**内部资料，注意保密**

# ROS机器人开发技术

# 中国大学MOOC

2018年7月V1.0



**中科重德智能有限公司**

## 前言

TF是一个ROS世界里的基本而十分重要的概念，指的是坐标转换，在现实生活中，我们做出各种行为模式都可以很快的完成，像拿起身边的物品，但是在机器人世界里，同样的过程就会变得精确而复杂，一个机器人想要做到一件事情，比如抓取东西，就需要知道自己的位置和姿态，然后每个部件所需要的位置和姿态和同样的物体对于各部分的位置，而这么多的位置和姿态，就需要通过坐标的转换来实现，就是TF。

TF（Transform）是一个可以使用户追溯到多个坐标系的实时状态的工具包。通过实时的缓存坐标系树，Tf维持着树状结构坐标系间的关系，并且可以使用户能够在期望的时间点的任意两个坐标系间进行空间点、向量的转换。

urdf（Unified Robot Description Format）是统一机器人描述格式。ROS中的urdf功能包包含一个URDF的C++解析器，URDF文件使用XML格式描述机器人模型。

# 第六章 tf 和 urdf

### 学习要求：

1. 掌握坐标系系统的常见变换，能够交互的查看TF在模型中的变化，了解 TF broadcaster 和 TF listener，了解 TF in rospy。

2、了解 URDF 文件使用XML格式描述机器人模型的具体内容

本章节主要介绍TF和URDF。TF指的是机器人中的坐标系变换，对于后面的路径规划（SLAM）、机械臂的运动等相关学习内容，都要涉及到TF。URDF指的是统一机器人描述格式，里面会定义具体的一些机器人的模型，而URDF会反应出机器人的关节和链接关系。

**相关基本概念：**

TF、TF broadcaster、TF listener、TF tree、URDF

## 6.1 TF内容相关教学：

### 6.1.1 案例：利用demo引入TF的相关教学

首先，我们来演示一个案例：

我们所使用的ros版本：kinetic；编译系统：catkin。

在终端输入以下指令：

$ sudo apt-get install ros-kinetic-ros-tutorials ros-kinetic-geometry-tutorials ros-kinetic-rviz ros-kinetic-rosbash ros-kinetic-rqt-tf-tree

安装tutorial的节点，如果已经安装过会有如下截图：

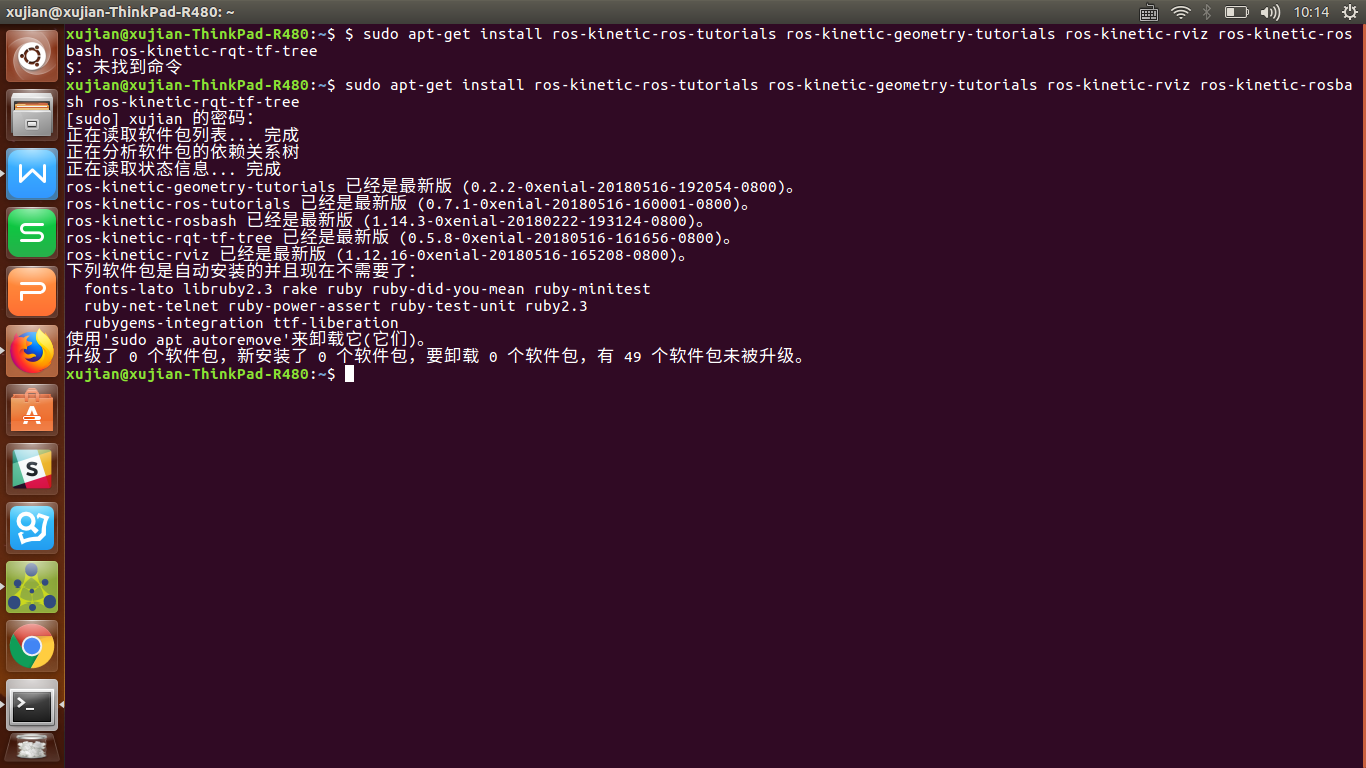


图1-1 安装节点图

之后，运行demo:

$ roslaunch turtle\_tf turtle\_tf\_demo.launch

出现：

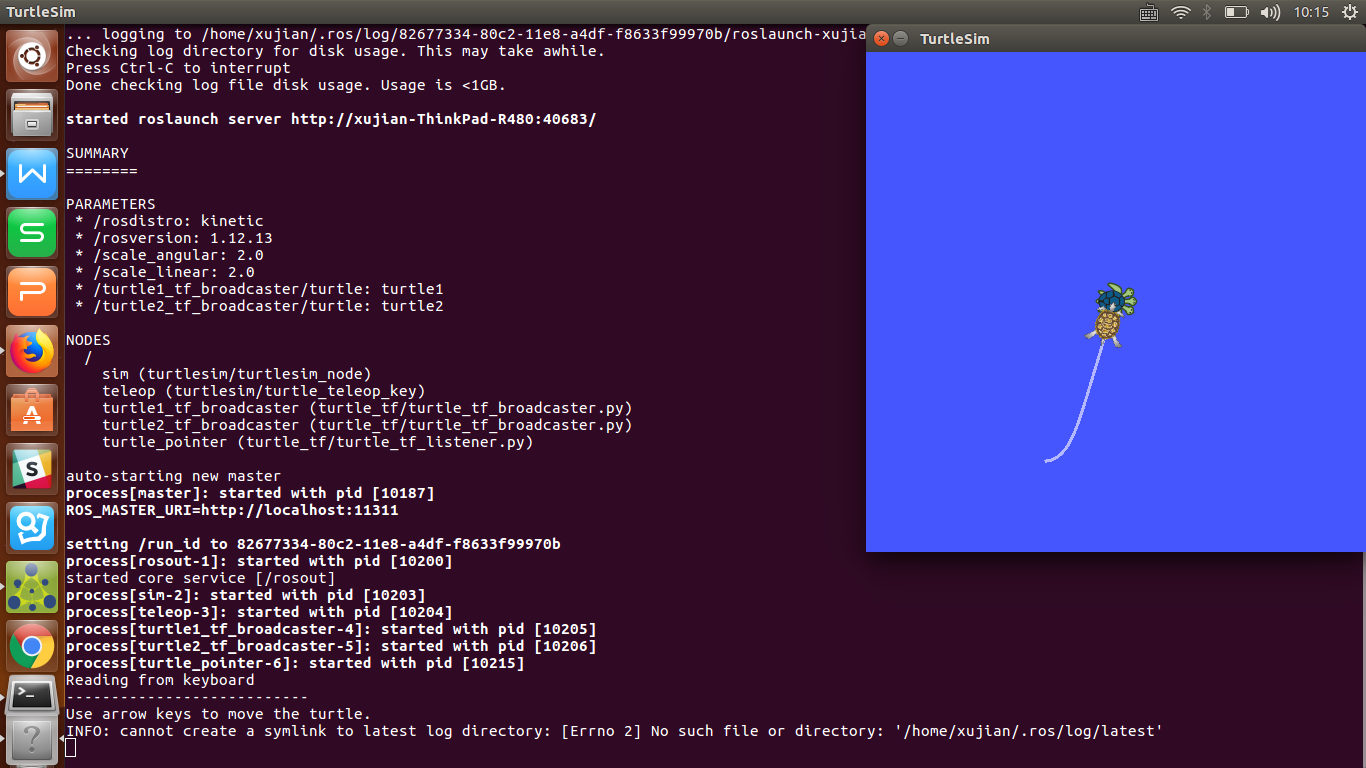


图1-2 运行DEMO示例图

一旦demo被运行起来以后，需要选中终端的端口，可以用键盘的方向键驱动中心的乌龟，我们会发现后面一个tutorial始终会跟着前面一个tutorial。如下图：

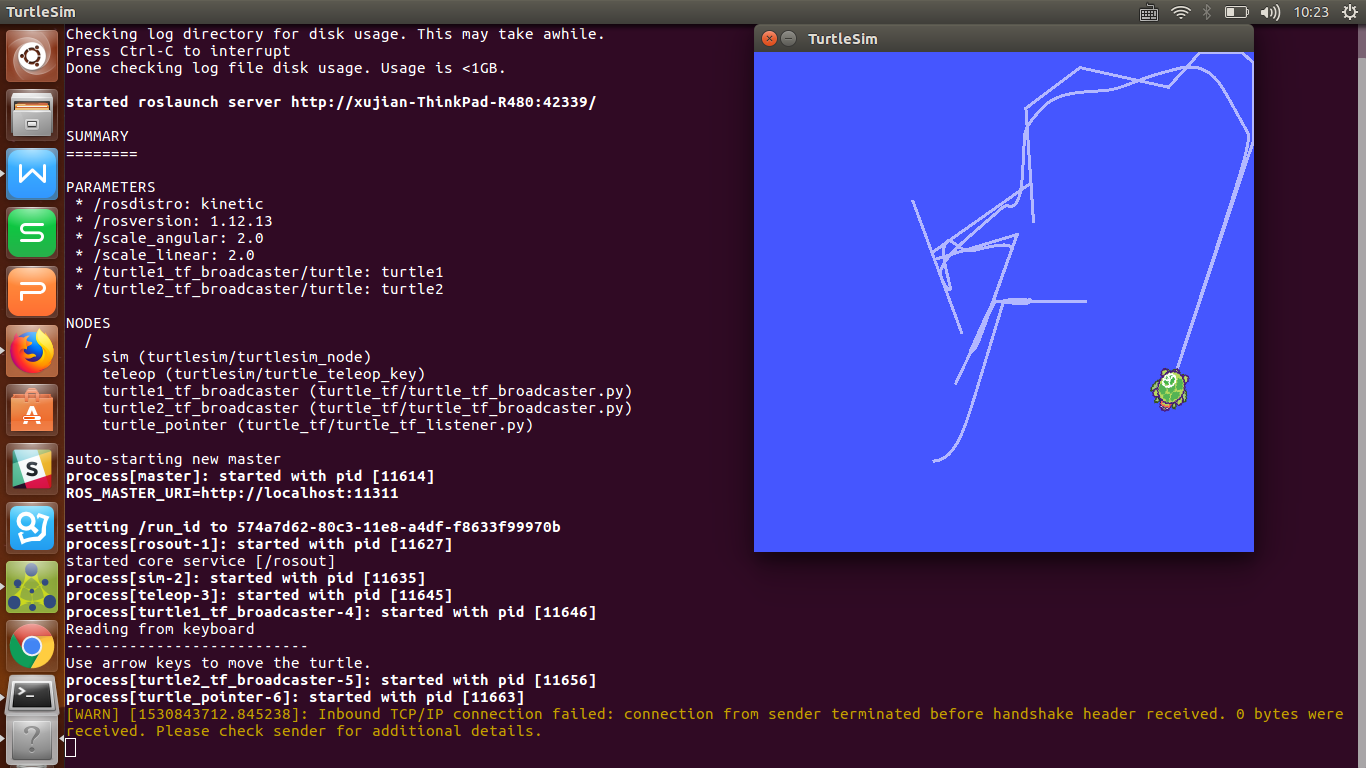


图1-3 DEMO运行驱动图

这个demo会用tf库创建三个坐标系：世界坐标系、turtle1 坐标系、turtle2 坐标系。这个demo使用了一个 tf broadcaster 来发布turtle的坐标系，并且使用了一个tf listener 来计算坐标系之间的差异，从而使得一个turtle能够跟上另外一个turtle。

### 6.1.2 坐标系系统的矩阵表示

TF涉及到坐标系变换，坐标系统是描述物质存在的空间位置（坐标）的参照系，通过定义特定基准及其参数形式来实现。坐标是描述位置的一组数值。按坐标的维度一般分为一维、二维和三维坐标。为了描述或确定位置，必须建立坐标系统，坐标只有存在于某个坐标系统才有实际的意义与具体的位置。

坐标变换包括两方面：位置和姿态，位置的获取可以通过向量，而姿态需要借助于欧拉角和四元数，为了更好的理解这些内容，回顾一下相关的数学知识。

#### 数学知识的回顾

坐标系的定义，简单来说坐标系就是一组基准数，就像是一把想象的尺子，通过这个尺子就可以找到你所在的准确位置，坐标系在机器人中常用的是空间直角坐标系、柱坐标系和球坐标系，因为后两种相对于空间直角坐标系而言还是使用的少，所以以下内容以空间直角坐标系为主要内容，而且后文的坐标系都默认为空间直角坐标系。外观如图1。

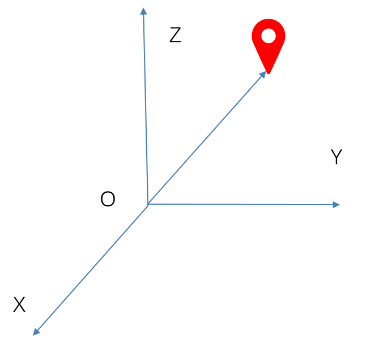


图2-1 空间直角坐标系

之所以需要坐标变换是因为机器人中存在很多的坐标系，如图2 pr2中的坐标系，可以看到，在这张pr2的坐标系抽象出来的ROS数据结构图中，ROS中机器人模型包含大量的部件，这些部件统称为link，每一个link（部件）上固定附着着一个frame（坐标系），link和frame是相互关联在一起的，frame是link的一个想象属性，可以改变它的位置和姿态，以便于处理。

图片包含 LEGO, 玩具, 天空, 室内

已生成极高可信度的说明

图2-2 pr2中的坐标系

从底座到头部的摄像头每个关节都会有一个坐标系，而这些坐标系并不是完全一致且静态的，随着机器人的运动，这些坐标系相对于固定的坐标系都是运动的，甚至运动的坐标系相对于周围的坐标系也是运动的，如何维护处理这些坐标系的关系，实现坐标系间的转换就很重要。

坐标系的变换指的是相对于不同的坐标系，某个物体的位置表示是不同的，需要根据坐标系间的关系来进行修改物体的位置表示。比如图3中，每个颜色的人可以代表着一个坐标系，那么红色人的位置相对于蓝色人和相对于黑色人是不同的，如果已知红色人在蓝色人中的坐标，也知道了蓝色人相对于黑色人的坐标关系，就可以通过这两者的变换，得到关于红色人在黑色人中的位置，这样做的优点是可以通过数学的坐标运算简化机器人的运动控制。实际上，这个人的图也构成了一个坐标系的树状关系，之后会提到。

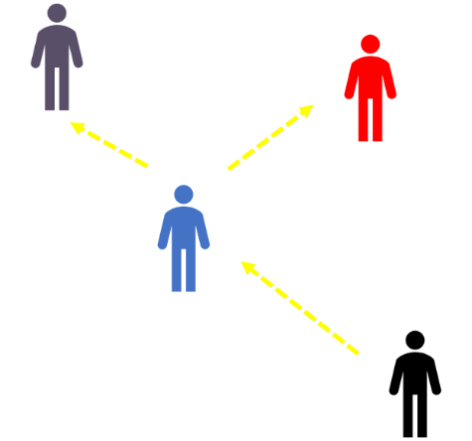


图2-3 坐标系转换的介绍图

矩阵和向量是坐标变换的主要对象。

矩阵可以看作是向量的组合，既可以是列的组合，也可以是行的组合，矩阵是一个数的表而不代表具体的某种含义，但是在机器人中，矩阵可以看成是坐标系或者坐标系变换关系。矩阵的数学表达如下：

Matrix(矩阵):

其中a代表了具体的数值，向量分为行向量和列向量，代表着有方向的一个位移，向量中的三个数分别在空间对应着x, y, z轴的三个值，从原点指向坐标值对应的点。向量的数学表示如下：

Vector(向量): [a, b, c] 或者

将向量和矩阵反映到空间直角坐标系，其对应关系如图3。

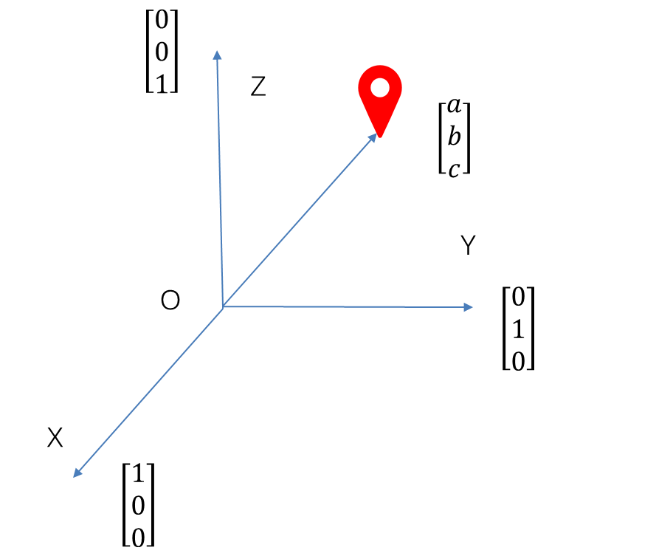


图2-4 向量在空间的数学表示

矩阵的本来面貌可以看作是像这样的三元一次方程组，和其实是一个向量

因为所处的坐标系不一样，所以导致了表示不一样

只要经过这个矩阵，就可以实现一个坐标系向另一个坐标系的转化

这也是tf的核心功能之一

当了解了这些矩阵的基本知识以后，可以做一个小结

1、矩阵表示的是坐标系间的转换关系

2、同一个向量在不同的矩阵下，表示方式是不一致的

#### 关于四元数和欧拉角:

Tf中常常会涉及到四元数和欧拉角，这里做一个了解。

欧拉角（euler angle）

欧拉角是用来唯一地确定定点转动明体位置的三个一组独立角参量，由章动角θ、进动角ψ和自转角φ组成。原始坐标系可以用矩阵表示为

*R*

如果坐标系只绕x轴旋转，对应的矩阵会变成

*R*

同样的，如果坐标系只绕y轴和z轴旋转，矩阵会变成

*R R*

假如坐标系进行了像图4一样的旋转，那么只需要将矩阵依次左乘，就可以得到最终的变换矩阵：

*R= R*

理论上，可以将任何一种变换，分解成欧拉角的旋转，但实际上，存在一些情况，比如万向节的锁死问题，就会造成欧拉角和旋转不是一一对应的情况。所以在ROS中引入了四元数表示旋转。

图片包含 天空, 地图, 滑雪, 文字

已生成极高可信度的说明

图2-5 欧拉角的变化原理图

四元数：

一个四元数可以表示为q = w + xi + yj + zk，其中的i, j ,k均为复数因子，这里不多做解释复数的概念，四元数可以完全表示出三维旋转。我们使用q = ((x, y, z)，w) = (v, w)，其中v是向量，w是实数，这样的式子来表示一个四元数，可以使用一个四元数q=((x,y,z)sinθ2, cosθ2) 来执行一个旋转。具体来说，如果我们想要把空间的一个点P绕着单位向量轴u = (x, y, z)表示的旋转轴旋转θ角度。所以，我们空间的任意一个旋转，都可以拿一个四元数与之对应。

如下面图6所示：

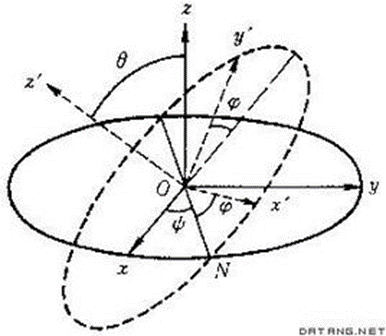


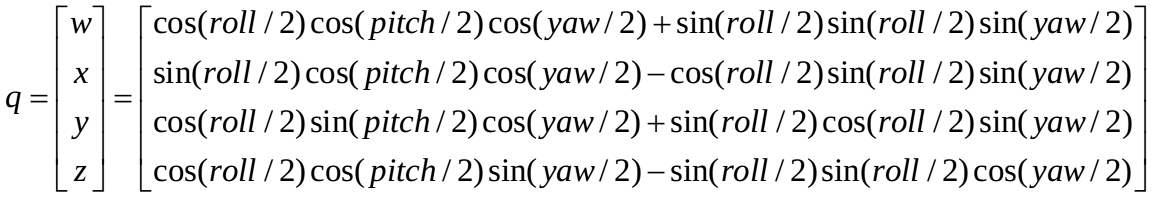
图2-6 四元数示意图

这里对这张图做一个简单说明，首先可以看到（x, y, z)坐标系，需要将该坐标系旋转到(x’, y’ ,z’)位置时，可以按照欧拉角每次仅绕一个轴旋转，旋转三次得到，也可以通过四元数，即找到两个坐标系间的一个旋转轴N，使坐标系仅绕N轴旋转θ角得到。

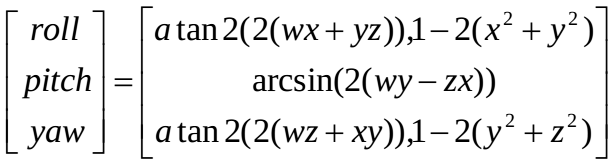
感兴趣的同学可以参考讲义最后的《参考资料》的网址。

欧拉角和四元数具体的转换公式如下：（下面的roll、pitch、yaw表示欧拉角的三个分量）

欧拉角转四元数



四元数转欧拉角



这个转换关系了解就可以，在tf中有函数可以直接实现这个运算过程。

刚体的位姿表示

刚体首先是不会变形的，或者说可以认为是不会变形的。刚体发生位移和旋转，对应的坐标系也会有变化，表示刚体的位姿就借助于坐标系的平移和旋转。

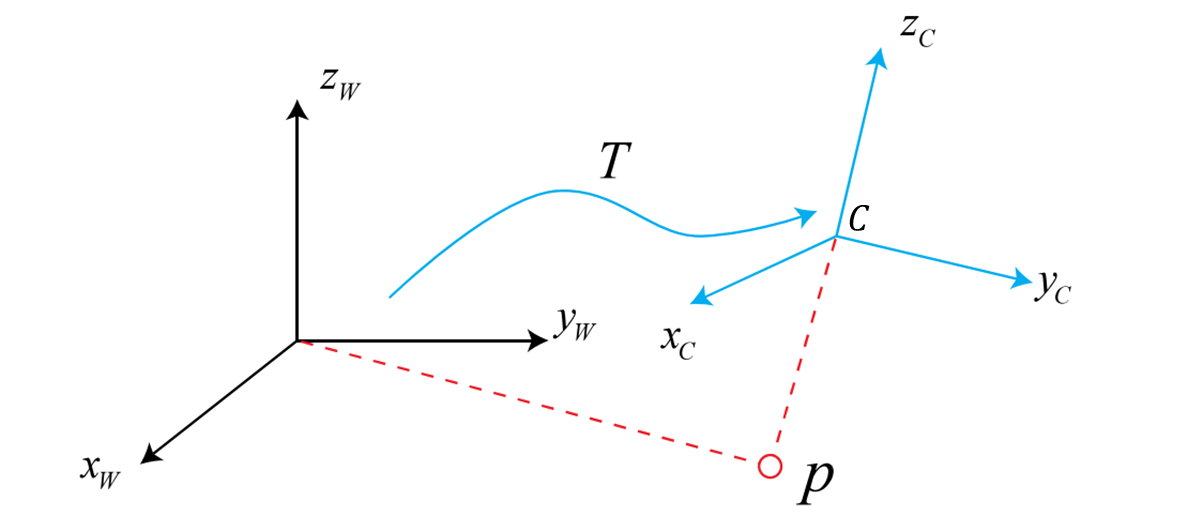


图2-7 刚体位姿图

在机器人运动的过程当中，我们通常会设定一个惯性坐标系（或者叫世界坐标系），姑且

认为这个坐标系是固定不动的。例如：X ,Y ,Z 是固定不动的世界坐标系，X ,Y ,Z 是机器人坐标系。存在一个向量P，在世界坐标系下的坐标是P ，在移动机器人坐标系下的坐标是P ，通常情况下，我们通过传感器已知移动机器人坐标系统下的坐标P ，来求P在世界坐标系下的坐标PW。为了求PW ，我们必须知道机器人坐标系, , 相对与世界坐标, , 做了哪些变换。

我们定义世界坐标系经过变换矩阵T之后得到机器人坐标系（这可以通过计算里程和 IMU的数据进行测量出来）（这也就说明了为什么在机器人刚刚启动的时候odom和base\_link 坐标系必须是重合的，不然没有办法计算旋转矩阵），另外一般情况下，移动机器人运动是 一个刚体运动，也就是说机器人的形状和大小不会因为坐标系不同而改变，这种变换叫做欧氏变换。一个欧氏变换可以由旋转和平移两个部分组成。首先我们考虑旋转问题，

假设在世界坐标系下的单位正交基(, , )，在移动机器人坐标系下的单位正交基(),

那么， 根据向量P的模可知：

因此， ，我们将记做旋转矩阵R，因此上面的表述可以简化为。接下来是平移部分，假设平移部分是经过平移向量t后得到的，那么可以得到。所以通过旋转矩阵R和平移向量**t**，可以描述从世界坐标系到移动机器人坐标系的坐标变换。但是这种方式存在一个问题，对于连续的位置变换，这不是一个线性的表达方式，比如机器人坐标系随着时间不断变换，假设我们经历了两次变换R ,t 和R ,t 且满足：从a到b的变

换b = R a + t ，从b到c的变换c = R b + t .那么从a到c的变换是c = R (R a + t )+ t .并不是我们希望的的形式c = Ra + t，为了格式的统一和方便处理，将非齐次矩阵通过形式的变换写成齐次矩阵的形式。

### 6.1.3 TF和坐标系系统的关系

TF是一个让用户随时间跟踪多个参考系的功能包，它使用一种树型数据结构，根据时间缓冲并维护多个参考系之间的坐标变换关系，可以帮助用户在任意时间，将点、向量等数据的坐标，在两个参考系中完成坐标变换。

TF主要的功能如下：

一个机器人系统通常有很多三维的参考系，而且会随着时间的推移发生变化，例如全局参考系（world frame），机器人中心参考系（base frame），机械夹参考系（gripper frame），机器人头参考系（head frame）等等。TF可以以时间为轴，跟踪这些参考系（默认是10秒之内的），并且允许用户提出如下的申请：

五秒钟之前，机器人头参考系相对于全局参考系的关系是什么样的？

机器人夹取的物体相对于机器人中心参考系的位置在哪里？

机器人中心参考系相对于全局参考系的位置在哪里？

TF可以在分布式系统中进行操作，也就是说一个机器人系统中所有的参考系变换关系，对于所有节点组件，都是可用的，所有订阅tf消息的节点都会缓冲一份所有参考系的变换关系数据，所以这种结构不需要中心服务器来存储任何数据

TF的树形关系图是TF的核心，也是TF和坐标系关系的直接表现，其形状如图 所示，图中的每一个圆圈代表一个frame，对应着机器人上的一个link，任意两个frame之间都必须是连通的，如果有一个环节出现断裂，就会引发error系统报错，所以完整的tf tree不能有任何的断层的地方，这样我们才能查清楚任意两个frame之间的关系，再看这张图片可以看到两个frame间会有一个broadcaster，这就是为了使得两个frame之间能够正确连通，中间会有一个Node来发布消息来broadcaster，broadcaster就是一个发布命令的函数类，两个frame产生相对运动，broadcaster就会公布相关的消息。

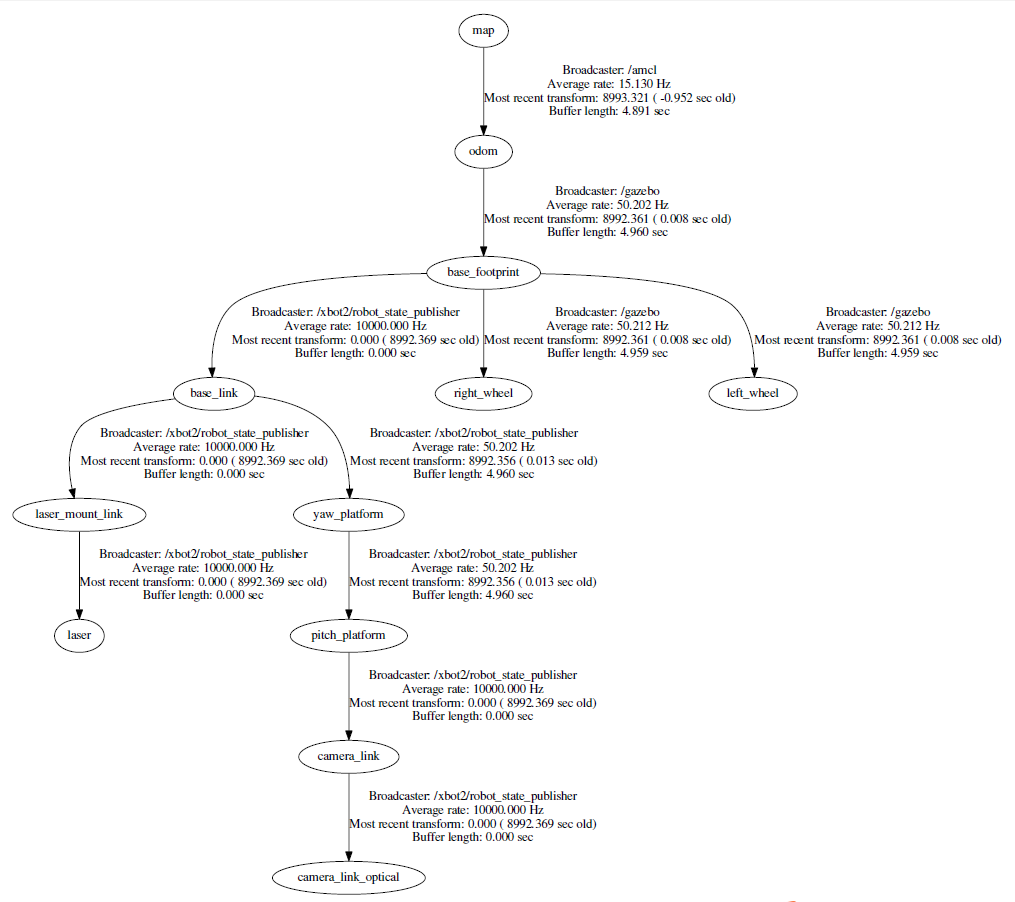


图3-1 TF tree图

TF的消息格式是TransformStamped.msg，格式规范如下：

std\_mags/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

flaot64 z

float64 w

观察标准的格式规范，首先hearder定义了序号、时间以及frame的名称，接着还写了child\_frame，这两个frame之间要做那种变换就是由geometry\_msgs/Transform来定义，Vector3三维向量表示平移，Quaternion四元数表示旋转，像图 TF tree中的两个frame之间的消息，就是由这种格式来定义的，odom就是frame\_id, baselink\_footprint就是child\_frame\_id。我们知道，一个topic上面可能会有多个node在向上发送消息。最终，许多的TransformStamped.msg发往tf，行程TF tree，这就是TF的消息格式。

### 6.1.4 tf in rospy

#### 数据与运算函数:

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，都可以用Tuple，List，Numpy Array表示

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

第一个平移数据使用Tuple表示的，同时也可以用List表示成t=[1.0, 1.5, 0]，也能用numpy.array(1.0, 1.5, 0)来表示。这些数据类型没有特殊对应，全部通用，所以减少了数据间的类型转换。

tf.transformations 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

#### TF类:

1. tf.Broadcaster类:

Broadcaster类函数主要有两种，见表

|  |  |
| --- | --- |
| 格式 | 作用 |
| sendTransform(translation,rotation,time,child,parent) | 向tf发布详细的消息 |
| sendTransformmessage(transform) | 向tf发布封装好的消息 |

表4-1 broadcaster类函数表

第一种sendTransform(translation,rotation,time,child,parant)把transform的平移和旋转填好，打上时间戳（什么时候的tf），然后表示从父到子的frame流），然后发向/tf的topic。第二种是发送封装好的message给/tf，方式不同，功能一致，这些在实训中也会具体看到。

2、tf.Listener类

Listener是对应broadcaster的接受转换的函数类，主要包括三种，具体用法见表

|  |  |
| --- | --- |
| 格式 | 作用 |
| canTransform(self,target\_frame,source\_frame,time) | Frame是否相通 |
| waitForTransform(self,target\_frame,source\_frame,time,timeout) | 阻塞或者说等待知道frame相通 |
| lookupTransform(self,target\_frame,source\_frame,time) | 查看相对的tf，返回(trans,quat) |

表4-2 listener类函数图

Tf.transformListener类中主要包含以上三种方法，它的构造函数不需要填值。注意这里的time参数，依然是使用rospy.Time(0)而非rospy.Time.now()。具体的原因和区别会在实训课中详细列举。

除了这三类主要的函数还有其他的一些辅助函数。

|  |  |
| --- | --- |
| 格式 | 作用 |
| 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) | 返回所有的tf名称 |
| fromTranslationRotation(translation,rotation) | 根据平移和旋转返回4\*4矩阵 |
| transformPoint(target\_frame,point\_msg) | 将PointStamped消息转换到新的frame |
| tranformPose(target\_frame,pose\_msg) | 将PoseStamped消息转换到新的frame |
| TransformQuanternion(target\_frame,quat\_msg) | QuaternionStamped返回相同类型 |
| … | … |

表4-3 Listener类辅助函数表

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]

Rviz

作用：使坐标系在三维空间可视化

用法：详见rviz，不再重复。

Rviz不是专门的tf工具，其用法也不局限于此。

### 参考资料

* 网站：<http://www.ros.org/>
* 掌握机器人 ROS 编程 作者：Lentin Joseph 译者:Carl Zhang
* 欧拉角介绍：<https://baike.baidu.com/item/%E6%AC%A7%E6%8B%89%E8%A7%92/1626212?fr=aladdin>
* rospy 介绍：<https://www.bilibili.com/video/av24585414/?p=26>
* TF python: <http://docs.ros.org/api/tf/html/python/tf_python.html>