《ROS机器人开发技术》

课程讲稿

|  |  |  |
| --- | --- | --- |
| 课程名称 | ： | TF授课 |
| 教师姓名 | ： | XXX |
| 提交时间 | ： | 2018年7月27日 |

中国大学MOOC制

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 课程  名称 | 《ROS机器人开发技术》 | | 章 | | 第6章 | | | 课程类型 |
| 节 | | 第一节 | | | 授课(√ )  实训( ) |
| 名称 | TF | | | | | | |
| 教师 |  | 课时 | | | | 1课时 | | |
| 参考  资料 | 1. Unix环境高级编程（第2版），[美] 史蒂文斯，拉戈著尤晋元，张亚英，戚正伟译，人民邮电出版社 2. <http://wiki.ros.org> 3. http://ros.org/docs/html/python | | | | | | | |
| 教学  目的  要求 | 掌握：TF的原理及TF in python (rospy)中的使用 | | | | | | | |
| 教学  重点  难点 |  | | | PPT页面 | | | 时间分配 | |
| 教学重点 | | | | | | | |
| 1. 坐标系系统的常用变换并能够查看这些变换 | | | 3~19页 | | | 15分钟 | |
| 1. 了解tf broadcaster类和tf listener类 | | | 21~23页 | | | 5分钟 | |
| 1. 了解TF in rospy中的使用命令 | | | 35~42页 | | | 5分钟 | |
| 教学难点 | | | | | | | |
| * + - 1. 坐标系的矩阵表示 | | | 8~9页 | | | 5分钟 | |
| * + - 1. 欧拉角和四元数 | | | 25~33页 | | | 5分钟 | |
|  | * + - 1. 正则表达式中常用元字符的组合使用 | | | 40~41页 | | | 5分钟 | |

|  |  |  |
| --- | --- | --- |
| 教学方法 | | |
| 本授课以课堂讲授为主，与课堂演示方式相结合 | | |
| 教学内容 | 操作演示 | 知识点 |
| **PPT第1-5页：**  本章我们讲ROS中TF库的使用。  对TF的简单引入，说明这节课的重要性，TF是坐标转换，而每个机器人中都必然有坐标系和坐标转换，正是坐标系间的变换实现了机器人的运动控制。本章的内容可视化的比较少，属于比较抽象的部分，等到学完这章进入urdf后，会直观的看到这节的内容。 |  |  |
| **PPT第6页:**  这是一个任务简述。介绍课程完成后需要达成的目标。主要是对tf有一个概念的了解，明白tf的原理及使用场合，对tf中的主要命令如broadcaster类有一个直观印象。这些内容都很实用，在后续的实训课程中都会练习到。 |  |  |
| **PPT第7页:**  PPT的目录，对课程内容的划分为四个小部分，对于每个部分的介绍性引入。比如DEMO实例是做什么，为什么需要坐标系的矩阵表示，tf是怎样实现坐标系间的变换的和rospy中tf命令的写法。 |  |  |
| **PPT第8页:**  这个实例是一个经典的例子。  可以直观的让同学们对坐标系在机器人中的作用有个印象，关于ROS 版本我再提一下，以防有的同学的版本是不一致的。我这里使用的例子是在kinetic 上运行的，编译系统是catkin。  如果哪位同学的版本不同可以通过一会儿的参考网站，找对应的资料。网站地址：<http://wiki.ros.org/tf>。  好，我们开始DEMO实例  首先打开ubuntu的终端，将第一行的代码输入终端。  这行代码的作用是设置运行环境，安装这个例子  我们可以看到安装完成的一个界面，如果已经安装过，就会提示已是最新版本.这儿会有一个乌龟的对话框弹出，提示选中终端对话框，移动方向键驱动乌龟移动。我们可以看到，动着的乌龟是我们驱动的乌龟。而另一个乌龟就像是着了魔一样，一直追逐着这个乌龟。还有一点，同学们会在实训课里感觉到，这个乌龟背对着你，或者侧对着你的时候，你会经常搞错乌龟该往那个方向转。  而这也是坐标系变换需要解决的问题！  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。 | 1.设置环境：sudo apt-get install ros-kinetic-ros-tutorials ros-kinetic-geometry-tutorials ros-kinetic-rviz ros-kinetic-rosbash ros-kinetic-rqt-tf-tree #安装节点  2.运行DEMO（实例）：roslaunch turtle\_tf turtle\_tf\_demo.launch  3. 打开launch文件，可以看到打开了那些节点，有turtle1\_tf\_broadcaster, turtle2\_tf\_broadcaster, turtle\_pointer等,launch文件路径，计算机/opt/ros/share/turtle\_tf/ | **引入socket** |
| **PPT第9页:**  这幅图片就是刚才的程序运行结果。  这里需要解释一下原理：  小乌龟1，2上固定连接着一个坐标系（turtle1， turtle2），这幅图里看不到的有一个世界坐标系（world frame）。就是这几个坐标系的不断发布位置，监听位置，才能实现坐标变换  让小乌龟2始终“追随”小乌龟1 |  | **socket起源** |
| **PPT第10页：**  从上个例子我们可以看出，tf的作用针对的就是坐标系。  实际上，tf是英文单词transform的缩写，transform的意思是转换。  顾名思义也比较容易理解tf命令的主要对象就是坐标系。而它的主要功能就是坐标系的转换和坐标系间的维护。其中，坐标系信息包括位置和姿态，这就引出了我们要将的下一个内容 |  | **socket概述** |
| **PPT第11页：**  坐标系相信大家都不陌生，比如战争剧里的导弹发射，就需要指定的位置坐标。而我们日常生活中使用的导航，也是坐标的应用。  对于机器人中的坐标主要是图中的这种三维直角坐标系，我们要定位一个位置，比如说当前这个图标的位置，只需要知道它的坐标值就足够了。  这样准确的数量描述，才能让机器人准确到达位置。  比如说有四个人分散在一个区域里，如果说要将这四个人的位置在同一个坐标系下定位，他们每个人只需要具体的一组坐标数值就足够了，那么就需要不断地发送信号探测他们的位置，但是如果这四个人的相对距离不变，只是发生角度变化，就像太阳系一样，那么通过转角就可以更快更准的定位他们的位置，在机器人中，大多都恰是这种情况。  通过下面这个例子，我们可以更为清楚的看到这一点 |  |  |
| **PPT第12页:**  第一张图片是PR2机器人实物，可以看到它正在从冰箱里取出一瓶饮料，这对于我们人而言这是个稀松平常的事，我相信没有任何一个地方会因为你会打开冰箱，那一瓶饮料而欢呼喝彩。  但是对于机器人而言，这是一个很复杂的过程！  首先，机器人需要知道自己在哪儿，有没有到了冰箱跟前，然后得判断饮料在哪儿，该拿那瓶？  第二张图就是PR2（ros团队的研发的机器人）的三维模型。  从这张图里，可以看到很多的坐标系，每个关节都会有一个连接在上面的坐标系。这个坐标系是完全跟随着所在的部件运动的。我们称这个部件为link，指的就是连接的部件。  为什么会有这么多的坐标系？  （等待回答）  原因就在于方便！  是的！ 很难想象啊，我们搞这么多东西，居然是为了简化对象。这么说，对于连接在一个件上的固定点，它的位置很容易得到  只需要知道这个件的转动角度，固定点距离坐标的距离，我们就能用矩阵表达清楚的说出一个点的位置，就像是那四个人一样。但是对于这个件上连接的另一个旋转的件，就得再次考虑旋转件的角度，距离。  而如果多建几个坐标系，就可以用一个坐标系和坐标系的相乘  得到这种一节又一节的清楚的关系。 |  | **TCP/IP概念** |
| **PPT第13页:**  Vector(向量): [a, b, c] 或者  Matrix(矩阵):  每个坐标系都会有三个作为参考的轴，线性代数里称之为基或者基轴。因为我们用到的基本都是空间直角坐标系，所以我们可以狭义的认为，坐标系都长右图这个样子。  向量英语单词是vector ，这会是一个常见的词，它的数学描述就是这样，一个括号带着三个数。  这三个数就分别代表了向量相对于坐标系的位置，一般我们将向量写成后面这种竖着的形式，称为列向量。  矩阵是一个数表，这些数并不代表着什么特定的含义，他是一个关系的合集。比如我们用到的矩阵，就表示一个向量在另一个坐标系中的关系。 |  | **TCP/IP层次结构** |
| **PPT第14页:**  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  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。 |  | **应用层介绍** |
| **PPT第15页:**  欧拉角的说明。  欧拉角是描述坐标系旋转的一种方式，最左边这张图代表着原始坐标系，当坐标系只绕Z轴旋转，可以形成一个新的坐标系。对于这个坐标系和他的父坐标系（父坐标系这里就指原始坐标系），他们之间的变化，对应的数学表达就是R(z,theta)，可以看到Z轴代表的第三列，是没有发生变化的，只是x, y 轴代表的坐标跟随转动的角度发生了变化。  如果以第一次旋转为基准，再次仅围绕Y轴旋转一个角度，  就会得到R(y,theta)，第三次也是一样的过程。这里有一点要注意，R(y,theta)的起始是第一次旋转的基础，而不是在原来的坐标系。  通过这样一个过程，可以将任意的一个旋转角度，变换成三个的仅围绕一个坐标轴旋转，这就是欧拉角的好处。  当完成三次旋转后，原始坐标系和第三次旋转的关系就可以通过三个旋转矩阵的左乘得到。 | **图片包含 天空, 地图, 滑雪, 文字  已生成极高可信度的说明** | **欧拉角的说明** |
| PPT第16页: 下面进一步介绍一下欧拉角的缺陷。  在实践中，人们发现了欧拉角的一个万向节锁死问题。就像视频演示的这样，时间限制，这里不做太多的数学引入和解释，需要知道的是欧拉角在表示旋转使可能会出现不可控的情况，导致这种现象的原因，在数学上来看就是角度和实际情况的非一一对应性，即一种旋转可能对应的不只一个欧拉角，而一个欧拉角也没有准确的对应着同一种情况。所以ros中引入了更为复杂的旋转表示方式，即下面的四元数。 |  | 传输层如何识别应用程序 |
| PPT第17页: 因为欧拉角的缺陷，引入了四元数。    四元数的定义牵扯到了虚数因子，这个公式中的I, j, k 都是不能和实数直接计算的，这也正好维持了x, y, z的数值。  四元数也表示旋转，可以从数学表达上看出，它的表示方法2欧拉角多了一个参数，复杂度更高。四元数的表示与欧拉角相同，区别在于他的方式是一步到位，欧拉角通过三次只针对一个坐标轴的旋转，可以旋转成目标的样子。四元数将初始的状态和最后的状态间引入一个旋转轴，即三个虚数因子的系数（x, y, z)所表示的向量，绕着这个轴，旋转θ角即实数w所表示的值，就可以使初始的状态旋转到末了的状态。关于四元数的细节部分可以查看相关资料或者讲义材料 |  | 网络层介绍 |
| PPT第18页: 欧拉角和四元数的转换关系。  这里是从欧拉角向四元数转换的数学方法，也比较简单直观，可以自己推算一下，当然这些也都可以借助于tf工具实现。所以不必深究计算过程。 |  | 网络层协议 |
| PPT第19页: 刚体的位姿表示。  刚体的位置在机器人中是不断变化的，因此机器人每个件的位置也在不停的变化，相应坐标也会随着改变，根据向量可以表示平移，欧拉角和四元数表示旋转，就可以表示坐标系的平移和旋转，也就是位置和姿态。  坐标系改变的过程可以用图形反应为图中所示。 |  | 网络接口层 |
| PPT第20页: 刚体位姿表示的数学方法。  我们要描述坐标系W中向量P的位置（代表一个link）  就可以用向量来描述，向量得到一个具体的位置，但是还需要知道当前的姿态，即它的旋转情况。  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。 |  | **TCP/IP通信模型** |
| **PPT第21页：**  非齐次变换转齐次变换。  上面的变换已经可以描述刚体的位姿，但是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。这是一个非常简单的等价变换，可以简单得到验证。 |  | **TCP/IP数据封装与传递过程** |
| PPT第22页: TF和坐标系系统的关系。  这一小节，主要讲TF的功能和TF消息，TF tree以及对于tf中的主要类函数和工具做引入说明。  首先看到的这张图就是一个TF tree，其实就是demo实例中的tf\_tree ，可以到有三个坐标系统：World , turtle1 , turtle2。Tuttle1和turtle2通过world形成了联系，这是一个相对简单的tree。  实际上的tf\_tree要更为复杂 |  | **小结** |
| PPT第23页: TF消息。  这是一张更为复杂的tree，如图中所示，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。  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。  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，形成TF tree，这就是TF的消息格式 |  | **小结** |
| PPT第24页: 介绍broadcaster类。  功能：publish坐标系，具体用法会在rospy中详细写出。 |  | **IP地址介绍** |
| PPT第25页:介绍listener类。 功能：监听坐标系，从demo来看，使小乌龟能够不断“追逐”另一个小乌龟。 |  | **端口号介绍** |
| PPT第26页: 下面我们来看下tf tools。  View frame  功能：生成一个frame.pdf  配合evince frame.pdf可以立即查看生成的树文件。 |  | **IPv4 socket地址结构** |
| PPT第27页: Rqt\_tf\_tree  功能：使坐标树的广播关系实时反映（被广播）。 |  | IPv6 socket地址结构 |
| **PPT第28页:**  Tf\_echo  功能：任意两个坐标系间的转换关系 |  | socket通用地址结构 |
| PPT第29页: Rivz  功能：使坐标系在三维空间可视化 |  | 三种地址结构比较 |
| **PPT第30页:**  Tf的功能总结。  有时候指的是一套规范，也就是标准，这套标准就定义了数据格式，数据转化是什么样子的。本质上就是前面提到过的树状结构，tf tree。  Tf也可以理解为package ，其中包含了很多的工具，查看两个关节的tf，格式化，调试等。  还有就是一套接口，roscpp 和rospy这两个API。  总结起来就是：TF是坐标转换的一套标准，也是话题topic，还是可以查看消息和坐标系关系的工具，也可以是和其他编程语言的接口工具。 |  | 字节序转换概念 |
| **PPT第31页:**  TF in python(rospy)。  Tf在python中的用法相对于C++而言就简单很多。Python中的数据类型通用，所以不必担心数据的类型转换。Rospy中有一个tf的库，在用的时候，得先import tf，其中有一个transformations，提供了基本的数学运算函数，会使数学运算非常方便。 |  | 字节序转换函数 |
| PPT第32页： TF的相关数据类型。  内容与PPT相同。 |  | 字节处理函数 |
| PPT第33页: Transformation的基本数学运算函数。  内容同PPT，解释PPT即可 |  | IP地址转换函数 |
| PPT第34页: 介绍生成TF tree的具体方法。  这里再一次提到了TF tree，这张图是已经看过的robot\_sim\_demo运行起来的tf tree结构，再强调一遍，每一个圆圈代表一个frame,对应着机器人上的一个link，任意的两个frame之间都必须是联通的，如果出现某一环节的断裂，就会引发error系统报错．所以完整的tf tree不能有任何断层的地方，这样我们才能查清楚任意 两个frame之间的关系．仔细观察上图：  我们发现每两个frame之间都有一个broadcaster,  这是为了使得两个frame之间能够正确连通，中间都会有一个Node来发布消息来broadcaster.如果缺少Node来发布消息维护连通，那么这两个frame之间的连接就会断掉．  broadcaster就 一个publisher,如果两个frame之间发生了相对运动，broadcaster就会发布相关消息 |  | 函数示例 |
| PPT第35页: listener类的介绍。  这个类中主要的三种方法：  一个是cantransform：查看是否两个坐标系相通；第二个是waitfortransform : 阻塞等待直至两个frame相通；Lookuptransform是查看两个frame间的transform，这里的time一般填rospy.time(0)，而不是rospy.time.now()，这是因为坐标传输也是有延时的，你要查看的应该是最近的状态，而不是完全同步的状态，这部分内容在实训中会详细对比练习。 |  | 小结 |
| PPT第36页:除了上页提到的主要的三个函数，listener类还包括很多的辅助函数，详见PPT和讲义部分。 或者PPT底部的参考网址。 |  |  |
| PPT第37页： Broadcaster类  第二个比较重要的类就是tf.transformbroadcaster类。  第一个sendtransform就是把transform的平移和旋转都填好，打上时间戳，然后让他表示从那个parent frame到那个child frame；  还有一个是transformmessage ，直接把tf message发送出去，发送到tf。  到这里授课就完成了，这些命令都会在实训课中联系到，可以提前预习，练习。 |  |  |
| 知识点框图 | | |
| TF  坐标系的表示  DEMO实例演示  环境设置  运行方式  数学知识回顾  欧拉角和四元数  TF broadcaster函数  Tf in rospy  数学基本运算函数  参数类型  TF listener类函数  TF和坐标系 | | |