

《ROS机器人开发技术》

**课程名称:ROS机器人开发技术**

**教师姓名:XXX**

**提交时间:2018年7月x日**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 课程信息 | 课程名称 | | 名称 | | 章数 | 节数 | | 课程类型 |
| 《ROS机器人开发技术》 | | Navigation知识学习 | | 九 | 四 | | 授课( )  实训(√ ) |
| 教师 |  | | | 时长 | 1课时 | |
| 参考文献 | 1. ROS机器人程序设计（原书第二版） [西班牙]恩里克.费尔南德斯等著 ，刘锦涛 等译 | | | | | | | |
| 教学  目的  要求 | 学会配置导航包，理解路径导航的基本思想 | | | | | | | |
| 教学  重点  难点 | 重难点 | | | PPT页面 | | | 时间分配 | |
| 重点 | 1. Move\_base配置 | | 2~10页 | | | 20分钟 | |
| 1. Map\_server配置 | | 11~15页 | | | 10分钟 | |
| 1. AMCL实现 | | 22~31页 | | | 20分钟 | |
| 1. 导航功能实现 | |  | | |  | |
| 1. 插入costmap | |  | | |  | |
| 难点 | 1. Costmap | | 18~21页 | | | 10分钟 | |
| 1. AMCL | | 22~31页 | | | 20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2页: 同学们好，欢迎来到Navigation的实训章节。纸上得来终觉浅，ROS学习最重要的还是动手实践，有许多坑只有自己踩过才能真正学会，希望大家带着钻研的精神完成这部分内容。我们给大家准备了五个实训任务，分别是…，其中最后一个为扩展题目，相信大家经过这五个任务的训练，能真正把导航包应用到自己的实际机器人开发实践中！ |  |  |
| **PPT第3页:**  在实训前我们先下载安装一些必要的依赖包，请大家依次运行如下命令：ppt  当然，最后不要忘记source~ |  |  |
| **PPT第4页:**  我们先来进行第一个任务，任务描述：配置move\_base参数例如运行成本、机器人半径、到达目标位置的距离，机器人移动的速度，并在rviz中让机器人以1m的速度前进和正方形前进进行测试。 |  |  |
| **PPT第5页:**  第一步，加载机器人模型，运行命令：  第二步，加载空白地图，运行命令： |  |  |
| **PPT第6页：**  根据授课内容我们已经知道，move\_base中有全局路径规划和本地实时路径规划，这里我们再简单的回顾一下：ppt |  |  |
| **PPT第7页：**  在运行完两个launch文件后，我们调用rviz可以看到在无障碍地图中的机器人仿真效果。 |  |  |
| **PPT第8页：**  有了机器人之后，我们先对它进行一些简单的操作，使它前进一米，输入如下命令，可以看到在Rviz中的轨迹图如下: |  |  |
| **PPT第9页:**  我们接着让机器人走正方形路线，先输入如下命令让机器人回到原来位置(0,0,0)  然后按reset按键，把所有的箭头清除。接着运行走正方形路径的代码：  rosrun rbx1\_nav move\_base\_square.py  在rviz中可以看到： |  |  |
| **PPT第10页:**  在实际情况中，往往需要让机器人自动躲避障碍物。move\_base包的一个强大的功能就是可以在全局规划的过程中自动躲避障碍物，而不影响全局路径。障碍物可以是静态的（比如墙、桌子等），也可以是动态的（比如人走过）。  现在我们尝试在之前的正方形路径中加入障碍物。把之前运fake\_move\_base\_blank\_map.launch的中断Ctrl-C掉，然后运行：   roslaunch rbx1\_navfake\_move\_base\_map\_with\_obstacle.launch          然后就会看到在rviz中出现了障碍物。然后在运行之前走正方形路线的代码：          rosrun rbx1\_nav move\_base\_square.py           这回我们可以看到，在全局路径规划的时候，机器人已经将障碍物绕过去了，如右图： |  |  |
| **PPT第11页:**  我们来到第二个任务： map\_server配置  任务描述：将软件所博物馆实际地图导入到rviz地图中 |  |  |
| **PPT第12页:**  根据前面授课章节所述，map\_server发布两个图像，一个是具体的地图的图像，一个是地图的描述信息。左图是软件博物馆的地图图像，右边是地图的描述信息并添加了注释。 |  |  |
| PPT第13页: 第一步，打开实际地图，运行命令：  第二步，打开rviz工具，运行命令 |  |  |
| PPT第14页: 实际上这个任务很简单，我们来看一下具体代码原理，我们在roslaunch的时候运行了map:=Software\_Museum.yaml 这个命令，这个命令是为了把launch文件中的test\_map.yaml改为我们需要仿真的环境地图Software\_Museum.yaml，  所以大家在rviz中做仿真的时候也可以参考这种方式把地图进行替换。 |  |  |
| PPT第15页: Map\_server配置完成后我们可以在rviz中看到效果，并可以看到机器人在地图中的定位。 |  |  |
| PPT第16页: 在加载完地图后，我们要对机器人在地图中的位置进行定位，我们使用AMCL定位方法。 |  |  |
| PPT第17页： 大家依次运行如下命令，可以看到机器人在仿真环境中的AMCL效果图。  这里附上笔者在运行的时候报错的解决方式：  1)在编译整个包的时候可能会报错缺少karto的slam包，大家需要额外装上，具体安装过程可以参考网上教程  2)在roslaunch amcl\_demo.launch的时候会报错路径找不到，需要在对应的launch文件下手动添加amcl\_demo.launch的路径 |  |  |
| PPT第18页： 刚才我们运行了AMCL的node，只是看到了一个效果，对这个算法体会并不深刻，那么AMCL这个算法具体事怎么实现的呢，笔者在这里吧AMCL的节点中的代码做了注释，大家可以结合算法理解一下  (读一下注释) |  |  |
| PPT第19页: 我们进入第四任务，我将第四任务分成了两个子任务，一个是…，另一个是… |  |  |
| PPT第20页: 在任务2完成后，我们可以看到机器人在rviz中的定位，下面我们使用2D goal选项来手动指定一个目标点，可以看到导航效果。实现效果如图所示： |  |  |
| **PPT第21页：**  在实际应用中，我们往往希望机器人能够自主进行定位和导航，不需要人为的干预，这样才更智能化。在这一节的测试中，我们让目标点在地图中随机生成，然后机器人自动导航到达目标。  运行完命令之后，然后我们点击rviz上的“2D Pose Estimate”按键，然后左键在机器人上单击，让绿色的箭头和黄色的箭头重合，机器人就开始随机选择目标导航了。在监控窗口中，我们可以看到机器人发送的状态信息。 |  |  |
| PPT第22页: 完成前面四个任务，大家可以对不同的导航包如何使用有个基本的掌握了，下面我们要对导航中比较难以理解也是比较重要的costmap部分进行一个扩展任务，供有兴趣的同学学习。  任务五分为两个任务，分别是: 1.在catkin 工作空间中创建New layer 插件; 2.在程序中调用新创建的New layer |  |  |
| PPT第23页: 我们先学习子任务1，读ppt…  第二步的意思是新建了一个SimpleLayer类，基类是costmap\_2d::Layer。上面头文件里有两个主要的函数updateBounds 和 updateCosts。这两个函数更新costmap区域和更新costmap的值。updateBounds的参数origin\_x,origin\_y,origin\_yaw是机器人的当前位姿，不需要我们人为的去设定，ROS会自动传递这几个参数。 |  |  |
| PPT第24页: (读第三点)，这个程序的主要思路是，更新updateBounds中的mark\_x\_和mark\_y\_，这两个变量是用来存放障碍点位置的，他们是在世界坐标系定义的变量。然后在updateCosts里将这两个变量从世界坐标转换到地图坐标系，并在地图坐标系中设定障碍点，即在栅格地图master\_grid设定这个点的cost。  然后读ppt |  |  |
| PPT第25页: 读ppt |  |  |
| PPT第26页: 刚才我们创建了costmap layer插件，并不意味着ROS就会使用它，我们得在显式的global\_costmap 或者 local\_costmap中声明它。在调用自己写的这个layer之前，先看看系统默认的global\_costmap 和local\_costmap使用了哪些layer。假设你已经安装了rbx1 package：  roslaunch rbx1\_bringup fake\_turtlebot.launch  在新的终端输入：  roslaunch rbx1\_nav fake\_move\_base\_blank\_map.launch  你将看到如下图所示的一些信息：  你会看到pre-hydro 字样。说明当前的costmap是默认的配置，也就是static\_layer,obstacle\_layer,inflation\_layer这三层。 |  |  |
| PPT第27页: 下面我们来把创建的simple\_layer，放入全局global\_costmap中。要想使得ROS接受你的插件，要在参数中用plugins指明，也就是主要修改move\_base中涉及到costmap的yaml文件，下面给出我的修改：  1.配置costmap\_common\_params.yaml（具体参数参考所给的代码包，下同）  2.global\_costmap\_params.yaml  3.local\_costmap\_params.yaml  另外一个与路径规划相关的base\_local\_planner\_params,yaml不用修改  这里由于只在全局层添加simple\_layer，所以local\_costmap还是使用的默认layer插件。然后我们运行这个新配置的move\_base launch文件，你会发现simplerlayer已经添加进global\_costmap了，而local\_costmap还是默认的pre-hydro格式 |  |  |
| **PPT第28页:**  最后在rviz中看看global\_costmap中有没有加入这个障碍物点。下面是在软件博物馆环境的地图，红色标记的地方是人为加入的障碍物。按照上面的程序，障碍物应该出现在机器人正前方1m处。  障碍物的膨胀系数，可以用下面的命令进行动态调试：  好的，navigation实训部分的内容到这就结束了，希望大家能自己动手操作一遍，而且我们对要用的关键的代码都进行了注释，大家可以去研究一些参数配置的过程，这样可以对导航有更深入的理解，谢谢！ |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | | |