控制系统综合设计与实验

报告人： 张耀坤 M201872549

目录

01

实验一

Dobot机械臂综合实验

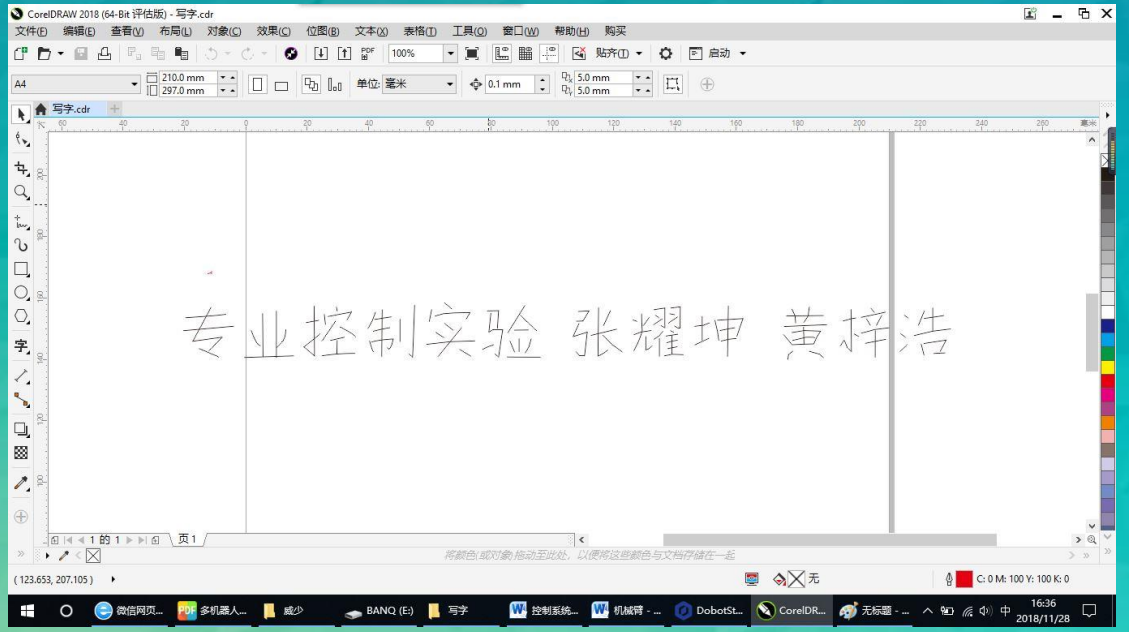
1.1实验目的：

(1)了解机械臂和传送带的组成、工作原理以及性能指标；
(2)了解机械臂的机械系统和控制系统的组成、相关软件的操作及应用；
(3)编写代码，独立实现机械臂文字书写、搬运装配的过程，掌握机械臂示教作业的方法；
(4)通过对这个实验的操作、调试，并结合实验环境搭建、编程、实验结果分析等环节，加深对本专业理论知识及应用技能技巧的理解和掌握，锻炼学生的研究能力、创新思维以及独立解决技术难题的能力。

1.2实验内容

机械臂书写实验

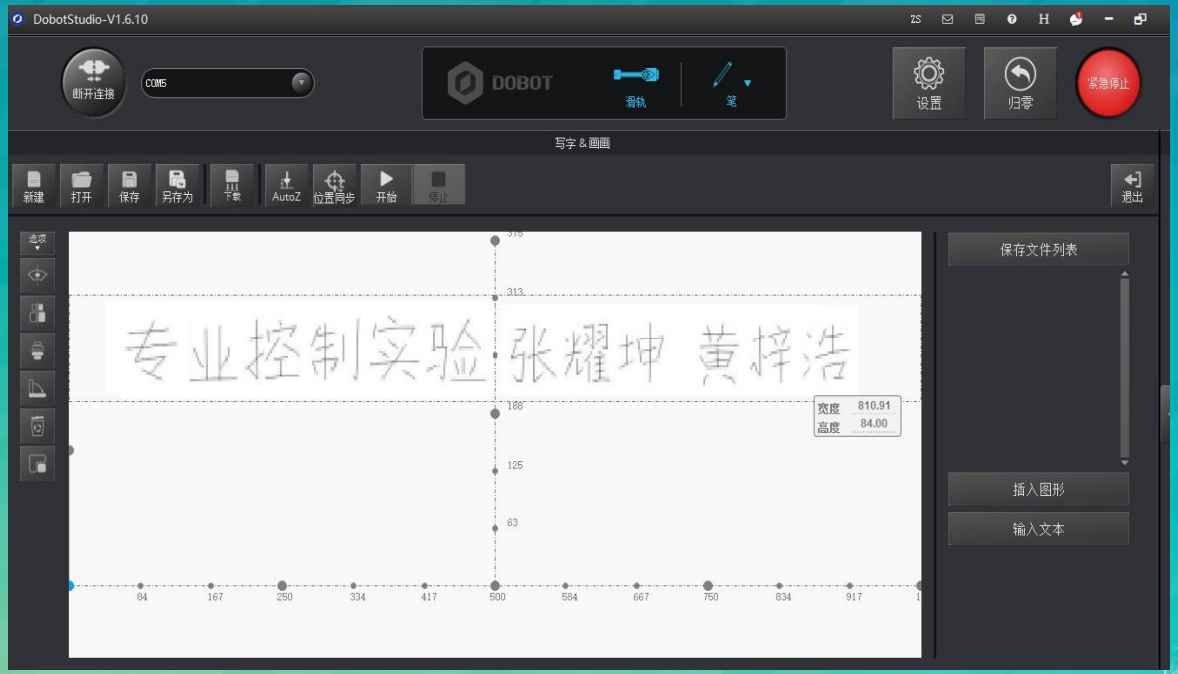
Dobot机械臂写字画画，首先需要将编辑好字体的文字图片转化为PLT格式（或者SVG格式），将字体内容转换为该格式有很多种方法，在本实验中我们以矢量绘图软件CorelDRAW X7来进行绘制。



1.2实验内容

机械臂书写实验

导入图案，设置参数：写字。进入Dobot操作平台的写字画画功能区，打开刚才做好的PLT文件，连接好机械臂和电脑，打开电源，连接电脑和机械臂的通讯，并进行归零操作。打开滑轨功能，并将末端设置为笔，同时打开制作好的PLT文件，将字放入两条横线之间，因为机械臂的臂长限制，能精确写字的部位为两条直线之间。



1.2实验内容

机械臂书写实验结果展示

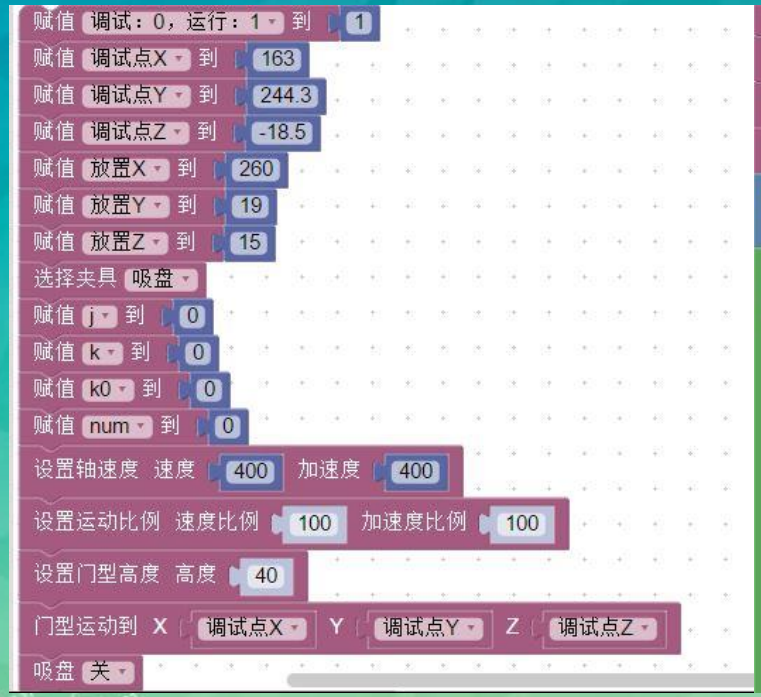
1.2实验内容

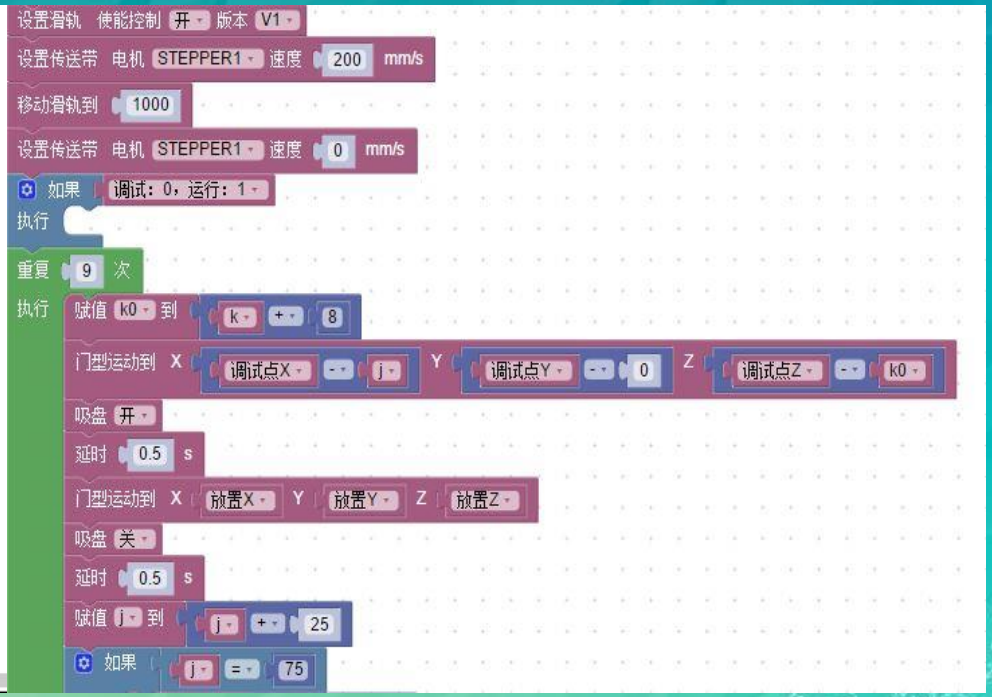
机械臂搬运实验

机械臂按照设计好的程序，通过吸盘把垂直放在桌面上的3×3木块按从上到下从左到右的顺序依次摆在传送带上。当9个木块全部摆放到传送带上后，机械臂再通过吸盘把传送带上木块运送到桌子上，并按照原来木块摆放的顺序摆放整齐。代码见附录（附有视频）。

1.2实验内容

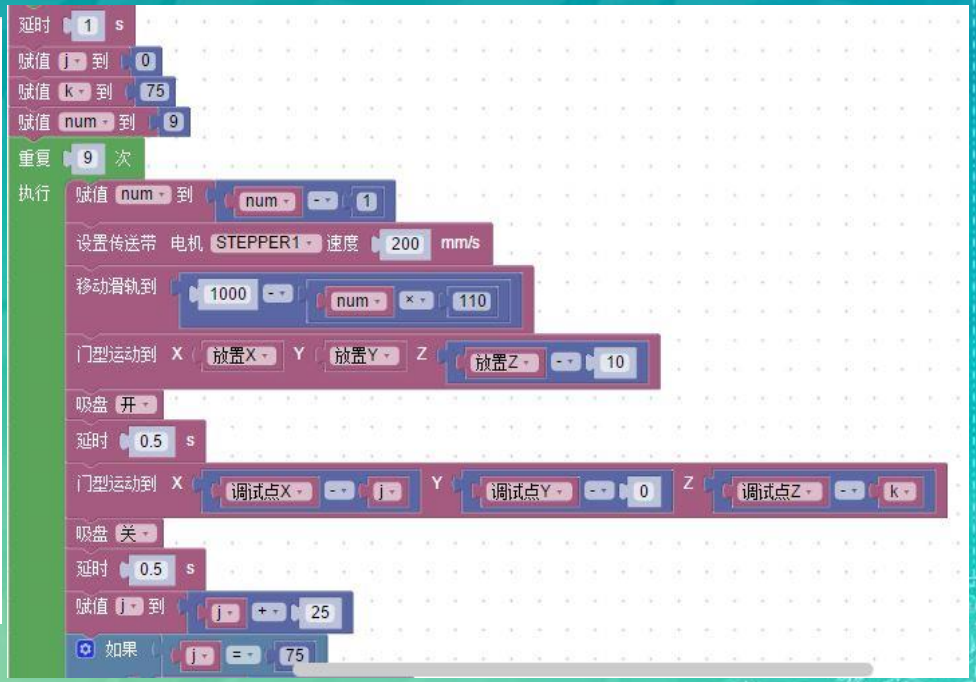
机械臂搬运实验程序





1.2实验内容

机械臂搬运实验程序



1.2实验内容

机械臂搬运实验结果展示

02

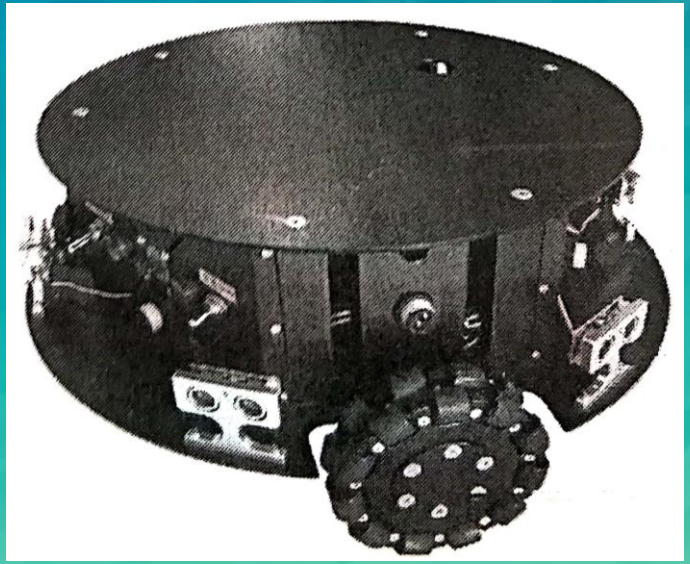
实验二

基于全向移动平台的Kinect二维SLAM建图

实验目的：

1）了解并熟悉Ubuntu系统的使用与操作方法，掌握在Ubuntu系统下安装机器人操作系统（ROS）的方法；
2）理解ROS的文件系统，掌握ROS架构组成及相关的概念，学会创建功能包、节点，发布与订阅话题等基本操作，并能够使用相关调试工具对系统进行调试；
3）了解全向移动机器人平台的硬件构成，掌握机器人平台的串口通信协议并实现与机器人平台的通信；
4）掌握Kinect视觉传感器的使用方法，包括驱动安装、数据获取与应用等；
5）了解二维SLAM建图的原理，掌握相关SLAM算法及其应用；
6）完成基于全向移动机器人平台和Kinect的二维SLAM建图综合实验；
通过对实验环境搭建、编程、调试及实验结果分析等环节，加深对相关理论知识及应用技能的理解和掌握，锻炼学生独立解决技术难题的能力。

实验装置：



三轮全向移动平台

Kinect V1

2.4实验过程：

第一步：ROS操作系统安装

在外置移动硬盘上安装Ubuntu16.04系统，然后安装Kinetic版本的ROS系统，并配置相关环境。

（1）安装ROS

在Ubuntu系统下打开一个终端，输入如下指令添加源文件：

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

再输入如下指令设置秘钥：

```
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 0xB01FA116
```

然后输入指令更新系统软件使之处于最新版：`$ sudo apt-get update`

最后获取安装包安装全功能版ROS：

```
$ sudo apt-get install ros-kinetic-desktop-full
```

2.4实验过程：

第一步：ROS操作系统安装

（2）配置ROS工作环境

首先初始化rosdep，使用指令如下：

$ sudo rosdep init

$ rosdep update

然后初始化环境变量

$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc

$ source ~/.bashrc

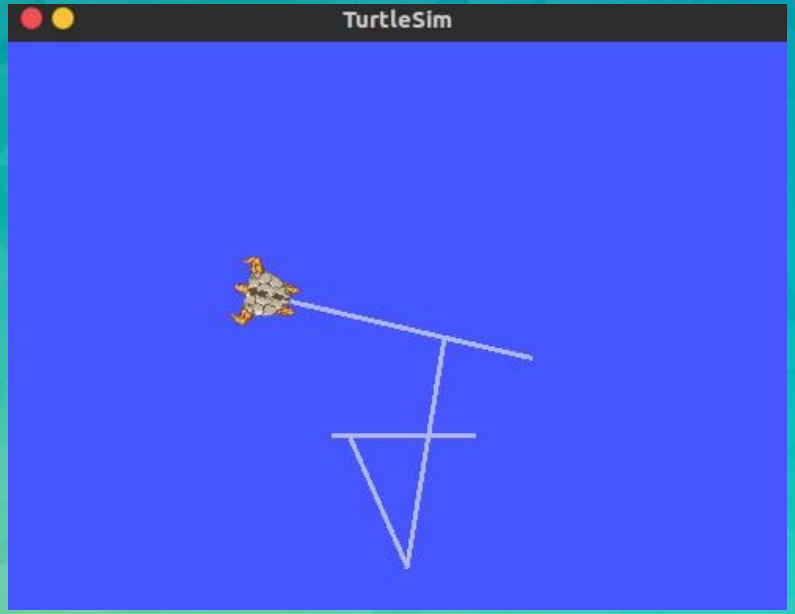
最后测试ROS是否安装成功：使用指令$ roscore启动ROS环境，终端中显示图4所示的信息，表明ROS安装成功并能够顺利启动。

```
hust@hust-ZYK:~$ roscore
... logging to /home/hust/.ros/log/90a422f4-189c-11e9-99a8-7429af526657/roslau
nch-hust-ZYK-3693.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://hust-ZYK:38045/
ros_comm version 1.12.14


SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES

auto-starting new master
process[master]: started with pid [3711]
ROS_MASTER_URI=http://hust-ZYK:11311/

setting /run_id to 90a422f4-189c-11e9-99a8-7429af526657
process[rosout-1]: started with pid [3724]
started core service [/rosout]
```

2.4实验过程：

第一步：ROS操作系统安装

（3）最后通过运行官方的小海龟测试程序熟悉ROS系统的操作。使用指令获取小海龟测试程序：$ sudo apt-get install ros-kinetic-turtlesim。
然后，在三个不同的终端分别执行以下两条指令：
$ rosrun turtlesim turtlesim_node
$ rosrun turtlesim turtle_teleop_key
通过键盘控制界面中出现的小海龟向前、向后以及旋转的运动，得到测试界面如右图所示。



2.4实验过程：

第二步：全向移动平台的键盘操控

由于要借助全向移动平台的运动对周围环境进行地图的构建，因此需要实现平台的键盘操控功能，按照实验原理部分的分析，首先要读取电脑键盘的键值，这一子功能作为一个节点；然后将键盘键值转化为移动平台的运动指令，并借助串口通信协议发送给移动平台，使之做相应的运动，这些子功能作为另一个节点。

（1）读取键盘键值的节点创建

（2）控制平台运动节点的创建

2.4实验过程：

第二步：全向移动平台的键盘操控

（1）读取键盘键值的节点创建

为了使平台具有更多的运动形式，下载键盘控制的ROS包，并将该功能包放置到当前工作空间中，使用如下指令操作：

$ git clone https://github.com/Forrest-Z/teleop_twist_keyboard.git

然后编译当前工作空间，然后通过使用如下指令启动键盘节点：

$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py。

2.4实验过程：

第二步：全向移动平台的键盘操控

（1）读取键盘键值的节点创建

```
hust@hust-ZYK:~/zykctk_ws$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py

Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5       turn 1.0
```

节点teleop_twist_keyboard的运行界面

实验过程：

第二步：全向移动平台的键盘操控

（1）读取键盘键值的节点创建

当我们按下键盘时，teleop_twist_keyboard节点会发布/cmd_vel主题发布速度信息，通过指令 $ rostopic echo /cmd_vel，可查看速度信息的具体情况如图7所示，可知，该速度信息包含三轴的位移速率和三轴的角速率。

```
hust@hust-ZYK:~/zykctk_ws$ source ~/zykctk_ws/dev
el/setup.bash
hust@hust-ZYK:~/zykctk_ws$ rostopic echo /cmd_vel
linear:
  x: 0.5
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 1.0
---
```

/cmd_vel话题内容

2.4实验过程：

第二步：全向移动平台的键盘操控

（2）控制平台运动节点的创建

在base_controller节点中订阅teleop_twist_keyboard节点发布的/cmd_vel话题，接收到速度指令，然后按照通信协议转换指令格式并写入到串口发送函数中，完成指令的传输。根据实际的运动情况，只用到X轴向的线速度和Z轴向的角速度信息。base_controller节点源码在附录中给出。

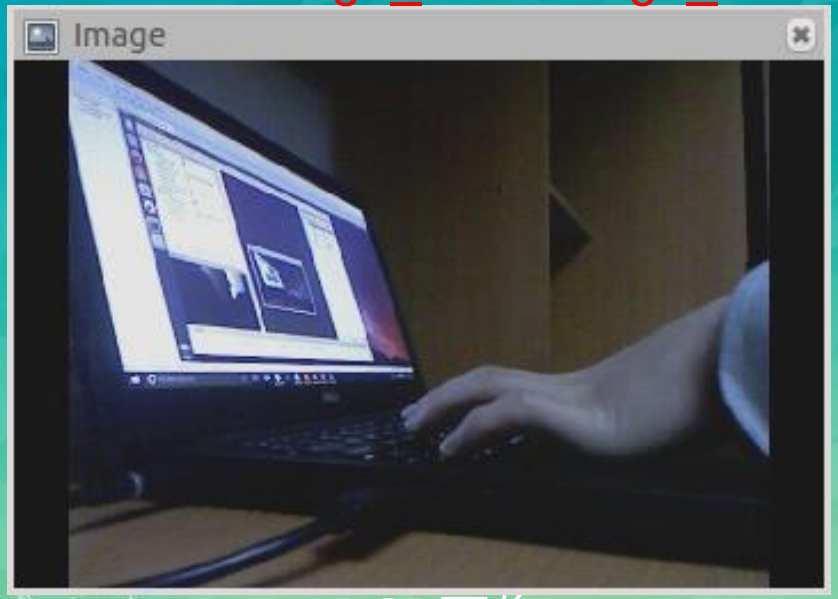
平台运动控制的节点的启动可通过如下指令操作：

```
$ rosrun base_controller base_controller
```

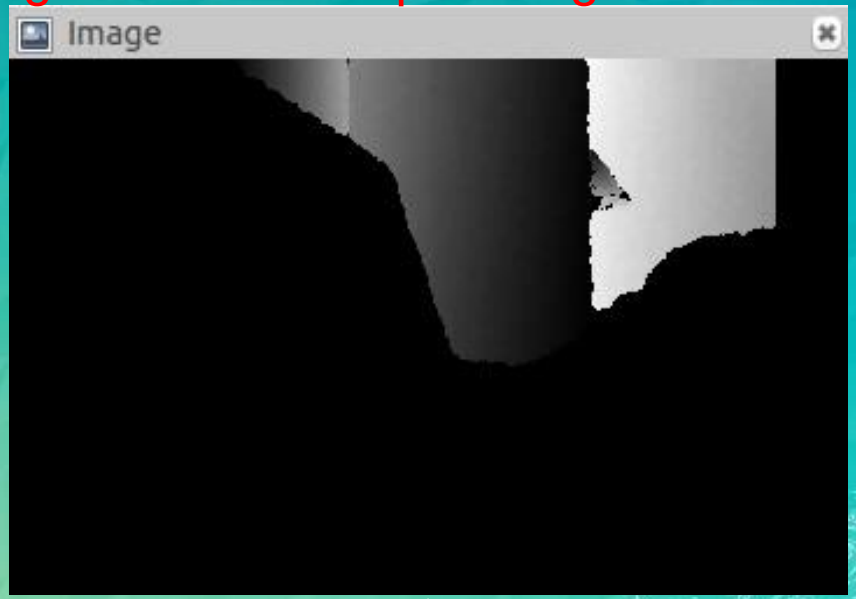
实验过程：

3.1安装Kinect驱动

```
$ sudo apt-get install  ros-kinetic-freenect-*
$ rosrun image_view image_view image:=/camera/rgb/image_color
$ rosrun image_view image_view image:=/camera/depth/image
```



RGB图像

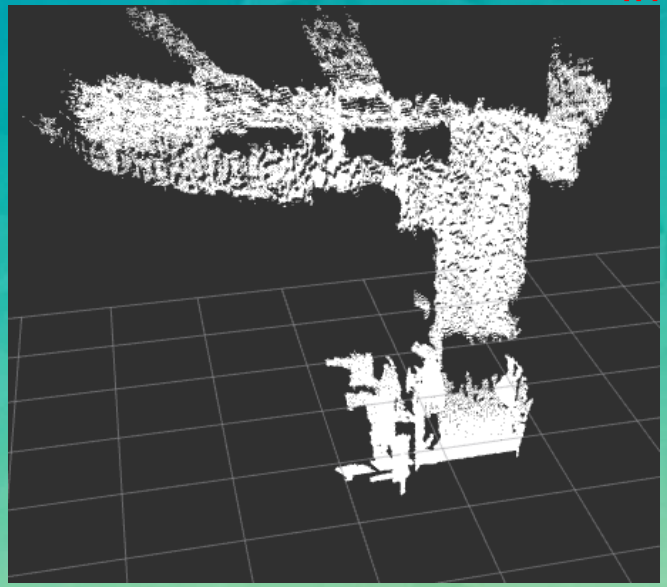


深度图像

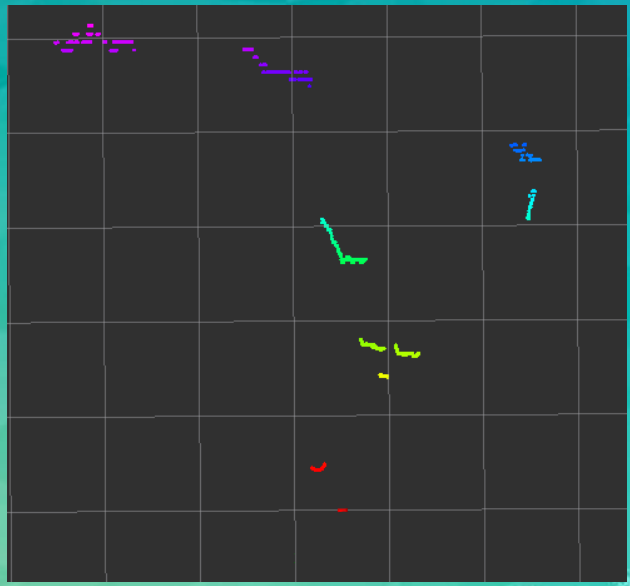
2.4实验过程：

3.2、将 Kinect深度信息转化为激光雷达数据

```
$ git clone https://github.com/ros-perception/depthimage_to_laserscan.git
$rosrun depthimage_to_laserscan depthimage_to_laserscan
                                    image:=/camera/depth/image_raw
```



点云图像



伪激光点

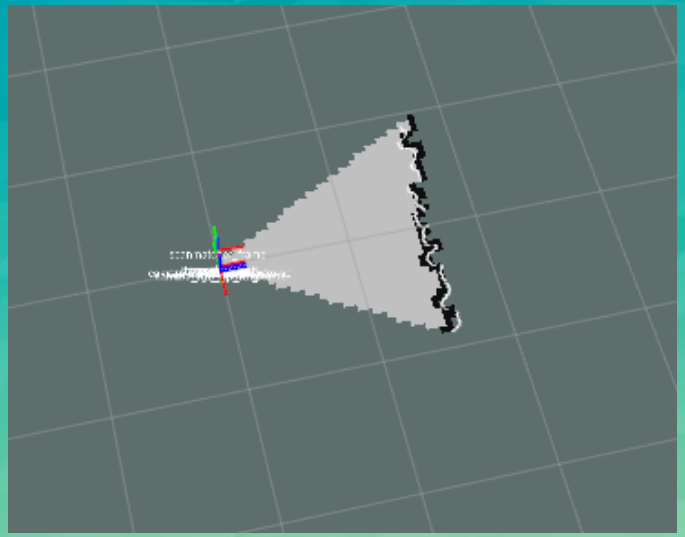
2.5实验结果展示：

Kinect传感器固定在全向移动平台上，通过键盘操控平台移动对周围环境进行地图创建。

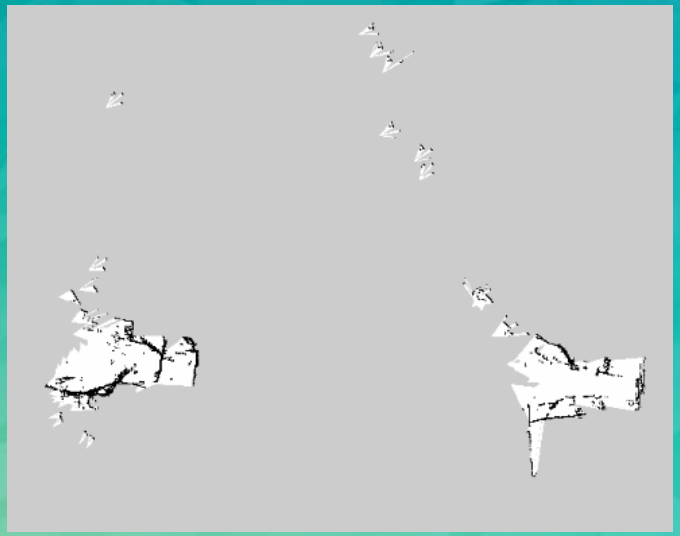
2.5实验结果展示：

使用Hector_slam算法

```
$ git clone https://github.com/DaikiMaekawa/hector_slam_example
$ roslaunch hector_slam_example hector_openni.launch
```



a) 未移动平台时创建的地图

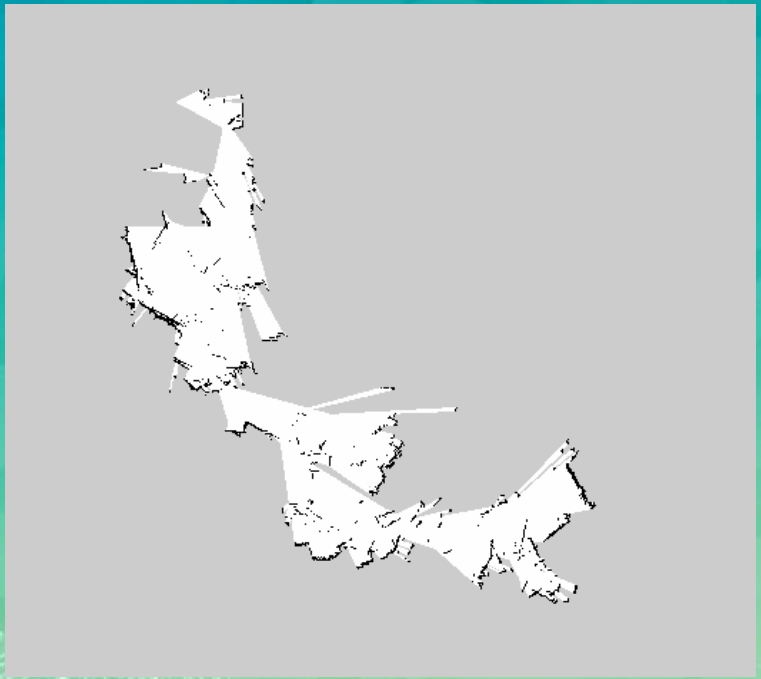


b)移动平台时创建的地图

2.5实验结果展示：

使用Gmapping算法

$ roslaunch base_controller zykslaunch.launch，同时启动其余各个功能节点，便可控制平台移动进行地图的创建。



由于未对所使用的全向移动平台的运动学模型进行分析，使得最终创建的地图和周围环境的实际情况有较大差异，以后有机会要在这方面作出改进。

使用Gmapping算法的建图结果

2.5实验结果展示：

二维SLAM建图实验结果

谢谢观看