## **相机标定实验指导内容**

**实验任务**

## 使用松灵小车，利用kalibr 工具标定相机的内参

1. （选做）利用imu\_utils工具实现imu的标定，并结合以上两个传感器的标定信息，再次利用kalibr实现对imu和相机的联合标定

**实验报告要求：**

按照群里的实验报告模板

将实验步骤及对应的相关截图放在报告中，并对最后的结果进行必要的分析

任务1的step3和step4尽量每个人都做，可以在报告中体现出来

**实验步骤：**

**实验1**

step1： 安装环境

同样地需要安装相关依赖：

sudo apt-get install -y \

git wget autoconf automake nano \

libeigen3-dev libboost-all-dev libsuitesparse-dev \

doxygen libopencv-dev \

libpoco-dev libtbb-dev libblas-dev liblapack-dev libv4l-dev

对于不同的Ubuntu版本，安装不同的python

# Ubuntu 16.04

sudo apt-get install -y python2.7-dev python-pip python-scipy \

python-matplotlib ipython python-wxgtk3.0 python-tk python-igraph

# Ubuntu 18.04

sudo apt-get install -y python3-dev python-pip python-scipy \

python-matplotlib ipython python-wxgtk4.0 python-tk python-igraph

# Ubuntu 20.04

sudo apt-get install -y python3-dev python3-pip python3-scipy \

python3-matplotlib ipython3 python3-wxgtk4.0 python3-tk python3-igraph

step2： 下载并编译kalibr

cd ~/kalibr\_workspace/src

git clone https://gitee.com/mario\_ly/Kalibr.git

cd ~/kalibr\_workspace

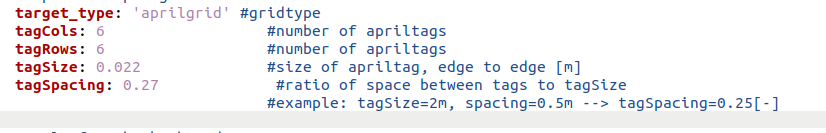
catkin build -DCMAKE\_BUILD\_TYPE=Release -j4

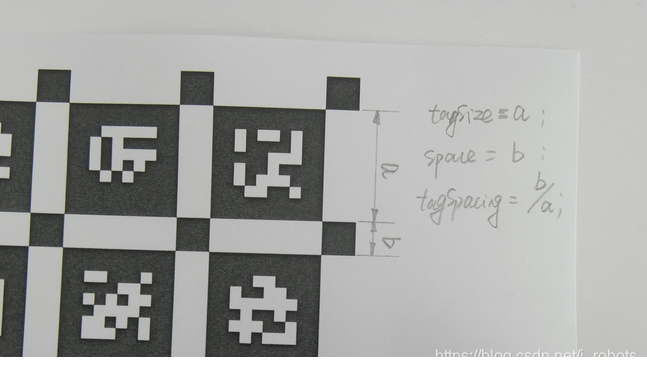
source ~/kalibr\_workspace/devel/setup.bash

注意：这里的catkin build -DCMAKE\_BUILD\_TYPE=Release -j4之前需要删除devel和build文件

step3：录制发布频率为4hz的相机话题

这里需要用上kalibr工具里官方的标定版April\_6X6（实验配每组一张），另外还需要官方里的april\_6x6.yaml配置文件，文件下载完成后里面的参数tagSize和tagSpacing需要和本次实验所使用到的标定板一致。可以参考下图计算





利用小车的相机对标定板发布小车相机相关的话题：roslaunch realsense2\_camera rs\_camera.launch

修改话题发布的频率：rosrun topic\_tools throttle messages /camera/color/image\_raw 4 /camera\_4hz/color/image\_raw

其中/camera\_4hz/color/image\_raw是修改后话题的名称，可以自主定

录制修改后话题的ros包：rosbag record /camera\_4hz/color/image\_raw -O scout\_4hz\_images.bag

step4:利用kalibr进行相机标定

cd kalibr\_workspace

source devel/setup.bash

#运行相机标定程序

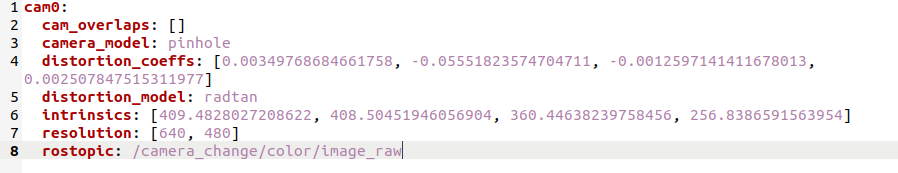
rosrun kalibr kalibr\_calibrate\_cameras --target /home/oseasy/kalibr\_workspace/src/kalibr/april\_6x6.yaml --bag /home/oseasy/rosbag\_data/scout\_4hz\_images.bag --bag-from-to 5 20 --models pinhole-radtan --topics /camera\_4hz/color/image\_raw --show-extraction

这里面的本文都写上了绝对路径，可以根据自己的实际情况修改

其中--bag-from-to 5 20意味着只取第5张到第20张的数据，避免开始和结束时的抖动

--models pinhole-radtan 是指相机模型为单孔相机

标定后会得到scout\_4hz\_images-camchain.yaml和scout\_4hz\_images-report-cam.pdf等文件，得到相机的内参，畸变参数和投影参数的误差范围，同时也会生成对应的可视化文件



以上完成了相机的标定

**实验2**

**IMU的标定**

### 这里使用的包是 [imu\_utils](https://github.com/gaowenliang/imu_utils) ，使用这个包可以校准IMU的噪声密度和随机游走噪声，这个功能包依赖ceres库，因此首先安装ceres库，如果已经安装了可以略过。

step1： 安装ceres库

首先在正式安装ceres库之前，先为其准备号相应的依赖包

Ubuntu18.04版本的所能适配依赖包：

sudo apt-get install liblapack-dev libsuitesparse-dev libcxsparse3 libgflags-dev libgoogle-glog-dev libgtest-dev

Step2：下载编译 ceres-solver

创建一个文件夹专门用于存放网上下载的包，我这边命名为“pkg”

mkdir pkg

cd pkg

git clone <https://gitee.com/mario_ly/ceres-solver.git>

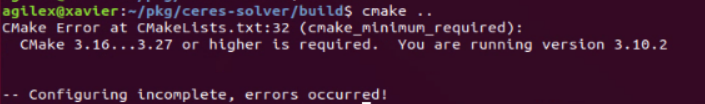
cd ceres-solver

mkdir build

cd build

cmake ..

这里可能会出现：

原因是这里的ceres库所需要的cmake版本最少为3.16（如果没有此错误可以略过）

解决方法：

更新cmake，本文更新的cmake版本为cmake-3.17.4

在官网<https://cmake.org/files/>里面找到

20231012203304

选择下载

下载完成后进行cmake更新与安装：

tar -zxvf cmake-3.17.3.tar.gz

cd cmake-3.17.3 # or wherever you downloaded cmake

./bootstrap --prefix=$HOME/cmake-install

make (这里也可以启动多线程进行编译，例如make -j4)

make install

sudo gedit ~/.bashrc

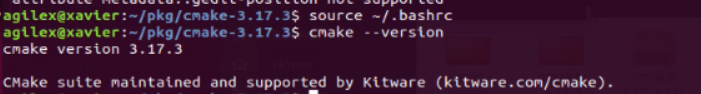
#把下面两个命令加入~/.bashrc中的最后

export PATH=$HOME/cmake-install/bin:$PATH

export CMAKE\_PREFIX\_PATH=$HOME/cmake-install:$CMAKE\_PREFIX\_PATH

#保存一下并运行

source ~/.bashrc



解决问题后重新回到ceres-solver下进行编译

cd ceres-solver

cd build

cmake ..

make -j4

sudo make install

**s**tep3：下载并编译 code\_utils并构建工作空间

首先构建工作空间

mkdir -p ~/kalibr\_workspace/src

cd ~/kalibr\_workspace

catkin\_make

下载code\_utils，并编译

sudo apt-get install libdw-dev

cd kalibr\_workspace/src

git clone https://github.com/gaowenliang/code\_utils.git

cd ..

catkin\_make

##### 一般可能会遇到错误，这个时候可以将在code\_utils下的CMakeLists.txt文件下代码修改如下：

修改set(CMAKE\_CXX\_FLAGS "-std=c++11")为set(CMAKE\_CXX\_FLAGS "-std=c++14")

如果还不能解决请参考此博客：<https://blog.csdn.net/qq_39607707/article/details/125061020>

step4： 下载编译 imu\_utils  
cd kalibr\_workspace/src

git clone https://github.com/gaowenliang/imu\_utils.git

cd ..

catkin\_make

##### 可能会遇到错误，这个时候可以将在code\_utils下的CMakeLists.txt文件下代码修改如下：

修改set(CMAKE\_CXX\_FLAGS "-std=c++11")为set(CMAKE\_CXX\_FLAGS "-std=c++14")

**step5:** imu标定

在松林小车上发布imu数据，后利用rosbag record /imu/data\_raw -O imu\_scout.bag进行录制，由于下载的代码里面对应的话题和小车的对应不上，因此需要修改，首先需要配置xsens.launch文件为如下内容：(指定IMU的topic)

这里的 <param name="imu\_topic" type="string" value= "/imu/data"/>

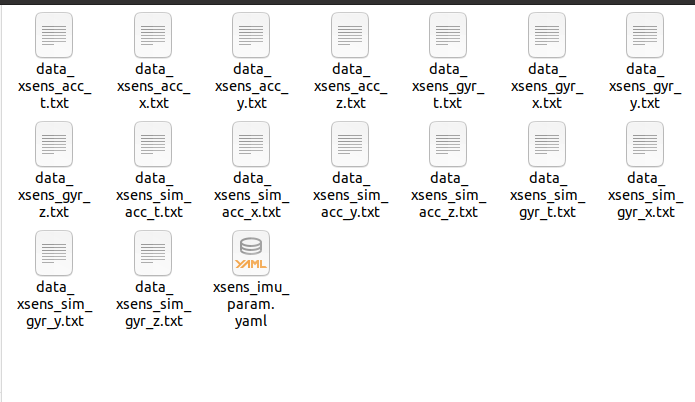
话题改为小车对应的"/imu/data\_raw"

进行imu标定：

rosbag play -r 20 imu\_scout.bag （这里的-r 20是指播放倍速20倍）这里录制了5分钟

roslaunch imu\_utils xsens.launch

可能会出现一直出不了结果的情况，多尝试几次，观察imu\_utils/data中是否有生成以下文件（原始的imu\_utils/data下有102个文件）



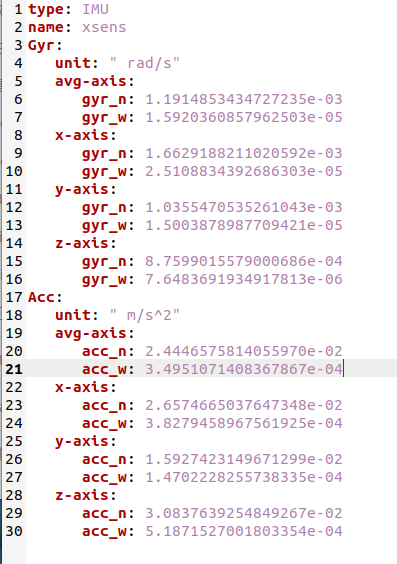
其中标定的结果如下，在xsens\_imu\_param.yaml中可以查看：

**相机和IMU联合标定**

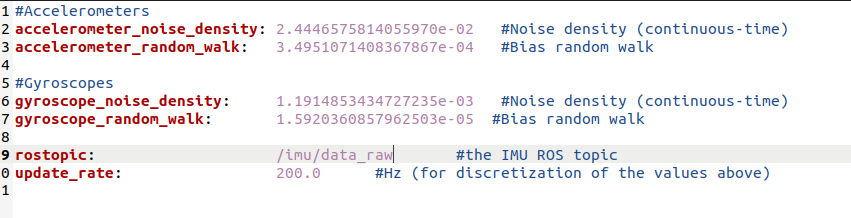
step1：同样地需要对标定版录制同时含有发布频率为4hz的图像话题以及imu话题的rosbag

这里的操作前面都有提到，就不再阐述，大家可以自己完成录制，需要注意的是录制过程中在小车视线不脱离标定版的情况下进行前后左右移动。

step2：这里需要将前面利用imu\_utils标定的imu标定数据转换为kalibr可以识别的形式



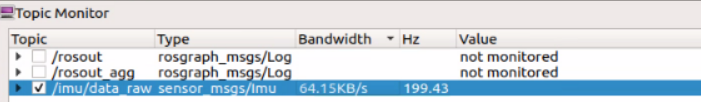
原来的xsens\_imu\_param.yaml



Kalibr可以识别的imu.yaml

主要工作是新建一个imu.yaml文件，里面涉及的参数内容就是把（avg-axis）统计平均值里面的gyr\_n->gyroscope\_noise\_density，gyr\_w->gyroscope\_random\_walk，acc\_n->accelerometer\_noise\_density，acc\_w->accelerometer\_random\_walk

添加对应的ros话题，udate\_rate对应的是imu数据发布话题的频率，不会查看发布频率的同学可以利用rosrun rqt\_topic rqt\_topic查看对应/imu/data\_raw发布的频率：



可以看到是200hz

step3:最后进行imu和相机的联合标定

cd kalibr\_workspace

source devel/setup.bash

#运行联合标定程序

rosrun kalibr kalibr\_calibrate\_imu\_camera --target /home/oseasy/kalibr\_workspace/src/kalibr/april\_6x6.yaml --bag /home/oseasy/rosbag\_data/camera\_imu.bag --bag-from-to 5 50 --cam /home/oseasy/rosbag\_data/camera\_biao-camchain.yaml --imu /home/oseasy/kalibr\_workspace/src/imu\_utils/data/imu\_data/imu.yaml --imu-models scale-misalignment --timeoffset-padding 0.1

注意，这里面本文写了自己的绝对路径，大家可以写自己的实际路径，这里选用的imu\_yaml是修改为kalibr可以识别的配置文件，

1） --target /home/oseasy/kalibr\_workspace/src/kalibr/april\_6x6.yaml 描述标定板的信息

2） --bag /home/oseasy/rosbag\_data/camera\_imu.bag 指定数据包

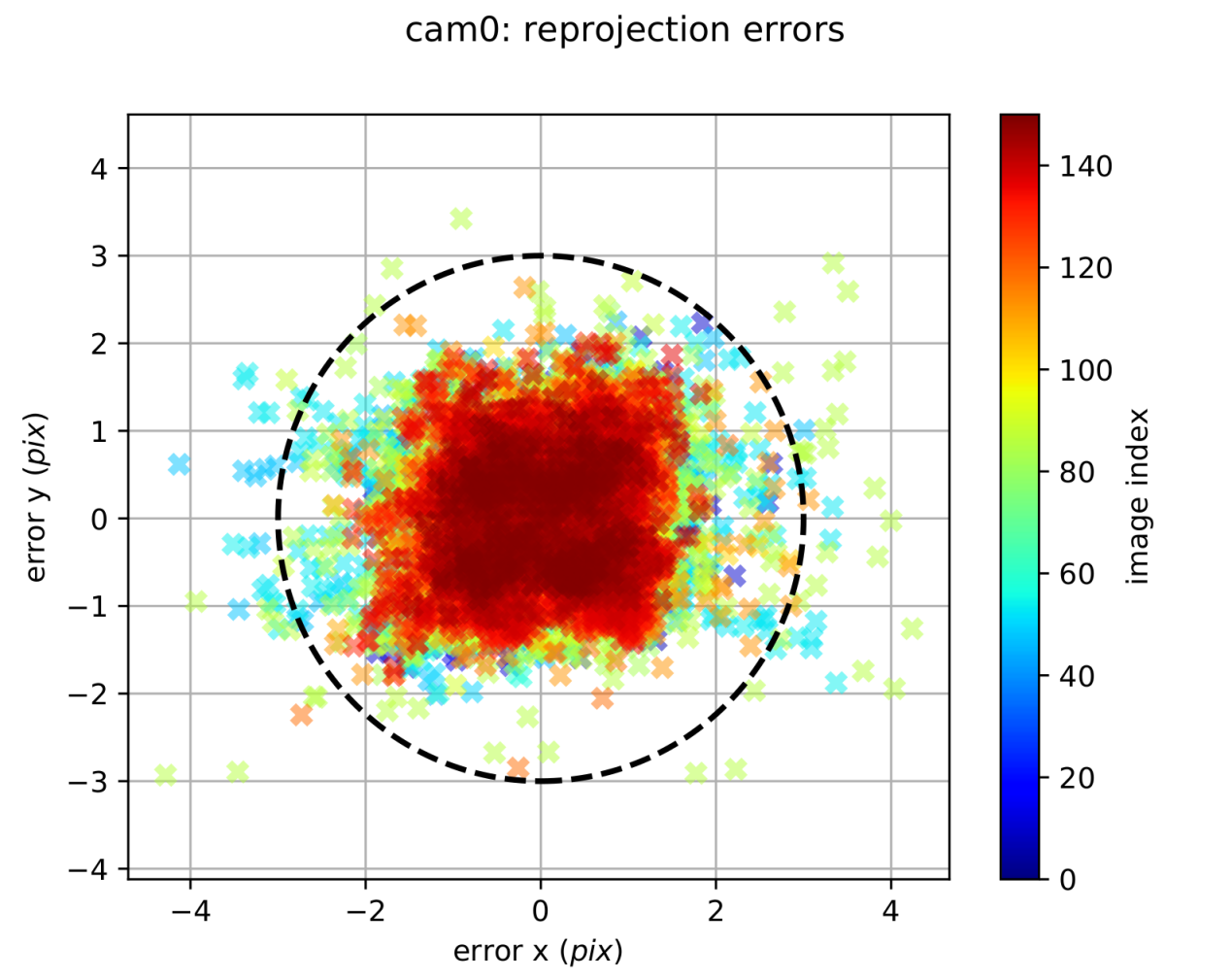
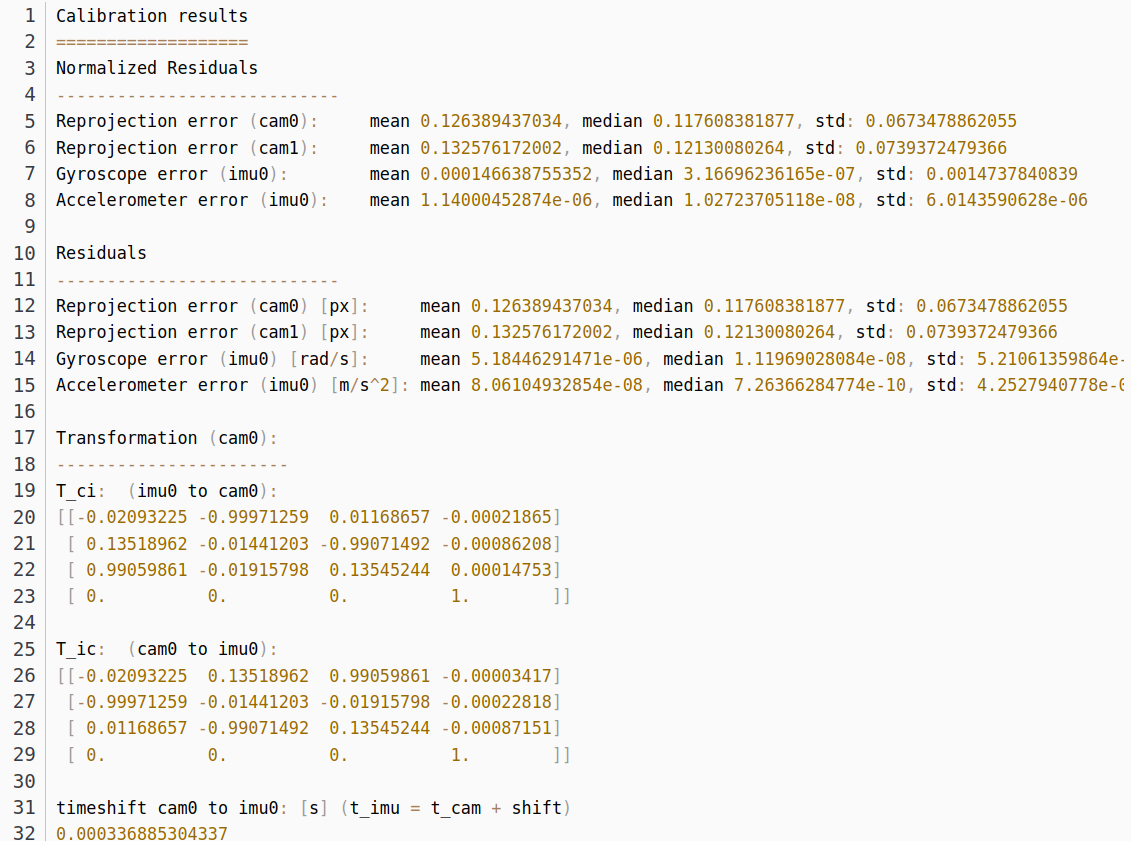
3） --bag-from-to 5 50 设定bag包开始时间和结束时间，避开拿起和放下IMU的时间段内的数据

4）--cam /home/oseasy/rosbag\_data/camera\_biao-camchain.yaml 相机参数文件

5）--imu /home/oseasy/kalibr\_workspace/src/imu\_utils/data/imu\_data/imu.yaml 设定IMU的信息

6）--imu-models scale-misalignment imu的参数模型

标定完成后会生成对应的文件，其中下图是camera\_imu-results-imucam.txt文件中的部分结果,最重要的是相机与imu之间的变换矩阵



重投影误差

如果校准的重投影误差在1个像素内就是比较好的结果了