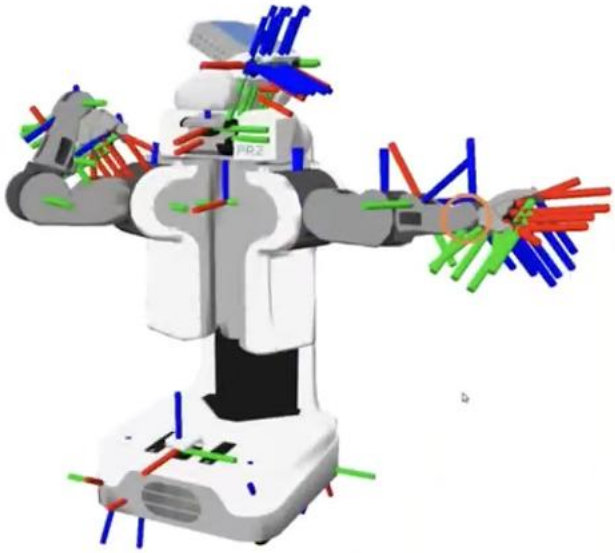
第五章： ROS2 中常用开发工具

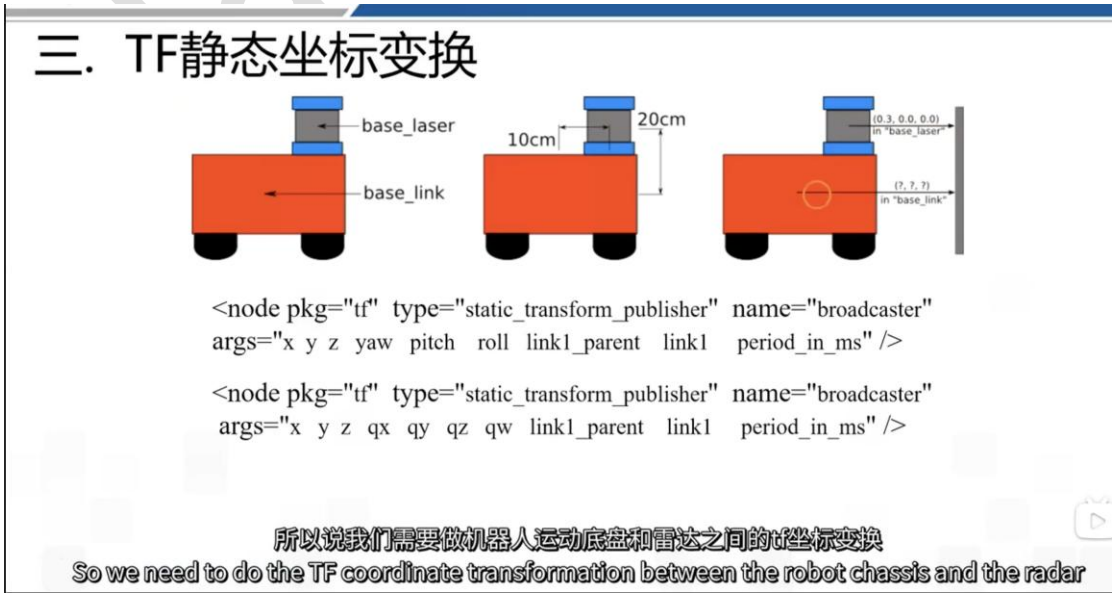
一． ROS 坐标管理系统

Transform：两个坐标之间的转换（旋转、平移）

（下图为 ROS 机器人——pr2）



如果想实现人和机器人握手，首先要知道手在哪，视觉摄像头可以实现此功能，但是他的坐标是手相对于摄像头的，我们还要知道手相对于机械臂的，这就需要坐标转换！



在这个例子中，base-laser 是雷达坐标，但是在实际运用中我们更看重是障碍物距离地盘的距离，所以需要 tf 坐标转换。

在这个例子中，雷达是固定在运动底盘上的不会产生相对运动，所以这种也叫做“TF 坐标静态变换”。（用在传感器偏多，固定组件）

一般基于欧拉角和四元素。（图中上面是欧拉角【常用】，下面是四元素【ROS】）

5.1 通过命令行使用 TF 工具

发布base_link到base_laser之间的变换： ros2 run tf2_ros static_transform_publisher --x 0.1 --y 0.0 --z 0.2 --roll 0.0 --pitch 0.0 --yaw 0.0 --frame-id base_link --child-frame-id base_laser

发布base_laser到wall_point之间的变换： ros2 run tf2_ros static_transform_publisher --x 0.3 --y 0.0 --z 0.0 --roll 0.0 --pitch 0.0 --yaw 0.0 --frame-id base_laser --child-frame-id wall_point

查询base_link到wall_point之间的关系： ros2 run tf2_ros tf2_echo base_link wall_point

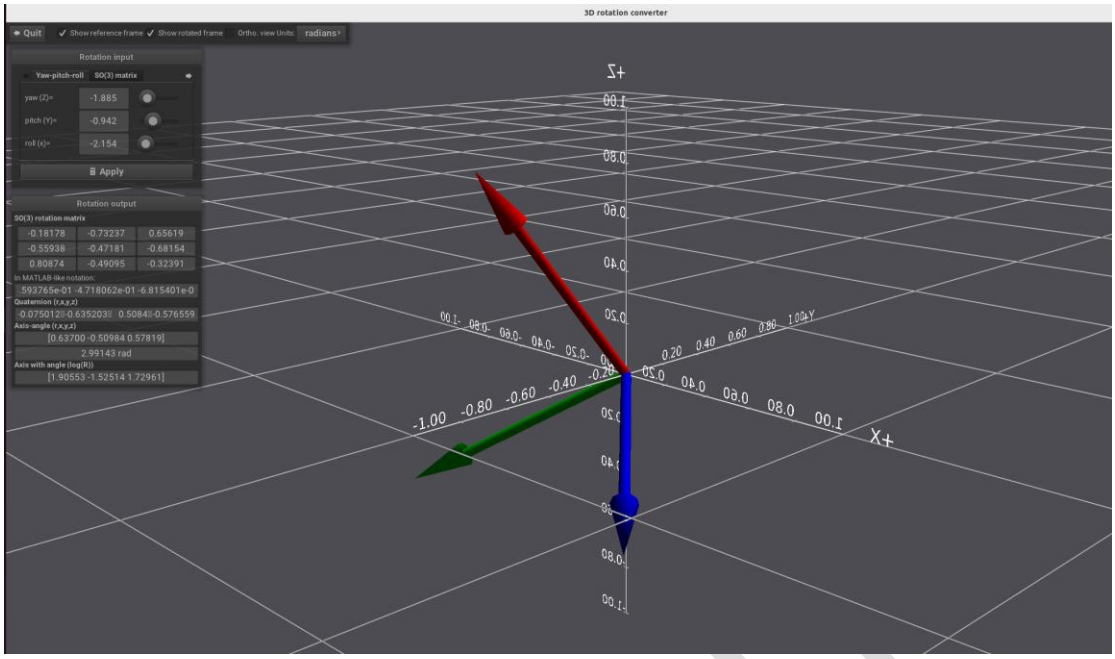
推荐工具安装

1. 3D-rotation-converter（直观看出旋转变化）

（鱼香 ROS 的命令无法安装，这里给出替代方案）

Sudo apt install mrpt-apps

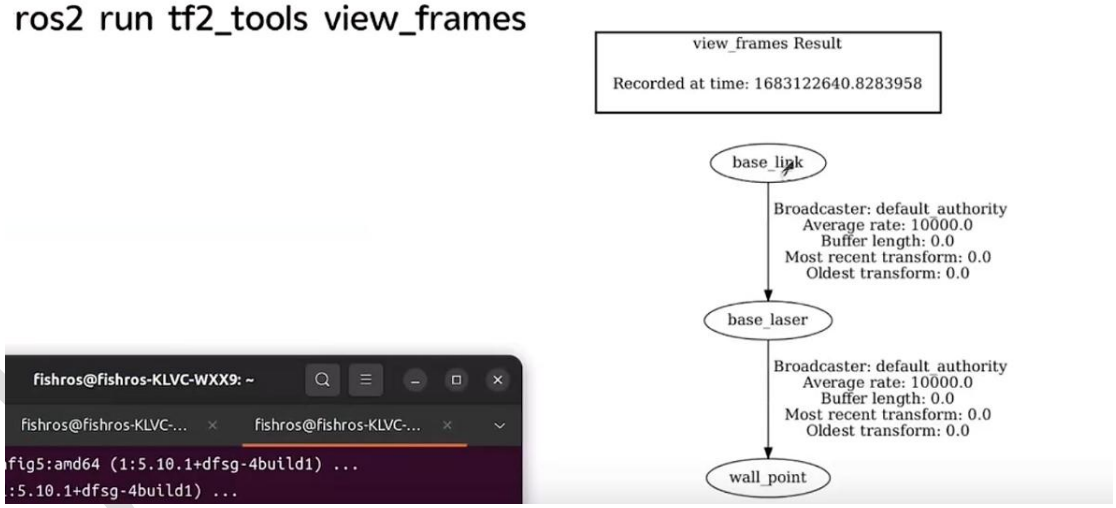
3D-rotation-converter



2. TF Tree （直观看出不同坐标系标准的关系）

(eg: base_link / base_laser / wall_point) pdf 文件

```
ros2 run tf2_tools view_frames
```



5.2 简单探究 TF 与话题通信关系

在 ROS 2 中，TF 的底层通信**是依赖话题系统来广播和接收坐标变换的**，即：

- TF 的变换信息通过话题发送，比如：
  - `/tf`（实时变换）
  - `/tf_static`（静态变换）

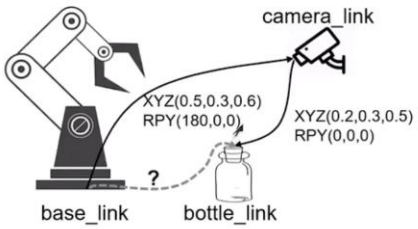
📌 **举例：**

- 一个 TF Broadcaster 会发布变换消息到 `/tf`，消息类型是 `tf2_msgs/msg/TFMessage`。
- TF Listener 会订阅 `/tf` 来接收这些变换数据，并缓存到变换树中。

➡️ 所以从技术角度看，**TF 本质上是话题通信的一种封装应用**，但专门用于管理坐标变换。

5.3 Python TF 手眼坐标变换

需求：
相机固定在右上方的camera_link处
机械臂的底座固定在base_link处
从base_link到camera_link的位置是固定不变的，平移分量（0.5，0.3，0.6），旋转分量（180，0，0）
相机通过识别得到瓶子bottle_link的坐标，其中平移分量（0.2，0.3，0.5），旋转分量（0，0，0）



相机（眼）通过机器视觉可以识别到 bottle、机械臂（手）；

机械臂可以在 roll 上面旋转 180 度。

基本想法：

Camera_link 和 base_link 关系通过静态坐标发布器发布

Camera_link 和 bottle_link 关系通过静态坐标发布器发布

然后再 df 监听

问题是实际情况中瓶子可能会变化，所以 Camera_link 和 bottle_link 关系用动态 TF 发布

5.3.1 从机械臂底座到相机——静态 TF 发布

例子中给的是欧拉角（RPY），所以需要把欧拉角转为四元素，这里鱼香 ROS 给了两个代码，第一个代码运行不了，但是下面也能实现部分功能平替，所以这里我们就先

装下面的： sudo pip3 install transform3d

和前面一样构建功能包

完整代码：

通过 sendTransform 发送

终端拆分左边用三段式将节点跑起来后右侧终端键入

ros2 topic list -> 应该能看到刚广播的话题（topic）: /tf_static

ros2 topic echo /tf_static : 监听

```python
import rclpy
from rclpy.node import Node
from tf2_ros import StaticTransformBroadcaster # 静态坐标广播器
from geometry_msgs.msg import TransformStamped # 坐标变换消息
from transforms3d.euler import euler2quat as quaternion_from_euler # 欧拉角转四元数
import math

class StaticTFBroadcaster(Node):
    def __init__(self):
        super().__init__('static_tf_broadcaster')
        self.static_broadcaster = StaticTransformBroadcaster(self) # 创建静态坐标广播器
        self.publish_static_tf() # 在静态坐标广播器中订阅这个函数

    # 发布静态坐标变换
    def publish_static_tf(self):
        '''
        发布静态坐标变换：从base_link 到 camera_link之间的变换关系
        '''

        # 发的是消息接口的对象 —— 创建一个 TransformStamped 消息
        transform = TransformStamped()
        transform.header.frame_id = 'base_link'
        transform.child_frame_id = 'camera_link'
        transform.header.stamp = self.get_clock().now().to_msg()

        # 设置平移部分 (看PPT)
```

```python
        transform.transform.translation.x = 0.5
        transform.transform.translation.y = 0.3
        transform.transform.translation.z = 0.6

        # 设置旋转部分
        q = quaternion_from_euler(math.radians(180),0, 0) # 欧拉角转四元数 (返回是元组)
        # 旋转部分赋值 (元组)
        transform.transform.rotation.x = q[0]
        transform.transform.rotation.y = q[1]
        transform.transform.rotation.z = q[2]
        transform.transform.rotation.w = q[3]
        # 发布静态坐标变换
        self.static_broadcaster.sendTransform(transform)
        self.get_logger().info('Static TF broadcasted: base_link -> camera_link')

def main():
    rclpy.init()  # 初始化ROS2
    node = StaticTFBroadcaster()  # 创建静态坐标广播器节点
    rclpy.spin(node)  # 保持节点运行
    node.destroy_node()  # 销毁节点
    rclpy.shutdown()  # 关闭ROS2
```

结果显示如下：

```
jackson@jackson-virtual-machine:~/chapt5/chapt5_ws$ colcon build
Starting >>> demo_python_tf
Finished <<< demo_python_tf [4.40s]

Summary: 1 package finished [5.81s]
upjackson@jackson-virtual-machine:~/chapt5/chapt5_ws$ source install/setup.bash
rosjackson@jackson-virtual-machine:~/chapt5/chapt5_ws$ ros2 run demo_python_tf static_tf_broadcaster
[INFO] [1750058593.568312905] [static_tf_broadcaster]: Static TF broadcasted: base_link -> camera_link
```

```
jackson@jackson-virtual-machine:~/chapt5/chapt5_ws$ ros2 topic list
/parameter_events
/rosout
/tf_static
jackson@jackson-virtual-machine:~/chapt5/chapt5_ws$ ros2 topic echo /tf_static
transforms:
- header:
    stamp:
      sec: 1750058593
      nanosec: 371839926
    frame_id: base_link
  child_frame_id: camera_link
  transform:
    translation:
      x: 0.5
      y: 0.3
      z: 0.6
    rotation:
      x: 6.123233995736766e-17
      y: 1.0
      z: 0.0
      w: 0.0
---
```

5.3.2 从水瓶到相机——动态 TF 发布

新建一个 dynamic_tf_broadcaster 在这里面复制粘贴刚刚写好的静态的代码略作改动

完整代码：不要忘了 setup 里配置 main

```python
import rclpy
from rclpy.node import Node
from tf2_ros import TransformBroadcaster # 动态坐标广播器
from geometry_msgs.msg import TransformStamped # 坐标变换消息
from transforms3d.euler import euler2quat as quaternion_from_euler # 欧拉角转四元数
import math

class TFBroadcaster(Node):
    def __init__(self):
        super().__init__('tf_broadcaster')
        self.broadcaster = TransformBroadcaster(self) # 创建动态坐标广播器
        self.timer = self.create_timer(0.01, self.publish_tf) # 定时器，每0.01秒调用一次函数

    # 发布静态坐标变换
    def publish_tf(self):
        '''
        发布静态坐标变换：从bottle_link 到 camera_link之间的变换关系
        '''

        # 发的是消息接口的对象 —— 创建一个 TransformStamped 消息
        transform = TransformStamped()
        transform.header.frame_id = 'camera_link'
        transform.child_frame_id = 'bottle_link'
        transform.header.stamp = self.get_clock().now().to_msg()

        # 设置平移部分 （看PPT）
        transform.transform.translation.x = 0.2
        transform.transform.translation.y = 0.3
        transform.transform.translation.z = 0.5

        # 设置旋转部分
        q = quaternion_from_euler(0,0,0) # 欧拉角转四元数 （返回是元组）
        # 旋转部分赋值 （元组）
        transform.transform.rotation.x = q[0]
        transform.transform.rotation.y = q[1]
        transform.transform.rotation.z = q[2]
        transform.transform.rotation.w = q[3]
        # 发布静态坐标变换
        self.broadcaster.sendTransform(transform)
        self.get_logger().info('Dynamic TF broadcasted: camera_link -> bottle_link')

def main():
    rclpy.init()  # 初始化ROS2
    node = TFBroadcaster()  # 创建静态坐标广播器节点
    rclpy.spin(node)  # 保持节点运行
    node.destroy_node()  # 销毁节点
    rclpy.shutdown()  # 关闭ROS2
```

这里一定注意父子坐标系：

问题	谁是父?	谁是子?
我要知道水瓶在相机中的位置	相机（父）	水瓶（子）
我要知道相机安装在机器人身上的什么地方	机器人底盘（父）	相机（子）
我要知道机械臂末端在地面中的位置	地面（父）	机械臂末端（子）

超简单判断方法 ✅

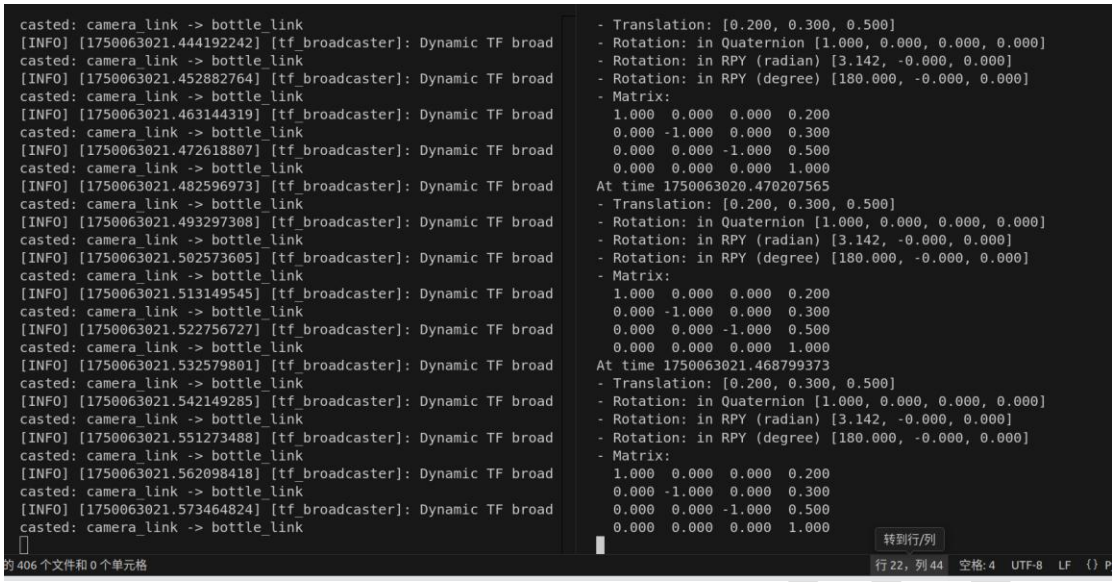
> 把句子变成"子坐标系 在 父坐标系 中的位置是啥"

举个例子：

> **水瓶 在 相机坐标系 中的位置是：x=1, y=2, z=0.5**

👉 所以：父 = 相机（camera_link），子 = 水瓶（bottle）

最终实现结果：（这里可以看到右侧的 At time 是在不断变化的）



5.3.3 通过 Python 查询 TF 关系

（查询 base_link 到 bottle_link 话题）—— 从机械臂到水瓶

新建一个 tf_listener.py

完整代码：

```python
import rclpy
from rclpy.node import Node
from tf2_ros import TransformListener, Buffer # 坐标变换监听器
from transforms3d.euler import quat2euler  # 四元数转欧拉角
import math

# 定义一个ROS2节点，用于监听和查询坐标变换
class TFBroadcaster(Node):
    def __init__(self):
        # 初始化节点，设置节点名称为'tf_broadcaster'
        super().__init__('tf_broadcaster')
        self.buffer = Buffer() # 创建坐标变换缓冲区
        # 创建一个坐标变换监听器，监听缓冲区中的坐标变换
        self.listener = TransformListener(self.buffer, self)
        # 创建一个定时器，每隔0.1秒调用一次`get_transform`方法
        self.timer = self.create_timer(0.1, self.get_transform)

    def get_transform(self):

        '''
        实时查询坐标关系 buffer
        '''
        try:
            # 从缓冲区中查询两个坐标系之间的变换关系
            # 'base_link' 是源坐标系，'bottle_link' 是目标坐标系
            # rclpy.time.Time(seconds=0.0) 表示查询最新的变换
            # rclpy.time.Duration(seconds=1.0) 表示查询超时时间为1秒
            result = self.buffer.lookup_transform('base_link', 'bottle_link',
                    rclpy.time.Time(seconds=0.0), rclpy.time.Duration(seconds=1.0))
```

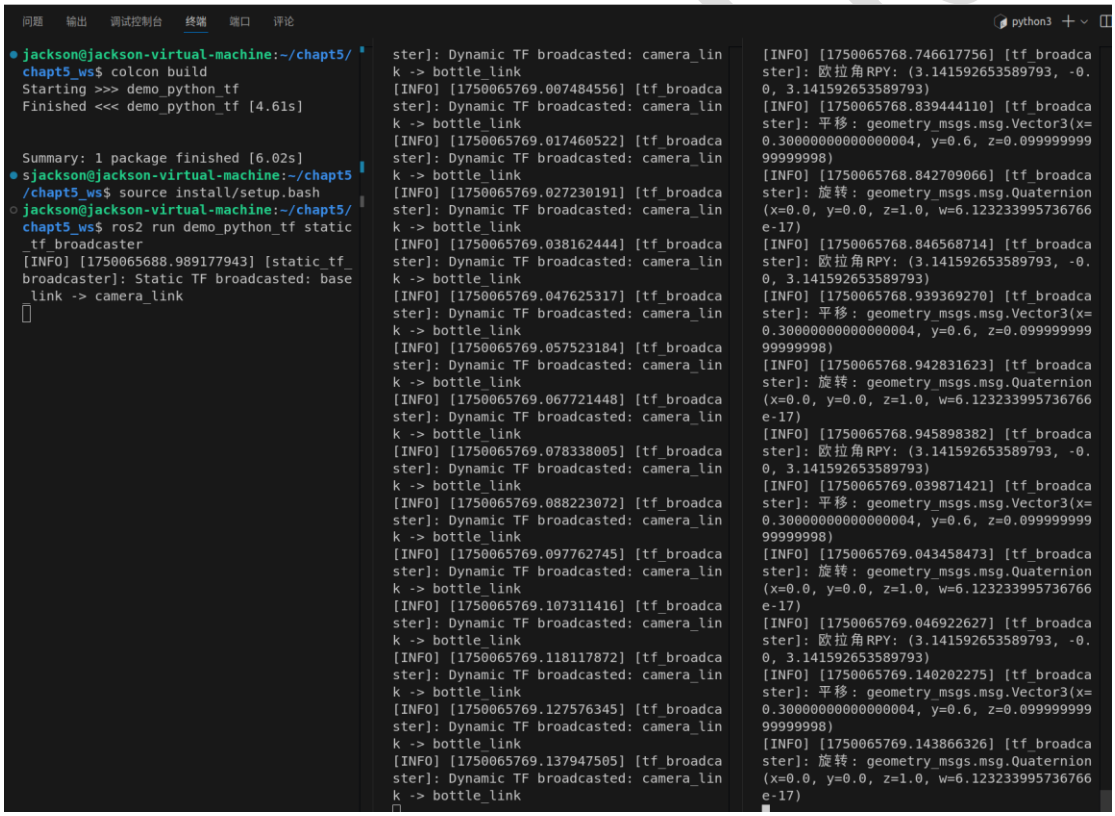
```python
            transform = result.transform # 获取变换关系中的坐标变换部分
            self.get_logger().info(f'平移: {transform.translation}')
            self.get_logger().info(f'旋转: {transform.rotation}')

            # 将四元数转换为欧拉角（RPY）
            rotation_euler = quat2euler([
                transform.rotation.x, transform.rotation.y,
                transform.rotation.z, transform.rotation.w])

            self.get_logger().info(f'欧拉角RPY: {rotation_euler}')
        except Exception as e:
            self.get_logger().error(f'获取坐标变换失败: 原因{str(e)}')

def main():
    rclpy.init()  # 初始化ROS2
    node = TFBroadcaster()  # 创建静态坐标广播器节点
    rclpy.spin(node)  # 保持节点运行
    node.destroy_node()  # 销毁节点
    rclpy.shutdown()  # 关闭ROS2
```

最终结果：



二． 常用可视化工具： rqt & RViz

此部分内容因为没有及时保存导致内容缺失，发现也并不难操作并且后续会经常用到，详细可参考下方 GPT 生成：

工具	主要用途	常用功能/插件	启动命令	保存方式
rqt	基于 Qt 的可视化工具集，调试/监控话题和节点	- rqt_graph（节点拓扑） - rqt_plot（实时曲线） - rqt_image_view（图像显示） - rqt_console（日志查看） - rqt_tf_tree（TF 树）	`rqt` 或 `rosrun rqt_graph rqt_graph`（ROS1） `rqt`（ROS2）	**Perspective**：保存/恢复布局
RViz/RViz2	3D 可视化工具，显示机器人模型、传感器数据和地图	- TF（坐标系） - RobotModel（URDF 模型） - LaserScan/PointCloud2（雷达/点云） - Map（栅格地图） - Path/Odometry（路径/里程计） - Image/Camera（相机图像）	`rviz`（ROS1） `rviz2`（ROS2）	**Config 文件**：保存为 `.rviz`，启动时 `-d` 加载