ROS 相机+IMU联合标定教程

Author: RangerOnMars guhao0521@gmail.com

add_small_Author: Kingupup 187972645lyh@gmail.com

使用imu_utils和code_utils标定IMU白噪声与随机游走

在正式开始标定前,我们需要先进行IMU白噪声与随机游走的标定。

1. 安装依赖

```
sudo apt-get install libdw-dev
```

2.编译

```
cd ~/kalibr_ws/src
git clone git@github.com:gaowenliang/imu_utils.git
git clone git@github.com:gaowenliang/code_utils.git
cd ..
catkin_make -DCMAKE_BUILD_TYPE=Release -j16
```

1.注意事项

需要先在Workspace下对code_utils完成编译,再对imu_utils进行编译

2.常见问题:

```
error: ‘integer_sequence’ is not a member of ‘std’
```

解决方法:对imu_utils与code_utils目录下CMakeLists.txt做如下修改

```
# set(CMAKE_CXX_FLAGS "-std=c++11")
set(CMAKE_CXX_STANDARD 17)
```

```
error: imu_utils/src/imu_an.cpp:68:19: error: aggregate ‘std::ofstream
out_t’ has incomplete type and cannot be defined
```

解决方法:为该文件添加头文件

```
#include <fstream>
```

```
error: #include “forward.hpp”未找到
```

解决办法：路径错误，修改文件路径

```
error: ‘CV_LOAD_IMAGE_GRAYSCALE’ was not declared in this scope
```

此处编译的为test文件，直接注释掉即可

3.标定

1.录制imu数据rosbag

长度约20min即可,录制时将陀螺仪置于坚硬稳定平面上,避免引入震动噪声.

2.修改/新建roslaunch文件

示例roslaunch位于imu_utils/launch下

```
<launch>
    <node pkg="imu_utils" type="imu_an" name="imu_an" output="screen">
        <!--imu数据topic-->
        <param name="imu_topic" type="string" value= "/imu_data"/>
        <!--imu名称-->
        <param name="imu_name" type="string" value= "bmi088"/>
        <!--标定数据保存目录-->
        <param name="data_save_path" type="string" value= "$(find
imu_utils)/data/bmi088/"/>
        <!--数据时间跨度,越长越好,需要短于rosbag时间长度-->
        <param name="max_time_min" type="int" value= "20"/>
        <param name="max_cluster" type="int" value= "100"/>
    </node>
</launch>
```

3.运行

```
roslaunch imu_utils bmi088.launch
rosbag play 2022-03-30-18-56-08.bag -r 200
```

Ubuntu 20.04 + ROS Noetic + OpenCV4 + Python3编译Kalibr

Kalibr包历史悠久,其不少模块甚至构建于OpenCV2与Python2时代,而在Python3和OpenCV4大行其道的今天,如果不进行针对性的环境配置.直接编译Kalibr的源码会出现较多较为棘手的版本兼容问题,该问题截至笔者写这篇Tutorial(2022.3)时仍未得到很好的解决.

幸运的是,ori-drs为我们提供了一份已进行部分修改的kalibr代码,该版本已进行了一些必要的修改,可以直接在Ubuntu20.04 + ROS Noetic + OpenCV4 + Python3环境上进行编译,出现的问题远少于ethz-asl的版本.

1.安装依赖

```
sudo apt update
sudo apt-get install python3-setuptools python3-rosinstall ipython3
libeigen3-dev libboost-all-dev doxygen libopencv-dev ros-noetic-vision-
opencv ros-noetic-image-transport-plugins ros-noetic-cmake-modules python3-
software-properties software-properties-common libpoco-dev python3-
matplotlib python3-scipy python3-git python3-pip libtbb-dev libblas-dev
liblapack-dev libv4l-dev python3-catkin-tools python3-igraph
libsuitesparse-dev libgtk-3-dev
pip3 install wxPython opencv-python matplotlib pycryptodomex gnupg scipy
python-igraph pycairo
```

2.编译

```
mkdir ~/kalibr_ws/src
cd ~/kalibr_ws/src
git clone --recursive https://github.com/ori-drs/kalibr

cd ~/kalibr_ws
source /opt/ros/noetic/setup.bash
catkin_make -DCMAKE_BUILD_TYPE=Release -j16
```

3.标定

1.录制ROSBag

在标定前，首先需要进行ROSBag的录制，ROSbag内需要包含Image和IMU的Topic。标定时请注意一下几点:

> 1.标定时运动云台,沿Roll,Yaw,Pitch轴,竖直,水平,前后方向各来回移动三次。
> 2.移动时需轻柔缓慢,尽可能避免相机拖影。
> 3.Image的录制非常占用空间，录制前请为你的电脑留出10-20G左右的空间。

1.5.修改kalibr源码使其支持CompressedImage（可选）

如果Image嫌大可以尝试换成CompressedImage，kalibr源码不支持使用CompressedImage,需要自己动手改一下源码,在ImageDatasetReader.py的getImage函数中按如下方式修改即可。但需要注意的是压缩耗时可能较

长，如果不采用异步或多线程的方法进行优化会导致录制的帧率不高。

```
  def getImage(self, idx):
    topic, data, stamp = self.bag._read_message(self.index[idx].position)
    if self.perform_synchronization:
      timestamp = acv.Time(self.timestamp_corrector.getLocalTime(
          data.header.stamp.to_sec()))
    else:
      timestamp = acv.Time(data.header.stamp.secs,
                           data.header.stamp.nsecs)
    if data._type == 'mv_cameras/ImageSnappyMsg':
      if self.uncompress is None:
        from snappy import uncompress
        self.uncompress = uncompress
      img_data = np.reshape(self.uncompress(np.fromstring(
          data.data, dtype='uint8')), (data.height, data.width), order="C")
    #以下为添加部分
    elif data._type == 'sensor_msgs/CompressedImage':
        np_arr = np.fromstring(data.data, np.uint8)
        img_data = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
        img_data = cv2.cvtColor(img_data, cv2.COLOR_BGR2GRAY)
```

2.准备标定YAML文件

1.标定目标 可以参照kalibr官方文档，以棋盘格标定板为例:

```
target_type: 'checkerboard' #标定板类型
targetCols: 6               #标定板横向格数
targetRows: 7               #标定板纵向格数
rowSpacingMeters: 0.06      #方格宽度
colSpacingMeters: 0.06      #方格高度
```

2.IMU
没什么好说的，把上面用IMU_Utils标定出来avg-axis的参数替换掉下面对应的数，换成自己的topic就行。

```
#Accelerometers
accelerometer_noise_density: 3.7998244145368565e-02   #Noise density
(continuous-time)
accelerometer_random_walk:   5.4525307354616904e-04   #Bias random walk

#Gyroscopes
gyroscope_noise_density:     4.0517270774554264e-03   #Noise density
(continuous-time)
gyroscope_random_walk:       3.8019931032411956e-05   #Bias random walk

rostopic:                    /imu_data      #the IMU ROS topic
update_rate:                 200.0      #Hz (for discretization of the
values above)
```

3.相机参数
也没什么好说的，具体参照官方文档就可以了。

```
cam0:
  cam_overlaps: []
  camera_model: pinhole
  distortion_coeffs:
  - -0.059247
  - 0.042376
  - 0.001615
  - 0.000831
  distortion_model: radtan
  intrinsics:
  - 1673.885947
  - 1678.230815
  - 649.000553
  - 483.866587
  resolution:
  - 1280
  - 1024
  rostopic: /img_data
```

3.标定

把下面的参数换成自己对应的文件名，直接标就完事了。

```
rosrun kalibr kalibr_calibrate_imu_camera --bag BAGFILE --target
TARGET_YAML --imu IMU_YAML --cams CHAIN_YAML
```

以下有几个常用的参数也可以进行添加。

```
--max-iter MAX_ITER 优化的最大迭代次数，设大点一般来说可以让结果更精确一些，但会增加标
定耗时。
--bag-from-to bag_from_to bag_from_to 只选择bag文件中的这段时间内的数据进行标定，
可以用来去掉录制开始和录制结束时的数据。
```

标定过程

1.对相机内参，以及畸变矩阵进行标定

1.打开大恒相机，注意话题名

```
source devel/setup.bash
rosrun daheng_mer139 camera.py
```

2.使用ros1相机标定功能包，进行相机标定，标定相机内参和畸变矩阵

```
source devel/setup.bash
rosrun camera_calibration cameracalibrator.py --size 7x10 --square 0.02
image:=/img_data camera:=/head_camera --no-service-check
```

参数含义

--size 表示棋盘格内角点数 --square 棋盘格边长 image:=/img_data 表示话题名，一定注意 camera：=/head_camera 表示相机名称，可以不用管 --no-service-check 需要加，不加的话有可能打不开标定窗口

移动标定板：

为了达到良好的标定效果，需要在摄像机周围移动标定板，并完成以下基本需求：
（1）移动标定板到画面的最左、右，最上、下方。
（2）移动标定板到视野的最近和最远处。
（3）移动标定板使其充满整个画面。
（4）保持标定板倾斜状态并使其移动到画面的最左、右，最上、下方 。
　当标定板移动到画面的最左、右方时，此时，窗口的x会达到最小或满值。同理，y指示标定板的在画面的上下位置，size表示标定板在视野中的距离，也可以理解为标定板离摄像头的远近。skew为标定板在视野中的倾斜位置。每次移动之后，请保持标定板不动直到窗口出现高亮提示。
直到条形变为绿色。当calibration按钮亮起时，代表已经有足够的数据进行摄像头的标定，此时请按下calibration并等待一分钟左右，标定界面会变成灰色，无法进行操作，属于正常情况。

3.打开陀螺仪接受数据程序

```
source devel/setup.bash
rosrun c_board_imu imu.py 注意数据位
```

4.进行陀螺仪数据录制

```
source devel/setup.bash
rosbag record /imu
/imu为话题名，一定要注意，后面可接录制包的名称，若不接则自动以日期时间命名
至少20min，40min打底，1h较好
```

5.查看录制包的信息

```
source devel/setup.bash
rosbag info 包名
```

注意查看包的话题名称，以及时间等信息

6.进行imu的随机噪声标定

- collect the data while the IMU is Stationary, with a two hours duration;
- (or) play rosbag dataset;

```
 rosbag play -r 200 包名
```

- roslaunch the rosnode;

```
roslaunch imu_utils 对应的launch文件 注意话题名称
```

7.同时录制imu和相机数据包

```
source devel/setup.bash
rosrun c_board_imu imu.py
```

```
source devel/setup.bash
rosrun daheng_mer139 camera.py
```

```
source devel/setup.bash
rosbag record /imu_data /img_data  会占用较大内存40GB左右注意空间
```

8.使用kalibr功能包进行标定

```
source devel/setup.bash
rosrun kalibr kalibr_calibrate_imu_camera --bag BAGFILE --target
TARGET_YAML --imu IMU_YAML --cams CHAIN_YAML
```