P3AT 机器人实验手册

黎振胜

2017 年 6 月 13 日

1 必读

大家好，这是我离校之前对使用 P3AT 机器人的一些方法介绍和总结，不全面但都是一些很基本和重要的东西，使用机器人之前把这个浏览一遍可以少走弯路，希望对大家以后的实验有帮助。

另外这算是我第二次使用 LaTeX 进行写作，权当作一次练习，希望大家以后也多多尝试。

最后，希望后面的师弟能够把这份文档维护下去，比如加入图片。既可以学习 LaTeX, 又可以学习 Git，何乐而不为。

1.1 基本使用

机器人为了正常连接激光设备，需要为激光分配固定 IP 地址，配置如下。如果需要电脑上网，则需要将网卡配置为自动获取 IP。这使用起来比较难受，但是目前没有找到解决办法。

```
# set ip config in network manager
# Address   192.168.0.4
# Netmask   192.0.255.255(automatically become 192.0.0.0, could
   ↪ work but I don't know why)
# Gateway   192.168.0.1
```

1.1.1 硬件

灯提示及声音提示

1.1.2 软件

自己看 readme 安装也行，看后面的记录也行。

1.2 充电

1.2.1 原装电池

1.2.2 高性能电池

1.3 里程计校准

1.4 资料清单

1.5 注意事项

重新整理后，分文件夹进行介绍

2 基于 ROS 的 P3AT 控制软件

命令和代码仅为示例，请在了解 Linux 基本原理和工具的基础上阅读使用。

2.1 SDK 和工具

2.1.1 libaria

aria 是 Adept 为 P3AT 机器人提供的核心开发包，能够屏蔽硬件细节，管理激光，声纳，摄像头等附件，此节介绍如何安装和使用这个 SDK

安装 直接用 dpkg 命令安装.deb 包，这里安装的是 libaria2.9.0，命令如下

```
sudo dpkg -i libaria_2.9.0+ubuntu12+gcc4.6_amd64.deb
sudo apt -f install
```

查看手册 SDK 安装后到其目录下/usr/local/aria 可以看到软件开发手册，点进去仔细阅读，尤其是开头的部分，如下图所示，应全部阅读。

运行示例程序 在.../examples 目录下有一个 cpp 项目，使用 make 构建，包含基本的开发示例，开发文档中对这些文件进行了介绍。按以下命令编译运行示例程序。

```
cd /usr/local/aria/examples
make simplemotioncommands
./simplemotioncommands
```

2.1.2 MobileSim

MobileSim 是一款基于 Stage 的仿真器，使用这个工具可以从某种程度上代替真正物理机，方便进行软件调试，注意这个工具是二维的，功能有限如果需要模拟更多传感器和执行器，推荐使用 Gazebo。

安装 也是安装 deb 包，命令如下

```
sudo dpkg -i mobilesim_0.7.3+ubuntu12+gcc4.6_amd64.deb
sudo apt -f install
```

使用 在 MobileSim 安装目录下的 rademe 中可以看到详细的使用方法，请浏览阅读。为了方便使用，在.bashrc 中配置环境变量，示例命令如下

```
# in bashrc
alias MobileSim="/usr/local/MobileSim/MobileSim"
# in shell
MobileSim -m cmee.map -r p3at
```

2.2 配置 zsROS

2.2.1 安装 ROS

到 wiki 页面查询安装方法，在 14.04Ubuntu 上安装 Indigo 版本[①] 建立工作空间。全部按照默认方法建立 catkin_ws[②]，配置.bashrc 环境变量，示例如下:

```
source /opt/ros/indigo/setup.bash
export EDITOR='subl'
export ROS_WORKSPACE=/home/zs/catkin_ws
export ROS_PACKAGE_PATH=${ROS_WORKSPACE}:${ROS_PACKAGE_PATH}
source ${ROS_WORKSPACE}/devel/setup.bash
alias MobileSim="/usr/local/MobileSim/MobileSim"
```

subsubsection 安装附件 ROS 包 安装一些没有预装的 ROS 包，命令如下

```
sudo apt install libsdl1.2-dev libsdl-image1.2-dev ros-indigo-
   ↪ navigation ros-indigo-axis-camera ros-indigo-flir-ptu-
   ↪ driver ros-indigo-serial
sudo apt install ros-indigo-freenect-stack ros-indigo-openni*
   ↪ libfreenect-dev
```

[①] http://wiki.ros.org/indigo/Installation/Ubuntu

[②] http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

```
sudo apt install ros-indigo-joy-listener ros-indigo-joy-teleop
   ↪ ros-indigo-teleop-tools ros-indigo-teleop-twist-joy ros-
   ↪ indigo-teleop-twist-keyboard
sudo apt install ros-indigo-lms1xx
sudo apt install ros-indigo-rosbridge-suite
```

2.2.2 编译 zsROS 包

这个包是我整理设计的，就在 github 下存放着，注意下载 link-withmap 分支的代码，这个代码是最新的，编译方法如下。

```
cd ./catkin_ws
cakkin_make
# 如果出错，则按下列步骤重新编译
cakkin_make
cd ./build
make
# 注意不要使用多线程编译，使用单线程就好，我遇到过这种错误，原因
   ↪ 未知
```

2.3 手柄控制

此处讲述如何在本地使用手柄控制机器人，此处的机器人既可以指 MobileSim 模拟器，也可以指真机，因为都是运行在 Ubuntu 上，所以原理一致。

连接 XBOX 手柄 手柄插入 USB 接口，在/dev/input 中查看手柄对应的设备 ID，如 js0

必要时修改 launch 文件 根据对应的手柄设备 ID，修改`E:\Zhensheng\documents\GitHub\zsROS\zs_launch\launch\zs_joy_rob.launch` 文件。如果对其他参数有兴趣也可修改，launch 文件全文如下

```
<launch>
  <node pkg = "joy" type = "joy_node" name = "joy_node" output="
     ↪ screen">
    <param name="dev"               value="/dev/input/js0" type="
       ↪ string"/>
    <param name="deadzone"          value="0.05"/>
    <param name="autorepeat_rate"   value="0.0"/>
    <param name="coalesce_interval" value="0.001"/>
```

```
  </node>
  <node pkg = "teleop_twist_joy" type = "teleop_node" name = "
     ↪ teleop_node" output="screen">
    <param name="enable_button"         value="0"/>
    <param name="enable_turbo_button"   value="-1"/>
    <param name="axis_linear"           value="1"/>
    <param name="scale_linear_turbo"    value="1.0"/>
    <param name="axis_angular"          value="0"/>
    <param name="scale_angular"         value="1.0"/>
    <remap from="cmd_vel" to="RosAria/cmd_vel" />
  </node>
</launch>
```

启动 ROS 系统 依次启动调试工具，ROSARIA 传感驱动节点，RViz 可视化工具，最后启动手柄驱动节点。命令示例如下

```
# 启动ros master
roscore
# 启动调试工具
roscd zs_launch_files/
./zs_tools.sh cmee
# 启动ROSARIA
roslaunch zs_launch_files zs_rosaria.launch
# 启动RViz
roslaunch zs_launch_files zs_p3at_rviz.launch
# 启动手柄驱动结点
roslaunch zs_launch_files zs_joy_rob.launch
```

2.4 记录里程计信息

这里介绍使用 rosbag 记录里程计信息

rosbag 基本使用 bag 实际上是将一个 node 的行为录制下来，然后可以重新播放；使用命令行能够：录制，从包重新发布，获取包的概括信息，检查包的消息类型，使用 Python 表达式过滤包中信息，压缩解压缩包，重新索引包；如果录制高带宽的东西，例如图像信息，建议在本地录制和存放文件；bag 文件默认名字为日期，可以添加前缀。

录制信息 简单讲，就是指定主题进行录制，命令示例如下

```
rosbag record rosout tf cmd_vel
rosbag record -a #所有
rosbag record --duration=30 /chatter #持续30s,5m,2h
rosbag record --split --size=1024 /chatter #空间达到1024M后分文件
    ↪ 存储
rosbag record --split --duration=30 /chatter #持续时间到30s后分文
    ↪ 件存储
rosbag record -o session1 /chatter #为文件名字做前缀
rosbag record -O session2_090210.bag /chatter #为文件命名
rosbag record --node=/joy_teleop #录制该节点左右消息
rosbag record -l 1000 /chatter  # 录制该主题1000个消息限制
```

重放 如果两个包被播放，它们将作为一个包对待。比如你录制一个包，在一个小时之后再录制一个包，然后同时播放，这个时候你会经历 1 个小时的空白。示例命令如下

```
rosbag play recorede1.bag
rosbag play --clock recorded1.bag #发布时钟时间
rosbag play --clock --hz=200 recorded1.bag #时钟频率默认100
rosbag play -r 10 recorded1.bag # 发布速度翻倍
rosbag play -d 5 recorded1.bag #每一次advertise后等待5秒
rosbag play -u 240 recorded1.bag #只播放240s
rosbag play -l recorded1.bag # 循环播放
```

使用 MATLAB 进一步处理 方法是先导出为 CSV 文件，再存为文件。示例命令如下

```
# 先做实验显示
rostopic echo -b log_file.bag /topic_name
rostopic echo -p /topic_name #使用matlab友好格式进行显示
# 这个命令就可以了
rostopic echo -b file.bag -p /topic > data.txt
```

注意 如果图方便，可以使用 rqt_bag，这是一个基于 qt 的图形界面工具

2.5 使用 kinect

这个很复杂，基本思路是使用libfreenect2 这个库，在这个库的基础上使用iai-kinect2 这个 ROS 包，但是库包并不受官方支持，所以很不稳定，必须根据每台计算机不断调试才能成功。以下仅为示例代码，本人电脑为 AMD 显卡。

```
# 安装libfreenect2
cd libfreenect2
cd depends; ./download_debs_trusty.sh
sudo apt-get install libusb-1.0-0-dev
sudo apt-get install libturbojpeg libjpeg-turbo8-dev
sudo apt-get install libglfw3-dev
sudo apt-get install beignet-dev
sudo apt-get install libva-dev libjpeg-dev
sudo apt-get install libopenni2-dev
cd ..
mkdir build
cd ./build
cmake ..  -DENABLE_CXX11=ON
make -j4
sudo make install
# 安装在 /usr/local/
# 测试库文件的有效性
# 最后可以运行程序，在build下面有个bin文件夹，放置生成的输出文
   ↪ 件，插上kinect，然后运行。此时黄灯变成白色的，表示有驱动。
   ↪ 注意：只能用于USB3的接口，好在台式机和笔记本都有3.0的口。改
   ↪ 成如下的超级命令即可。
sudo ./bin/Protonect
./Protonect cl 查看依赖错误信息
# 复制iai-kinect2ROS包到catkin_ws
catkin_make
# 测试ROS包正常工作
roslaunch kinect2_bridge kinect2_bridge.launch
```

2.6 与远程人机交互平台配合使用

修改 bashrc 文件 根据具体情况，添加下位机（本机）IP 地址，设置环境变量，示例如下。注意，如果不是作为下位机运行 ROS，则环境变量得改回来。

```
# 假设下位机IP地址为10.0.126.3
export ROS_IP=10.0.126.3
```

```
export ROS_MASTER_URI=http://10.0.126.3:11311
```

启动 ROS 系统 如果是仿真的话，则直接启动即可，如果机器人已经在远程运行，则需要使用 SSH 等远程登陆工具运行。示例命令如下

```
# 启动ros master
roscore
# 启动调试工具（仿真环境下启动，真机条件下无需启动）
roscd zs_launch_files/
./zs_tools.sh cmee
# 启动ROSARIA（真机条件下需要打开串口权限）
# sudo chmod a+rwx /dev/ttyS0
roslaunch zs_launch_files zs_rosaria.launch
# 启动代理结点
roslaunch zs_launch_files zs_proxy.launch
# 启动导航避障模块
roslaunch zs_launch_files zs_move_base.launch
# 在必要时启动RViz可视化模块
# roslaunch zs_launch_files zs_p3at_rviz_ex.launch
```

3 使用远程人机交互平台

已经把所有软件安装到老板笔记本电脑上，下面是我的记录

3.1 软件安装

你们看到这个文档的时候环境应该已经配置好了，但我把它记下来以备不时之需。

3.1.1 安装 LabVIEW2014

安装软件 安装软件, 大家都会装吧，只需注意一点，安装所有软件功能，咱们实验室的 LabVIEW 包含功能特别多，全部装上去，安装过程如下

安装更新 使用前最好彻底更新，方法是【帮助】-【检查更新】。网络条件不好，记住一个一个装。

升级 VI Package Manager 升级到最新版本，当前是 2016，目录为.../软件依赖项/labview2014/vipm-windows.exe

3.1.2 安装 LabVIEW 附加包

使用 VI Package Manager 进行安装，可以在线安装，也可离线安装，文件夹中已经有离线包。

lvh__toolbox-2.0.0.35.vip 这个是安装 lvh_xbone 基础包，直接安装即可

lvh__xbone-2.0.0.6.vip 这个是 LabVIEW 中的 xbox 手柄驱动程序，安装之后学习示例程序就可以了

ni__lib__state__pattern__actor-1.1.0.10.vip 这个是操作者框架的状态模式编程工具包，由于 LabVIEW2014 自带 Actor Framework，所以可直接安装。安装目录在`D:\ProgramFiles(x86)\NationalInstruments\LabVIEW2014\vi.lib\ActorFramework\`。[③]

ROS for LabVIEW 这个是 Tufts University 的师生制作的 LabVIEW 与 ROS 通信工具，其基于 ROS 底层通信协议（xml-RPC）, 可以在 LabVIEW 端启动 roscore，新建节点，发送订阅消息等。现在 GitHub 上开源，但是早已停止维护，其功能在新版本的 ROS 中部分无法使用。安装方法是将文件夹复制到`D:\ProgramFiles(x86)\NationalInstruments\LabVIEW2014\vi.lib\` 下。

3.1.3 安装自然语言处理运行环境

Anaconda2-2.5.0-Windows-x86__64.exe 这是 Python 运行环境

jdk-8u31-windows-i586.exe java 运行环境，用于运行基于科大讯飞的语音识别窗口程序

aiml-0.8.6 与语音生成相关的包，使用 python setup.py 方法安装

PyAIML-master AIML 的 Python 接口

pymdptoolbox-master 马尔科夫决策过程工具箱的 Python 接口，与对话管理器的状态机有关

[③]见`https://forums.ni.com/t5/Actor-Framework-Discussions/Implementing-the-State-Pattern-in-Actor-Framework/td-p/3409456?tstart=0`

3.1.4 安装语义推理运行环境

swipl-w64-723.exe swi-prolog 是最受欢迎的 Prolog 开发环境，安装本环境用于运行基于语义的路径规划，安装完成记得添加 PATH 环境变量

pyswip-0.2.3 这是 swip 的 Python 接口包在线安装，也可离线安装，文件夹中已经有离线包。

3.2 软件使用

LabVIEW 运行的是上位机软件，通过与 ROS 通信，达到控制 P3AT 移动机器人的目的，由于 ROS 为服务端，所以务必先启动 P3AT ROS 系统。需要注意的是，ROS 同样也有一套自己的控制界面系统 rqt，熟悉 ROS 的同学敬请把这些移植到 Ubuntu 上。这一套是我基于 LabVIEW 实现的。

运行 ROS 系统 按照如2.6所述方法启动下位机 ROS 系统

打开上位机软件 进入`E:\Zhensheng\documents\GitHub\zsLV`, 打开 zsLV.lvproj，进入 LabVIEW 项目管理器，打开`\LaunchCOUI.vi`，在控件中输入下位机 IP 地址，启动 vi，会弹出控制面板和地图两个用户界面，当控制面板上接收到机器人的电压值时，说明连接成功，用户也可以查看控制面板上的消息日志记录来查看软件使用情况。

注意 在使用上位机的同时，同样可以使用 ROS 的 RViz 工具对 ROS 系统进行监控以弥补上位机对数据可视化的不足，上位机程序主要目的是验证自然语言处理以及语义规划算法的使用效果。