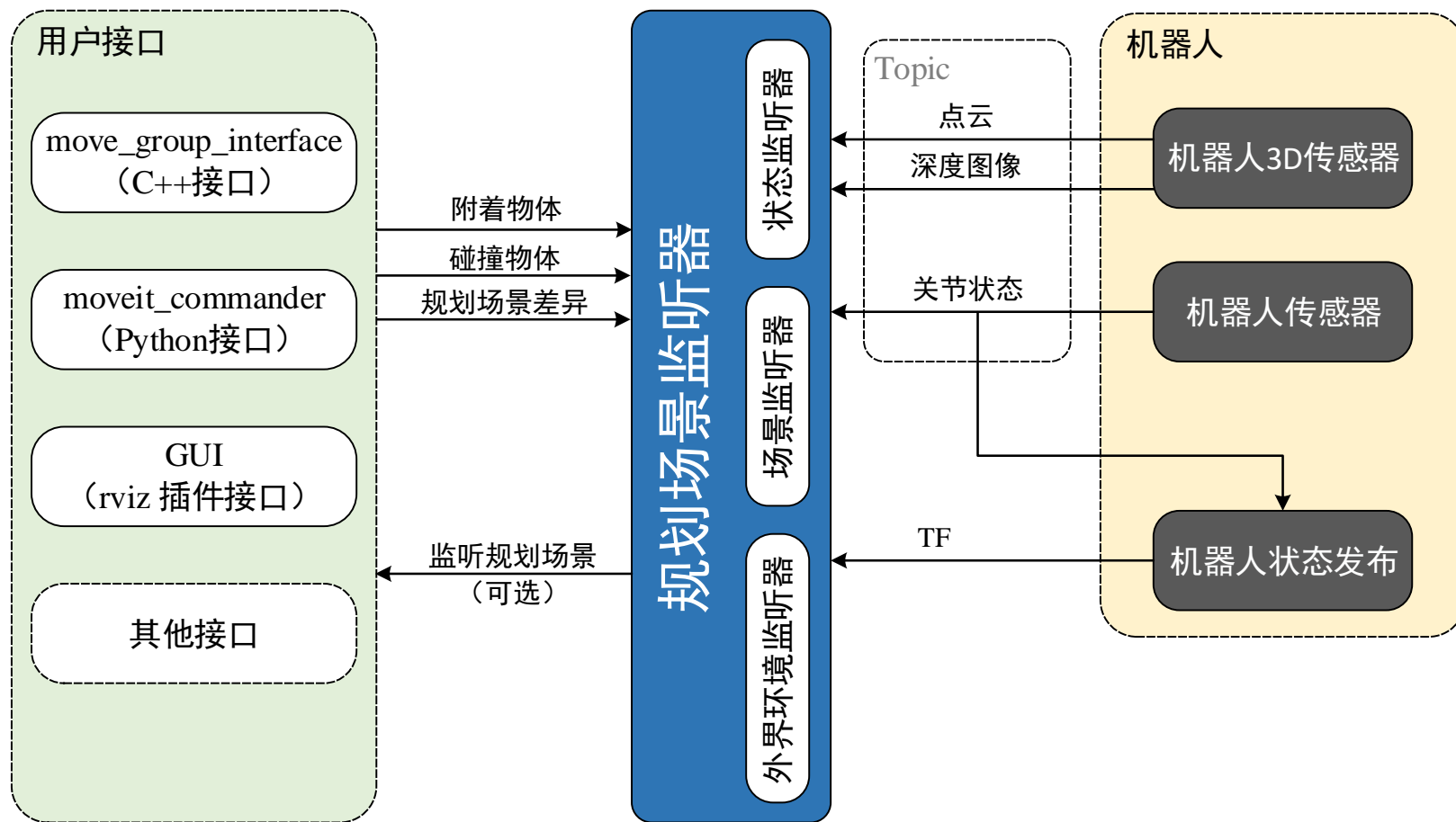
(Sensor Information)

机器人的传感器数据;

- 外界环境信息

(World geometry information)

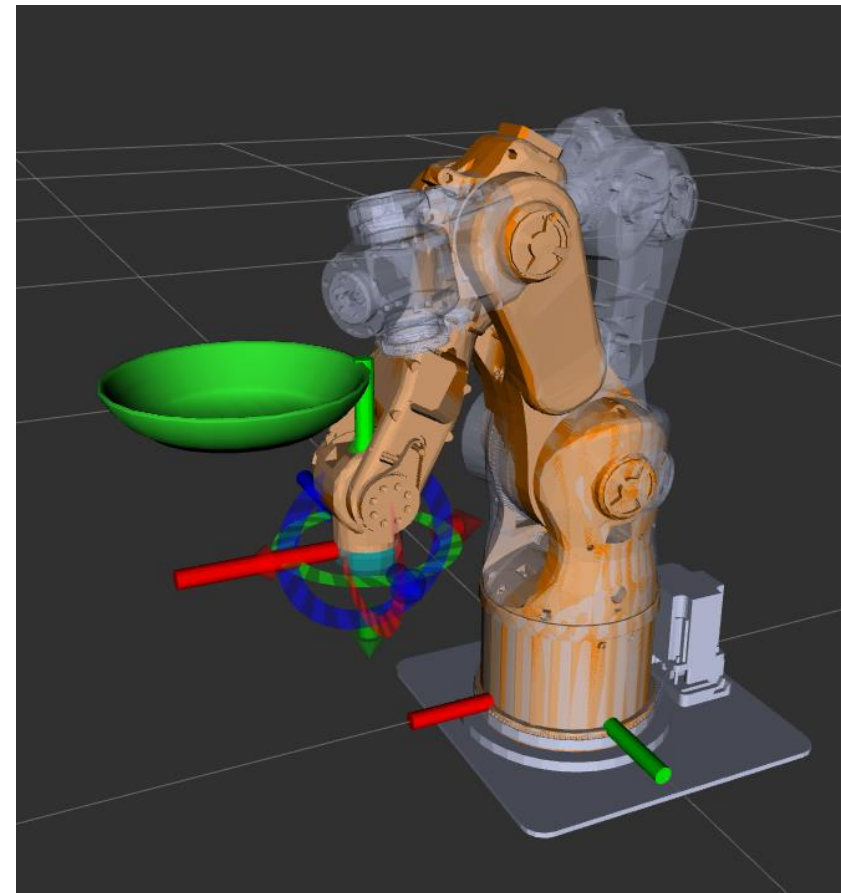
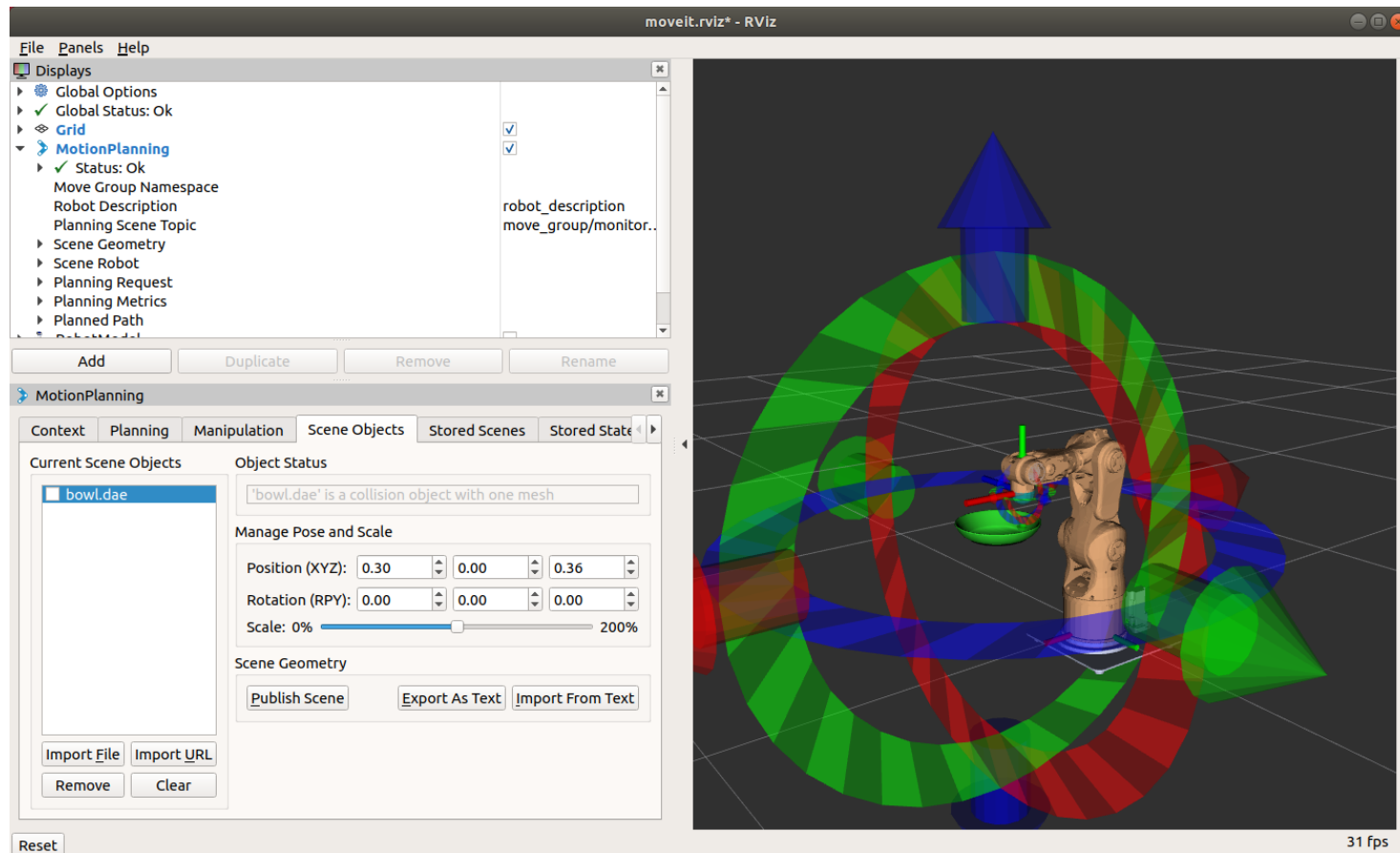
周围环境信息。



规划场景模块的结构



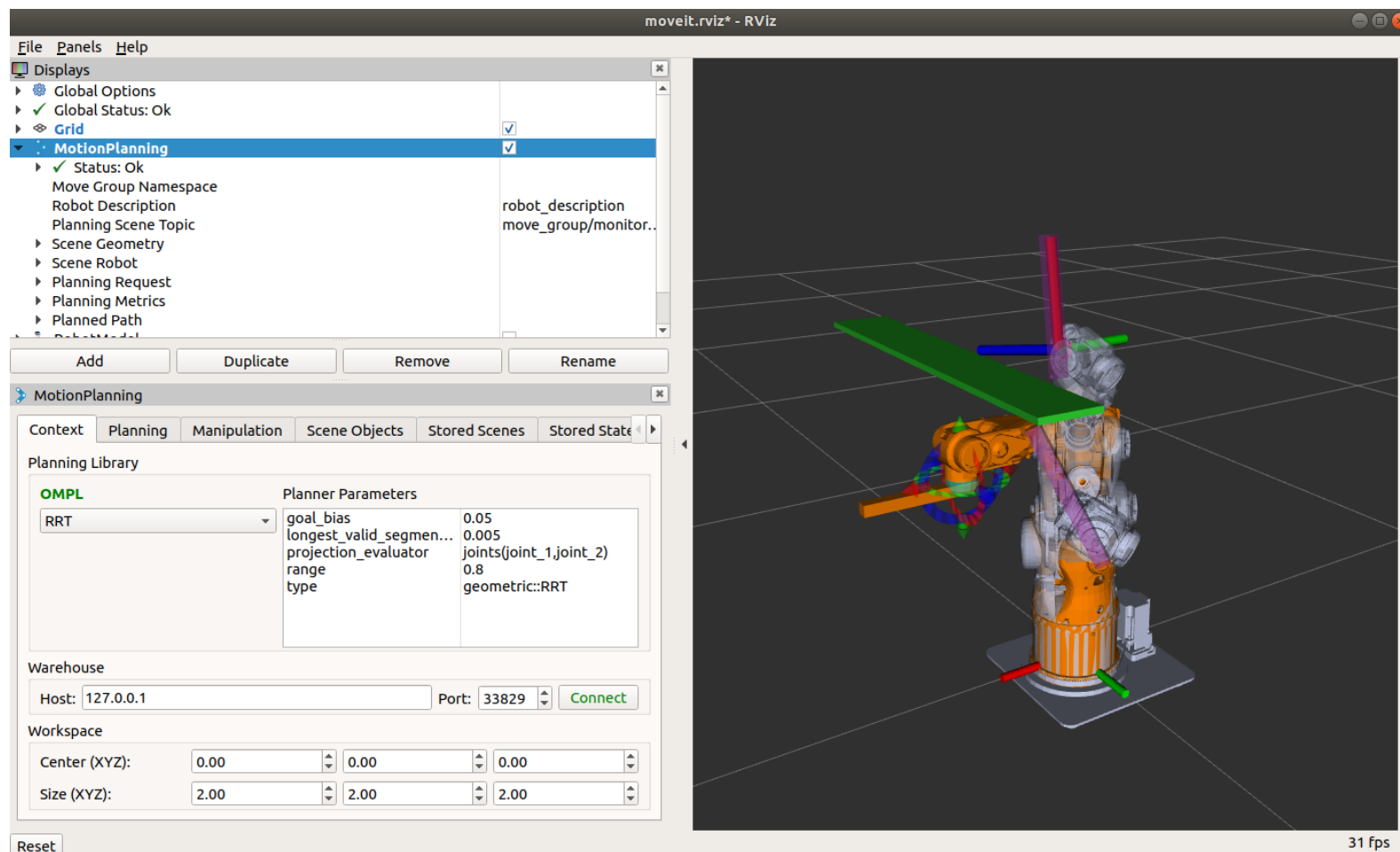
4. 自主避障运动



通过MoveIt!可视化插件添加模型，在运动规划时会考虑碰撞检测



4. 自主避障运动



附着物体
避障例程

```
$ roslaunch probot_anno_moveit_config demo.launch
```

```
$ rosrunc probot_demo moveit_attached_object_demo.py
```



4. 自主避障运动

移除场景中之前运行残留的物体

```
scene.remove_attached_object(end_effector_link, 'tool')
scene.remove_world_object('table')
scene.remove_world_object('target')
```

设置桌面的高度

```
table_ground = 0.6
```

设置table和tool的三维尺寸

```
table_size = [0.1, 0.7, 0.01]
tool_size = [0.2, 0.02, 0.02]
```

设置tool的位姿

```
p = PoseStamped()
p.header.frame_id = end_effector_link
```

```
p.pose.position.x = tool_size[0] / 2.0 - 0.025
p.pose.position.y = -0.015
p.pose.position.z = 0.0
p.pose.orientation.x = 0
p.pose.orientation.y = 0
p.pose.orientation.z = 0
p.pose.orientation.w = 1
```

将tool附着到机器人的终端

```
scene.attach_box(end_effector_link, 'tool', p, tool_size)
```

将tool附着到机器人的终端

```
scene.attach_box(end_effector_link, 'tool', p, tool_size)
```

将table加入场景当中

```
table_pose = PoseStamped()
table_pose.header.frame_id = 'base_link'
table_pose.pose.position.x = 0.25
table_pose.pose.position.y = 0.0
table_pose.pose.position.z = table_ground + table_size[2] / 2.0
table_pose.pose.orientation.w = 1.0
scene.add_box('table', table_pose, table_size)
```

```
rospy.sleep(2)
```

更新当前的位姿

```
arm.set_start_state_to_current_state()
```

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 = [0.827228546495185, 0.29496592875743577, 1.1185,
                  0.0, 0.0, 0.0]
arm.set_joint_value_target(joint_positions)
```

控制机械臂完成运动

```
arm.go()
rospy.sleep(1)
```

控制机械臂回到初始化位置

```
arm.set_named_target('home')
arm.go()
```

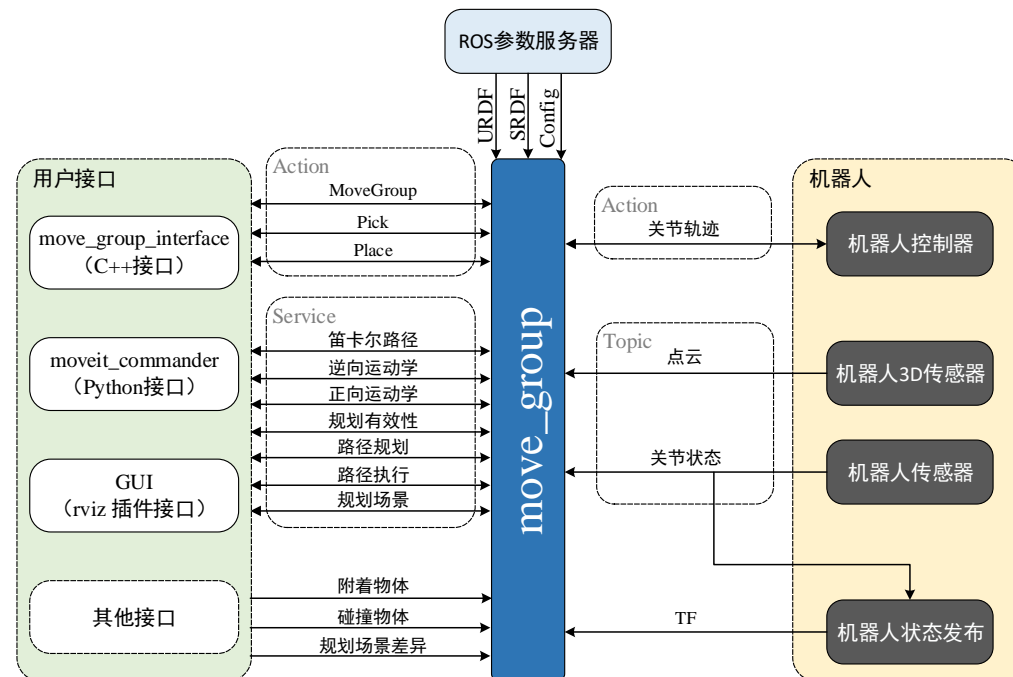



Movelt!的编程接口

- C++、Python
- 编程流程：规划组→目标位姿→运动规划→运动执行

Movelt!基础编程

- 关节空间运动
 - set_joint_value_target
 - set_pose_target
- 笛卡尔空间运动
 - compute_cartesian_path
- 自主避障规划
 - rviz添加障碍物
 - add_box
 - attach_box





1. 编写一个程序，实现以下运动控制功能：

- (1) 在关节空间下，机械臂从起点A运动到B点(关节位置描述)，再从B点运动到C点（终端姿态描述）；
- (2) 在笛卡尔空间下，机械臂从C点依次直线运动到D点和E点，最后回到初始A点，运动结束。

2. 编写一个程序，实现以下自主避障功能：

- (1) 为机械臂终端添加一个附着物体，运动范围内添加一个障碍物体，让机械臂在运动时完成避障过程。



- MoveIt! API Document
<http://moveit.ros.org/code-api/>
- MoveIt! Tutorials
http://docs.ros.org/kinetic/api/moveit_tutorials/html/index.html
- MoveIt! Motion Planning Framework
<https://moveit.ros.org/>
- ROS探索总结（二十六）——MoveIt编程
<http://www.guyuehome.com/455>

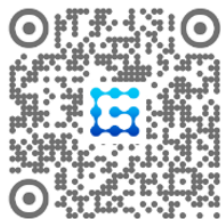




Thank You

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