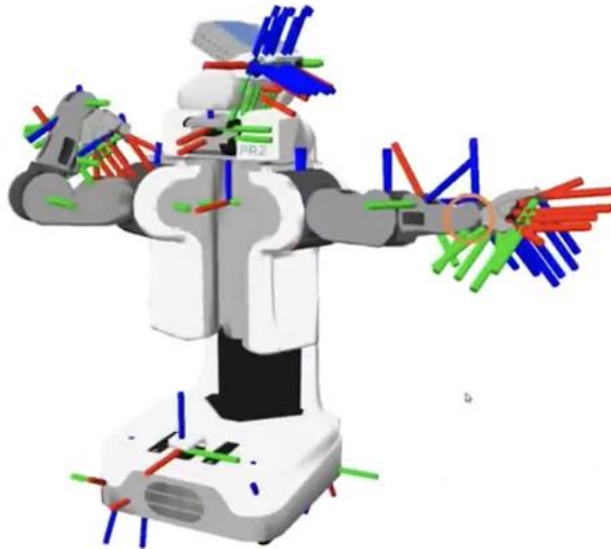


第五章： ROS2 中常用开发工具

一. ROS 坐标管理系统

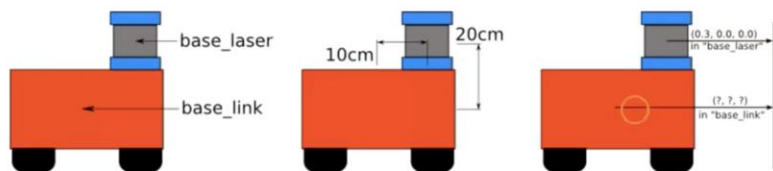
Transform：两个坐标之间的转换（旋转、平移）

（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！

三. TF静态坐标变换



```
<node pkg="tf" type="static_transform_publisher" name="broadcaster"
args="x y z yaw pitch roll link1_parent link1 period_in_ms" />
```

```
<node pkg="tf" type="static_transform_publisher" name="broadcaster"
args="x y z qx qy qz qw link1_parent link1 period_in_ms" />
```

所以说我们需要做机器人运动底盘和雷达之间的tf坐标变换

So we need to do the TF coordinate transformation between the robot chassis and the radar

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`

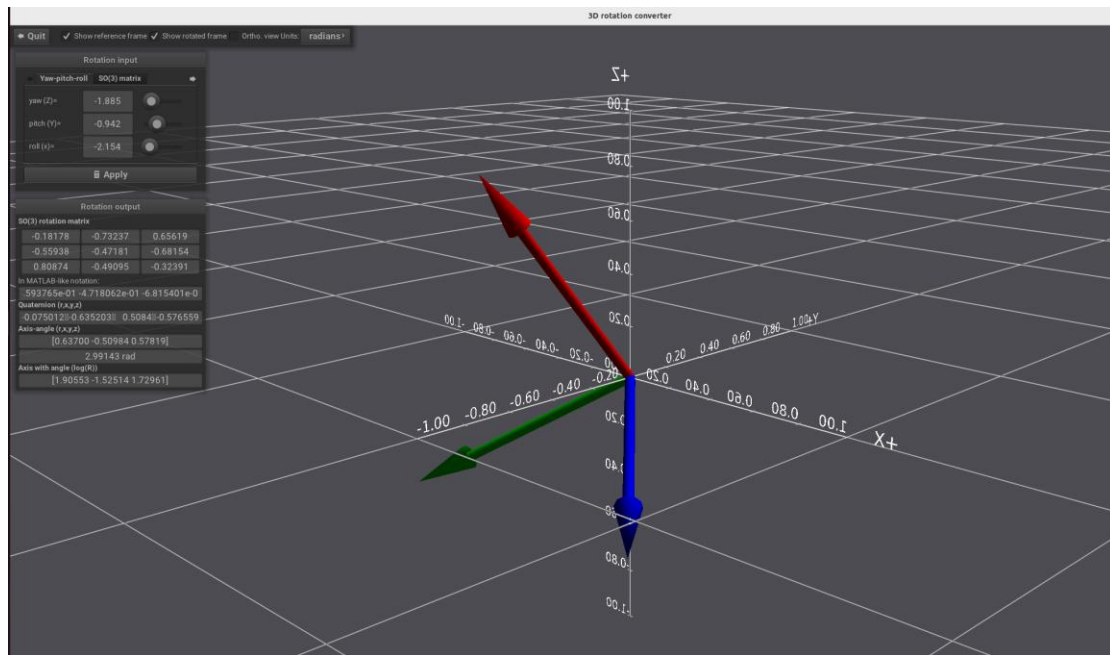
推荐工具安装

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`

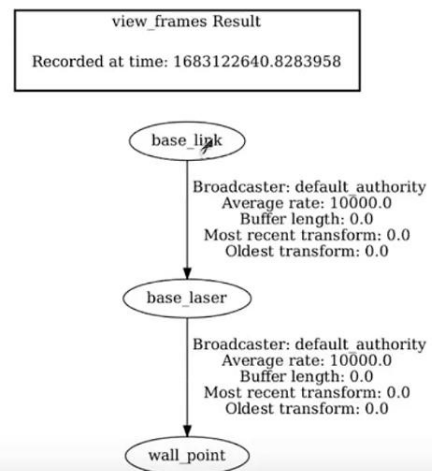
3D-rotation-converter



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



```

fishros@fishros-KLVC-WXX9: ~
fishros@fishros-KLVC-... x fishros@fishros-KLVC-... x
fig5:amd64 (1:5.10.1+dfsg-4build1) ...
:5.10.1+dfsg-4build1) ...
  
```

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

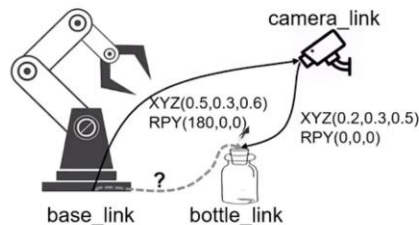
需求：

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者发布

Camera_link 和 bottle_link 关系通过静态坐标发布者发布

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

```
1 import rclpy
2 from rclpy.node import Node
3 from tf2_ros import StaticTransformBroadcaster # 静态坐标广播器
4 from geometry_msgs.msg import TransformStamped # 坐标变换消息
5 from transforms3d.euler import euler2quat as quaternion_from_euler # 欧拉角转四元数
6 import math
7
8 class StaticTFBroadcaster(Node):
9     def __init__(self):
10         super().__init__('static_tf_broadcaster')
11         self.static_broadcaster = StaticTransformBroadcaster(self) # 创建静态坐标广播器
12         self.publish_static_tf() # 在静态坐标广播器中订阅这个函数
13
14     # 发布静态坐标变换
15     def publish_static_tf(self):
16         '''
17         发布静态坐标变换：从base_link 到 camera_link之间的变换关系
18         '''
19
20         # 发的是消息接口的对象 — 创建一个 TransformStamped 消息
21         transform = TransformStamped()
22         transform.header.frame_id = 'base_link'
23         transform.child_frame_id = 'camera_link'
24         transform.header.stamp = self.get_clock().now().to_msg()
25
26         # 设置平移部分 (看PPT)
```

```

27     transform.transform.translation.x = 0.5
28     transform.transform.translation.y = 0.3
29     transform.transform.translation.z = 0.6
30
31     # 设置旋转部分
32     q = quaternion_from_euler(math.radians(180),0, 0) # 欧拉角转四元数 (返回是元组)
33     # 旋转部分赋值 (元组)
34     transform.transform.rotation.x = q[0]
35     transform.transform.rotation.y = q[1]
36     transform.transform.rotation.z = q[2]
37     transform.transform.rotation.w = q[3]
38     # 发布静态坐标变换
39     self.static_broadcaster.sendTransform(transform)
40     self.get_logger().info('Static TF broadcasted: base_link -> camera_link')
41
42 def main():
43     rclpy.init() # 初始化ROS2
44     node = StaticTFBroadcaster() # 创建静态坐标广播器节点
45     rclpy.spin(node) # 保持节点运行
46     node.destroy_node() # 销毁节点
47     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
ist
/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: 6.1232333995736766e-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

```

1 import rclpy
2 from rclpy.node import Node
3 from tf2_ros import TransformBroadcaster # 动态坐标广播器
4 from geometry_msgs.msg import TransformStamped # 坐标变换消息
5 from transforms3d.euler import euler2quat as quaternion_from_euler # 欧拉角转四元数
6 import math
7
8 class TFBroadcaster(Node):
9     def __init__(self):
10         super().__init__('tf_broadcaster')
11         self.broadcaster = TransformBroadcaster(self) # 创建动态坐标广播器
12         self.timer = self.create_timer(0.01, self.publish_tf) # 定时器, 每0.01秒调用一次函数
13
14         # 发布静态坐标变换
15         def publish_tf(self):
16             '''
17             发布静态坐标变换: 从bottle_link 到 camera_link之间的变换关系
18             '''
19
20             # 发的是消息接口的对象 — 创建一个 TransformStamped 消息
21             transform = TransformStamped()
22             transform.header.frame_id = 'camera_link'
23             transform.child_frame_id = 'bottle_link'
24             transform.header.stamp = self.get_clock().now().to_msg()
25
26             # 设置平移部分 (看PPT)
27             transform.transform.translation.x = 0.2
28             transform.transform.translation.y = 0.3
29             transform.transform.translation.z = 0.5
30
31             # 设置旋转部分
32             q = quaternion_from_euler(0,0,0) # 欧拉角转四元数 (返回是元组)
33             # 旋转部分赋值 (元组)
34             transform.transform.rotation.x = q[0]
35             transform.transform.rotation.y = q[1]
36             transform.transform.rotation.z = q[2]
37             transform.transform.rotation.w = q[3]
38             # 发布静态坐标变换
39             self.broadcaster.sendTransform(transform)
40             self.get_logger().info('Dynamic TF broadcasted: camera_link -> bottle_link')
41
42 def main():
43     rclpy.init() # 初始化ROS2
44     node = TFBroadcaster() # 创建静态坐标广播器节点
45     rclpy.spin(node) # 保持节点运行
46     node.destroy_node() # 销毁节点
47     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)

```
casted: camera_link -> bottle_link
[INFO] [1750063021.444192242] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link
[INFO] [1750063021.452882764] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link
[INFO] [1750063021.463144319] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link
[INFO] [1750063021.472618807] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link
[INFO] [1750063021.482596973] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link
[INFO] [1750063021.493297308] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link
[INFO] [1750063021.502573605] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link
[INFO] [1750063021.513149545] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link
[INFO] [1750063021.522756727] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link
[INFO] [1750063021.532579801] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link
[INFO] [1750063021.542149285] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link
[INFO] [1750063021.551273488] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link
[INFO] [1750063021.562098418] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link
[INFO] [1750063021.573464824] [tf broadcaster]: Dynamic TF broad
casted: camera_link -> bottle_link

- Translation: [0.200, 0.300, 0.500]
- Rotation: in Quaternion [1.000, 0.000, 0.000, 0.000]
- Rotation: in RPY (radian) [3.142, -0.000, 0.000]
- Rotation: in RPY (degree) [180.000, -0.000, 0.000]
- Matrix:
  1.000 0.000 0.000 0.200
  0.000 -1.000 0.000 0.300
  0.000 0.000 -1.000 0.500
  0.000 0.000 0.000 1.000
At time 1750063020.470207565
- Translation: [0.200, 0.300, 0.500]
- Rotation: in Quaternion [1.000, 0.000, 0.000, 0.000]
- Rotation: in RPY (radian) [3.142, -0.000, 0.000]
- Rotation: in RPY (degree) [180.000, -0.000, 0.000]
- Matrix:
  1.000 0.000 0.000 0.200
  0.000 -1.000 0.000 0.300
  0.000 0.000 -1.000 0.500
  0.000 0.000 0.000 1.000
At time 1750063021.468799373
- Translation: [0.200, 0.300, 0.500]
- Rotation: in Quaternion [1.000, 0.000, 0.000, 0.000]
- Rotation: in RPY (radian) [3.142, -0.000, 0.000]
- Rotation: in RPY (degree) [180.000, -0.000, 0.000]
- Matrix:
  1.000 0.000 0.000 0.200
  0.000 -1.000 0.000 0.300
  0.000 0.000 -1.000 0.500
  0.000 0.000 0.000 1.000

转到行列
行 22, 列 44 空格: 4 UTF-8 LF {} P
的 406 个文件和 0 个单元格
```

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：

```
1 import rclpy
2 from rclpy.node import Node
3 from tf2_ros import TransformListener, Buffer # 坐标变换监听器
4 from transforms3d.euler import quat2euler # 四元数转欧拉角
5 import math
6
7 # 定义一个ROS2节点，用于监听和查询坐标变换
8 class TFBroadcaster(Node):
9     def __init__(self):
10         # 初始化节点，设置节点名称为'tf_broadcaster'
11         super().__init__('tf_broadcaster')
12         self.buffer = Buffer() # 创建坐标变换缓冲区
13         # 创建一个坐标变换监听器，监听缓冲区中的坐标变换
14         self.listener = TransformListener(self.buffer, self)
15         # 创建一个定时器，每隔0.1秒调用一次`get_transform`方法
16         self.timer = self.create_timer(0.1, self.get_transform)
17
18     def get_transform(self):
19         '''
20         实时查询坐标关系 buffer
21         '''
22         try:
23             # 从缓冲区中查询两个坐标系之间的变换关系
24             # 'base_link' 是源坐标系, 'bottle_link' 是目标坐标系
25             # rclpy.time.Time(seconds=0.0) 表示查询最新的变换
26             # rclpy.time.Duration(seconds=1.0) 表示查询超时时间为1秒
27             result = self.buffer.lookup_transform('base_link', 'bottle_link',
28                                                  rclpy.time.Time(seconds=0.0), rclpy.time.Duration(seconds=1.0))
```

```

30         transform = result.transform # 获取变换关系中的坐标变换部分
31         self.get_logger().info(f'平移: {transform.translation}')
32         self.get_logger().info(f'旋转: {transform.rotation}')
33
34         # 将四元数转换为欧拉角 (RPY)
35         rotation_euler = quat2euler([
36             transform.rotation.x, transform.rotation.y,
37             transform.rotation.z, transform.rotation.w])
38
39         self.get_logger().info(f'欧拉角RPY: {rotation_euler}')
40     except Exception as e:
41         self.get_logger().error(f'获取坐标变换失败: 原因{str(e)}')
42
43 def main():
44     rclpy.init() # 初始化ROS2
45     node = TFBroadcaster() # 创建静态坐标广播器节点
46     rclpy.spin(node) # 保持节点运行
47     node.destroy_node() # 销毁节点
48     rclpy.shutdown() # 关闭ROS2

```

最终结果:

```

问题  输出  调试控制台  终端  端口  评论
python3  + ~ 11

• jackson@jackson-virtual-machine:~/chapt5/
chapt5_ws$ colcon build
Starting >>> demo_python_tf
Finished <<< demo_python_tf [4.61s]

Summary: 1 package finished [6.02s]
• sjackson@jackson-virtual-machine:~/chapt5/
/chapt5_ws$ source install/setup.bash
• jackson@jackson-virtual-machine:~/chapt5/
chapt5_ws$ ros2 run demo_python_tf static
tf broadcaster
[INFO] [1750065688.989177943] [static_tf_
broadcaster]: Static TF broadcasted: base
link -> camera_link
[]

ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.007484556] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.017460522] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.027230191] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.038162444] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.047625317] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.057523184] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.067721448] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.078338005] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.088223072] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.097762745] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.107311416] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.118117872] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.127576345] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065769.137947505] [tf_broadca
ster]: Dynamic TF broadcasted: camera_lin
k -> bottle_link
[INFO] [1750065768.746617756] [tf_broadca
ster]: 欧拉角RPY: (3.141592653589793, -0.
0, 3.141592653589793)
[INFO] [1750065768.839444110] [tf_broadca
ster]: 平移: geometry_msgs.msg.Vector3(x=
0.30000000000000004, y=0.6, z=0.099999999
99999998)
[INFO] [1750065768.842709066] [tf_broadca
ster]: 旋转: geometry_msgs.msg.Quaternion
(x=0.0, y=0.0, z=1.0, w=6.123233995736766
e-17)
[INFO] [1750065768.846568714] [tf_broadca
ster]: 欧拉角RPY: (3.141592653589793, -0.
0, 3.141592653589793)
[INFO] [1750065768.939369270] [tf_broadca
ster]: 平移: geometry_msgs.msg.Vector3(x=
0.30000000000000004, y=0.6, z=0.099999999
99999998)
[INFO] [1750065768.942831623] [tf_broadca
ster]: 旋转: geometry_msgs.msg.Quaternion
(x=0.0, y=0.0, z=1.0, w=6.123233995736766
e-17)
[INFO] [1750065768.945898382] [tf_broadca
ster]: 欧拉角RPY: (3.141592653589793, -0.
0, 3.141592653589793)
[INFO] [1750065769.039871421] [tf_broadca
ster]: 平移: geometry_msgs.msg.Vector3(x=
0.30000000000000004, y=0.6, z=0.099999999
99999998)
[INFO] [1750065769.043458473] [tf_broadca
ster]: 旋转: geometry_msgs.msg.Quaternion
(x=0.0, y=0.0, z=1.0, w=6.123233995736766
e-17)
[INFO] [1750065769.046922627] [tf_broadca
ster]: 欧拉角RPY: (3.141592653589793, -0.
0, 3.141592653589793)
[INFO] [1750065769.140202275] [tf_broadca
ster]: 平移: geometry_msgs.msg.Vector3(x=
0.30000000000000004, y=0.6, z=0.099999999
99999998)
[INFO] [1750065769.143866326] [tf_broadca
ster]: 旋转: geometry_msgs.msg.Quaternion
(x=0.0, y=0.0, z=1.0, w=6.123233995736766
e-17)

```

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)	<pre>rqt 或 roslaunch rqt_graph rqt_graph (ROS1) rqt (ROS2)</pre>	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)	<pre>rviz (ROS1) rviz2 (ROS2)</pre>	Config 文件: 保存为 .rviz，启动时 -d 加载