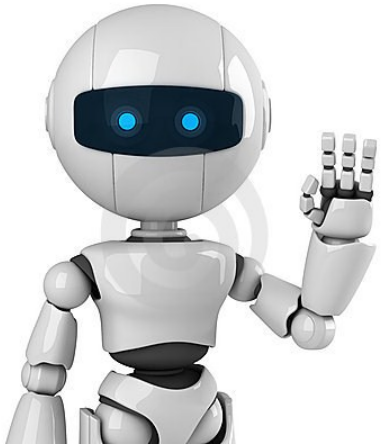
基于moveit!的机械臂仿真



东北大学 机器人科学与工程学院

张云洲

2018年11月

内容提纲

1. Moveit!系统结构

2.move_group核心组件

3. KDL运动学package

4. Ompl运动规划package

5. FCL碰撞检测package

6. Trajectory Processing 轨迹处理package

8.Moveit!与实际机器人的连接:ros_control

7.消息类型

9. 实验一:建立第一个机械臂仿真模型

10.实验二:机械臂操控仿真

11. 实验三:物体操作仿真

MoveIt!的功能模块

- 运动规划（Motion Planning）：

要让一个机器人实现运动规划，需要先将其抽象到构形空间（C-Space）。MoveIt提供此功能。只需提供机器人URDF模型，即可调用运动规划库的规划算法（如OMPL，SBPL，CHMOP），自动生成机器人运动轨迹。

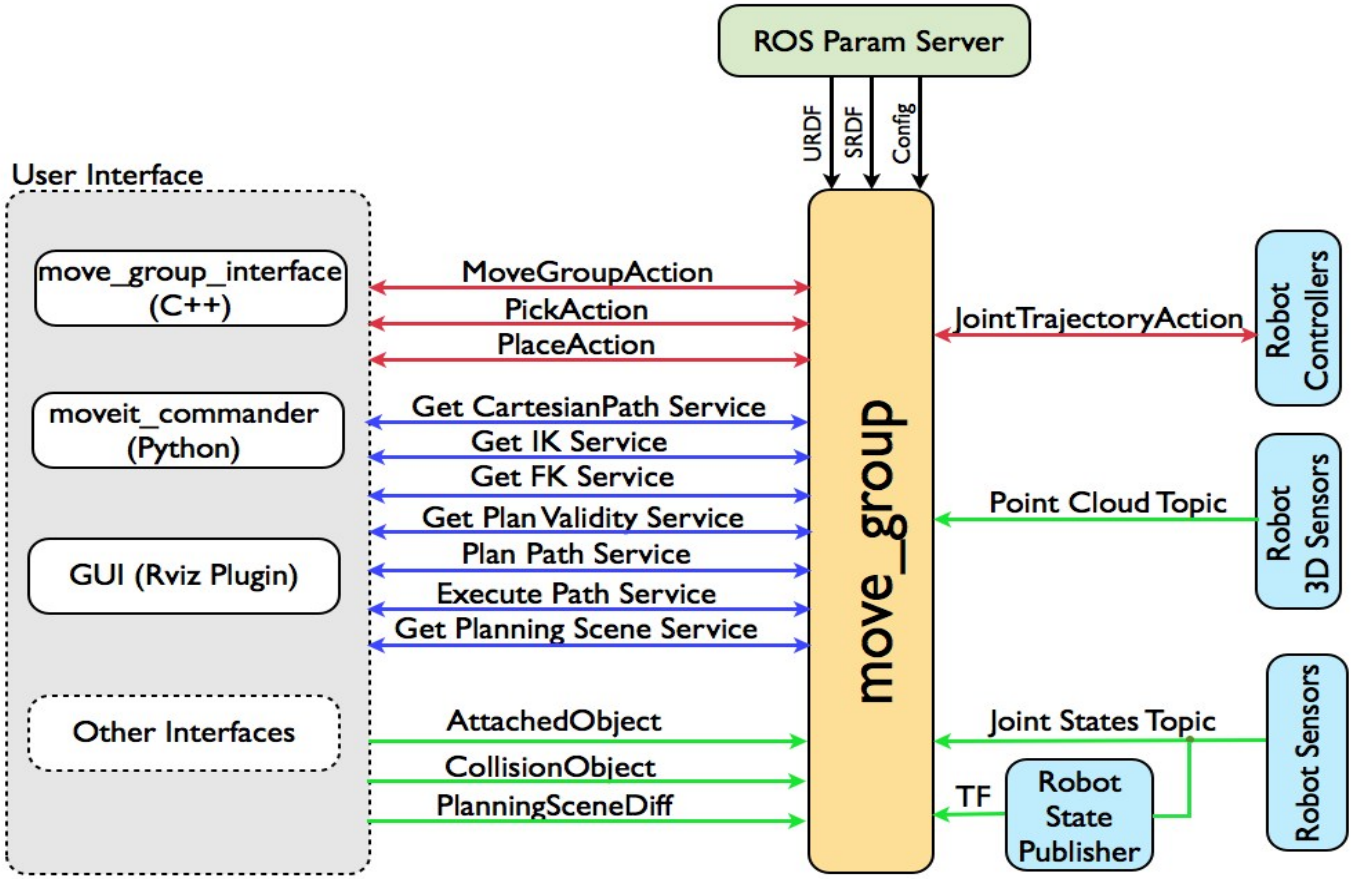
- 操作（Manipulation）：

目前还比较弱，是根据识别的物体生成一系列动作以抓取物体（pick-and-place），不涉及任何反馈、动力学、re-grasp等操作问题。

- 3D感知（Perception）：

并不意味着MoveIt整合了物体识别、环境建模等模块，而是它可以利用传感器（坑）采集的信息（点云或深度图像）生成用于碰撞检测的OctoMap。OctoMap这个东西挺好的，做SLAM的同学应该了解，它就是以八叉树形式表示点云，可以大大降低存储空间，它看起来就跟你们玩的minecraft差不多。同时，这些3D OctoMap也可以依据贝叶斯准则不断实时更新。这样，机器人就可以避开真实世界的障碍物了。

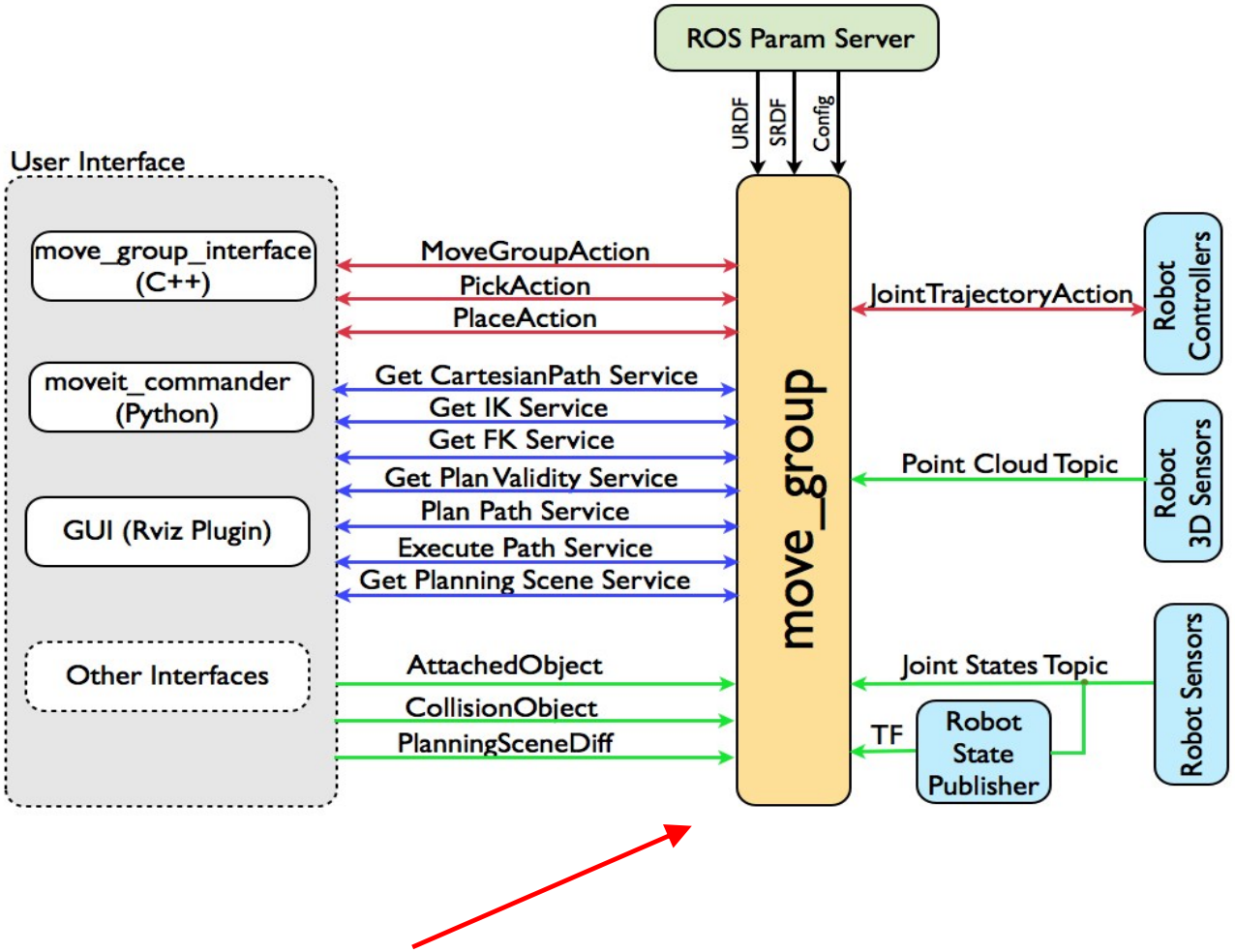
Moveit!系统结构



Moveit!系统结构示意图

MOVEit！结合了运动规划，操纵，三维感知，运动学，控制和导航的最新进展，提供一个易于使用的平台，开发先进的机器人应用程序，评估新的机器人设计和建筑集成的机器人产品。

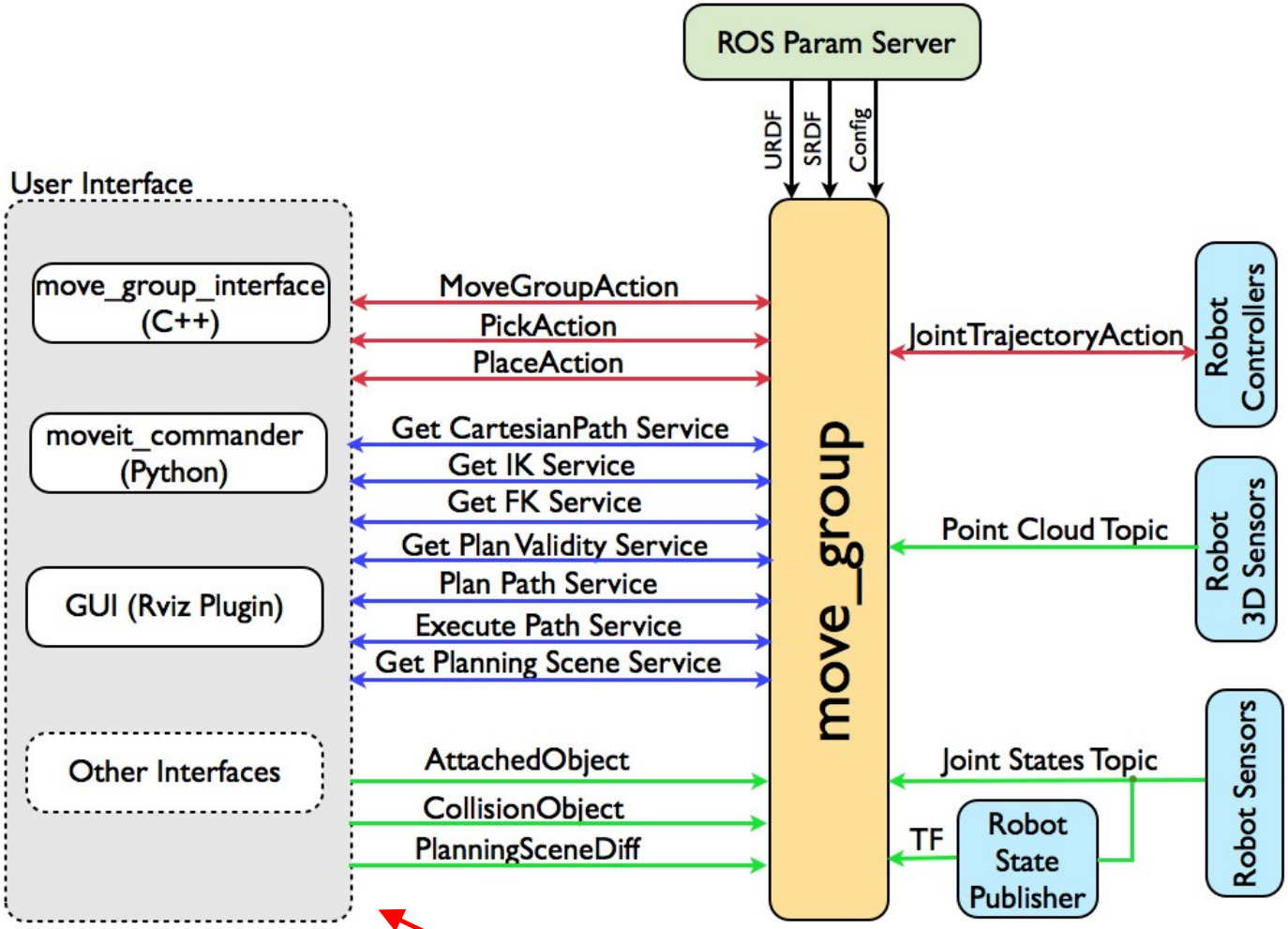
Moveit!系统结构



The move_group node（move_group节点）

最重要的是[move_group](move_group)节点，充当整合器：整合多个独立的组件，并提供ROS风格的Action和service。

Moveit!系统结构



User Interface（用户接口，三种接口可供调用）

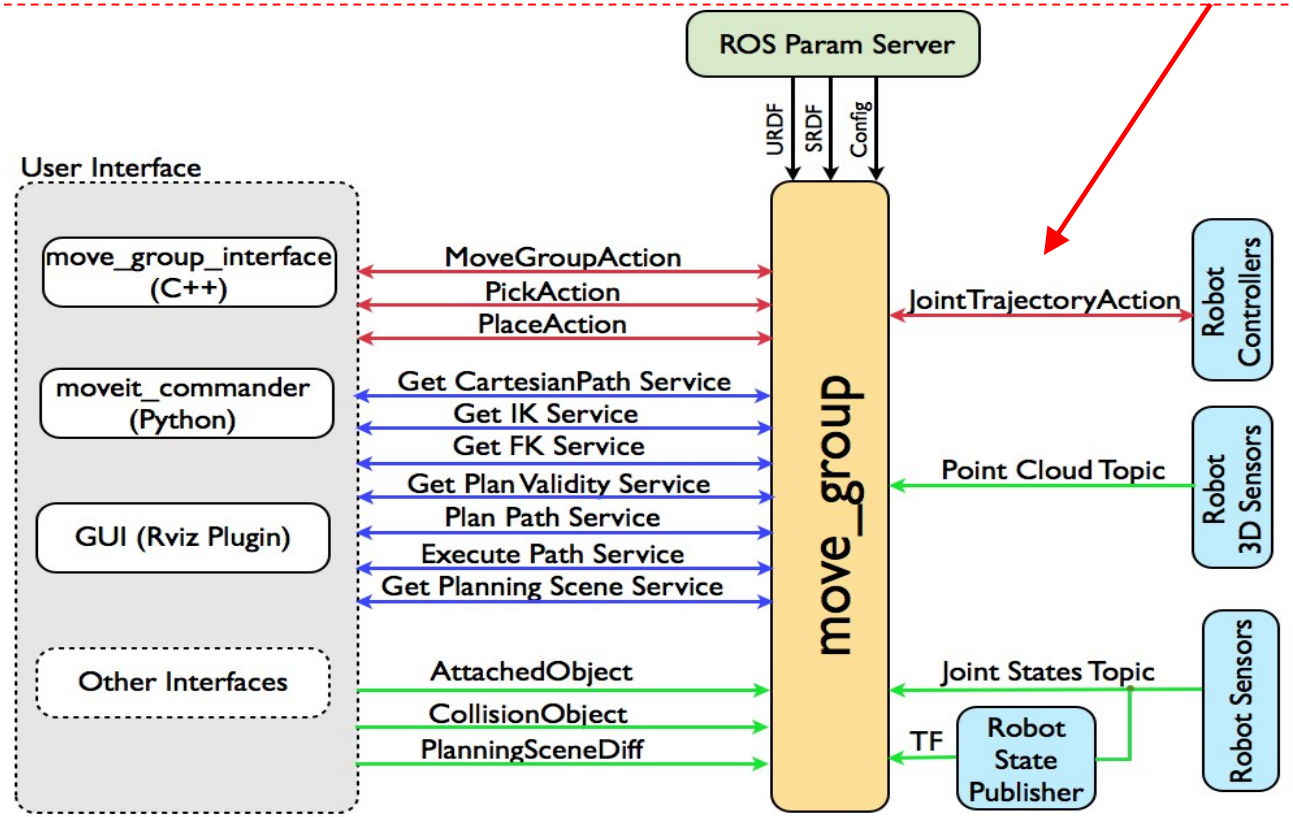
- C++， 利用move_group_interface包可以方便使用move_group。
- Python，利用moveit_commander包
- GUI（界面），利用Motion Planning 的 Rviz插件。

Moveit!系统结构

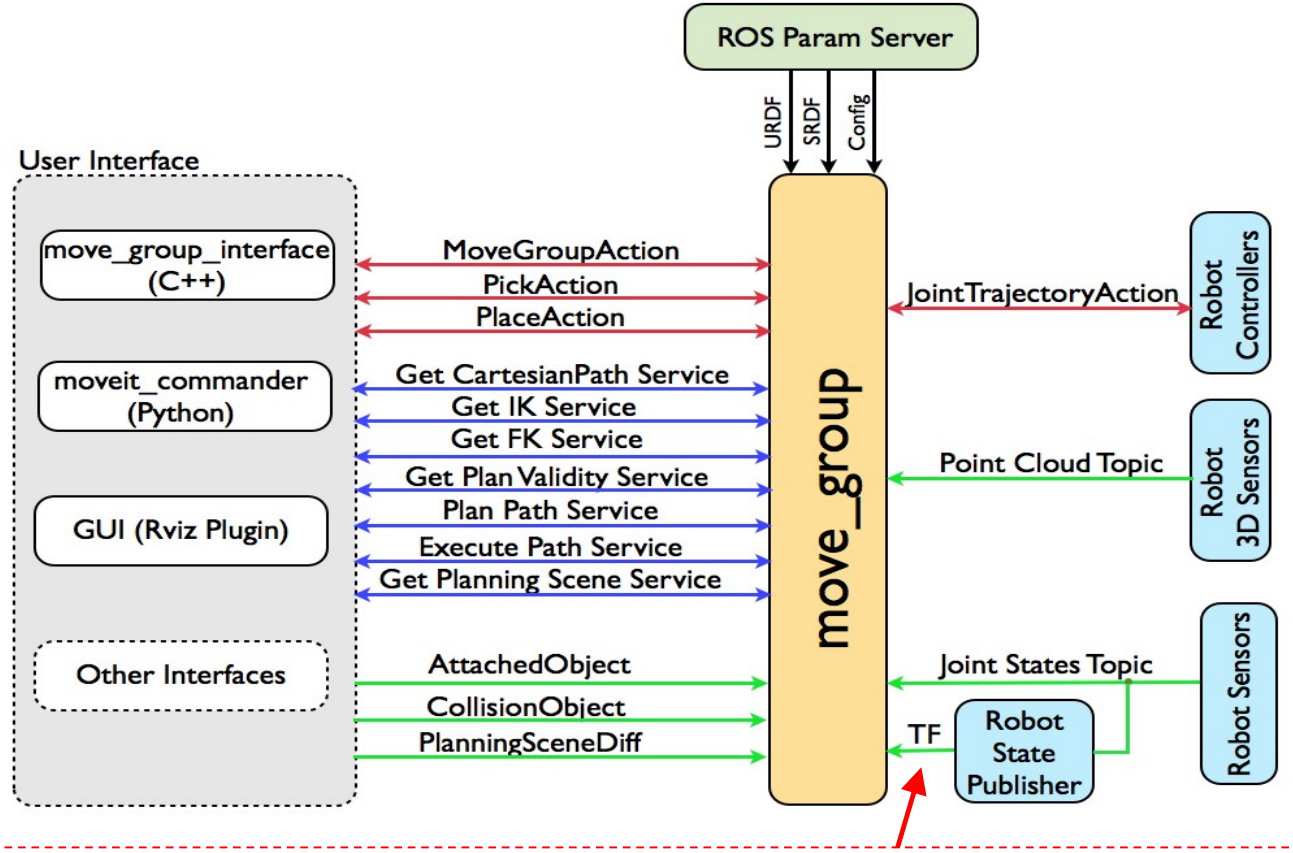
- Robot Interface（机器人接口）

  move_group通过ROS topics和actions与机器人通讯，获取机器人的状态（位置，节点等），获取点云或其他传感器数据再传递给机器人的控制器。

- Joint State Information（节点状态信息）
  - move_group监听 /joint_states 主题确定状态信息。例如：确定每个节点的位置。
  - move_group能够监听在这主题的多个发布器信息，即使是发布部分的信息（例如：独立的发布器可能是用于机械臂或移动机器人）。
  - move_group不会建立自己的节点状态发布器。这就需要在每个机器人单独来建立。



Moveit!系统结构



- Transform Information（变换信息）

Move_group通过ROS TF库来监视变换信息。这允许节点获取全局的姿态信息。

例如：navigation包发布机器人的map frame和base frame到TF，move_group可以使用TF找出这个变换信息，在内部使用。

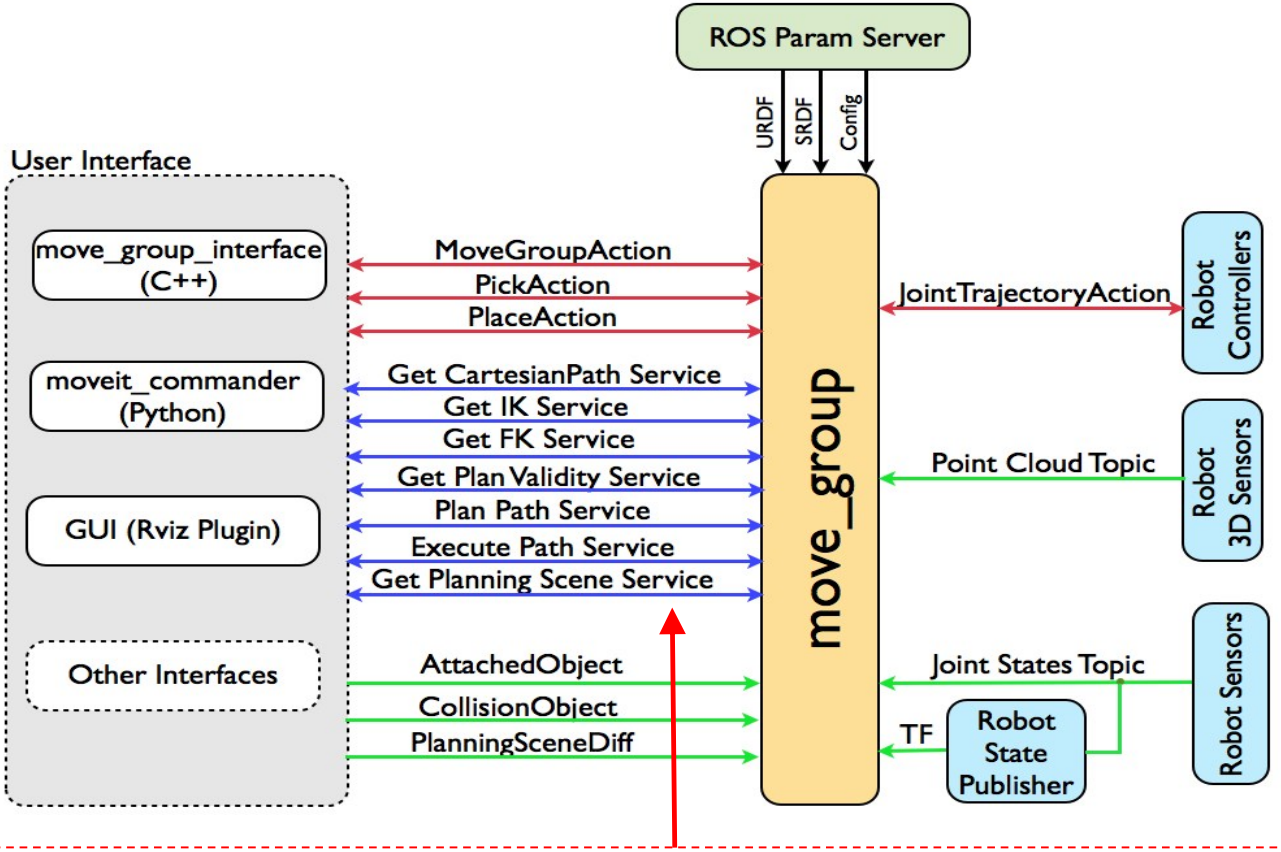
注：Move_group只是监听TF，需要启动robot_state_publisher才能发布TF。

- Controller Interface （控制器接口）

通过ROS的action接口，FollowJointTrajectoryAction接口来使用控制器。一个机器人的服务器服务于这个action-这个服务器不是有move_group提供。

move_group只会实例化客户端与机器人的控制器action服务器通讯。

Moveit!系统结构



- Planning Scene（规划场景）

  move_group使用规划场景监视器来维护规划场景。

  场景是世界的和机器人的状态的表现。

  机器人状态包含机器人刚性连接到机器人的所有物体。

- Extensible Capabilities（可扩展能力）

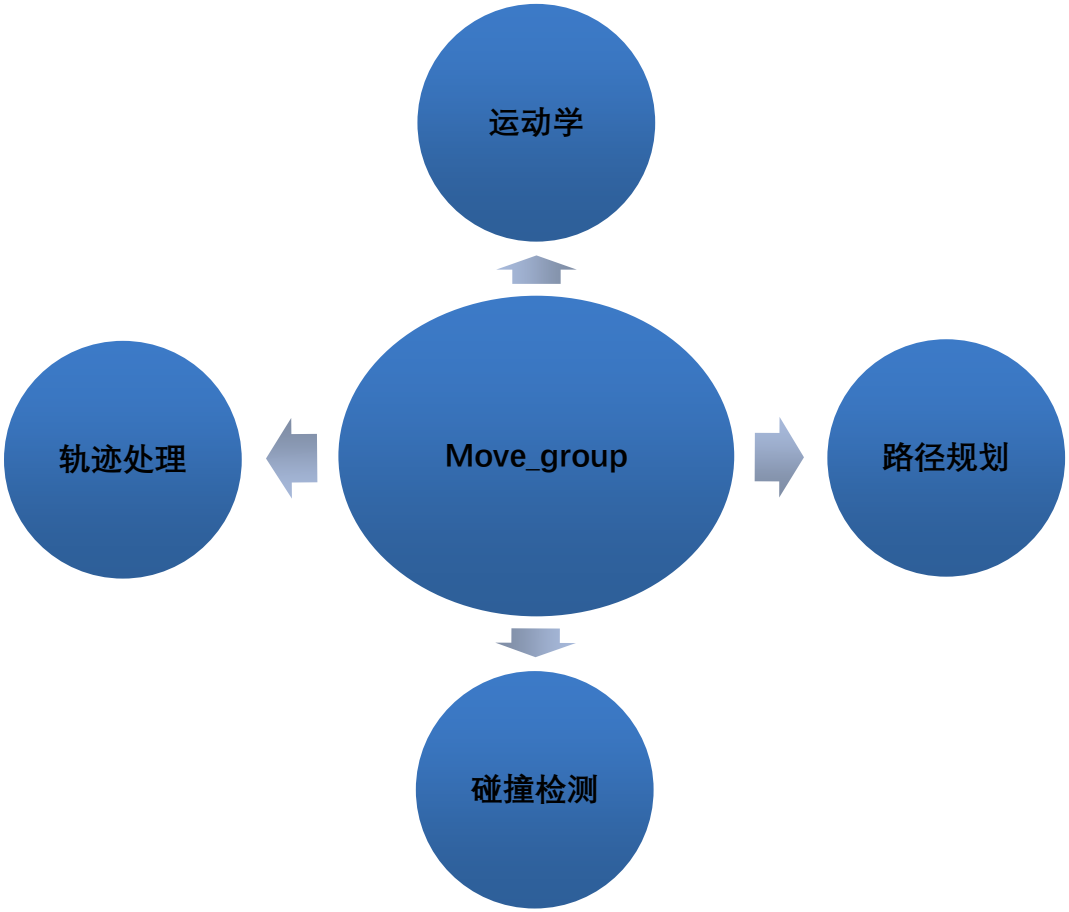
  move_group的结构被设计成容易扩展，独立的能力如抓放，运动学，运动规划。

  扩展自公共类，但实际作为独立的插件运行。

  插件可经由一系列的ROS yaml parameters 和ROS pluginlib库配置。。

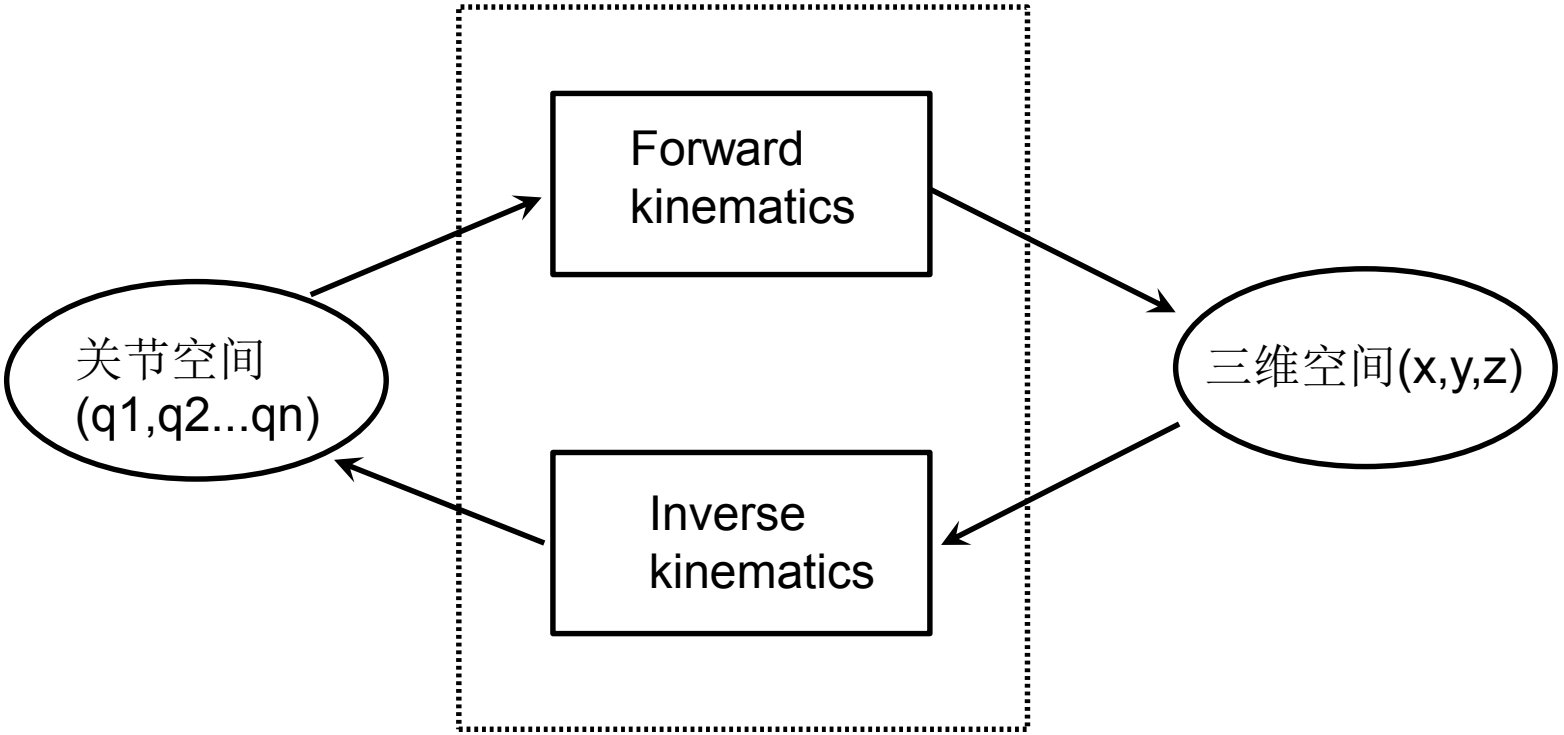
Move_group 核心组件

Move_group是一个开源框架，本体不包含任何算法。通过插件的形式实现具体的功能。开发者也可以把自己的算法以插件的形式插入到move_group中。



KDL Package

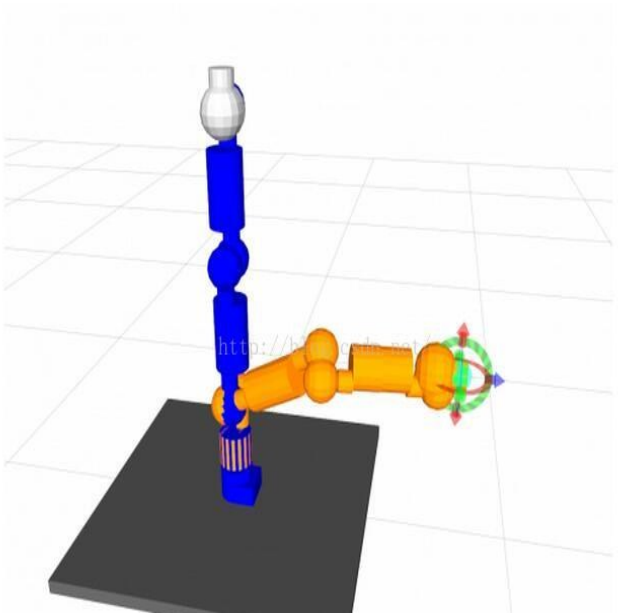
Kinematics and Dynamics Libraryonte(KDL)，是运动学与动力学的库。它可以很好的解决6自由度以上的单链机械结构的正逆运动学问题。

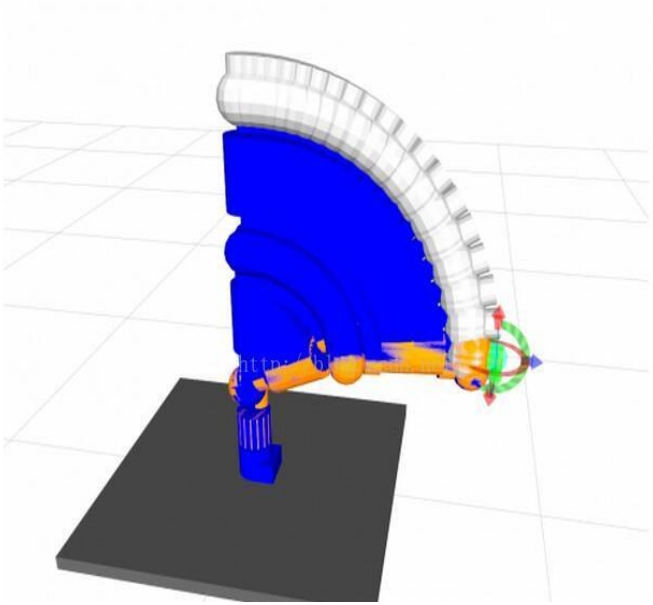


KDL库

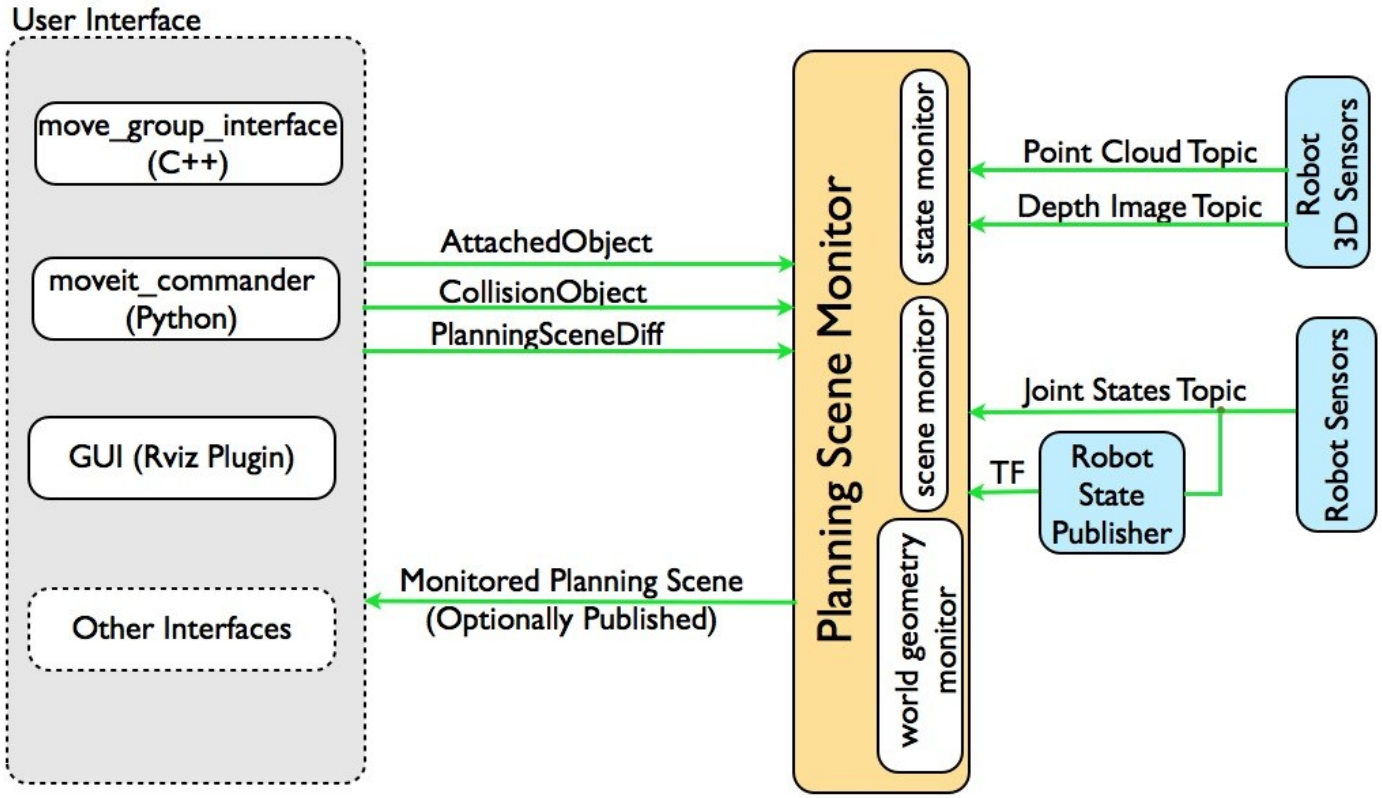
ompl Package

OMPL (Open Motion Planning Library)， 开源运动规划库，就是一个运动规划的C++库，其包含了很多运动规划领域的前沿算法。最出名的莫过于Rapidly-exploring Random Trees (RRT) 和 Probabilistic Roadmap (PRM)系列算法。





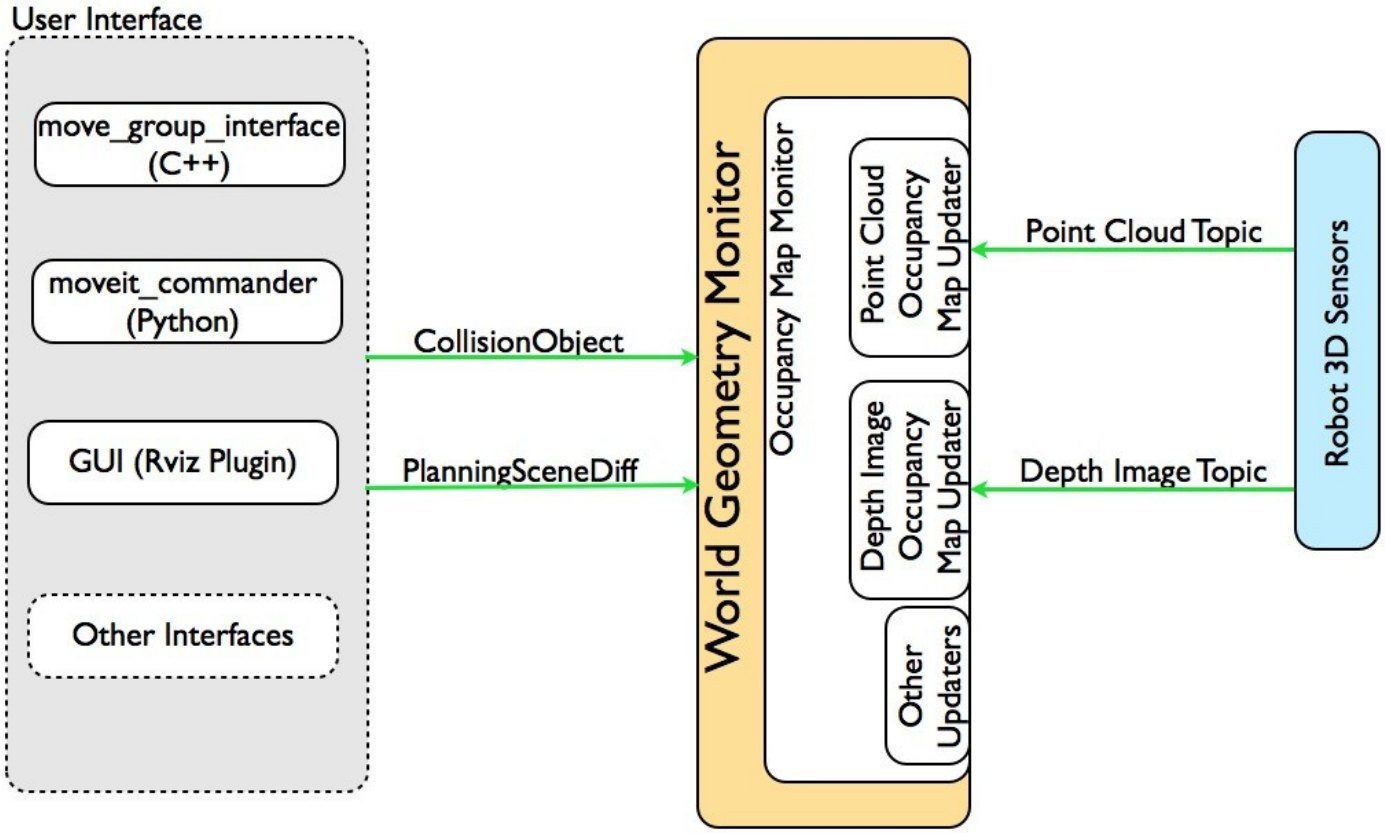
ompl Package



规划场景，用于显示机器人的世界，同时保存机器人自己的状态。它由Move_group节点内的规划场景监视器来维护。规划场景监视器监听：

- State Information（状态的信息）：joint_states 主题
- Sensor Information（传感器的信息）：using the world geometry monitor described below
- World geometry information（世界的几何图形信息）：from user input on the planning_scene topic (as a planning scene diff).

ompl Package



在MoveIt，3D感知是由occupancy map monitor处理，它使用插件结构处理不同的传感器输入。MoveIt有两个内置支持可以处理两种输入：

Point clouds: handled by the point cloud occupancy map updater plugin

Depth images: handled by the depth image occupancy map updater plugin

Octomap，Occupancy map monitor使用Octomap维持Occupancy map的环境。

Depth Image Occupancy Map Updater，深度图像栅格地图的更新器包括它自己的过滤器，例如：可以从深度图消除机器人的可见部分。

FCL Package

FCL(Flexible Collision Library),碰撞检测库。根据机械臂的位置与周围环境信息判断是否发生碰撞，为路径规划提供依据。

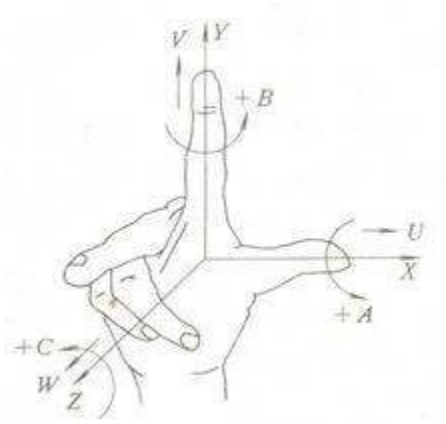
Trajectory Processing package

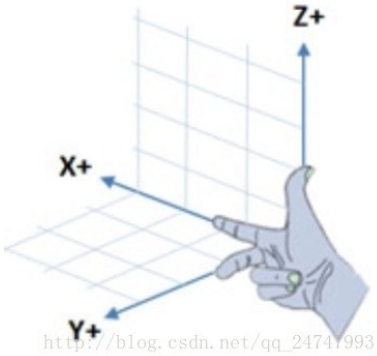
运动规划器一般只会生成路径，这个路径不带时间信息。MoveIt包含轨迹处理程序。它对结合路径和时间参数化的关节限制的速度和加速度来生成轨迹。

坐标系定义

在ROS的机器人控制和导航中，所有的坐标系都以**右手笛卡尔坐标系**定义。常见的坐标系有：世界坐标系、机体坐标系、里程计坐标系等。

ROS系统使用公制作为计量系统，线速度通常使用米/秒(m/s)来作为单位，角速度通常使用弧度/秒(rad/s)来作为单位。





x轴: 机器人正前方，y轴: 正左方，z轴: 正上方

。

右手笛卡尔坐标系

消息类型

在ROS中,通过控制关节的转动实现机械臂的运动

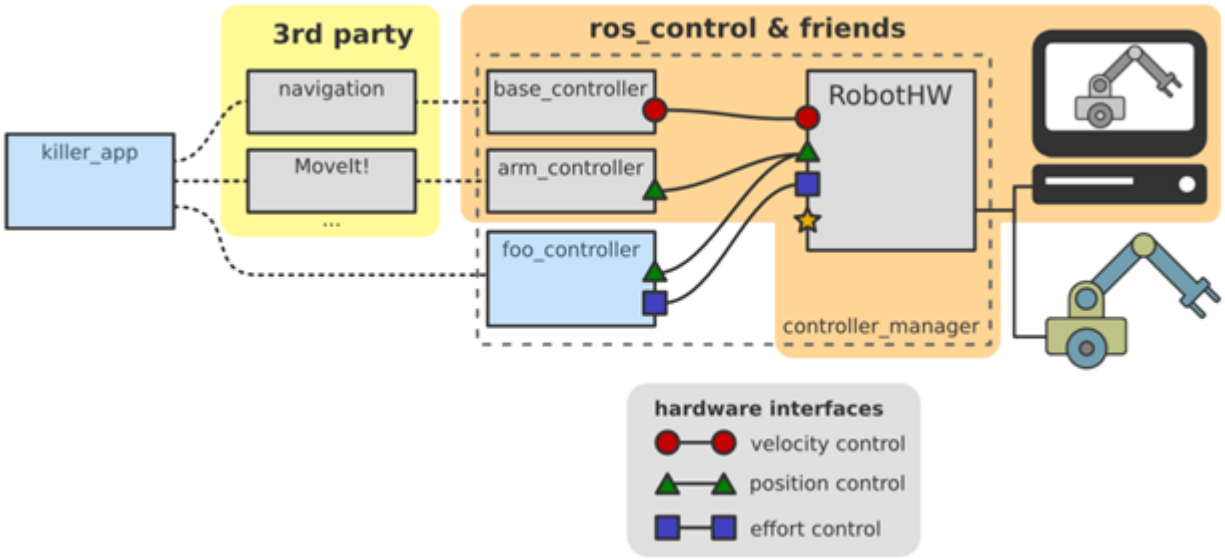
发布话题：/joint_states

Header header

string[] name　　//关节名称
float64[] position　//关节位置
float64[] velocity　//关节速度
float64[] effort　　//关节力矩

Moveit!与实际机器人的连接:ros_control

实际机器人和moveit!之间的通信方式



ros_control是ROS为用户提供的应用与机器人之间的中间件，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。

实验前的准备

1.安装moveit

```
sudo apt-get install ros-indigo-moveit-full
```

2.安装pr2基础插件

```
sudo apt-get install ros-indigo-moveit-full-pr2
```

3.安装一些编译过程中要用到的package

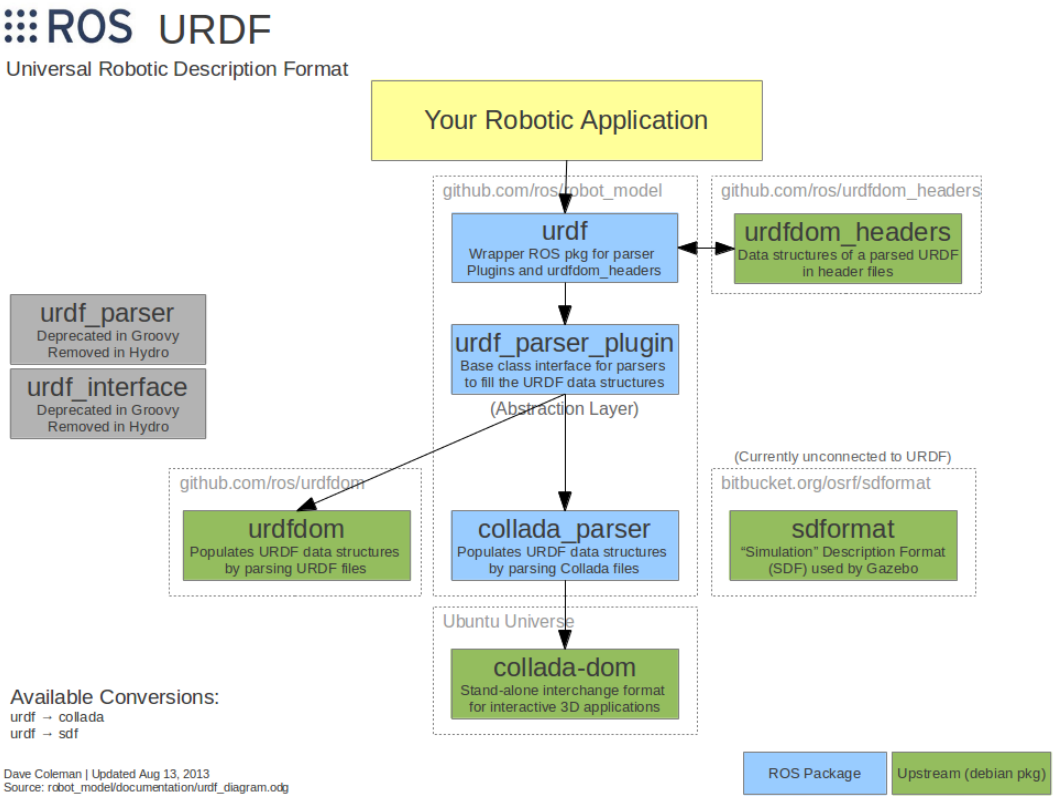
```
sudo apt-get install ros-indigo-moveit-visual-tools
sudo apt-get install ros-indigo-arbotix-*
source /opt/ros/indigo/setup.bash
```

4.下载仿真package并进行编译

```
cd ~/catkin_ws/src
git clone https://gitee.com/neu103_robot/easy_demo.git
git clone https://gitee.com/neu103_robot/moveit_tutorials.git
git clone https://gitee.com/neu103_robot/rbx2.git
catkin_make
source ~/catkin_ws/devel/setup.bash
```

urdf定义

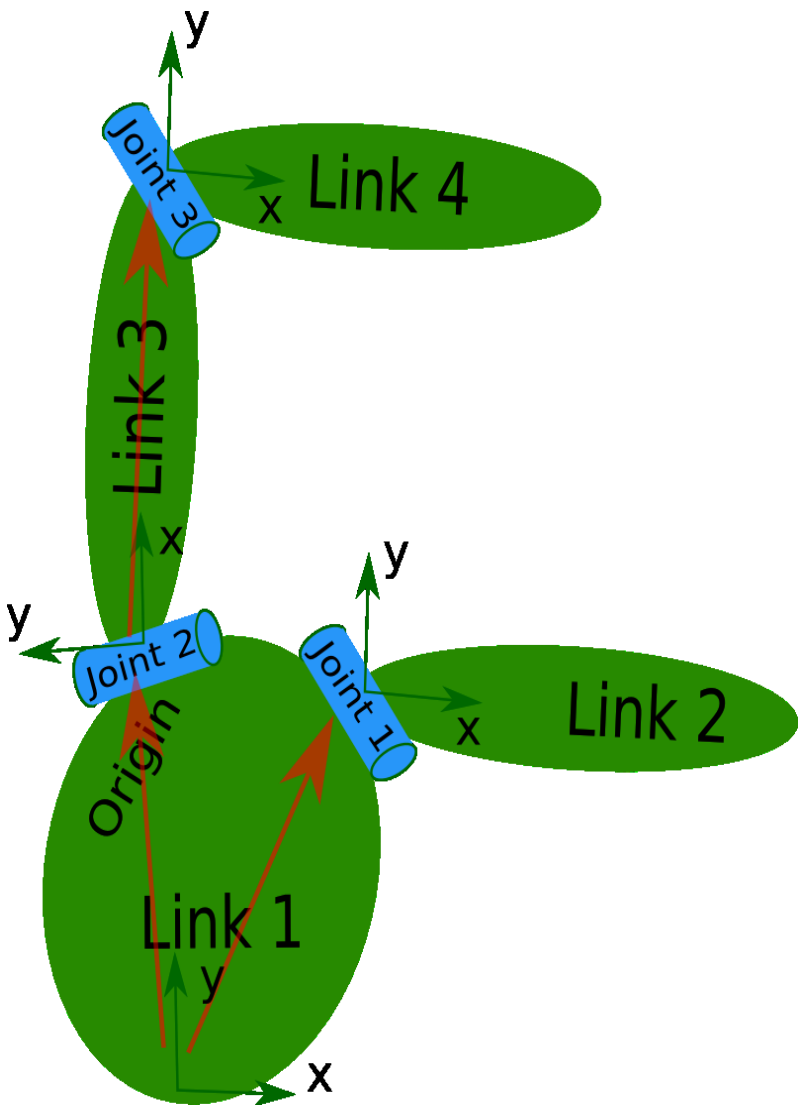
Unified Robot Description Format，统一机器人描述格式，简称为URDF。URDF文件使用XML格式描述机器人模型。



URDF package

功能:将URDF文件解析为3D模型并显示

urdf文件基本结构




```
1 <robot name="pr2">
2   <link> ... </link>
3   <link> ... </link>
4   <link> ... </link>
5
6   <joint>  ....  </joint>
7   <joint>  ....  </joint>
8   <joint>  ....  </joint>
9 </robot>
```

URDF 基本结构
机器人的描述包括link(连杆)和joint(关节)

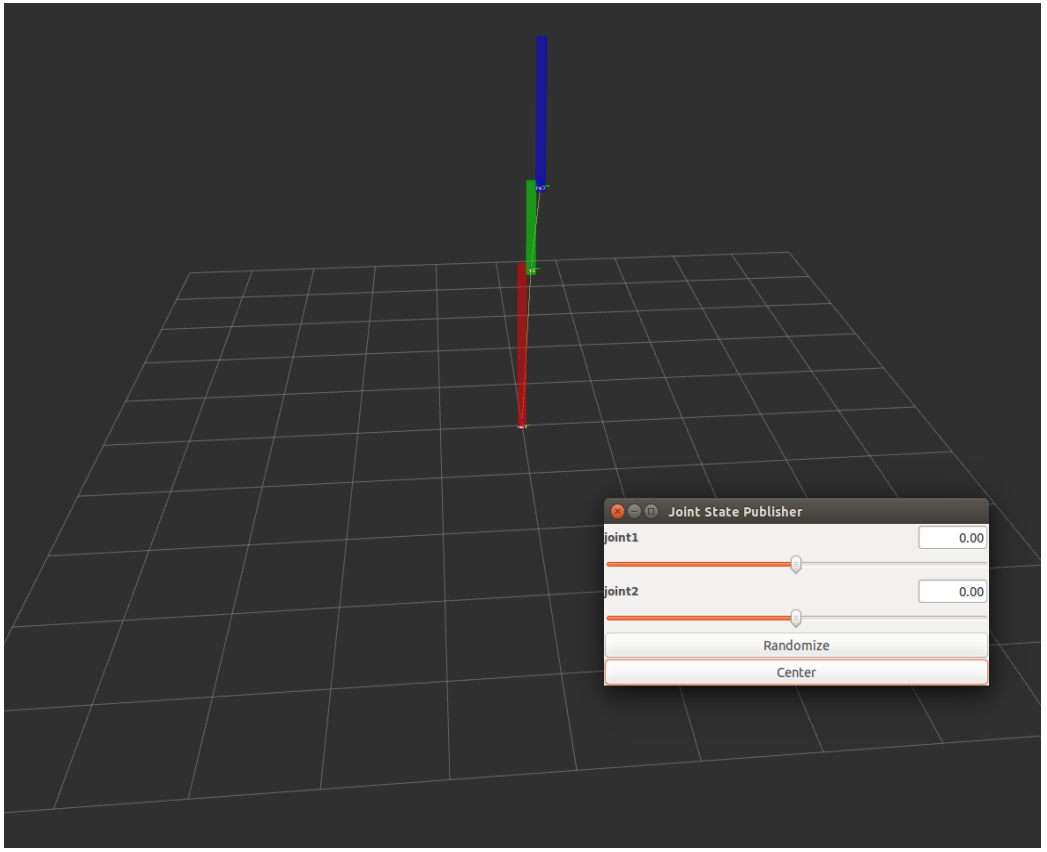
实验一：构建机械臂模型

■实验目的

利用URDF构建简单的机械臂模型

■技术要点

URDF文件格式



机械臂模型

实验一：构建机械臂模型

■实验步骤

1.启动viz
roslaunch easy_demo demo.launch

2.查看节点关系图
rosrun rqt_graph rqt_graph

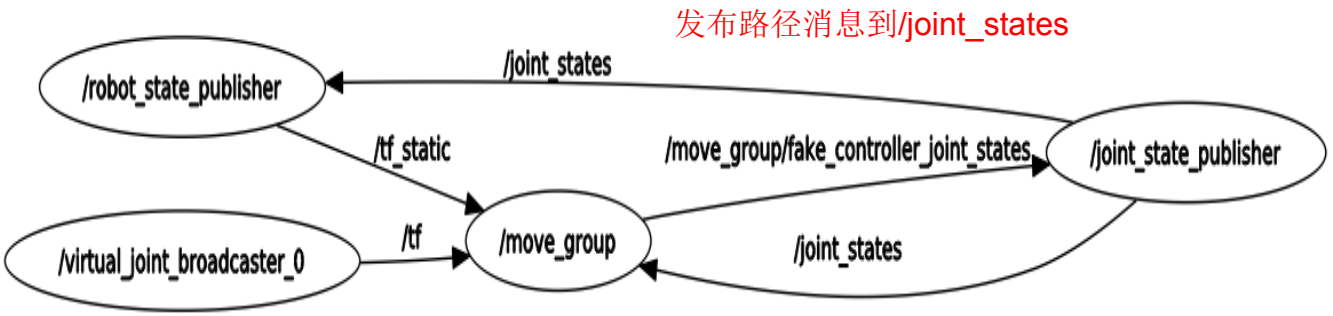
3.查看节点输出
rosopic echo /joint_states

4.在joint_states_publisher中拖动joint values，观察节点输出

实验二：机械臂操控仿真

■实验目的

控制机械臂运动



机械臂操控节点关系图

实验二：机械臂操控仿真

■实验步骤

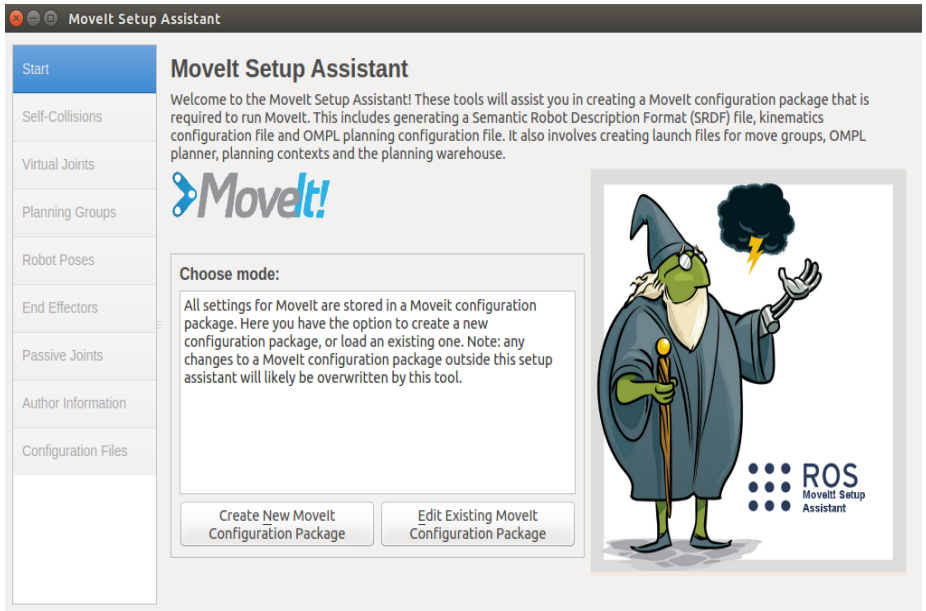
1.启动Setup assistant

roslaunch moveit_setup_assistant setup_assistant.launch

2.在Setup assistant配置仿真机器人

配置助手(Setup Assistant)是一个图形化的用户接口，用于配置整合机器人，让我们能使用MoveIt!控制机器人。

核心功能：通过urdf文件生成Semantic Robot Description Format (SRDF)文件

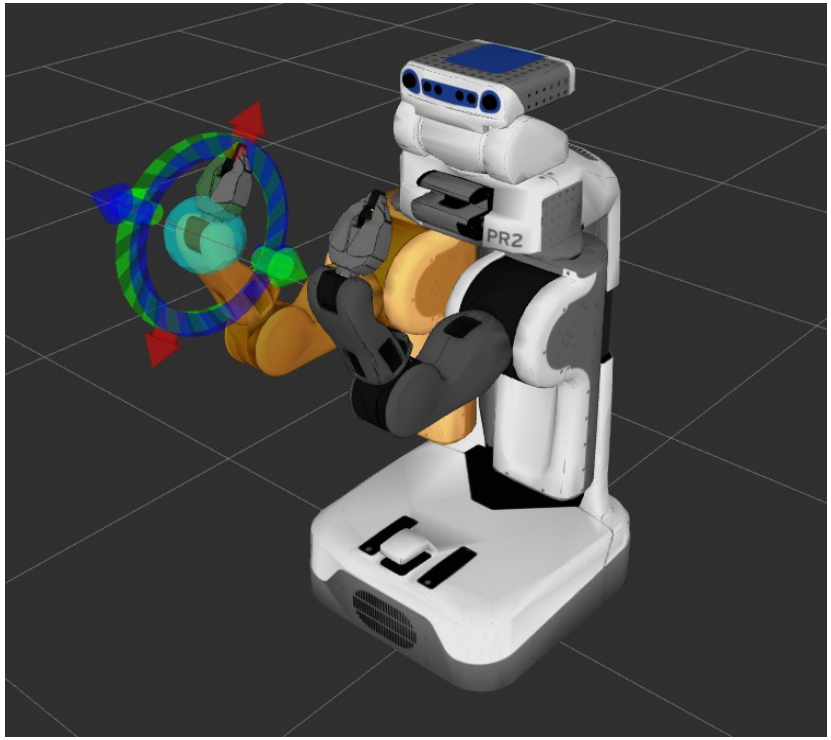


实验二：机械臂操控仿真

■实验步骤

3.在rviz中查看机器人
roslaunch pr2_moveit_config demo.launch

4.通过程序进行运动规划
roslaunch moveit_tutorials move_group_interface_tutorial.launch



仿真效果图

正逆运动学求解实验

■实验目的

求解机械臂正逆运动学

■技术要点

理解关节空间与笛卡尔空间

■实验步骤

启动运动学解算demo节点：

roslaunch moveit_tutorials kinematic_model_tutorial.launch

实验三：物体操作仿真

■实验目的

物体抓取与放置

■实验步骤

1.启动仿真模型

roslaunch rbx2_bringup pi_robot_with_gripper.launch sim:=true

2.启动moveit配置

roslaunch pi_robot_moveit_config move_group.launch

3.启动rviz

rosrun rviz rviz -d `rospack find \rbx2_arm_nav`/config/pick_and_place.rviz

4.启动物体操作仿真程序

rosrun rbx2_arm_nav moveit_pick_and_place_demo.py