

東北大學

信息科学与工程学院
计算机视觉课程设计实验报告

东北大学信息科学与工程学院

2021 年 12 月 13 日

一、相机标定实验

(1) 实验内容和实验步骤

➤ 相机标定实验需要写的内容:

1) 拍摄的图片;



图 1: 拍摄的图片

2) 标定的结果, 包括相机的内参矩阵、径向和切向畸变参数、重投影误差;

```
<data>
  5.0973440493282772e+02 0. 3.195000000000000e+02 0.
  5.0973440493282772e+02 2.395000000000000e+02 0. 0. 1.</data></camera_matrix>

<data>
  1.6305493474984151e-01 -8.7924295536443997e-01 0. 0.
  1.0833657276054698e+00</data></distortion_coefficients>

<data>
  1.83506593e-01 2.62221247e-01 4.30098504e-01 1.77741289e-01
  1.00446701e-01 2.01398283e-01 5.19664586e-01 3.92103583e-01
  3.02759111e-01 1.98299050e-01 2.10376397e-01 1.76716611e-01
  3.26387435e-01 1.90870747e-01 4.69741106e+00 2.30128944e-01
  9.57330167e-02 2.42778152e-01 1.53544828e-01 9.00925770e-02
  1.03062563e-01 9.14743915e-02 1.23013936e-01 9.54304412e-02
  1.64654732e-01</data></per_view_reprojection_errors>

<avg_reprojection_error>9.6812517208994886e-01</avg_reprojection_error>
```

3) 解释相机内参、畸变参数的含义。

定义相机深度 z_c , 像素坐标 P_x , 相机外参 K , 相机内参 RT , 世界坐标 P_w , 则有:

$$z_c \cdot P_x = K \cdot RT \cdot P_w \quad (1)$$

即通过相机外参, 将世界坐标系转换为相机坐标系, 再通过相机内参可以将相机坐标系转换为像素坐标系。

畸变参数有 5 组, k_1, k_2, k_3 为径向畸变参数, p_1, p_2 为切向畸变参数, 径

向畸变发生在相机坐标系转像物理坐标系的过程中，切向畸变产生的原因是透镜不完全平行于图像。

(2) 主要仪器设备和编程软件

系统：基于 x86 的 Ubuntu20.04 LTS

软件：OpenCV 4.5.4

代码托管：Github

标定板：8x11 棋盘标定板（15mm）

(3) 实验数据记录和结果分析

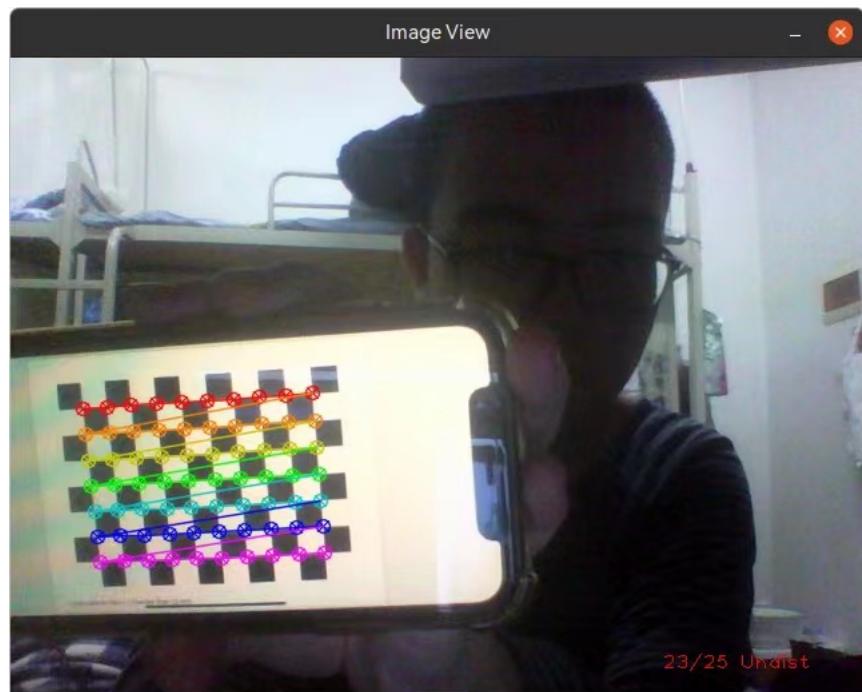


图 2：OpenCV 标定预览

```

1  <?xml version="1.0"?>
2  <opencv_storage>
3  <calibration_time>"2021年11月12日 星期五 22时42分44秒"</calibration_time>
4  <nr_of_frames>25</nr_of_frames>
5  <image_width>640</image_width>
6  <image_height>480</image_height>
7  <board_width>10</board_width>
8  <board_height>7</board_height>
9  <square_size>7.</square_size>
10 <fix_aspect_ratio>1.</fix_aspect_ratio>
11 <!-- flags: +fix_aspectRatio +fix_principal_point +zero_tangent_dist +fix_k4 +fix_k5 -->
12 <flags>6158</flags>
13 <fisheye_model>0</fisheye_model>
14 <camera_matrix type_id="opencv-matrix">
15   <rows>3</rows>
16   <cols>3</cols>
17   <dt>d</dt>
18   <data>
19     5.0973440493282772e+02 0. 3.195000000000000e+02 0.
20     5.0973440493282772e+02 2.395000000000000e+02 0. 0. 1.</data></camera_matrix>
21 <distortion_coefficients type_id="opencv-matrix">
22   <rows>5</rows>
23   <cols>1</cols>
24   <dt>d</dt>
25   <data>
26     1.6305493474984151e-01 -8.7924295536443997e-01 0. 0.
27     1.0833657276054698e+00</data></distortion_coefficients>
28 <avg_reprojection_error>9.6812517208994886e-01</avg_reprojection_error>
29 <per_view_reprojection_errors type_id="opencv-matrix">
30   <rows>25</rows>
31   <cols>1</cols>
32   <dt>f</dt>
33   <data>
34     1.83506593e-01 2.62221247e-01 4.30098504e-01 1.77741289e-01
35     1.00446701e-01 2.01398283e-01 5.19664586e-01 3.92103583e-01
36     3.02759111e-01 1.98299050e-01 2.10376397e-01 1.76716611e-01

```

图 3: 得到的 out_camera_data.xml 的文件截图

二、4步相移法结构光实验

(1) 实验内容和实验步骤

➤ 4步相移法结构光实验需要写的内容:

1) 包裹相位、绝对相位、相位差;

包裹相位: 相位提取不管用 FTP 还是 PMP, 最终都用反正切来求相位, 因此求出来的相位值都是分布在 $[-\pi, \pi]$ 或者实 $[0, 2\pi]$ 这一个区间里。因此, 提取出来的相位是被截断的, 被称为包裹相位。

绝对相位: 绝对相位 = 包裹相位 + $2k\pi$ (k 为条纹级数)。

相位差: 两个相位之间的差值。

2) 解释相位差和深度的关系, 或者解释如果运用绝对相位求解出物体上某

个点的三维坐标

可根据 CCD 的绝对相位值可以求得投影仪对应的绝对相位值为：

$$u^p = \frac{\varphi}{N \times 2\pi} \times W$$

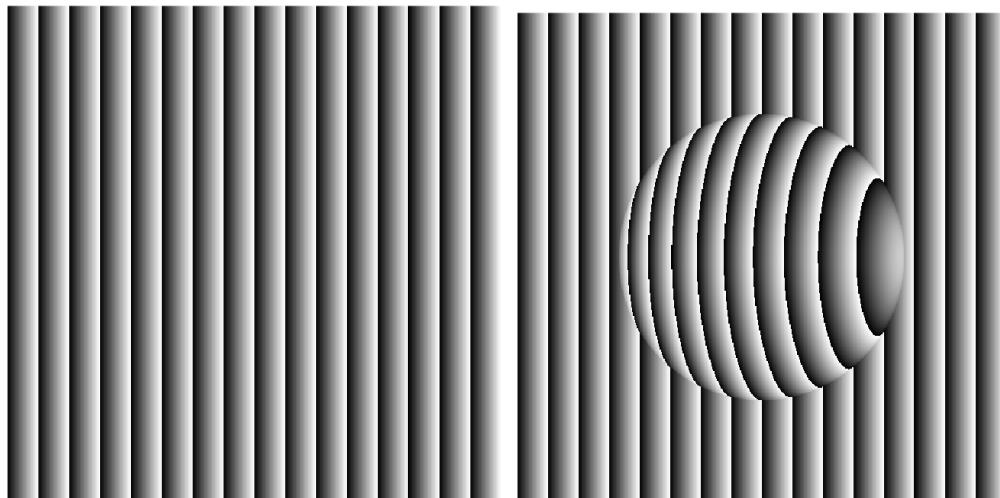
u^p 为三维点在投影仪上的绝对相位值， φ 为三维坐标点在相机中的绝对相位值， N 为光栅图像的条纹周期数， W 为投影仪在水平方向上的像素值。被测物体上 A 点，对应于投影仪坐标系中的相位为 u^p 的直线。得知三维点在投影仪上的绝对相位 u^p 后，根据图像上的像素坐标到三维坐标的转化关系，可以求得绝对相位线对应的平面。相机的像素点对应的直线和投影仪绝对相位对应的平面的交点，即为被测点的三维坐标。

3) 代码可复制在附录中

(2) 主要仪器设备和编程软件

Matlab R2020b on macOS 12.1.0

(3) 实验数据记录和结果分析



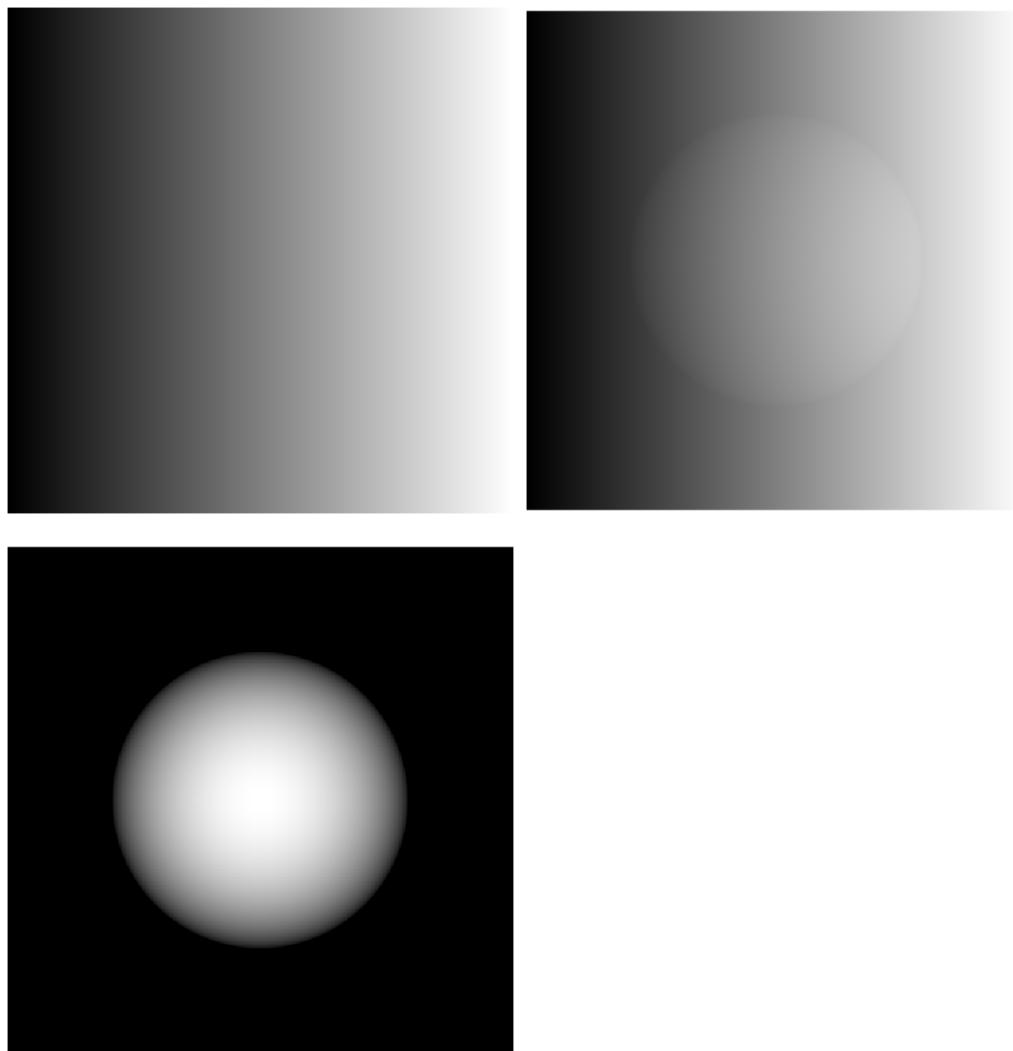


图 4：图片的变化

附录：代码

```
img1=double(imread("1.png"));
img2=double(imread("2.png"));
img3=double(imread("3.png"));
img4=double(imread("4.png"));
imga=double(imread("a.png"));
imgb=double(imread("b.png"));
imgc=double(imread("c.png"));
imgd=double(imread("d.png"));
phase1=atan2(img3-img1,img4-img2);
phase2=atan2(imgc-imga,imgd-imgb);
imshow(phase1,[ ]);
imshow(phase2,[ ]);
phase11=unwrap(phase1,[],2);
```

```
phase22=unwrap(phase2,[],2);
imshow(phase11[])
imshow(phase22[])
minus=phase22-phase11;
imshow(minus[])
```

三、PC 端 ROS 系统安装及测试

(1) 安装过程记录

系统：Windows10、Ubuntu 16.04

软件：ros-kinetic

第一步，安装Vmware虚拟机，下载Ubuntu 16.04镜像。

第二步，安装Ubuntu 16.04并打开。

第三步，打开终端，添加下载源、设置密钥、更新源。

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
curl -sSL
'http://keyserver.ubuntu.com/pks/lookup?op=get&search=0xC1CF6E31E6BADE88
68B172B4F42ED6FBAB17C654' | sudo apt-key add -
sudo apt-get update
```

第四步，安装ros-kinetic

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

第五步，初始化rosdep（需要更改源文件）

```
sudo rosdep init
```

```
rosdep update
```

第六步，环境配置

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

```
source ~/.bashrc
```

第七步，安装rosinstall

```
sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-
essential
```

第八步，启动roscore

```
roscore
```

第九步，启动小海龟

```
rosrun turtlesim turtlesim_node
```

第十步，启动小海龟控制器

```
rosrun turtlesim turtle_teleop_key
```

(2) bug 调试记录

在初始化 rosdep 时会发生错误，原因在于 roedep 初始化时会下载 GitHub 仓库中上的最新内容，而这些内容由于网络等原因无法被下载到本地。对此，我们只需要将几个初始化文件中的 GitHub 地址更改为其他可以下载的仓库。

四、机器人运动实验

(1) 机器人接线步骤

1. 检查大锂电池的电压是否达到 19V。
2. 将大锂电池的 DC OUT 端与小电脑的电源口相连。
3. 将蓝色电池的两个输出的电源线分别接到机器人的底盘的电源孔和相机引出来的电源线。
4. 将小车上的几个 USB 口都接到扩展坞上。
5. 显示屏的 mini HDMI 接电脑的 HDMI 口，控制线接电脑或者扩展坞 USB 口，取出无线键盘和无线鼠标，将接收器接到电脑或者扩展坞 USB 口上。

(2) 机器人运动步骤

1. 首先需要将底盘的总开关打开，将开关拨到右侧。
2. 底盘初始化：在底盘总开关另一侧的左手边找到一大两小的按钮，先按一下大按钮重置，再长按任意一个小按钮到底盘的轮子转动。
3. 电脑打开终端输入：

```
roscore
```

打开 ros 核心。

4. 再开一个终端，输入：

```
roslaunch turbot3_bringup minimal.launch
```

启动机器人的底盘。

5. 再开一个终端，输入：

```
roslaunch turbot3_teleop keyboard.launch
```

然后便可以使用键盘的 WASD 键操控小车的运动。

6. 再开一个终端，输入：

```
roslaunch rgbdslam freenect+rgbdslam.launch
```

配合小车的运动，便可以开始建图。

五、PC 端 rgbdslamV2 编译安装过程（重点）

(1) 安装编译过程

第一步，建立工作空间。

```
mkdir -p ~/robot_ws/src  
cd ~/catkin_ws  
catkin_make
```

第二步，下载 RGBD-SLAM

```
cd ~/robot_ws/src  
git clone https://github.com/felixendres/rgbdslam_v2
```

第三步，安装 g2o 依赖项

```
sudo apt-get install libsuitesparse-dev
```

第四步，下载并提取 eigen 3.2.10 头文件

```
cd ~  
wget http://bitbucket.org/eigen/eigen/get/3.2.10.tar.bz2
```

第五步，下载经过 RGBD-SLAMv2 作者修改的 g2o

```
cd ~/robot_ws/src  
git clone https://github.com/felixendres/g2o.git  
cd ~/robot_ws/src/g2o  
mkdir ~/robot_ws/src/g2o/build  
cd ~/robot_ws/src/g2o/build
```

第六步，配置 g2o 头文件

第七步，编译 g2o

```
cd ~/robot_ws/src/g2o/build  
cmake ../  
make  
sudo make install
```

第八步，下载 PCL

```
cd ~  
wget https://github.com/PointCloudLibrary/pcl/archive/pcl-1.8.0.tar.gz  
tar -xvzf pcl-pcl-1.8.0.tar.gz
```

第九步，用 C++ 2011 支持配置编译 PCL

```
cd ~/pcl-pcl-1.8.0  
gedit CMakeLists.txt
```

第十步，编译并安装 PCL

```
cd ~/pcl-pcl-1.8.0
```

```
mkdir build  
cd build  
cmake ..
```

第十一步，配置 RGBD-SLAM

第十二步，构建 SiftGPU 库

```
cd ~/robot_ws/src/rgbdslam_v2/external/SiftGPU  
sudo apt-get install libglew-dev  
sudo apt-get install libdevil1c2 libdevil-dev  
make
```

第十三步，建构 RGBD-SLAM

```
cd ~/robot_ws  
catkin_make
```

第十四步，添加 robot_ws 工作空间到.bashrc 文件下

```
$ echo "source ~/robot_ws/devel/setup.bash" >> ~/.bashrc$  
source ~/.bashrc
```

第十五步，下载数据集并修改配置

```
cd ~  
mkdir bag-data  
cd ~/bag-data  
wget http://vision.in.tum.de/rgbd/dataset/freiburg1_rgbd_dataset_freiburg1_xyz.bag
```

第十六步，测试 TUM 数据集

打开 roscore 与 rgbdslam.launch

然后在 TUM 数据集所在目录下运行

```
cd ~/bag-data  
rosbag play rgbd_dataset_freiburg1_xyz.bag
```

(2) bug 调试记录

在运行 rgbdslam.launch 后会报错，原因在于 g2o 库需要使用作者修改的版本，以及 PCL 需要使用 1.8 版。因此 PCL 1.8.0 需要在安装 RGBDSLAMv2 前安装，并且将 RGBDSLAMv2 的配置文件中对 PCL 1.7 的依赖更改为 PCL 1.8.0 版本，这样就不会产生冲突。

六、机器人实际场景稠密建图

简要流程概括

机器人实际场景稠密建图分为两部分，第一部分是通过 ros 与机器人建立联系并能够控制机器人的运动。第二部分是通过 RGBDSLAMv2，通过机器人上的 kinetic 摄像头对场景进行建图。

具体到第二部分需要启动 RGBDSLAMv2，然后通过控制机器人的运动，在保证拍摄画面没有同学来回穿越的情况下便可以实现对场景的建图。

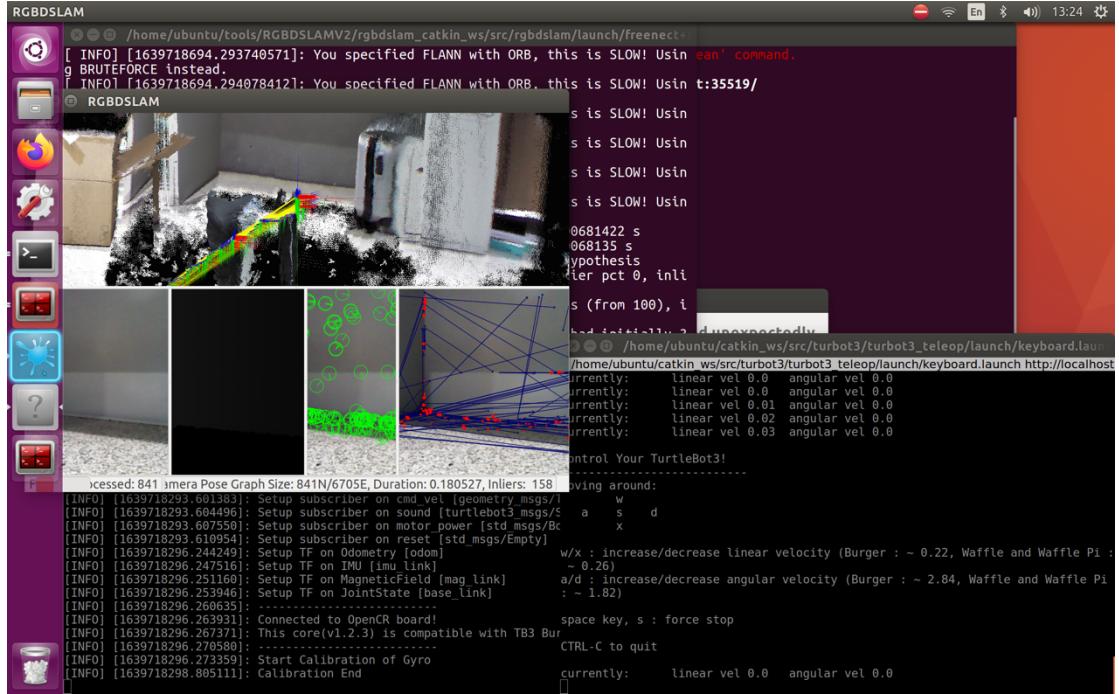


图 5：稠密建图的截图

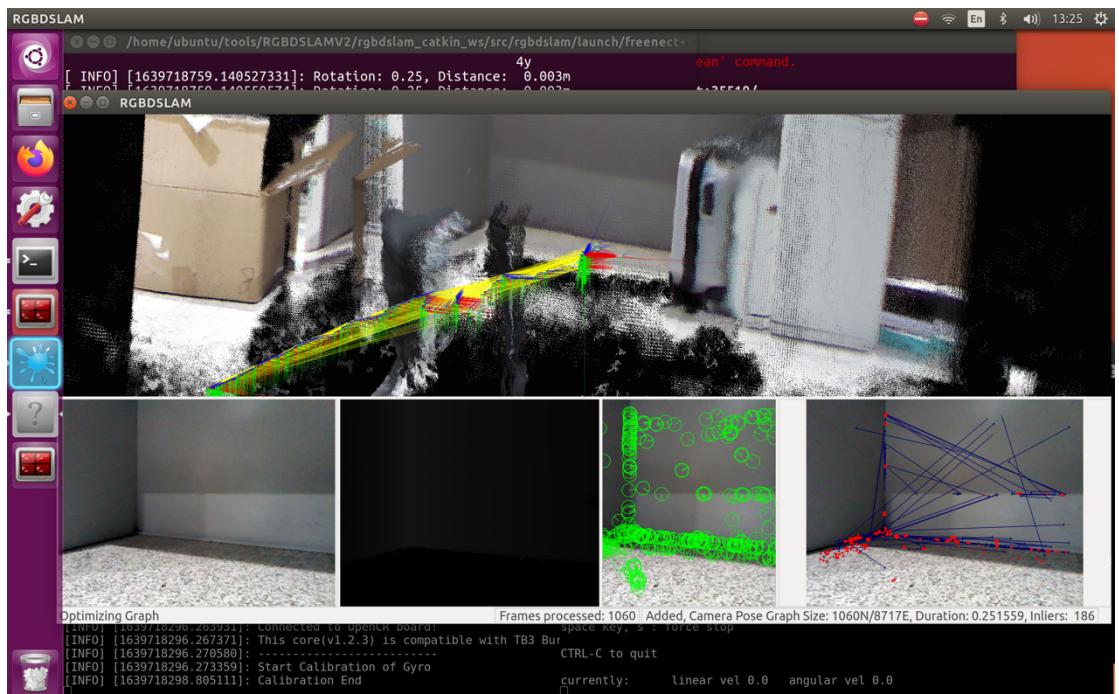


图 6：稠密建图的截图

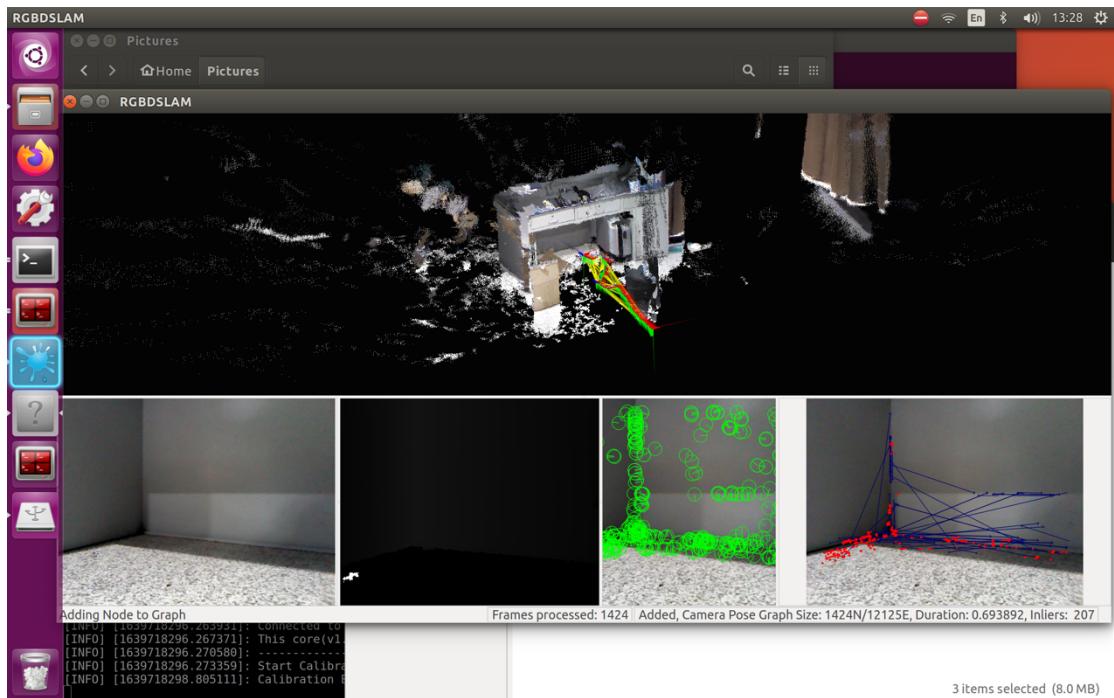


图 7：稠密建图的截图

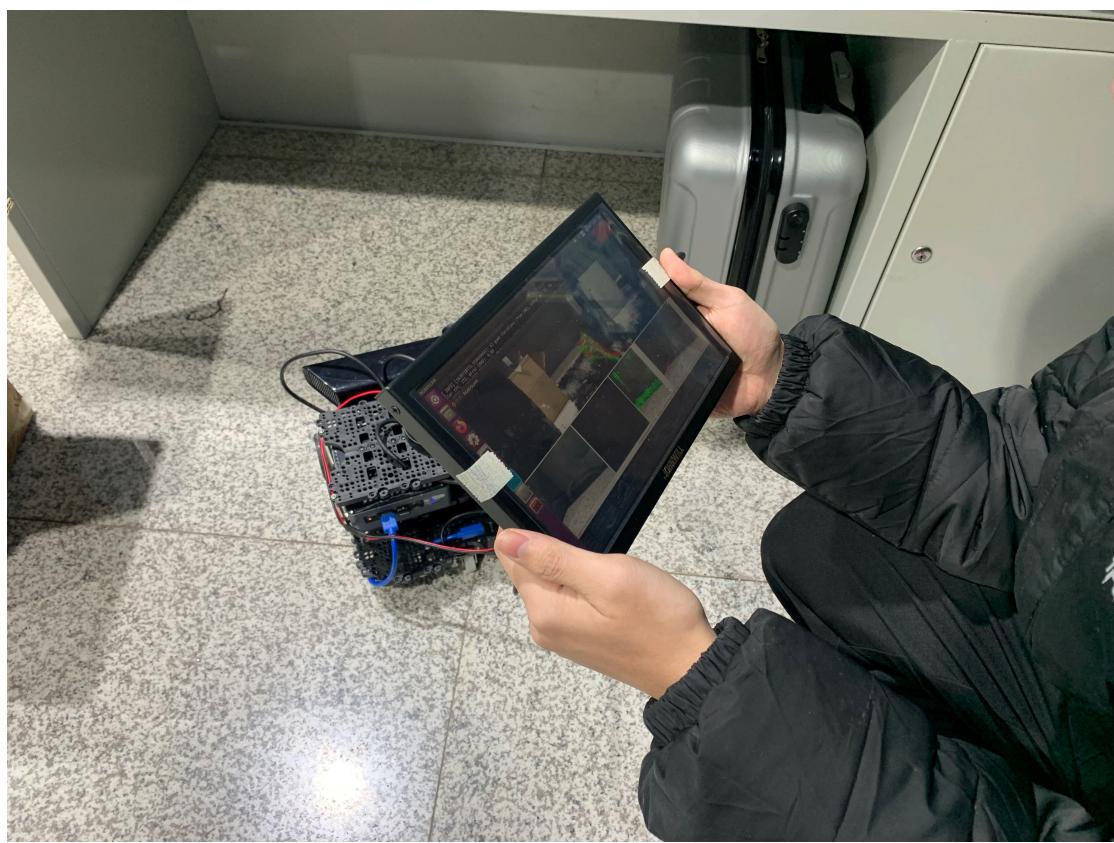


图 8：稠密建图的实际过程