机器人软件工程学

实验报告

姓　　名：　郑可欣

学　　号：　1911717

组　　号：　第五组 KameLiDAR

1. 任务要求

重点考察在运用仿真、语音、导航、图像、机械臂（创新加分项）的能力，以及对以上基础模块融合运用的创新能力。

2. 任务设计

- 建图：在 Gazebo 下搭建仿真场景，通过键盘操控 TurtleBot 机器人运动，通过 GMapping 实现进行机器人的定位与建图。
- 导航：在起始位置，通过语音启动 RViz 模块，告知机器人前往目标点抓取某物，随后使用 AMCL 进行导航，机器人从起始位置移动至目标点。
- 物体识别：到达目标点后使用 YOLO 对待抓取物体进行物体识别，通过坐标变换定位目标物体。
- 抓取：Gazebo 与 Moveit! 结合，利用获得的物体位姿控制机械臂完成对指定物体的抓取。

3. 安装依赖

- turtlebot_gazebo

```
sudo apt-get install ros-melodic-gazebo-ros-pkgs ros-melodic-gazebo-ros-control
sudo apt-get install ros-melodic-turtlebot*
```

- turtlebot_arm

```
git clone https://github.com/turtlebot/turtlebot_arm.git
```

- YOLO_v3

```
git clone https://github.com/leggedrobotics/darknet_ros.git
```

4. 语音控制

- 首先打开语音控制模块，这里我们使用的是 PocketSphinx：

```
roslaunch pocketsphinx kws.launch dict:=/home/longway/turtlebot_ws/src/pocketsphinx/demo/voice_cmd.dic kws:=/home/longway/turtlebot_ws/src/pocketsphinx/demo/voice_cmd.kwlist
```

```
rostopic echo /kws_data
rosrun pocketsphinx voice_control_example.py
```

5. Gazebo 场景构建与建图

- 打开 Gazebo 后，按 Ctrl+B 绘制新地图，或者直接 insert 现有的模型：

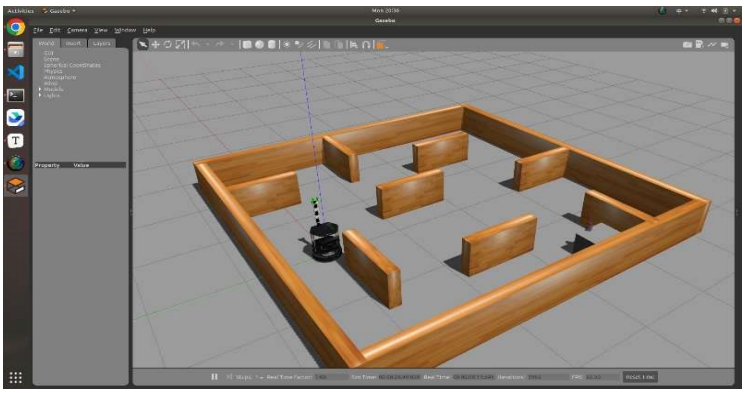
```
roslaunch turtlebot_gazebo turtlebot_world.launch
```

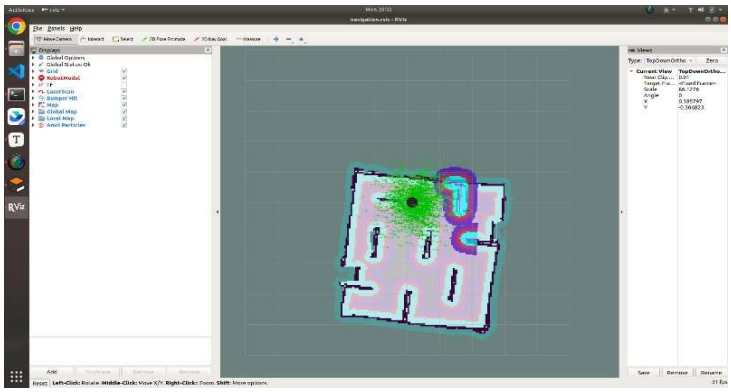
- 机器人定位建图：

```
roslaunch turtlebot_teleop keyboard_teleop.launch
roslaunch turtlebot_gazebo gmapping_demo.launch
roslaunch turtlebot_rviz_launchers view_navigation.launch
rosrun image_view image_view image:=/camera/rgb/image_raw
```

- 保存地图：

```
rosrun map_server map_saver -
f /home/longway/turtlebot_ws/src/turtlebot_simulator/turtlebot_gazeb
o/maps/coke
```





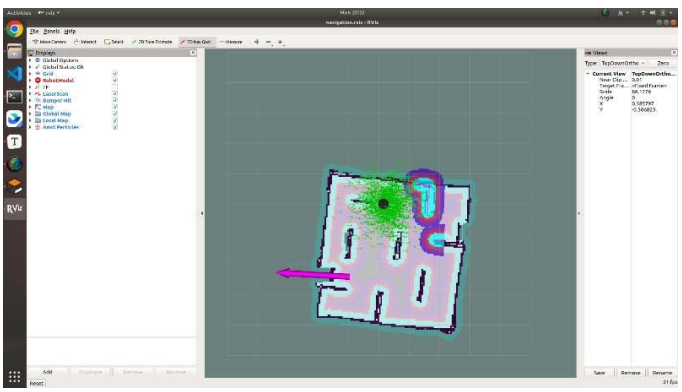
6. 机器人定点导航

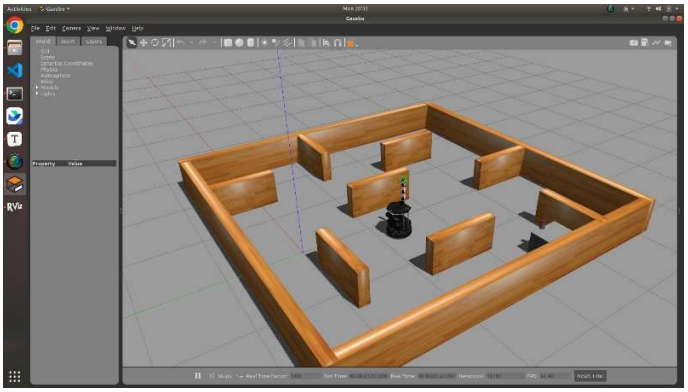
- 打开我们预先构建好的 gazebo 仿真世界，我们使用了 turtlebot2 和 turtlebot_arm 来进行实验：

```
roslaunch turtlebot_gazebo turtlebot_world.launch
```

- 接下来开启导航模块，这里使用了自适应蒙特卡洛定位 amcl 算法。
- 这个地方我们下达了一个语音指令，使用语音控制来唤起可视化工具 rviz，用来给机器人一个初始定位，和指定目标点。我们的目标是机器人能移动到我们放置的可乐模型前面。

```
roslaunch turtlebot_gazebo amcl_demo.launch map_file:=/home/longway/
turtlebot_ws/src/turtlebot_simulator/turtlebot_gazebo/maps/coke.yaml
roslaunch turtlebot_rviz_launchers view_navigation.launch
# rosrun my_pkg navigation.py
```





7. 物体识别与机械臂抓取

- 将 turtlebot_arm 连接到底盘上。在 ~/turtlebot_ws/src/turtlebot/turtlebot_description/robots/kobuki_hexagons_kinect.urdf.xacro 中添加：

```
<xacro:include filename="$(find turtlebot_arm_description)/urdf/turt
lebot_arm.xacro"/>
```

- 当机器人移动到目标点后，开启物体检测模块，YOLO 用 boundingbox 框出相机检测到的可乐的位置。
- 利用 YOLO 算法完成图像中的物体检测（设定目标物体）：

```
roslaunch darknet_ros yolo_v3.launch
```

- 我们编写了一个代码用来得到可乐的坐标信息，它从 ros 的 topic 中订阅三种

数据，一个 Kinect 的 rgb 信息，一个点云深度信息，一个 boundingbox 信息。代码通过信息确定可乐的中心坐标，然后使用 tf 函数得到可乐到机器人的坐标变换，以供机械臂抓取使用。

- 机器人在三维空间中定位目标物体：

  物体检测框 -- 物体像素坐标 (u, v) -- 对应的点云坐标 (xc, yc, zc)

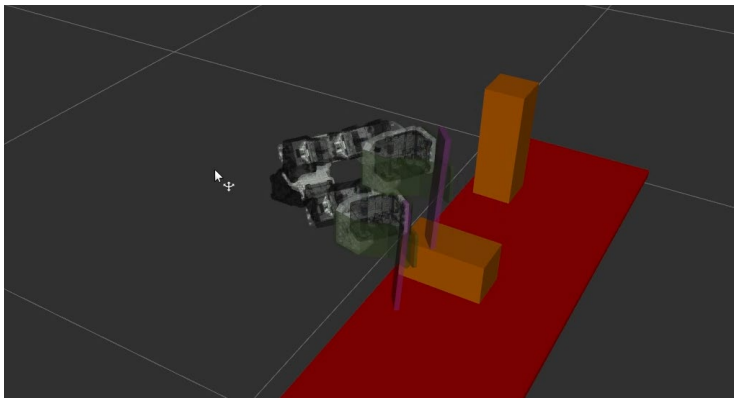
```
rosrun darknet_ros test_darknet.py
```

- Moveit! 与 Gazebo 联合仿真：

```
roslaunch turtlebot_arm_moveit_config moveit_planning_execution.launch
roslaunch turtlebot_gazebo turtlebot_world_moveit.launch
```

- 调用 moveit 中配置好的 ik 求解器和控制器完成抓取操作：

```
rosrun turtlebot_arm_moveit_demos pick_and_place.py
```



9. 总结与展望

9.1 实验总结

在这次实验中我学会了很多：

1. 掌握了 Gazebo 仿真方法，并学会了如何通过仿真模型来进行实验；
2. 通过仿真实验和具体设备实验，我对 Ubuntu 以及 ROS 系统的层级结构和通信方式的了解更加深入具体，对每一部分的作用更加熟悉；
3. 学习到了使用 Moveit! 设置及调试运行机械臂的方法；
4. 通过实际实验和仿真实验的对比我认识到很多时候理论和实际情况差距很大，理论并不完全适用于实际，这个时候我们应当慢慢尝试探索，在实践中总结新的经验来解决实践中遇到的各种问题。

9.2 不足与展望

9.2.1 不足

1. 对于机械臂的了解还不够深入，在做机械臂抓取仿真实验，修改 URDF 文件和 launch 文件时感到比较困难；

2. 阅读和编写代码的能力需要增强。

9.2.2 展望

1. 在今后的学习中更加深入地学习 ROS 和其他机器人软件相关的具体知识，并加深对其每一方面的理解认识；

2. 能够利用空余时间多多联系自己编写代码；

3. 培养自己良好的实验习惯、实验方法。多请教老师同学，善于发现问题，解决问题，培养自己的实验性思维。

10. 参考资料

https://github.com/HilbertXu/ros_task_frame

https://www.guyuehome.com/7639

https://www.ncnynl.com/category/Turtlebot-arm-code/