

# 使用 imu\_calib 实现轻舟机器人 IMU 自动校准

AI 航团队

**imu\_calib** 包含一个 ROS 包，包含用于计算和应用校准参数到 IMU 测量的工具。

## 用法

程序包包含两个节点。**do\_calib** 和 **apply\_calib** 第一种计算加速度计校准参数并将其保存到 YAML 文件中，并且只需要运行一次。运行此节点以生成 YAML 校准文件后，第二个节点将使用该文件将校准应用于未校准的 IMU 主题以生成已校准的 IMU 主题

## 一、do\_calib

计算加速度计校准参数。由于它需要键盘输入，因此应直接在终端中使用 **roslaunch** 运行，而不是从启动文件运行。收到第一条 IMU 消息后，节点将提示您将 IMU 保持在特定方向，然后按 **Enter** 键以记录测量结果。完成所有 6 个方向后，节点将计算校准参数并将其写入指定的 YAML 文件。

底层算法是基于和类似于 STMicroelectronics 应用笔记 [AN4508](#) 中所述的最小二乘法校准方法。由于算法的性质，要获得良好的校准，需要将 IMU 沿其每个轴进行相当准确的定位。

### 主题

#### 订阅的主题

- **imu** (**sensor\_msgs / Imu**)  
未经校准的原始 IMU 测量

### 参数

- **~calib\_file** (字符串，默认值: "imu\_calib.yaml")  
将向其中写入校准参数的文件
- **~measurements** (整数，默认值: 500)  
每个方向要收集的测量数量
- **~reference\_acceleration** (双精度，默认值: 9.80665)  
由于重力而产生的预期加速度

## 二、apply\_calib

应用由 **do\_calib** 节点计算的加速度计校准参数。也可以选择（默认情况下启用）在启动时计算陀螺仪偏差并将其减去。

### 主题

#### 订阅的主题

- **raw** (**sensor\_msgs / Imu**)  
未经校准的原始 IMU 测量

### 发表的话题

- **corrected** (**sensor\_msgs / Imu**)  
校正后的 IMU 测量值

### 参数

- **~calib\_file** (字符串，默认值: "imu\_calib.yaml")  
从中读取校准参数的文件

- ~calibrate\_gyros (bool, 默认值: true)  
是否在启动时计算陀螺仪偏差, 然后将其减去
- ~gyro\_calib\_samples (整数, 默认值: 100)  
用于计算陀螺仪偏差的测量数量

### 三、轻舟机器人示例代码解读

参考 qinzhou\_ws 代码包, 轻舟机器人在 qinzhou\_bringup 节点中接收 stm32 驱动板发来的传感器数据, 并将解算后的 imu 数据发布到/raw 话题。

```
memcpy(&tempaccelX, str+12, 2);
memcpy(&tempaccelY, str+14, 2);
memcpy(&tempaccelZ, str+16, 2);

memcpy(&tempgyroX, str+18, 2);
memcpy(&tempgyroY, str+20, 2);
memcpy(&tempgyroZ, str+22, 2);

memcpy(&tempmagX, str+24, 2);
memcpy(&tempmagY, str+26, 2);
memcpy(&tempmagZ, str+28, 2);

accelX = (float)tempaccelX/2048*9.8;
accelY = (float)tempaccelY/2048*9.8;
accelZ = (float)tempaccelZ/2048*9.8;

gyroX = (float)tempgyroX/16.4/57.3;
gyroY = (float)tempgyroY/16.4/57.3;
gyroZ = (float)tempgyroZ/16.4/57.3;

magX = (float)tempmagX*0.14;
magY = (float)tempmagY*0.14;
magZ = (float)tempmagZ*0.14;

//发布imu函数
void actuator::pub_9250() {
    sensor_msgs::Imu imuMsg;
    sensor_msgs::MagneticField magMsg;

    ros::Time current_time = ros::Time::now();

    imuMsg.header.stamp = current_time;
    imuMsg.header.frame_id = "imu_link";
    imuMsg.angular_velocity.x = gyroX;
    imuMsg.angular_velocity.y = gyroY;
    imuMsg.angular_velocity.z = gyroZ;
    imuMsg.angular_velocity_covariance = {
        0.04, 0.0, 0.0,
        0.0, 0.04, 0.0,
        0.0, 0.0, 0.04
    };

    imuMsg.linear_acceleration.x = accelX;
    imuMsg.linear_acceleration.y = accelY;
    imuMsg.linear_acceleration.z = accelZ;
    imuMsg.linear_acceleration_covariance = {
        0.04, 0.0, 0.0,
        0.0, 0.04, 0.0,
        0.0, 0.0, 0.04
    };
    pub_imu.publish(imuMsg);
}
```

### 3.1 计算加速度计校准参数（同教程 308 imu 标定部分）

打开终端，启动 qingzhou\_bringup.launch 文件：

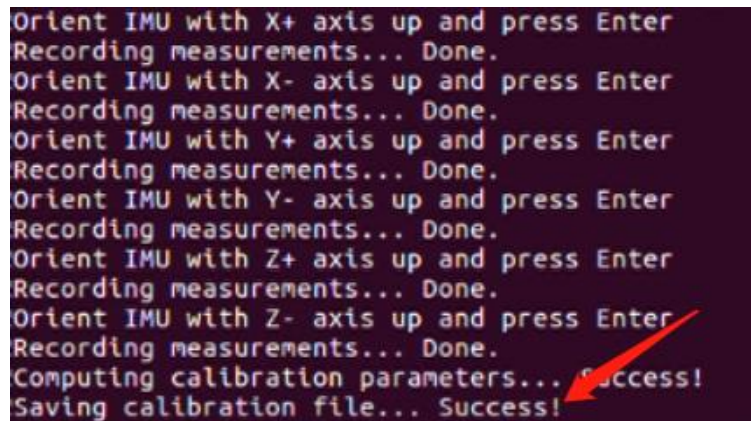
```
roslaunch qingzhou_nav qingzhou_bringup.launch
```

新开另一个终端进入到指定目录：

```
cd qingzhou_ws/src/qingzhou_odom/imu_calibrate/launch
```

运行标定文件 `roslaunch imu_calib do_calib`

接下来会自动完成标定，在弹出需要按 Enter 时按下 Enter，共需要 6 次，直到显示 success，完成标定。



```

Orient IMU with X+ axis up and press Enter
Recording measurements... Done.
Orient IMU with X- axis up and press Enter
Recording measurements... Done.
Orient IMU with Y+ axis up and press Enter
Recording measurements... Done.
Orient IMU with Y- axis up and press Enter
Recording measurements... Done.
Orient IMU with Z+ axis up and press Enter
Recording measurements... Done.
Orient IMU with Z- axis up and press Enter
Recording measurements... Done.
Computing calibration parameters... Success!
Saving calibration file... Success!

```

标定参数被保存到 qingzhou\_ws/src/qingzhou\_odom/imu\_calibrate/launch 下的 imu\_calib.yaml 文件下

```

SM:
- 0.5663376516754989
- -2.387897501838048
- -0.9890343888104273
- -0.6336476736116295
- 6.824109414938346
- -0.2247785356092069
- 0.3454817262669186
- -5.410153975877225
- 0.6087232475082894
bias:
- -10.59398169678908
- 0.3083624974181918
- 3.984632681078423

```

### 3.2 启动 apply\_calib 完成参数校准

参考 qingzhou\_ws/src/qingzhou\_nav/launch 目录下的 qingzhou\_bringup.launch 文件。

```

<node pkg="imu_calib" type="apply_calib" name="apply_calib" output="screen" respawn="false">
  <param name="calib_file" value="$(find imu_calib)/../launch/imu_calib.yaml" />
  <param name="calibrate_gyros" value="true" />
  <remap from="corrected" to="imu/data_raw" />
</node>

```

由上图看出，我们启动了 apply\_calib 节点，该节点将订阅我们在 qingzhou\_bringup 节点发布的 /raw 话题，进行校准后输出到 /corrected 话题中，此处我们将 corrected 话题重映射为 /imu/data\_raw 话题，这也是为我们下节中的滤波做准备，下节将订阅重映射后的话题。

校准前后话题显示类似下图, 其中 `orientation` 的数据由于我们没有发布, 所以数据为零, 将会在下节滤波的时候计算得到。

```
learningx@learningx: ~  
learningx@learningx:~$ rostopic echo /raw  
header:  
  seq: 3543  
  stamp:  
    secs: 1613622296  
    nsecs: 8025  
  frame_id: "imu_link"  
orientation:  
  x: 0.0  
  y: 0.0  
  z: 0.0  
  w: 0.0  
orientation_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]  
angular_velocity:  
  x: -0.00638488060273  
  y: 0.0095773209041  
  z: -0.0180904950411  
angular_velocity_covariance: [0.04, 0.0, 0.0, 0.0, 0.04, 0.0, 0.0, 0.0, 0.04]  
linear_acceleration:  
  x: 0.555078125  
  y: 0.55986328125  
  z: 9.95791015625  
linear_acceleration_covariance: [0.04, 0.0, 0.0, 0.0, 0.04, 0.0, 0.0, 0.0, 0.04]  
---  
  
learningx@learningx: ~  
learningx@learningx:~$ rostopic echo /raw/data_raw  
header:  
  seq: 3451  
  stamp:  
    secs: 1613622296  
    nsecs: 840082660  
  frame_id: "imu_link"  
orientation:  
  x: 0.0  
  y: 0.0  
  z: 0.0  
  w: 0.0  
orientation_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]  
angular_velocity:  
  x: -0.000106414676712  
  y: -0.000989656493424  
  z: -0.000404375771506  
angular_velocity_covariance: [0.04, 0.0, 0.0, 0.0, 0.04, 0.0, 0.0, 0.0, 0.04]  
linear_acceleration:  
  x: 1.68138977044  
  y: 0.162697769122  
  z: 4.56583191546  
linear_acceleration_covariance: [0.04, 0.0, 0.0, 0.0, 0.04, 0.0, 0.0, 0.0, 0.04]  
---
```

1. 同学们在使用过程中, 如果发现内容有疏漏或者不严谨的地方, 请与我们联系, 将会有轻舟积分送上! QQ: 270220858
2. 内容如有雷同, 侵权!

2021 年 2 月