

车载软件开发基础

松灵小车任务1



**2023至2024学年第 1 学期**

|  |  |  |
| --- | --- | --- |
| 学号 | 姓名 | 团队作用 |
| 202324131109 | 阳沐云 | 主要负责完成任务（2）：编程实现用里程计计算小车移动距离,协作完成任务（1），撰写报告。 |
| 202324131117 | 李文鑫 | 主要负责完成任务（1）：编写程序，显示移动过程中，各类传感器数据,协作完成任务（2），撰写报告。 |
| 任课教师 | 刘骥 | |
| 成 绩 |  | |

|  |  |
| --- | --- |
| 任务书 | |
| 任务内容 | 实验分组进行，每组人数不超过4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4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。 |

**一、人员分工**

**阳沐云：**按照分工，主要负责完成任务（2）：编程实现用里程计算小车移动距离,协作完成任务（1），撰写报告。按照分工，主要完成内容包括：show\_odom.cpp的编写，以及合作完成show\_camera.cpp、show\_pl.cpp、show\_speed.cpp的编写。

**李文鑫：**按照分工，主要负责完成任务（1）：编写程序，显示移动过程中，各类传感器数据,协作完成任务（2），撰写报告。主要完成内容包括：show\_camera.cpp、show\_pl.cpp、show\_speed.cpp的编写，以及合作完成show\_odom.cpp的编写。

**二、需求分析**

本次任务一共编写了四个cpp文件，完成需要实现的功能。

1. **show\_speed.cpp：命令行窗口显示小车的线速度和角速度**

订阅IMU数据、计算角速度、计算线速度、输出速度信息、保存和更新先前的时间和方向。

1. **show\_camera.cpp: OpenCV显示深度相机的RGB图像和深度图**

图像订阅与处理、图像转换、图像显示、异常处理、消息传递

1. **show\_pl.cpp:  PCL显示激光雷达的点云数据**

初始化与ROS节点设置、订阅LIDAR数据、LIDAR数据解析与存储、可视化LIDAR点云、处理ROS回调

1. **show\_odom.cpp: 用里程计（odometry）计算小车移动距离**

实时里程计算、里程累加、时间同步

**三、程序设计（组成程序的模块、类、函数和关键算法）**

**① show\_speed：命令行窗口显示小车的线速度和角速度**

该程序用于计算小车运行的角速度和线速度。以下是程序的简要介绍：

**所用函数：**

1. imuCallback(const sensor\_msgs::Imu::ConstPtr& msg)：IMU消息的回调函数，接收IMU数据并进行处理。

* 通过消息中的时间戳获取当前时间。
* 获取IMU数据的角速度信息，并计算角速度的合速度。
* 获取IMU数据的姿态信息（四元数），将其转换为欧拉角。
* 计算角速度的方向（yaw角度变化率）。
* 根据假设的车轮半径，计算线速度。
* 打印线速度和角速度信息。
* 更新上一次的时间戳和yaw角度。

**关键算法：**

1. 计算角速度的合速度：通过将三个方向上的角速度平方和的平方根，得到角速度的合速度。
2. 将四元数转换为欧拉角：使用tf库的Quaternion和Matrix3x3类，将四元数转换为欧拉角，其中yaw角表示姿态的航向角。

该程序通过订阅IMU数据的话题，获取角速度和姿态信息，并根据这些信息计算线速度和角速度。最后，将计算得到的线速度和角速度打印到控制台。整个程序在ROS系统中以节点的形式运行，通过ros::spin()函数实现循环监听消息的功能。

**② show\_camera.cpp: OpenCV显示深度相机的RGB图像和深度图**

该程序用于显示相机采集的RGB图像和深度图像。以下是程序的简要介绍

**所用函数：**

1. rgbImageCallback(const sensor\_msgs::ImageConstPtr& msg)：RGB图像消息的回调函数，接收相机采集的RGB图像数据并进行处理。

* 将ROS图像消息转换为OpenCV图像。
* 使用OpenCV的imshow函数显示RGB图像。
* 使用OpenCV的waitKey函数等待用户按键输入。

1. depthImageCallback(const sensor\_msgs::ImageConstPtr& msg)：深度图像消息的回调函数，接收相机采集的深度图像数据并进行处理。

* 将ROS图像消息转换为OpenCV图像。
* 使用OpenCV的imshow函数显示深度图像。
* 使用OpenCV的waitKey函数等待用户按键输入。

**关键算法：**

1. 将ROS图像消息转换为OpenCV图像：使用cv\_bridge库中的toCvCopy函数，将ROS图像消息转换为OpenCV图像，以便进行后续的处理和显示。

该程序通过订阅RGB图像和深度图像的话题，接收相机采集的图像数据，并使用OpenCV库显示图像。整个程序在ROS系统中以节点的形式运行，通过ros::spin()函数实现循环监听消息的功能。

**③ show\_pl.cpp:  PCL显示激光雷达的点云数据**

该程序用于显示激光雷达采集的点云数据。以下是程序的简要介绍：

**所用函数：**

1. lidarCallback(const sensor\_msgs::PointCloud2::ConstPtr &data)：激光雷达数据的回调函数，接收激光雷达采集的点云数据并进行处理。

* 将ROS点云消息转换为PCL点云数据。
* 标记新的点云数据可用。

1. main(int argc, char \*\*argv)：主函数，程序的入口。

* 初始化ROS节点。
* 创建ROS节点句柄。
* 订阅激光雷达数据的话题。
* 创建PCL点云可视化窗口。
* 设置定时器，每100毫秒更新一次点云数据。
* 在循环中检查是否有新的点云数据可用，如果有则显示点云数据。
* 处理ROS回调函数。
* 控制程序的运行频率。

**关键算法：**

1. 将ROS点云消息转换为PCL点云数据：使用pcl\_conversions库中的fromROSMsg函数，将ROS点云消息转换为PCL点云数据，以便进行后续的处理和显示。

该程序通过订阅激光雷达数据的话题，接收激光雷达采集的点云数据，并使用PCL库进行点云数据的处理和显示。整个程序在ROS系统中以节点的形式运行，通过循环等待新的点云数据并更新显示。

**④ show\_odom.cpp: 用里程计（odometry）计算小车移动距离**

该程序用于监听并计算里程计信息。以下是程序的简要介绍：

**所用函数：**

1. odomCallback(const nav\_msgs::Odometry::ConstPtr& msg)：里程计信息的回调函数，接收松灵小车的里程计信息并进行处理。

* 获取当前时间戳。
* 计算时间差（dt）。
* 获取线速度信息。
* 计算松灵小车在这段时间内移动的距离。
* 累加总里程。
* 更新上一个时间戳。
* 输出总里程。

**关键算法：**

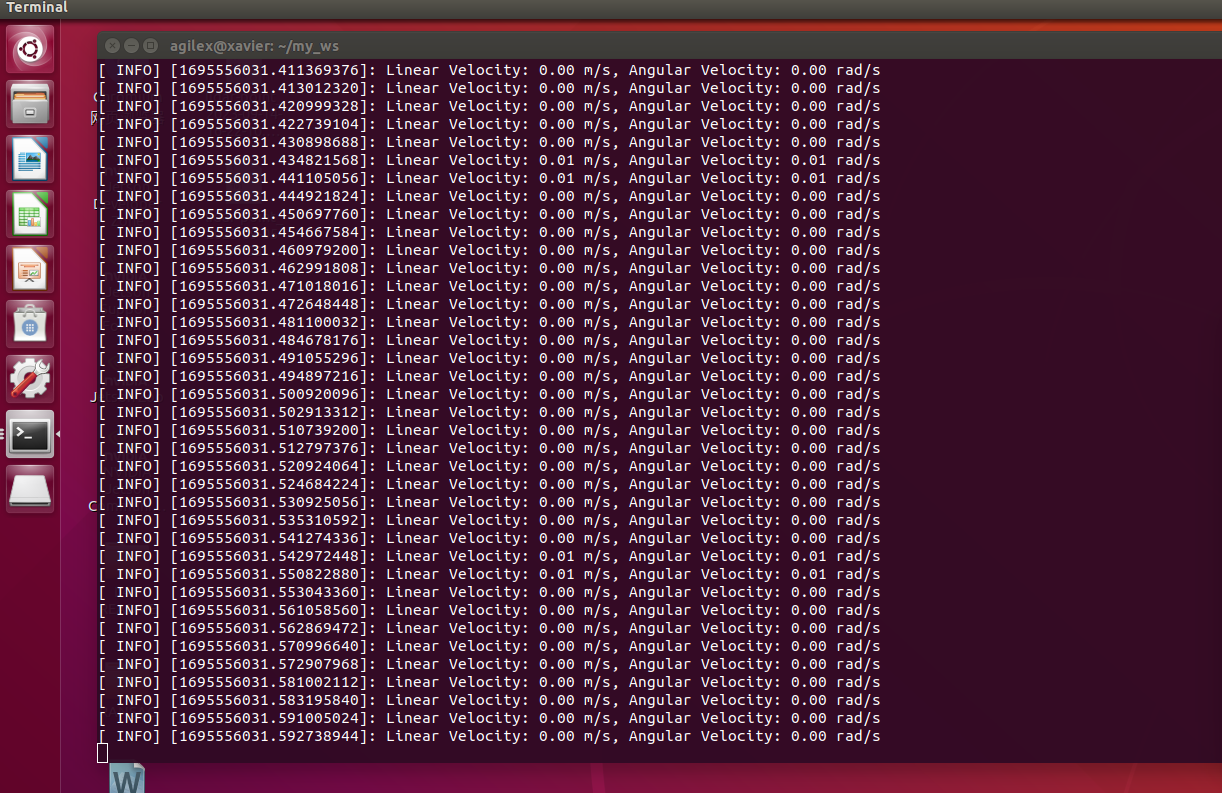
1. 时间差计算：通过获取当前时间戳和上一个时间戳，计算时间差（dt），以确定松灵小车移动的时间间隔。
2. 距离计算：通过获取线速度信息和时间差，计算松灵小车在这段时间内移动的距离（线速度乘以时间差）。
3. 总里程累加：每次计算得到的距离都会累加到总里程中。

该程序通过订阅松灵小车的里程计话题（/odom），接收里程计信息，并根据线速度和时间差计算松灵小车的移动距离，然后累加到总里程中。程序在ROS系统中以节点的形式运行，通过ros::spin()函数实现循环监听里程计信息的功能。

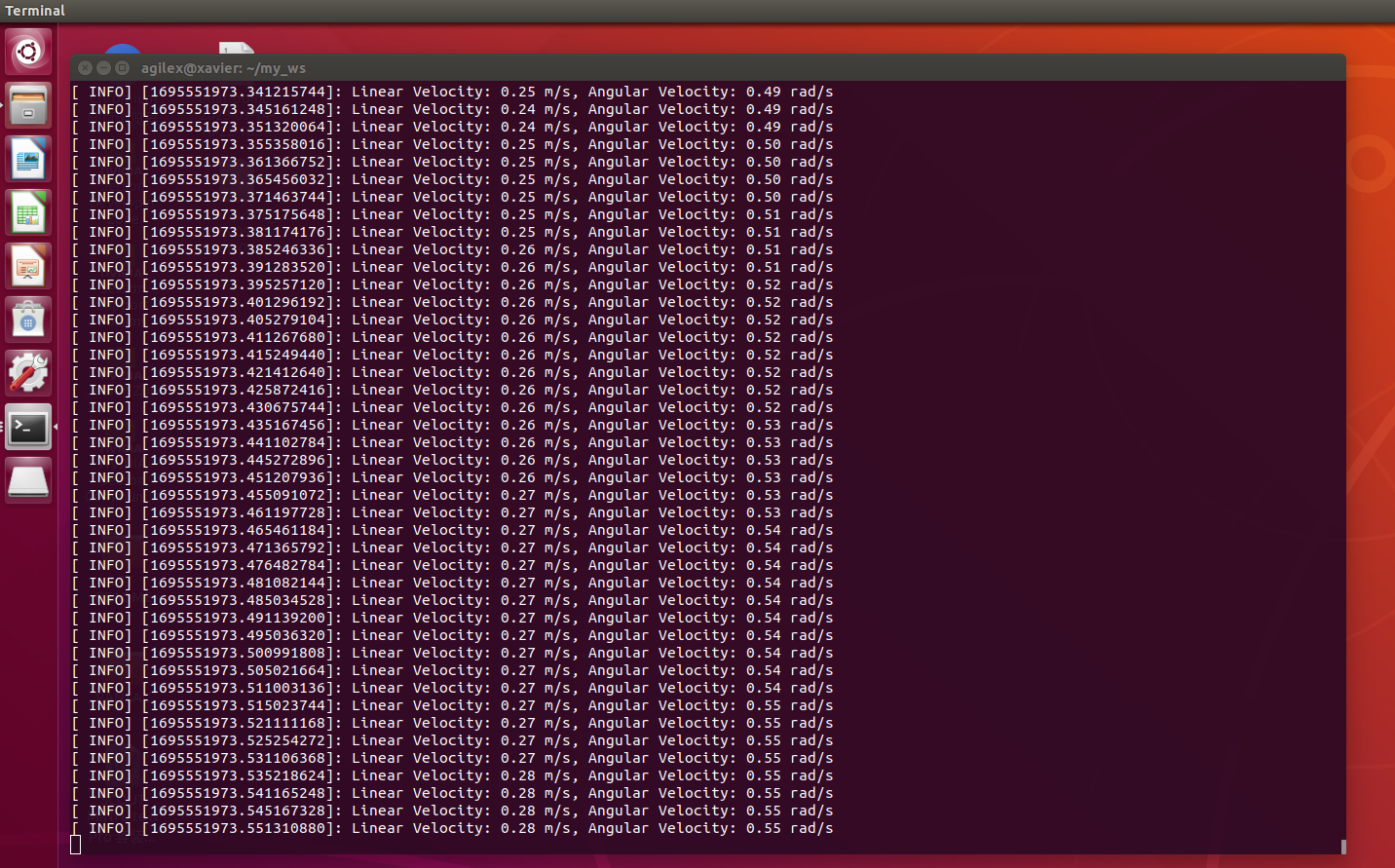
**四、程序效果展示**

**1)命令行窗口显示小车的线速度和角速度**

小车静止时的线速度与角速度：

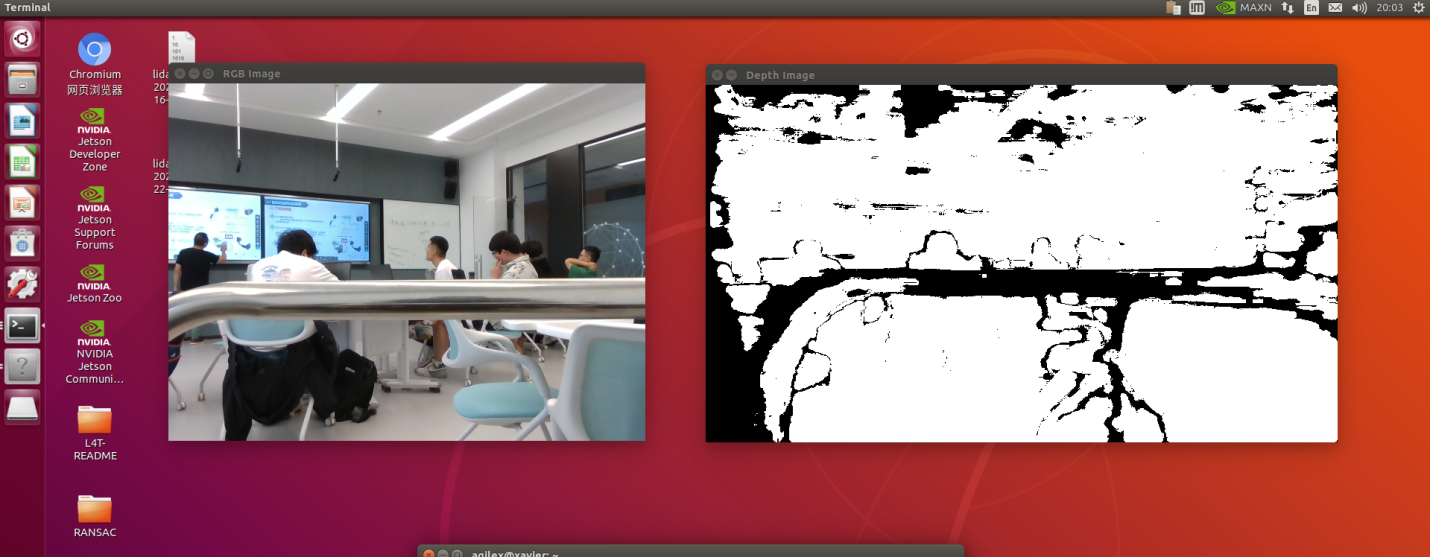
****

小车行驶时的线速度与角速度：



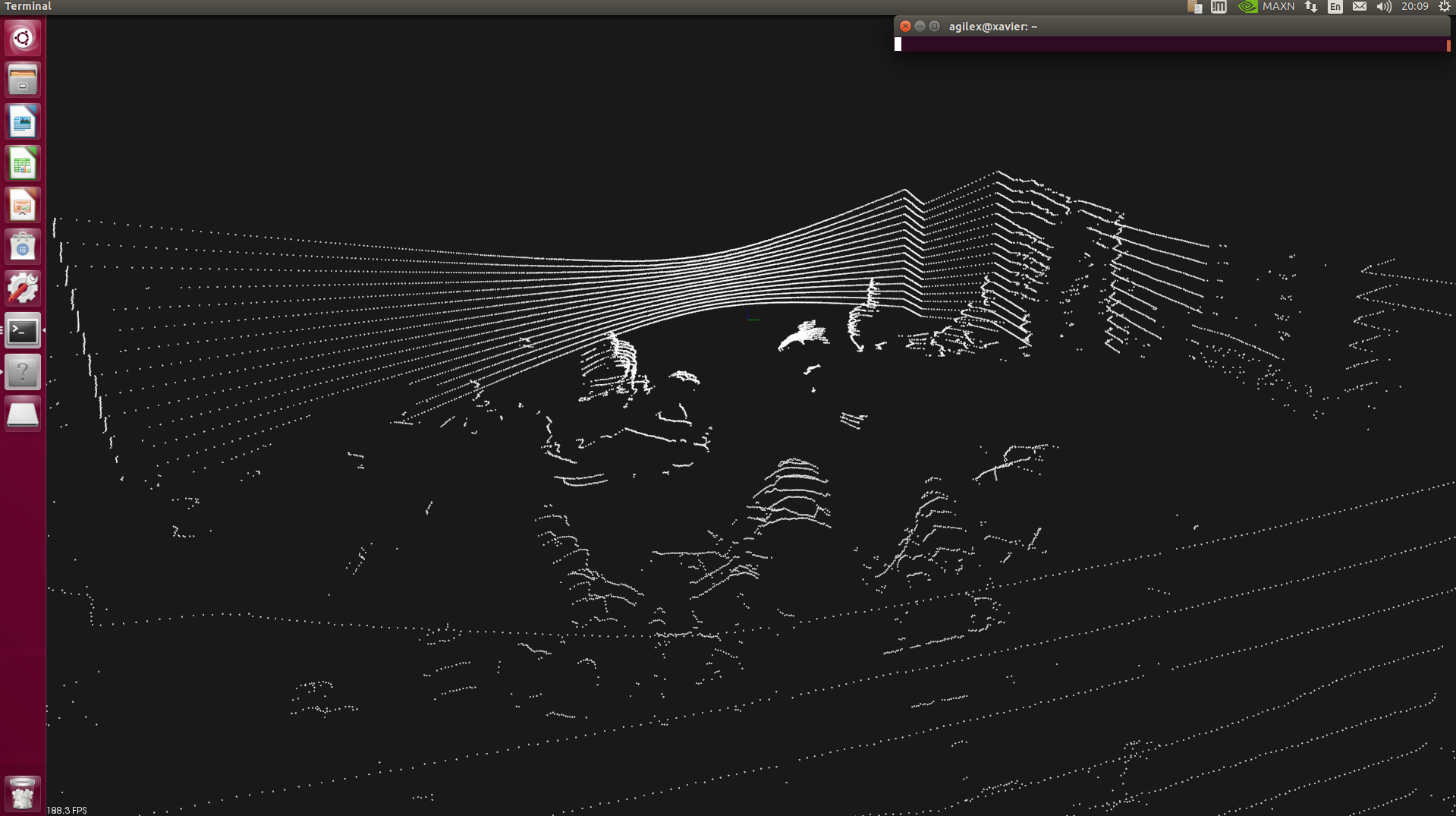
**2)** **OpenCV显示深度相机的RGB图像和深度图**

通过小车深度相机获取得到的RGB图像（左）与对应的深度图（右）

****

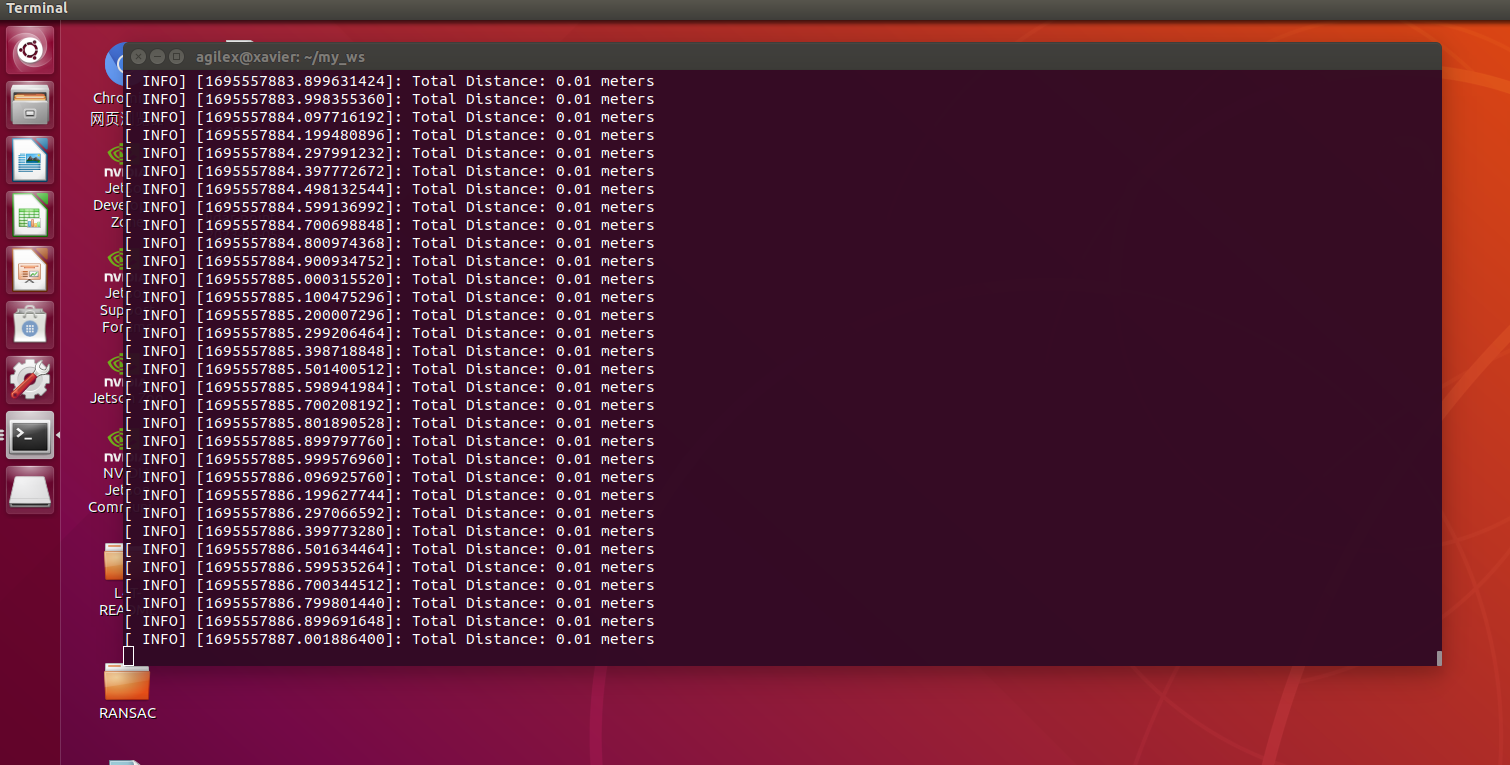
**3)** **PCL显示激光雷达的点云数据**

通过激光雷达获取得到的教室内点云数据可视化：

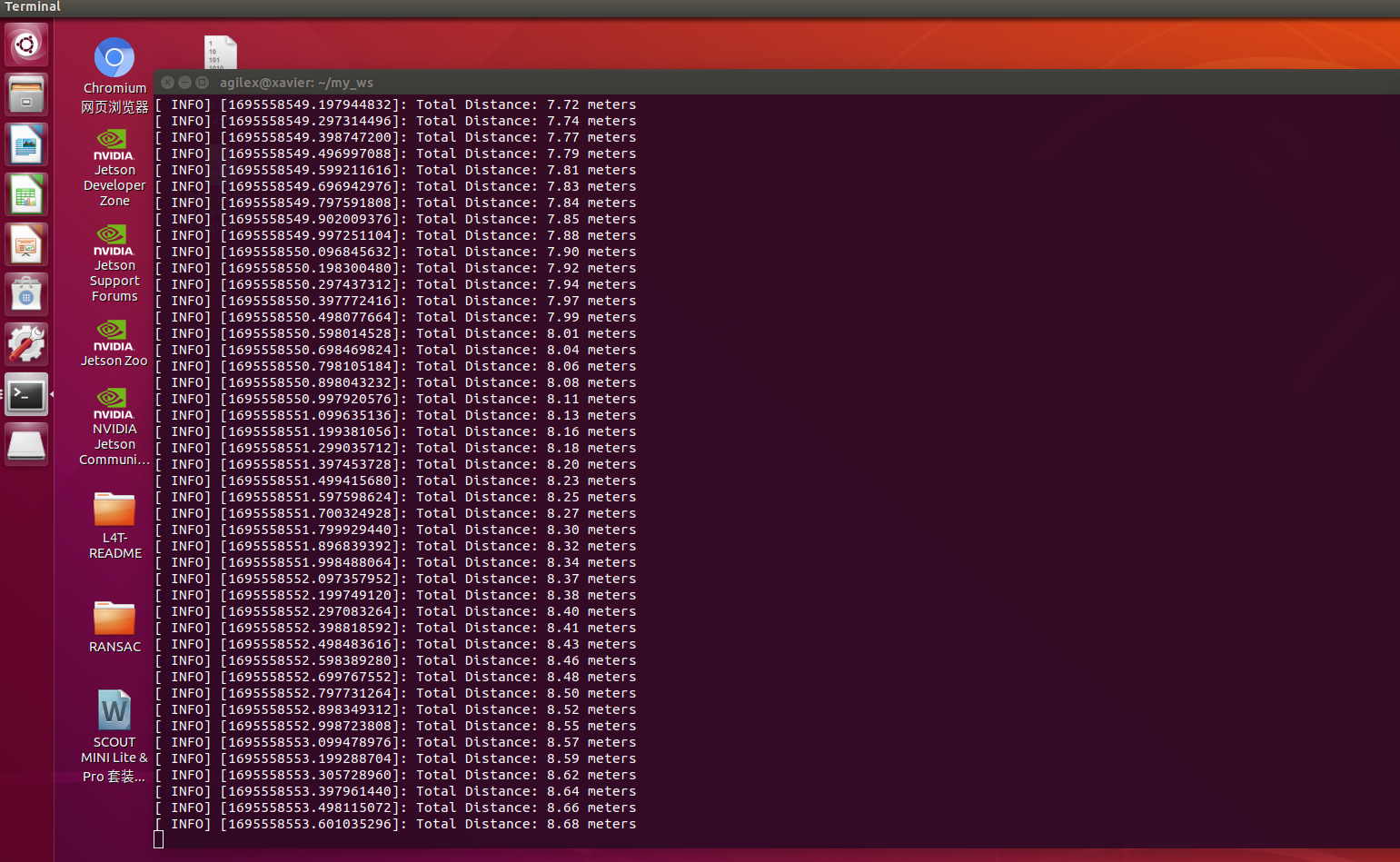
****

**4)编程实现用里程计（odometry）计算小车移动距离**

小车静止时移动距离：

****

小车行驶时移动距离：

****

**五、总结和分析**

在任务过程中，我们发现以下关键点：

1. 开启传感器时，需要保证ROS程序的正确编写和配置，以便成功获取各传感器的数据。此外，需要注意各传感器的初始化过程
2. 控制小车移动时，ROS程序需要正确解析并处理遥控器发出的信号。我们使用了ROS的订阅和发布机制来实现对遥控器信号的接收和处理。
3. 在计算小车移动距离时，我们需要准确读取里程计的数据并建立合适的模型来描述里程计读数与小车移动距离之间的关系。在实验过程中，我们发现里程计的读数与小车实际移动距离之间存在一定的误差，这可能与里程计的精度有关。为了减小误差，我们可以进行多次实验并进行数据拟合，以获取更精确的模型参数。