# 手眼标定&&IMU-kinfu融合结果测试报告

## Changelog

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 变更人 | 变更说明 | 变更时间 |
| V1.0 | 张琛 | 初稿 | 2016/01/11 |

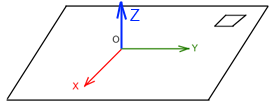
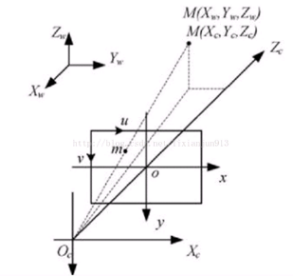
## 引言

### 编写目的

本测试报告尝试：

* + - 1. 利用CAPG-IMU硬件设备的原始输出数据（加速度，磁通，陀螺仪，共9轴数据），进行**IMU姿态估计**；
      2. 使用opencv中calibrateCamera算法，求解Kinect红外相机在某一时刻，机身相对于视野中棋盘格定标板的**姿态矩阵（红外相机内外定标）**；
      3. 将此IMU设备与Kinect硬件进行捆绑固定，求解IMU设备坐标系与Kinect红外相机坐标系之间的旋转关系，即，**手眼标定**；
      4. 分别使用手眼标定输出的旋转矩阵，以及观察估计得到的两坐标系旋转矩阵，将IMU姿态转换到相机坐标系下，与kinfu中ICP算法估计的**姿态融合**，并测试融合生成的模型精度。

### 问题描述

a) b) c)

图 1 a)为IMU输出姿态时所用的参考系；b)为Kinect相机坐标系；c)为capg-imu与kinect固定在一起的实物图

如上文所述，本测试主要进行4方面的测试。

a) 对于**IMU姿态估计**，问题描述如下：

**输入：**

acc\*3, mag\*3, gyro\*3, 共9轴数据，capg-IMU硬件输出无单位的值，e.g., acc合加速度值可能为 1.2, 而非 ≈9.8m/s2。所以需要预处理normalize过程。

**输出：**

每一时刻，IMU九轴数据计算得到一个机身到世界坐标系（地球ENU）的旋转矩阵。

b) 对于**红外相机内、外定标**，问题描述如下：

**输入：**

使用Kinect红外相机采集的，一组静止状态下，场景中含有棋盘格定标板的灰度图像。实际数据为保留最完整信息，采取视频拍摄的方式，保存为oni文件（OpenNI格式）；然后从视频中手动选取画面稳定时刻的关键帧，作为外定标输入。

**输出：**

* 相机内参矩阵
* 畸变系数
* 每一帧关键帧图像对应的相机姿态外参
* 使用此内参进行棋盘格外定标，得到的反投影误差RMS

c) 对于**手眼标定**算法，问题描述如下：

相机与IMU硬件的坐标系定义如图1所示。

**输入：**

。其中，，是世界（标定板）坐标系到相机坐标系变换矩阵序列， 是IMU模块坐标系到世界（地球ENU）坐标系变换矩阵序列。

**输出：**

从IMU模块坐标系到相机坐标系的旋转矩阵 ，使得，或者说。另外，将目前设备固定方式下，观察估计得到的二者相对旋转矩阵 与Y进行对比，以验证标定流程的正确性。

(1)

d) 姿态融合问题描述如下：

**输入：**

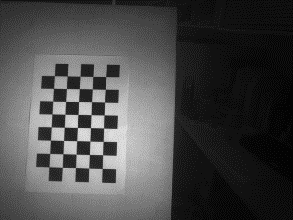
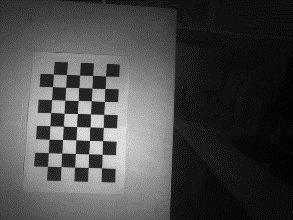
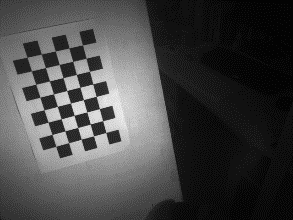
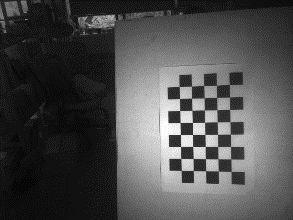
* 一组深度图序列，
* 以及每一帧深度图对应时刻的IMU姿态，
* 矩阵。

**输出：**

* 原始的kinfu结果；
* 使用矩阵作为融合迭代初值，其中 如公式(1)所示；
* 使用实际手眼标定输出Y，求，进而计算得到的kinfu融合结果。

### 测试数据

IMU、红外相机各自外定标，以及二者的手眼标定，需要同时采集IR灰度图（需要卤素灯补光降噪）与IMU数据。

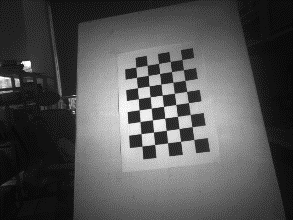
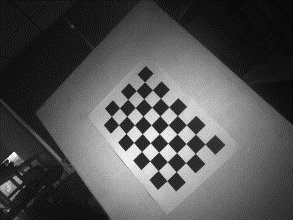
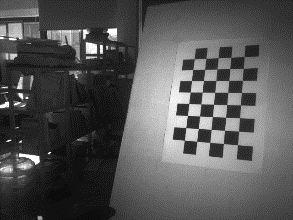
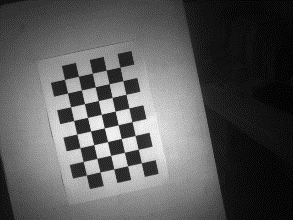
   

图 2 使用kinect采集的红外图像，遮住kinect红外发射器以避免高亮光斑干扰图像。图象分辨率：640\*480

采集数据步骤如下：

1. 必须首先开启“zsens”IMU数据采集程序，正确加载（自动或手动）节点的calibrate配置文件，并开始采集；然后打开NiViewer进行IR视频数据录制。手持kinect设备，保持标定板全部位于相机视野中，移动使视角不断变换。在每个新视角下，握持设备静止一小段时间，然后移动到下一个视角。
2. 用NiViewer打开已经录制的oni视频文件，设定主数据流为IR流。挑选每个视角下，设备稳定的时间段内，棋盘格角点较为清晰的帧，保存单帧。同时将帧序号，及其对应时间戳保存到 \*.frames2calib.txt

目前整个预处理流程中，我使用的具体数据预处理工具集见笔记：<http://www.evernote.com/l/AY-J8TZ8Bx9IB4wxTWdLWUDU5zLgyDjEzdw/>

### 测试环境

* 操作系统：Windows10
* 编程语言：C++, Python, Matlab,
* 开发环境：VS2010, ipython-notebook, matlab R2015a
* 第三方库与工具: opencv, openni, python{numpy, lxml}

## 测试过程与结果

### IMU姿态估计

* 使用@杜宇移植修改后的madgwick 代码实现。其主要工作如下：

1. 预处理：IMU raw-data 简单**归一化**到(-1, +1)，e.g., 由 (-1.2, +2.5) 缩放到 (-1, +1)。预处理输出包括：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| minAx | minAy | minAz | minMx | minMy | minMz | zeroGx | zeroGy | zeroGz |
| maxAx | maxAy | maxAz | maxMx | maxMy | maxMz |  |  |  |

2. 将madgwick C代码移植为C++，隐藏包装全局变量；

3. 在真实时序输入之前，使用第0帧数据，迭代1000次，以期望算法内部参数较早收敛；

* 为测试算法调用、设备工作的正确性，采集数据如下：

1. 设备**始终静止**，采集一个时间段内数据，观察数据首尾两帧之间的旋转量（角度），判断标归一化之后是否仍有姿态漂移；

2. 设备初始状态为一边紧贴某固定物体，绕X轴旋转+360°，回到一边紧贴固定物体的**原姿态**，然后继续分别绕Y轴，Z轴做同样的操作，观察数据首尾两帧之间的旋转量。

首尾帧矩阵A[0], A[-1]之间的旋转量（角度）计算方法伪码如下：

|  |
| --- |
| deltaRmat = A[-1]/A[0];  angle, axis = Rodrigues(deltaRmat);  //angle即所求 |

表 1 设备静止首尾帧旋转量

|  |  |  |  |
| --- | --- | --- | --- |
| 测试序号 | 数据文件 | A[0], A[-1]间旋转量（角度） | 备注描述 |
| 1 | CaptureSession.2427.ynorth.static.xml | 2.436° | phyId=…2427; Acc-Y轴大致朝北; 设备始终保持静止 |
| 2 | CaptureSession.2427.ynorth.static.2.xml | 21.2° | 如上，但在设备启动之后，开始采集之前大幅转动几下 |
| 3 | CaptureSession.2427.ynorth.static.3.xml | 0.9° | 如上，开始采集之前大幅转动，但静止之后，做 reset session 处理 |

表 2 依次绕XYZ三轴旋转+360°首尾帧旋转量

|  |  |  |  |
| --- | --- | --- | --- |
| 测试序号 | 数据文件 | A[0], A[-1]间旋转量（角度） | 备注描述 |
| 1 | CaptureSession.2427.ynorth.xyz360.xml | 13.4° | 原始数据 |
| 2 | CaptureSession.2427.ynorth.xyz360.p.xml | 0.9° | 对姿态后处理数据 |

* 小结：

1. 原始数据要归一化

2. 采集之前需要reset session

3. 要使用后处理后的姿态输出

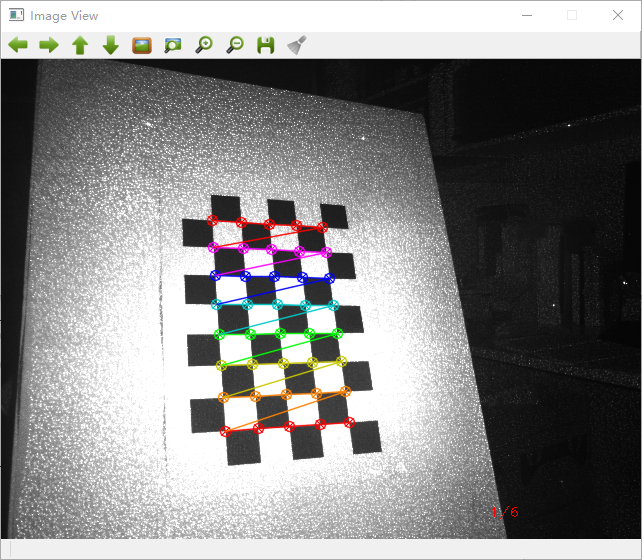
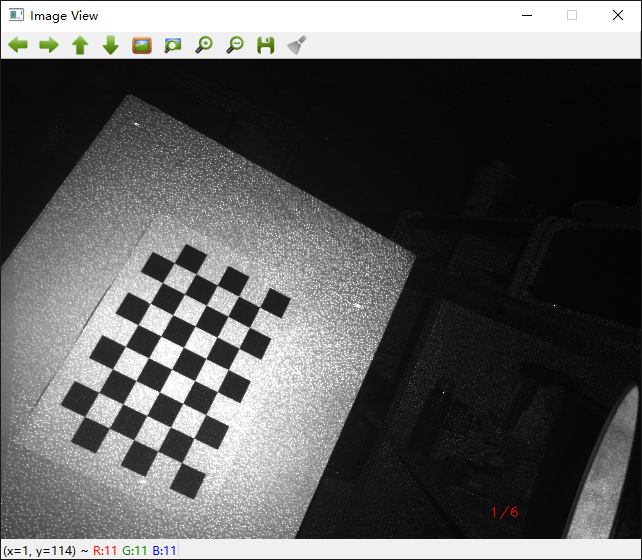
### 红外相机内定标

使用opencv棋盘格标定算法，求解相机内参，并求出每一帧标定图像的外参，进而计算反投影角点的像素坐标与自动找到的角点坐标间的RMS像素误差。

表 3 相机内定标结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试序号 | 数据集 | 图像帧数 | RMS-err | 内参矩阵 | 畸变系数 | 备注描述 |
| 1 | CapturedFrames.ir-cb-imu-precalib-20160103 | 14 | 1.97e-001 | 5.8e+002 0. 3.2e+002  0. 5. 8e+002 2.5e+002  0. 0. 1. | -1.9e-001  9.1e-001  6.8e-003  3.1e-003  -1.6e+000 | 有角点帧数13。 棋盘格角点个数：5\*8 |
| 2 | CapturedFrames.ir-cb-imu-precalib-20160111 | 6 | 1.55e-001 | 6.5e+002 0. 3. 7e+002  0. 6.5e+002 2.0e+002  0. 0. 1. | -2.0e-002  -1.4e+000  6.0e-003  4.2e-003  6.2e+000 | **有角点帧数3**（1,4,6）；  但是RMS居然小于上面序号1结果 |
| 3 | CapturedFrames.ir-cb-imu-precalib-20160111/t | 1 | 1.56e-001 | 6.7e+002 0. 4.0e+002  0. 6.0e+002 2.0e+002  0. 0. 1. | -2.9e-002  -1.5e+000  6.4e-003  -4.8e-003  5.4e+000 | 取序号2中仅第一帧，有角点 |

测试序号2中，找不到角点的图像帧可能因为Kinect红外发射器干扰噪声较大，下图是其数据集中有角点、找不到角点的图像帧示例。

测试序号2数据中，a) 第1帧 b) 第3帧

* 小结：

发现并非标定图像帧数、拍摄视角越丰富，误差就一定越小。如，表3中，序号2只有三帧有效（有角点）帧，但是RMS误差略小于序号1。

### 手眼标定

详细测试结果见《手眼标定算法(AX=XB)测试报告》[1]，截取其结果列表（表6）如下：

表 4 实际数据（kinect-IR & capg-imu）上的手眼标定结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试序号 | 相机、IMU姿态数据文件(预处理后) | 标定结果 | 指标 (单位: 角度°) | 若不标定，直接用 矩阵，得到 |
| 1 | ir-cb-imu-precalib.oni.frames2calib.cvAndImuExtr.csv | -0.2790 0.8740 0.3978  -0.1239 0.3781 -0.9175  -0.9523 -0.3052 0.0028 | 34.7231° | 36.4712° |
| 2 | ir-cb-imu-precalib-**20160103**.oni.frames2calib.cvAndImuExtr.csv | -0.1781 0.9826 -0.0530  0.0995 -0.0356 -0.9944  -0.9790 -0.1824 -0.0914 | 36.4103° | 36.6123° |
| 3 | ir-cb-imu-precalib-**20160111**.oni.frames2calib.cvAndImuExtr.csv | 0.4474 -0.6388 0.6259  0.2599 -0.5768 -0.7745  0.8557 0.5092 -0.0920 | 2.452° | 0.99959 |

* 小结：

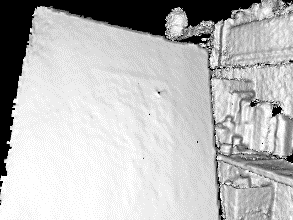
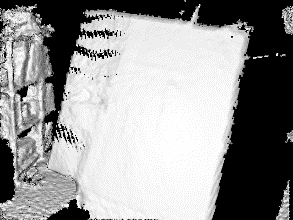
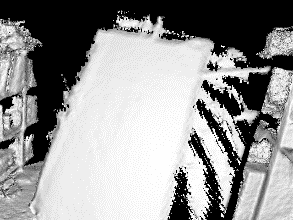
表4测试序号3，其相机内定标只有三帧有效图像（对应表3测试序号2），但是其反投影误差最小（1.55e-1），且手眼标定误差也最小。

### 姿态融合测试

目前主要对比原始pcl-kinfu算法，与使用IMU姿态值（手眼标定之后）做kinfu相机姿态修正的方法。

表 5 pcl-kinfu与kinfu+IMU方法结果对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试序号 | 数据 | 算法 | 使用手眼标定结果 | 备注 |
| 1 | CapturedFrames.ir-cb-imu-precalib-20160111 | pcl-kinfu | - | 原始pcl-kinfu算法 |
| 2 | CapturedFrames.ir-cb-imu-precalib-20160111 | kinfu+IMU | -0.2790 0.8740 0.3978  -0.1239 0.3781 -0.9175  -0.9523 -0.3052 0.0028 |  |

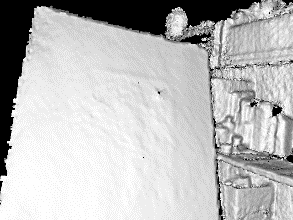
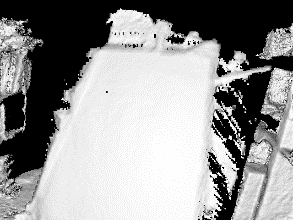
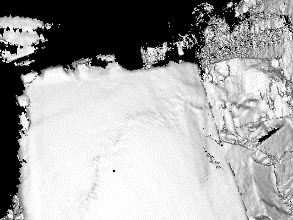
   

图 3 原始pcl-kinfu 对比kinfu+IMU的融合3D重建结果（第112, 127, 161, 567帧）。 a) 第一行，原始pcl-kinfu; b)第二行，kinfu+IMU结果

* 小结：

目前的测试结果上，kinfu+IMU的3D重建结果比原pcl-kinfu结果差。

## 参考文献

[1] 《手眼标定算法(AX=XB)测试报告》，张琛。