

车载软件设计基础

松灵小车任务1



**2023至2024学年第 1 学期**

|  |  |  |
| --- | --- | --- |
| 学号 | 姓名 | 团队作用 |
| 20203523 | 王一鸣 |  |
|  |  |  |
| 任课教师 |  | |
| 成 绩 |  | |

|  |  |
| --- | --- |
| 任务书 | |
| 任务内容 | 实验分组进行，每组人数不超过2人。在松灵小车上编写ROS程序，实现如下功能：  （1）开启松灵小车的激光雷达、深度相机等传感器，控制小车移动（遥控器），编写程序，显示移动过程中，各类传感器数据：   * 命令行窗口显示小车的**线速度和角速度** * OpenCV显示深度相机的**RGB图像和深度图** * PCL显示激光雷达的**点云数据**   （2）编程实现用里程计（odometry）计算小车移动距离   * 小车靜止不动，读取里程计数据，记为a控制小车前进n米距离(n=1、2、3均可），读取里程计数据，记为b * 建立小车移动距离与里程计读数a、b之间的关系模型（ 前两步应进行多次以拟合更精确的模型） * 控制小车移动，利用上一步构建的模型，计算小车移动的距离，并将计算结果与实际值进行对比 |
| 程序规范 | （1）所有程序代码采用C++编写，使用git进行源代码管理；  （2）类名、变量名、函数名应符合C++的命名规范，并在代码中前后保持一致；  （3）涉及面向对象的程序，例如自定义的类，应符合面向对象的设计原则；  （4）正确使用头文件和源文件，自定义的头文件应符合头文件的编写原则，例如用条件宏定义确保头文件不被多次引用、不在头文件中进行类和函数的实现（模板除外）；  （5）项目必须是ROS项目，符合ROS的项目的规范，正确编写CmakeLists.txt等文件；  （6）程序能够在松灵小车上运行。 |
| 报告要求 | （1）报告至少应该包括人员分工、需求分析、程序设计、程序效果展示、总结分析5个部分；  （2）人员分工介绍组员各自的工作情况；  （3）需求分析侧重描述程序所需要实现的功能，功能预期的效果；  （4）程序设计描述组成程序的模块、类、函数以及他们之间的相互关系，若有算法，可以描述算法流程；  （5）程序效果展示除了程序运行效果截图之外，应该有必要的文字说明；  （6）总结分析可以分析实现的效果与理想情况的差异，分析导致这些差异的原因，切忌不要写成心得体会；  （7）报告应该格式规范、排版整洁、少语病和错误。 |
| 作业提交 | （1）含有git仓库（有.git目录）的完整源代码；  （2）程序功能演示的讲解视频；  （3）任务报告。 |
| 评分标准 | 按照五级制打分，分为优秀、良好、中等、及格、不及格，各评分项占总成绩的比例为：  （1）任务完成情况占评分的60%；  （2）程序规范占评分的20%；  （3）报告占评分的20%。  评分老师根据各部分的完成情况，直接给出总成绩。 |

# 人员分工

# 需求分析

## 显示小车移动过程中的各类传感器数据

需要调用ROS的包获取小车的数据：

1. 小车线速度和角速度
2. OpenCV显示深度相机的RGB图像和深度图
3. PCL显示激光雷达的点云数据

## 用里程计(odeometry)计算小车移动距离

1. 读取里程计数据
2. 统计小车移动距离与里程计度数的数据
3. 分析误差来源

# 程序设计

## 获取传感器数据

启动激光雷达可以通过*/odom*消息获取线速度和角速度

启动深度相机可以通过*/camera*下一系列消息获得彩色图、深度图

## 计算小车移动距离

通过*/odom*获取小车的线速度，通过计算小车移动距离。

# 程序效果展示

## 分析小车移动数据

|  |  |
| --- | --- |
| ROS-TOPIC | CAR |
| 4.564711 | 4.78 |
| 7.682313 | 7.985 |
| 1.418964 | 1.521 |
| 9.052275 | 9.65 |

Python代码示例：

1. **import** numpy as np
2. **import** pandas as pd
3. **from** matplotlib **import** pyplot as plt
4. **from** scipy **import** optimize

7. **def** fitt(x, k):
8. **return** k \* x

11. data = pd.read\_csv("./data.CSV")
12. x0 = data["ROS-TOPIC"]
13. y0 = data["CAR"]
15. k, eps = optimize.curve\_fit(fitt, x0, y0)
16. **print**("curve fitting: y=%0.4fx  error is %0.6f" % (k, eps))
18. x = np.arange(x0[0], x0[len(x0) - 1], 0.01)
19. y = func(x, k)
20. plt.figure()
21. plt.scatter(x0.to\_list(), y0.to\_list(), 25, "blue")
22. plt.plot(x, y, "red")
23. plt.show()

# 总结分析

较好地完成了本方面任务。