10 TF&URDF

方宝富

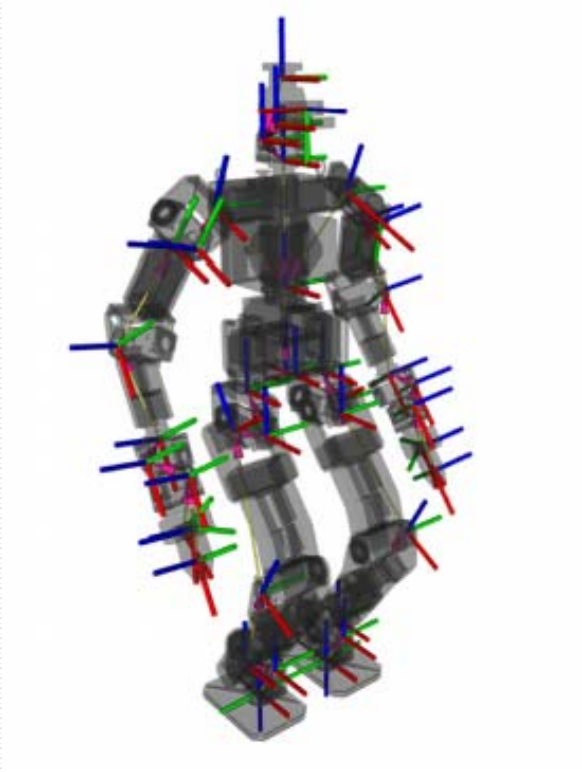
fangbf@hfut.edu.cn

提纲

- 坐标转换系统----TF
- 统一机器人描述格式----URDF

10.1 坐标转换系统：TF

10.1 坐标转换系统：TF



TF是一个ROS世界里的一个基本的也是很重要的概念，所谓TF(TransForm)，就是坐标转换.

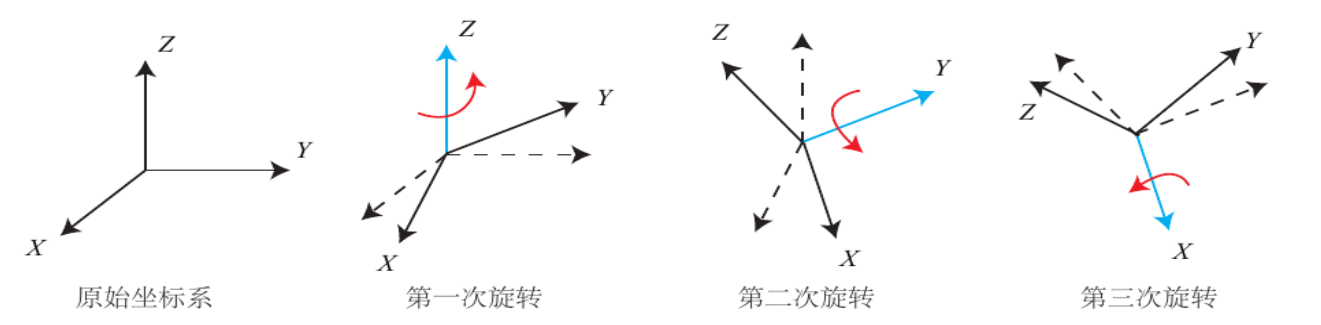
坐标转换TF在描述组成机器人的每个部分、障碍物和外部物体时是最有用的概念之一。

10.1 坐标转换系统：TF

当机器人的“眼睛”获取一组数据（物体的方位坐标），但是相对于机器人手臂来说，这个坐标只是相对于机器人头部的传感器，并不直接适用于机器人手臂执行，那么物体相对于头部和手臂之间的坐标转换，就是TF.

10.1 坐标转换系统：TF

1.欧拉角（euler angle）

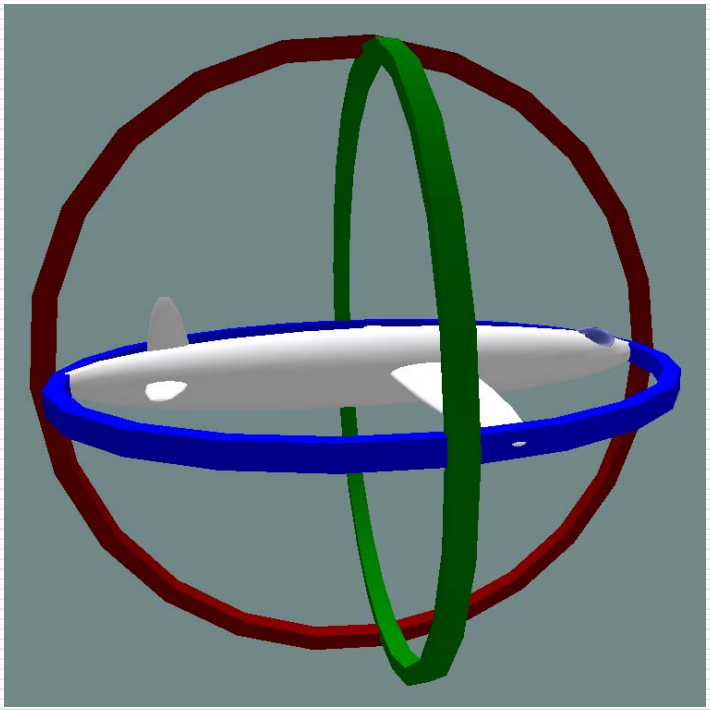


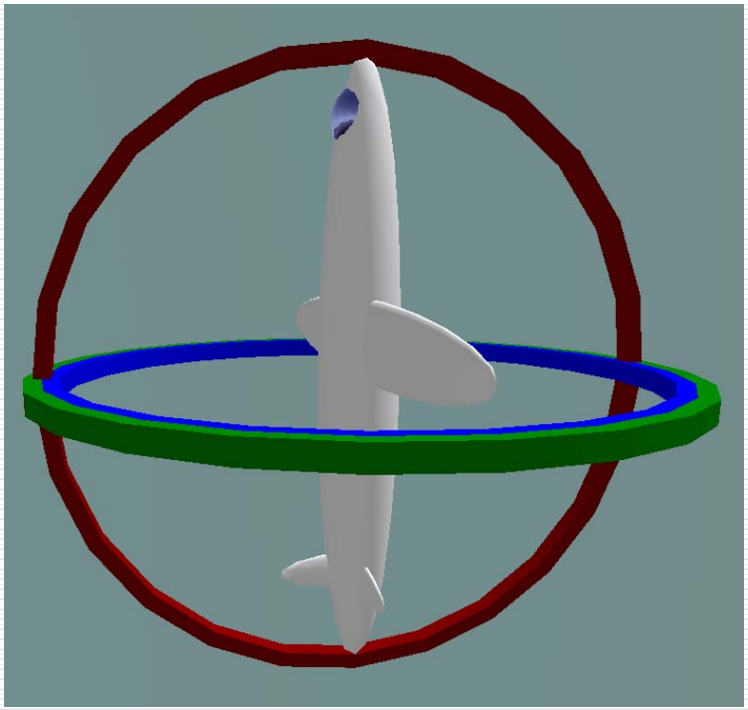
$$\boldsymbol{R}(z,\theta)=\begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{R}(y,\theta)=\begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

$$\boldsymbol{R}(x,\theta)=\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix}$$

$$\boldsymbol{R}=\boldsymbol{R}(x,\theta)\boldsymbol{R}(y,\theta)\boldsymbol{R}(z,\theta)$$

10.1 坐标转换系统：TF





10.1 坐标转换系统：TF

2.四元数（quaternion）

$$q = w + xi + yj + zk$$

$$q = [w, x, y, z]^T$$

$$\begin{cases} i^2 = j^2 = k^2 = -1 \\ ij = k, jk = -k \\ jk = i, kj = -i \\ ki = j, ik = -j \end{cases}$$

$$|q|^2 = w^2 + x^2 + y^2 + z^2 = 1$$

10.1 坐标转换系统：TF

欧拉角转四元数

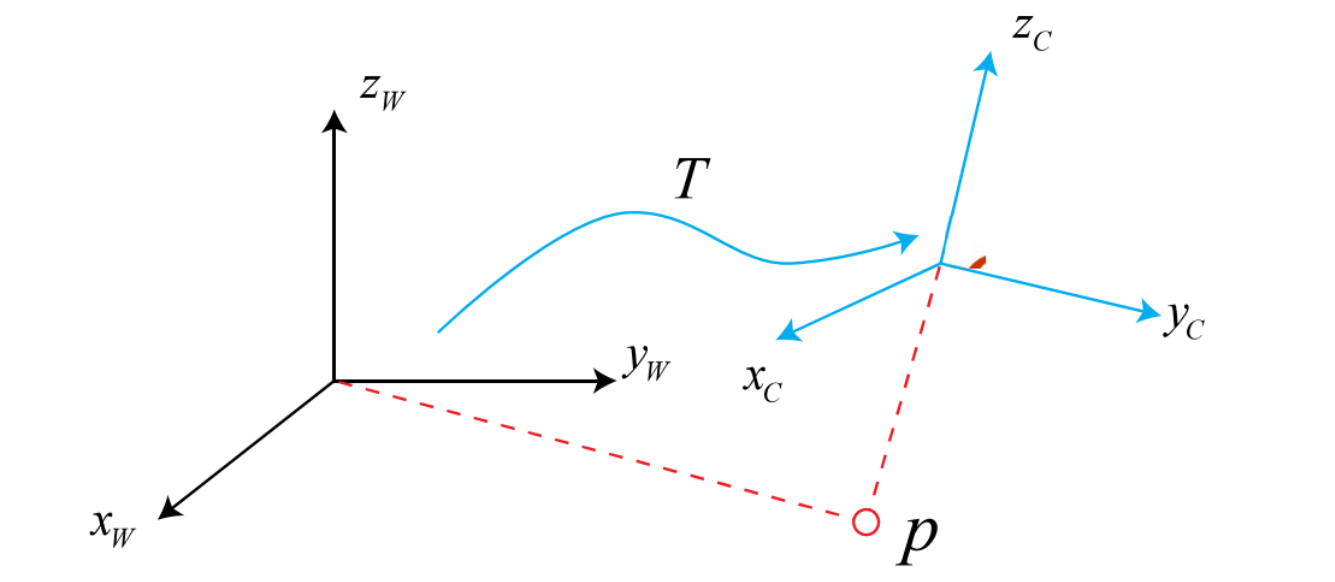
$$q=\begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix}=\begin{bmatrix} \cos(roll/2)\cos(pitch/2)\cos(yaw/2)+\sin(roll/2)\sin(roll/2)\sin(yaw/2) \\ \sin(roll/2)\cos(pitch/2)\cos(yaw/2)-\cos(roll/2)\sin(roll/2)\sin(yaw/2) \\ \cos(roll/2)\sin(pitch/2)\cos(yaw/2)+\sin(roll/2)\cos(roll/2)\sin(yaw/2) \\ \cos(roll/2)\cos(pitch/2)\sin(yaw/2)-\sin(roll/2)\sin(roll/2)\cos(yaw/2) \end{bmatrix}$$

四元数转欧拉角

$$\begin{bmatrix} roll \\ pitch \\ yaw \end{bmatrix}=\begin{bmatrix} a\tan 2(2(wx+yz)),1-2(x^2+y^2) \\ \arcsin(2(wy-zx)) \\ a\tan 2(2(wz+xy)),1-2(y^2+z^2) \end{bmatrix}$$

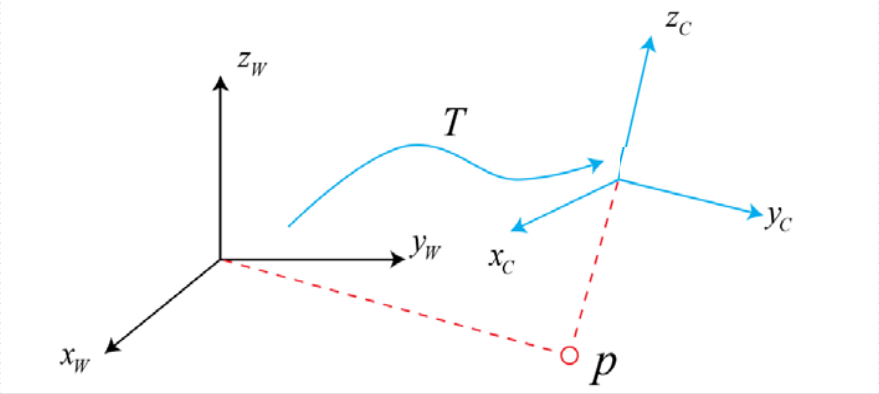
10.1 坐标转换系统：TF

3.刚体的位姿表示



10.1 坐标转换系统：TF

3.刚体的位姿表示



(1) 位置　　**(2) 方位**

$$^{w}C = \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \quad {}^{W}_{C}\mathrm{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

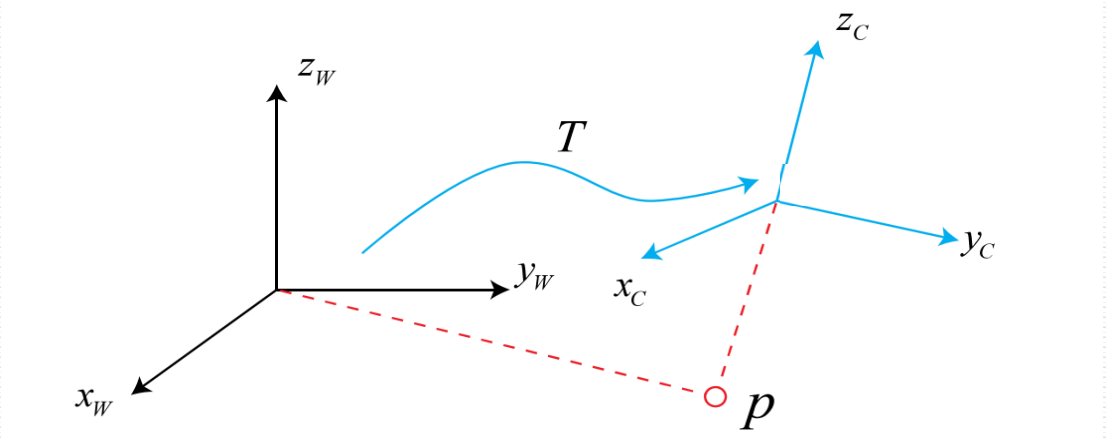
$$\boldsymbol{R}(x,\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix}$$

$$\boldsymbol{R}(y,\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

$$\boldsymbol{R}(z,\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ 0 & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

10.1 坐标转换系统：TF

3.刚体的位姿表示



非齐次变换

$$ {}^{w}p = {}^{W}_{C}R\ {}^{c}p + {}^{W}C $$

齐次变换

$$ \begin{bmatrix} {}^{w}p \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{W}_{C}R & {}^{W}C \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{c}p \\ 1 \end{bmatrix} $$

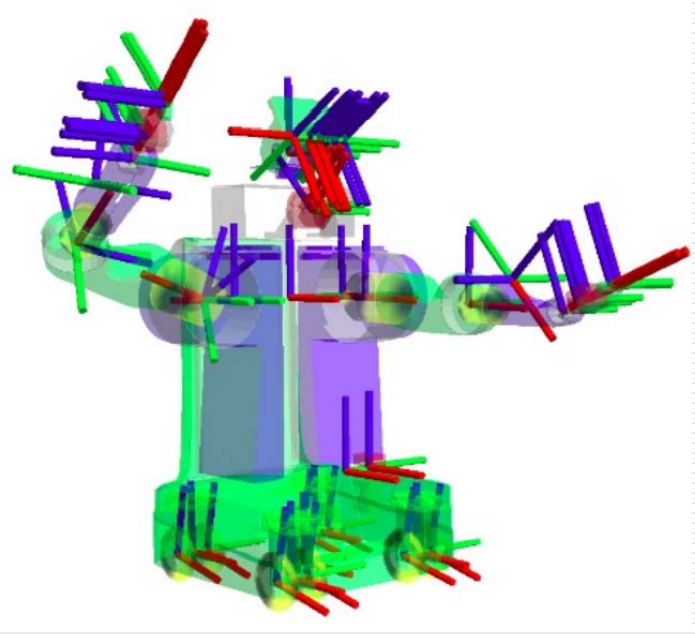
10.1 坐标转换系统：TF

tf

坐标转换的标准、话题、工具、接口

tf package, tf topic

10.1 坐标转换系统：TF

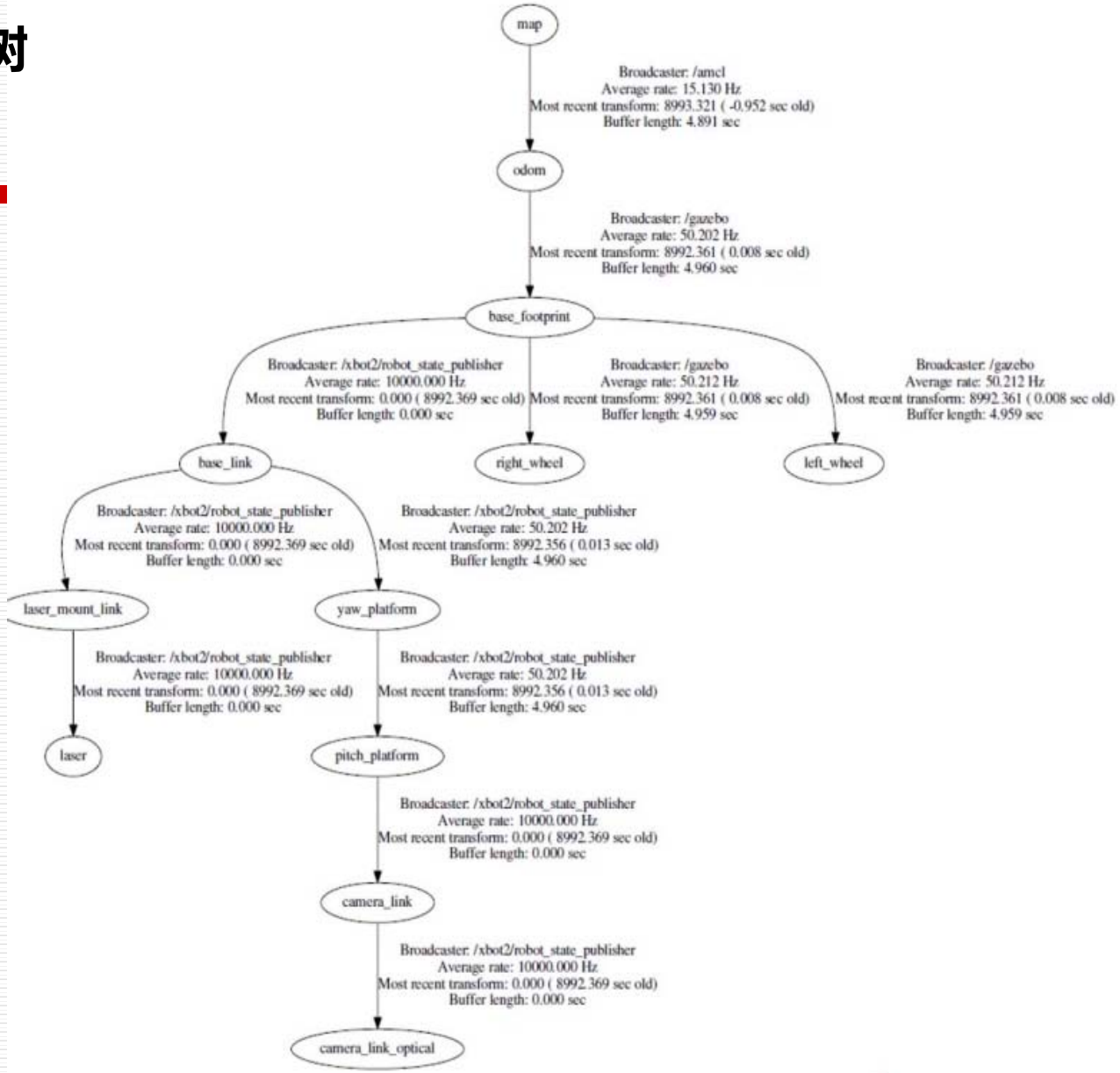


TF可以被当做是一种标准规范，这套标准定义了坐标转换的数据格式和数据结构，tf本质是树状的数据结构，所以我们通常称之为“tf tree”

观察左图，可以看到ROS数据结构的一个抽象图，ROS中机器人模型包含大量的部件，这些部件统称之为link,每一个link上面对应着一个frame, 即一个坐标系.

link和frame概念是绑定在一起的．像左图模型中有很多的frame,错综复杂的铺置在机器人的各个link上，维护各个坐标系之间的关系，就要靠着tf tree来处理，维护着各个坐标系之间的联通.

TF树



每一个圆圈代表一个frame,对应着机器人上的一个link

两个frame间会有一个broadcaster，这是为了使得两个frame能够连通，会有一个node broadcaster两个frame间的相对运动信息。可以看到两个frame之间的消息node和broadcaster，这些都是在发布tf消息。

10.1 坐标转换系统：TF

TF消息格式：TransformStamped.msg

-Header：

序号

时间

frame的名称

子frame名称

-变换规则

Vector3三维向量表示平移

Quaternion四元数表示旋转

```
std_msgs/Header header
        uint32 seq
        time stamp
        string frame_id
        string child_frame_id
geometry_msgs/Transform transform
        geometry_msgs/Vector3 translation
                float64 x
                float64 y
                float64 z
        geometry_msgs/Quaternion rotation
                float64 x
                float64 y
                float64 z
                float64 w
```

10.1 坐标转换系统：TF

tf/tfMessage.msg

tf2_msgs/TFMessage.msg

10.1 坐标转换系统：TF

tf/tfMessage.msg 或 tf2_msgs/TFMessage

```
geometry_msgs/TransformStamped[] transforms
    std_msgs/Header header
        uint32 seq
        time stamp
        string frame_id
      string child_frame_id
    geometry_msgs/Transform transform
        geometry_msgs/Vector3 translation
                float64 x
                float64 y
                float64 z
        geometry_msgs/Quaternion rotation
                float64 x
                float64 y
                float64 z
                float64 w
```

10.1 坐标转换系统：TF

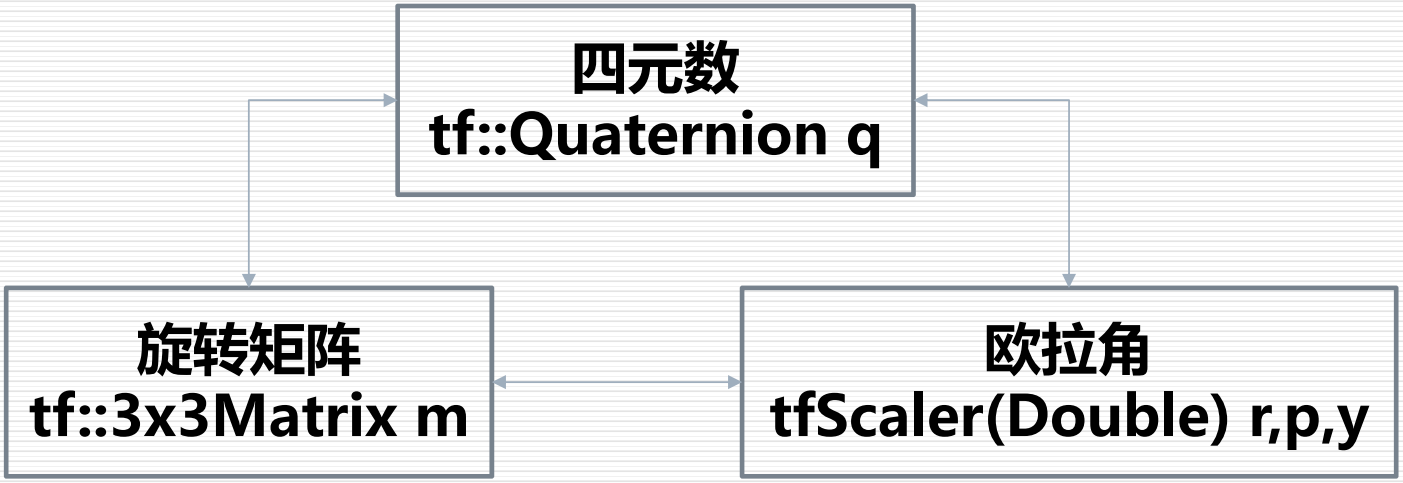
tf in c++

10.1 坐标转换系统：TF

TF相关数据类型

向量	tf::Vector3
点	tf::Point
四元数	tf::Quaternion
3x3矩阵（旋转矩阵）	tf::Matrix3x3
位姿	tf::Pose
变换	tf::Transform
带时间戳的以上类型	tf::Stamped<T>
带时间戳的变换	tf::StampedTransform

10.1 坐标转换系统：TF



10.1 坐标转换系统：TF

tf::TransformBroadcaster类

TransformBroadcaster ()

void sendTransform (const StampedTransform &transform)

void sendTransform (const std::vector< StampedTransform > &transforms)

void sendTransform (const geometry_msgs::TransformStamped &transform)

void sendTransform (const std::vector<geometry_msgs::TransformStamped>
&transforms)

10.1 坐标转换系统：TF

tf::TransformListener类

ros::Time(0);//最新的

ros::Time::now() //不太适合

void lookupTransform(const std::string &target_frame,

const std::string &source_frame, const ros::Time &time,

StampedTransform &transform) const

bool canTransform()

bool waitForTransform() const

10.1 坐标转换系统：TF

tf in python

10.1 坐标转换系统：TF

TF相关数据类型

向量、点、四元数、矩阵都表示成类似数组形式

Tuple，List，Numpy Array通用

例如

t = (1.0, 1.5, 0)　#平移

q = [1, 0, 0, 0]　#四元数

m = numpy.identity(3)　#旋转矩阵

10.1 坐标转换系统：TF

tf.transformations

import tf

基本数学运算函数

euler_matrix(ai, aj, ak, axes= ‘sxyz’) #欧拉角到矩阵

euler_from_matrix(matrix, axes= ‘sxyz’) #矩阵到欧拉角

euler_from_quaternion(quaternion, axes= ‘sxyz’) #四元数到欧拉角

quaternion_from_euler(ai, aj, ak, axes= ‘sxyz’) #欧拉角到四元数

quaternion_matrix(quaternion) #四元数到矩阵

quaternion_from_matrix(matrix) #矩阵到四元数

....

10.1 坐标转换系统：TF

tf.TransformListener类

canTransform(self, target_frame, source_frame, time)

#frame是否相通

waitForTransform(self, target_frame, source_frame, time, timeout)

#阻塞直到frame相通

lookupTransform(self, target_frame, source_frame, time)

#查看相对的tf,返回(trans, quat)

10.1 坐标转换系统：TF

tf.TransformListener类

chain(target_frame, target_time, source_frame, source_time, fixed_frame) #frame的连接关系

frameExists(self, frame_id) #frame是否存在

getFrameStrings(self) #返回所有tf的名称

fromTranslationRotation(translation, rotation) #根据平移和旋转返回4x4矩阵

transformPoint(target_frame, point_msg)#将PointStamped消息转换到新frame下

transformPose(target_frame, pose_msg)#将PoseStamped消息转换到新frame下

transformQuaternion(target_frame, quat_msg) #将QuaternionStamped…

#返回相同类型

10.1 坐标转换系统：TF

tf.TransformBroadcaster类

sendTransform(translation, rotation, time, child, parent) # 向/tf发布消息

sendTransformMessage(transform)　# 向/tf发布消息

10.1 坐标转换系统：TF

View frame

作用：订阅5秒topic的信息，根据这段时间的信息，生成一个tf tree的pdf图

用法：rosrun tf view_frames

rqt_tf_tree

作用：查看当前的tf tree，动态的查询tf tree，当前的变化都可以看到

用法：rosrun rqt_tf_tree rqt_tf_tree

tf_echo

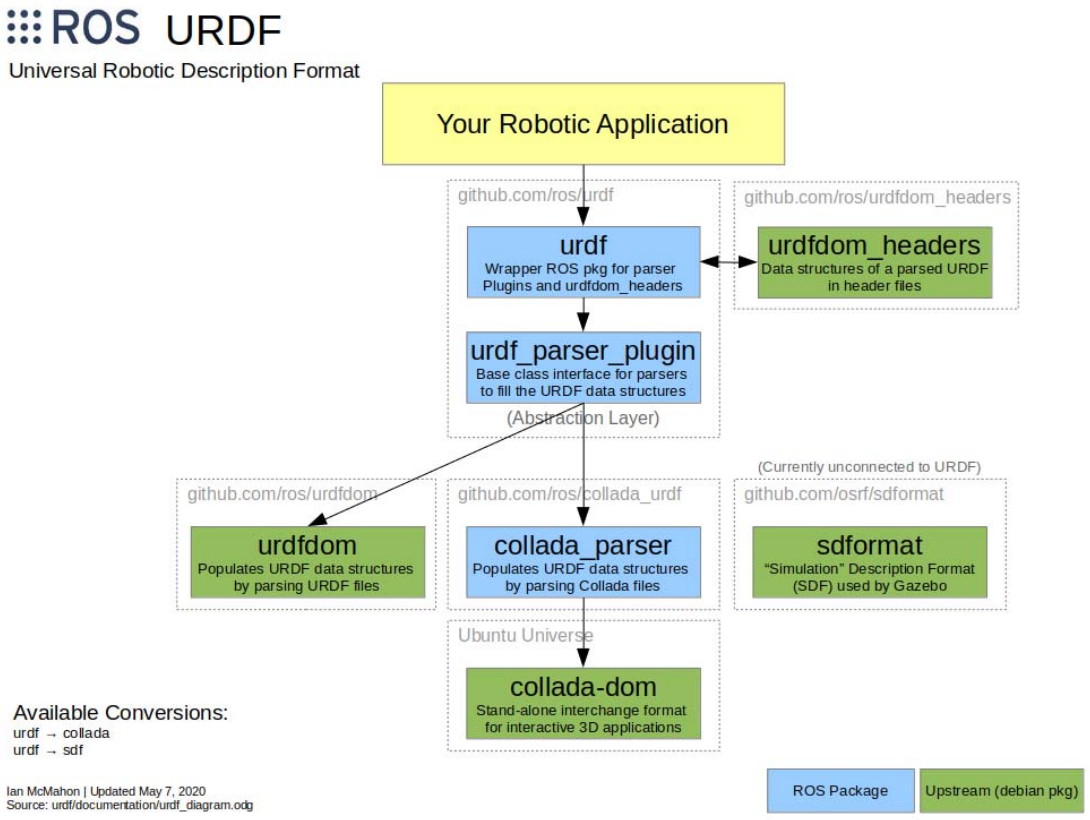
作用：查看两个frame间的变换关系

用法：rosrun tf tf_echo [reference_frame][target_frame]

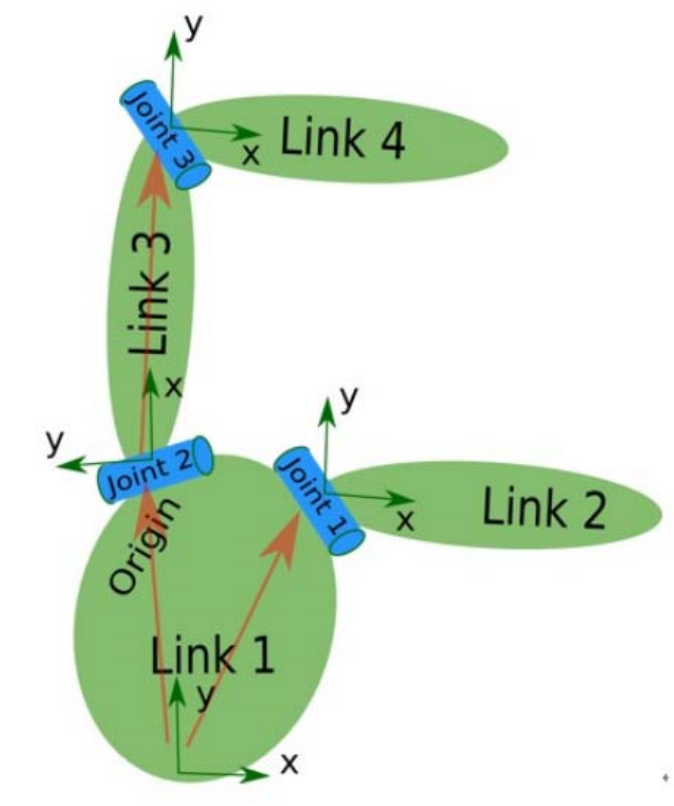
10.2 统一机器人描述格式：URDF

URDF(Unified Robot Description Format)

10.2 统一机器人描述格式：URDF



10.2 统一机器人描述格式：URDF



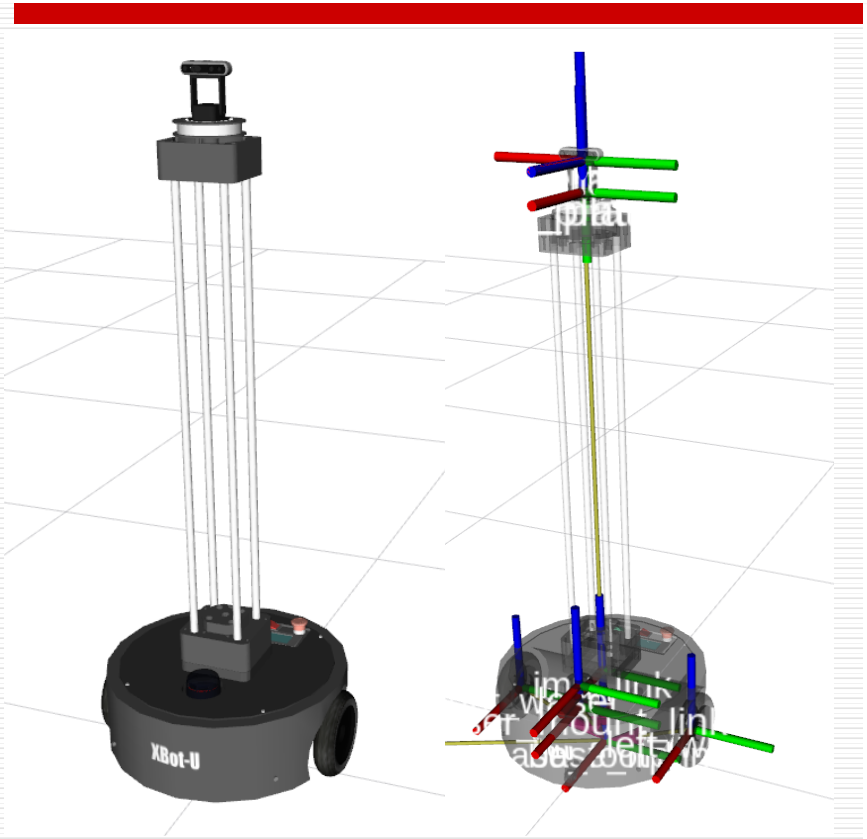
URDF（Unified Robot Description Format）统一机器人描述格式

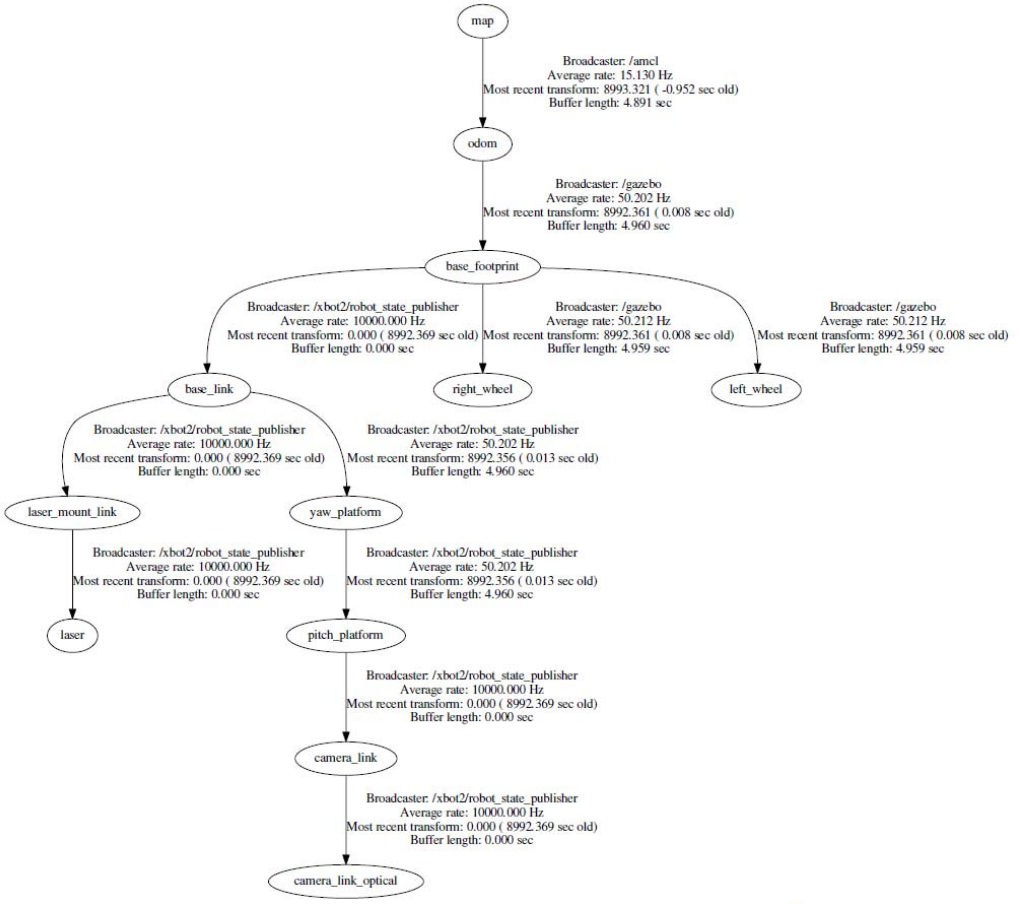
URDF使用XML格式描述机器人文件。

机器人的TF维护、可视化、仿真等功能都得由URDF的参与。

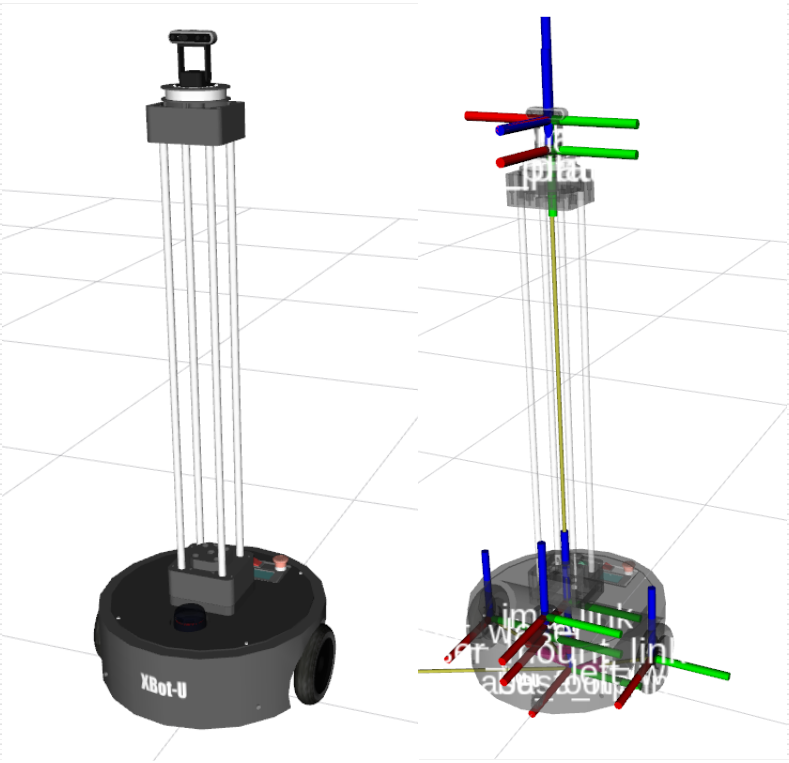
urdf package

10.2 统一机器人描述格式：URDF





10.2 统一机器人描述格式：URDF



xbot-u.urdf.xacro

```
  <xacro:macro name="xbot2_base">
  <link name="base_footprint"/>
  <joint name="base_footprint_to_base" type="fixed">…
  </joint>

  <link name="base_link">…
  </link>

  <link name="laser_mount_link"/>
  <joint name="base_to_laser" type="fixed">…
  </joint>

  <link name="imu_link"/>
  <joint name="base_to_imu" type="fixed">…
  </joint>

  <link name="left_wheel">…
  </link>

  <joint type="continuous" name="left_wheel_hinge">…
  </joint>

  <link name="right_wheel">…
  </link>

  <joint type="continuous" name="right_wheel_hinge">…
  </joint>

  <link name="yaw_platform">
    <inertial>
      <mass value="1e-5" />
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <inertia ixx="1e-6" ixy="0" ixz="0" iyy="1e-6" iyz="0" izz="1e-6" />
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="0 0 1.57079632" />
      <geometry>
```

10.2 统一机器人描述格式：URDF

```
<link>
    <inertial>        惯性属性
        <origin>      相对link的坐标
        <mass>        质量
        <inertial>    转动惯量
    <visual>          视觉属性
        <origin>      相对link的坐标
        <geometry>    形状
        <material>    材质
    <collision>       碰撞属性
        <origin>      相对坐标
        <geometry>    形状
```

```
<joint>
    <origin>      从父link到子link的变换
    <parent>      父link
    <child>       子link
    <axis>        关节轴
    <calibration>   视觉属性
    <dynamics>      动力学参数
    <limit>       关节限位
```

10.2 统一机器人描述格式：URDF

示例：构建简单小车模型

```
<robot name="mycar">
  <link name="base_link" />
  <link name="right" />
  <link name="left" />
  <link name="rplidar" />

  <joint name="right_joint" type="continuous">
    <parent link="base_link"/>
    <child link="right"/>
  </joint>

  <joint name="letf_joint" type="continuous">
    <parent link="base_link"/>
    <child link="left"/>
  </joint>

  <joint name="rplidar_joint" type="fixed">
    <parent link="base_link"/>
    <child link="rplidar"/>
  </joint>
</robot>
```

构建base_link作为小车的父坐标系；

在base_link基础上，再构建左轮，右轮 和雷达 link；

最后不同的link之间通过joint来连接。代码如图所示

10.2 统一机器人描述格式：URDF

示例：构建简单小车模型

```
<robot name="mycar">
  <link name="base_link" />
  <link name="right" />
  <link name="left" />
  <link name="rplidar" />

  <joint name="right_joint" type="continuous">
    <parent link="base_link"/>
    <child link="right"/>
    <origin rpy="1.57075  0  0" xyz="0 -0.2 0.07"/>
  </joint>

  <joint name="letf_joint" type="continuous">
    <parent link="base_link"/>
    <child link="left"/>
    <origin rpy="-1.57075 0 0" xyz="0 0.2 0.07"/>
  </joint>

  <joint name="rplidar_joint" type="fixed">
    <parent link="base_link"/>
    <child link="rplidar"/>
    <origin xyz="0.2 0 0.12"/>
  </joint>
</robot>
```

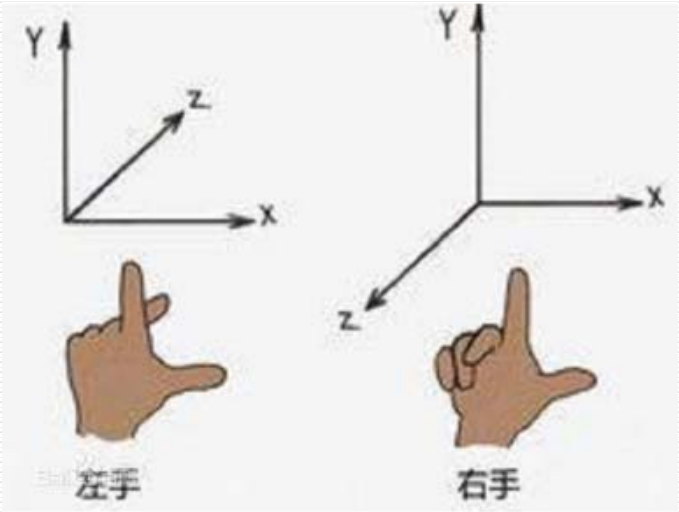
（2）添加机器人link之间的相对位置关系

在基础模型之上，我们需要为机器人之间link来设相对位置和朝向的关系，URDF中通过<origin>来描述这种关系

10.2 统一机器人描述格式：URDF

三维坐标系

三维坐标系使用的是右手坐标系，上图是tf_tree，xyz：0 -0.2 0.7，表示x轴平移0个单位，y轴平移-0.2个单位，z轴平移0.7个单位。rpy表示旋转的欧拉角对应的三个分量。



空间中的坐标系是右手坐标系

10.2 统一机器人描述格式：URDF

示例：构建简单小车模型

```
<robot name="mycar">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length=".06" radius="0.27"></cylinder>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0.1"/>
      <material name="white">
        <color rgba="1 1 1 1"/>
      </material>
    </visual>
  </link>

  <link name="right">
    <visual>
      <geometry>
        <cylinder length="0.04" radius="0.07"/>
      </geometry>
      <origin rpy="0  0  0" xyz="0 0 0"/>
      <material name="black">
        <color rgba="0 0 0 1"/>
      </material>
    </visual>
  </link>

  <link name="left">
    <visual>
      <geometry>
        <cylinder length="0.04" radius="0.07"/>
      </geometry>
      <origin rpy="0  0  0" xyz="0 0 0"/>
      <material name="black"/>
    </visual>
  </link>
```

（3）添加模型的尺寸，形状和颜色等

在已经设置好模型的link基础上，添加模型的形状（例如圆柱或长方体），相对于link的位置，颜色等。其中形状用<geometry>来描述，颜色用<color>来描述。

10.2 统一机器人描述格式：URDF

示例：构建简单小车模型