# 手眼标定算法(AX=XB)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 | 2015/12/21 |
| V1.1 | 张琛 | 1. 增加@王镇姿态估计算法（对比android系统算法输出）测试 2. 增加outlier剔除测试 3. 增加测试数据集 | 2015/12/24 |
| V1.2 | 张琛 | 增加Lobo82~88作测试集的测试 | 2015/12/27 |
| V1.2.1 | 张琛 | azimuth公式(11)改为 min(…) | 2015/12/28 |

## 引言

### 编写目的

本测试报告使用单个手机采集的[RGB+IMU]离线数据，利用opencv棋盘格标定相机算法，求解手机相机在棋盘格坐标系下的姿态矩阵A；使用华为项目中的IMU姿态估计算法，求解IMU单元在地球ENU坐标系下的姿态矩阵B。再利用手眼标定算法（论文[2][3]），求解相机坐标系与IMU坐标系之间的旋转矩阵，测试标定效果。

### 原理简述

一组[RGB+IMU]数据的采集、存储形式为：在有棋盘格定标板的场景中，相机在多个静止姿态下进行拍照；同时，快门按下的前后1s内，开启IMU单元采集13轴传感器数据（acc{x,y,z}+mag{x,y,z}+gyro{x,y,z}+quaternion{x,y,z,w}），因此每张图像(jpg)对应一个xml存储的IMU数据文件。

手眼标定算法的测试流程如下图1所示：



图 1 手眼标定及测试方法流程图

对于图1中步骤3，即核心标定算法，其原理简述为：

opencv棋盘格标定算法, 得到的是世界（标定板）坐标系到相机坐标系变换矩阵序列 ;

IMU得到的是IMU模块坐标系到世界（地球ENU）坐标系变换矩阵序列 ;

假设从相机坐标系到IMU模块坐标系之间的变换矩阵为 X

考虑如下变换：将棋盘格坐标系下的向量 变换到地球ENU坐标系下，得到向量。在第i, j帧，有如下表达：

(1)

(2)

(1)(2)联立得到：

(3)

即

(4)

分别记 , 需要注意，表示将相机第i帧向量坐标变换到第j帧坐标系的变换矩阵，而表示将IMU第i帧向量坐标变换到第j帧坐标系的变换矩阵，(4)可改写为：

(5)

另，记 ，即 表示从IMU模块坐标系到相机坐标系的变换矩阵，则(5)可改写为

(6)

### 问题描述

如图1所示，本文涉及的手眼标定问题描述为：

**输入：**

1. 原始数据：一组包含棋盘格定标板的图像序列，以及每张图像对应的IMU数据xml文件序列；
2. 核心算法输入：。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。

而 表示将相机第i帧向量坐标变换到第j帧坐标系的变换矩阵，表示将IMU第i帧向量坐标变换到第j帧坐标系的变换矩阵。对于有N张图像的一组标定数据，我们生成了 对 ，带入(6)式求解（实际代码中为了方便观察outlier数据序号并剔除之，我们令j=1,2,…,N，即生成了对 ）。

**输出：**

1. 核心算法输出：从IMU模块坐标系到相机坐标系的旋转矩阵 ，如公式(6)
2. 测试误差输出：

对于解得的 ，验证其标定效果，我们计算 ，标定结果越精确，应该越接近单位矩阵。

1. 我们将 转化为轴角表示，观察其角度值均值：

(7)

值越接近0表示结果越好；

1. 以及将 转化为四元数表示，观察其 分量均值：

(8)

即下文公式(8)，值越接近1表示结果越好；

1. 由 ，得到棋盘格在地球ENU坐标系下姿态矩阵M，进而求解棋盘格下倾角：

(9)

方位角（若Y轴做测量轴）：

(10)

将得到的两个角度与物理仪器测量值进行对比，分别得到下倾角误差downtiltErr，以及方位角误差azimuthErr。

(11)

### 标定 & 测试数据

如图2所示，一组测试数据中，包括不同相机姿态下，对同一个固定姿态的棋盘格定标板进行拍照的RGB图像数据。

   

   

图 2 华为项目采用的NVan标定数据

每张图像对应的IMU数据采用xml文件存储，具体存储文件结构见《时序数据文件格式》[5]。

### 测试环境与工具

* 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}

## 测试方法

### IMU姿态估计算法对比测试

本测试共使用3中算法方案进行IMU姿态估计：

1. Android手机设备自带IMU姿态四元数输出，这一结果使用了acc, mag, gyro共9轴数据做sensor fusion，其缺点是原始数据可能存在偏差(bias)，android的算法没有对这一偏差进行补偿。
2. @王镇算法使用acc, mag共6轴数据进行IMU姿态估计，并在算法之前对acc, mag各自原始数据进行偏差校正归一化。其优点是，偏差校正归一化之后的acc、mag值能够保证姿态估算更加精确，下倾角仅依赖acc数值，方位角仅依赖mag数值；缺点是，由于缺少gyro的校正融合，时序上旋转曲线存在“锯齿”。
3. @张琛实现的求解IMU姿态的补偿滤波算法（直接在手机上计算，并输出）。与方案①相比，此实现结果精度更高[4]，与方案②相比，因为使用了gyro三轴数据做补偿，其旋转曲线更加“平滑”。但是由于是在手机端进行较为低频（30Hz）的运算，相对于①，其旋转曲线仍然有锯齿感。

【注】目前在以@王镇数据{下倾角,方位角}物理测量值为主要误差指标(groundtruth) 的情况下, 仅对比测试①②两方案, ③因为数据采集方法不同, 尚未加入测试 //2015-12-22 22:48:47

测试使用公式(7)(8)(9)(10)误差指标，对三种方案优劣进行评估。

### 手动剔除outlier帧测试

某些图像&IMU数据对，可能存在因RGB图像角点寻找不精确，或者IMU姿态估计误差较大，导致手眼标定最后结果较差，我们尝试根据公式(7)，剔除 的图像&IMU数据对，然后重新进行手眼标定，并与之前未剔除坏数据时的误差进行对比。

### 不同AX=XB求解算法对比测试

本测试目前采用论文[2][3]的算法进行标定。

另外，算法matlab脚本接受的输入矩阵A, B格式均为：4\*n矩阵，即 个4\*4旋转矩阵，横向连接而成。

## 测试结果

### 结果演示

表 1各组数据集的姿态测量真值表（groundtruth）

|  |  |  |
| --- | --- | --- |
| 数据 | 下倾角 (单位: 角度) | 方位角 (单位: 角度) |
| NVan | z g 0.1 | y n 180.0 |
| Lobo81 | z g 90 | x n 30 |

注：”z g 0.1”表示棋盘格Z轴与重力方向夹角为0.1°，” y n 180.0”表示棋盘格Y轴与地磁北极方向夹角为180.0°，其余以此类推。

表 2 测试结果列表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 数据 | IMU姿态输入来源 | 是否剔除outliers | 算法 | 指标 (单位: 角度°) | 指标 (单位: 角度°) | downtiltErr (单位: 角度°) | azimuthErr (单位: 角度°) | 若不标定，得downtiltErr (单位: 角度°) |
| 1 | NVan | Android-Q | N | liang.m，论文 [2] | 8.2059 | 0.99624 | 1.5702 | **6.0205** | **2.1825** |
| 2 | NVan | Android-Q | N | park.m，论文 [3] | 8.2044 | 0.99624 | **1.5728** | **6.0064** | - |
| 3 | NVan | Android-Q | Y | liang.m | **4.2902** | 0.99903 | 1.6665 | 6.2179 | - |
| 4 | NVan | Android-Q | Y | park.m | **4.2919** | 0.99903 | 1.6407 | 6.154 | - |
| 5 | NVan | @王镇 | N | liang.m | 7.1262 | 0.99699 | 0.47532 | 5.1387 | **1.0859** |
| 6 | NVan | @王镇 | N | park.m | 7.1268 | 0.99699 | 0.47782 | 5.1368 | - |
| 7 | NVan | @王镇 | Y | liang.m | 3.5838 | 0.99929 | 0.7631 | 5.1314 | - |
| 8 | NVan | @王镇 | Y | park.m | 3.559 | 0.99929 | 0.62437 | 5.1327 | - |
| 9 | Lobo81 | Android-Q | N | liang.m | 5.4815 | 0.9985 | 3.4656 | 6.535 | **0.12836** |
| 10 | Lobo81 | Android-Q | N | park.m | 5.4856 | 0.9985 | 3.4384 | 6.5596 | - |
| 11 | Lobo81 | Android-Q | Y | liang.m | 3.4581 | 0.99938 | 4.0773 | 4.4767 | - |
| 12 | Lobo81 | Android-Q | Y | park.m | 3.4562 | 0.99938 | 4.0877 | 4.3959 | - |
| 13 | Lobo81 | @王镇 | N | liang.m | 3.802 | 0.99929 | 2.1765 | 7.4002 | **1.02** |
| 14 | Lobo81 | @王镇 | N | park.m | 3.8006 | 0.99929 | 2.1468 | 7.3896 |  |
| 15 | Lobo81 | @王镇 | Y | liang.m | 2.1153 | 0.99978 | 2.7346 | 5.1571 |  |
| 16 | Lobo81 | @王镇 | Y | park.m | 2.119 | 0.99978 | 2.7657 | 5.077 |  |

其中，测试序号3相对于序号1，剔除了 outliers ；当前判断outliers标准为：回测结果中 的帧视为异常帧。测试序号3（NVan数据）剔除的帧序号为：[9, 10, 12, 14, 15, 17:22, 24:26]；序号11（Lobo81）相对9剔除的帧序号为：[4:7, 9,10,11]。

各次测试输出的定标结果（矩阵Y）如表3所示，理想状态下，手机中相机与IMU坐标系之间的旋转矩阵为如下形式：

(12)

表 3 标定结果-矩阵Y值

|  |  |
| --- | --- |
| 序号 | 结果矩阵Y（IMU->camera） |
| 1 | -0.0021 -0.9999 0.0103  -0.9996 0.0018 -0.0275  0.0275 -0.0104 -0.9996 |
| 2 | 0.0114 -0.9999 -0.0014  -0.9997 -0.0114 -0.0210  0.0210 0.0016 -0.9998 |
| 3 | -0.0021 -0.9999 0.0103  -0.9996 0.0018 -0.0275  0.0275 -0.0104 -0.9996 |
| 4 | 0.0015 -1.0000 0.0086  -0.9997 -0.0017 -0.0261  0.0261 -0.0085 -0.9996 |
| 5 | 0.0181 -0.9998 -0.0127  -0.9998 -0.0180 -0.0031  0.0029 0.0128 -0.9999 |
| 6 | 0.0184 -0.9997 -0.0128  -0.9998 -0.0183 -0.0029  0.0026 0.0129 -0.9999 |
| 7 | 0.0179 -0.9998 -0.0059  -0.9998 -0.0179 0.0017  -0.0018 0.0059 -1.0000 |
| 8 | 0.0175 -0.9998 -0.0077  -0.9998 -0.0175 -0.0010  0.0008 0.0077 -1.0000 |
| 9 | 0.0444 -0.9985 0.0315  -0.9943 -0.0411 0.0987  -0.0973 -0.0357 -0.9946 |
| 10 | 0.0502 -0.9983 0.0304  -0.9940 -0.0469 0.0990  -0.0974 -0.0352 -0.9946 |
| 11 | -0.0389 -0.9960 0.0808  -0.9951 0.0460 0.0881  -0.0914 -0.0770 -0.9928 |
| 12 | -0.0446 -0.9957 0.0815  -0.9949 0.0516 0.0871  -0.0909 -0.0772 -0.9929 |
| 13 | 0.0133 -0.9997 0.0183  -0.9958 -0.0116 0.0908  -0.0906 -0.0194 -0.9957 |
| 14 | 0.0203 -0.9996 0.0183  -0.9957 -0.0186 0.0908  -0.0905 -0.0200 -0.9957 |
| 15 | -0.0579 -0.9957 0.0725  -0.9949 0.0636 0.0782  -0.0825 -0.0676 -0.9943 |
| 16 | -0.0636 -0.9953 0.0733  -0.9946 0.0693 0.0778  -0.0825 -0.0680 -0.9943 |

进一步地，我们使用测试序号5（NVan数据集，王镇姿态估计结果，liang[2]手眼标定算法，不进行outlier去除的方式）的标定结果，在同一设备在其他场景采集的数据上测试误差；测试同样采用liang[2]手眼标定算法，不进行outlier去除的方式。

待测试的数据集包括：{Lobo82, Lobo83, Lobo84, Lobo85, Lobo86, Lobo87, Lobo88}，误差度量结果如下表4所示。

表 4 测试序号5标定结果在其他测试数据上的误差度量

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试数据 | IMU姿态输入来源 | 指标 (单位: 角度°) | 指标 (单位: 角度°) | downtiltErr (单位: 角度°) | azimuthErr (单位: 角度°) |
| Lobo82 | Android-Q | 12.5434 | 0.99063 | 0.16418 | 6.5502 |
| Lobo82 | @王镇 | 11.8399 | 0.99152 | 1.6535 | 6.0025 |
| Lobo83 | Android-Q | 6.373 | 0.99727 | 0.80945 | 5.5317 |
| Lobo83 | @王镇 | 5.2585 | 0.99793 | 0.88689 | 4.4127 |
| Lobo84 | Android-Q | 2.0835 | 0.99975 | 0.27037 | 119.2092 |
| Lobo84 | @王镇 | 4.8135 | 0.99858 | 1.47 | 20.1227 |
| Lobo85 | Android-Q | 1.4351 | 0.99989 | 0.59879 | 122.3968 |
| Lobo85 | @王镇 | 2.4951 | 0.99966 | 0.78222 | 14.9589 |
| Lobo86 | Android-Q | 5.0938 | 0.99835 | 0.67971 | 79.5785 |
| Lobo86 | @王镇 | 6.7455 | 0.9973 | 0.25491 | 0.83014 |
| Lobo87 | Android-Q | 4.0858 | 0.99895 | 0.83356 | 3.2253 |
| Lobo87 | @王镇 | 4.4809 | 0.9988 | 0.38594 | 12.6376 |
| Lobo88 | Android-Q | 9.1529 | 0.99414 | 0.30468 | 0.63169 |
| Lobo88 | @王镇 | 7.5353 | 0.99536 | 0.61092 | 3.2863 |

从表4的误差度量结果来看，使用IMU不同的姿态计算方式（Android-Q vs. @王镇），在Lobo82~Lobo88数据上，对于主要误差指标 以及 downtiltErr，难以判断哪IMU姿态计算方式更优。

### 误差分析

从表1各次测试结果来看，{下倾角，方位角} 误差是可以接受的，并且标定结果是优于不标定直接用公式(10)理想坐标变换矩阵的（误差1.57°vs. 2.18°），这表明所求得的IMU-camera转换矩阵Y是正确的。

造成误差的原因可能为：

1. 手眼标定算法不同，求解精度不同；
2. ~~我使用了i~0变换矩阵，共K对（K为一组数据集中图像个数）作为算法输入，可能不够稳定（e.g., 假如第0帧恰好较差），应该用每两帧i~j之间的变换矩阵作为输入，共 对作为算法输入（建议by王镇）；~~

//代码已修改, 目前使用 对 作为输入。2015-12-21 02:29:27

1. IMU姿态求解算法不同，@王镇实现的姿态估计算法比android系统输出的IMU姿态更精确；
2. 以 为标准，剔除outlier之后，对误差 有明显的改善效果（参见序号7 vs. 5, 15 vs. 13）；但是对应的误差downtiltErr却升高，目前原因不确定。
3. 从downtiltErr看，数据NVan上，标定效果有明显提升：序号1:: 2.1825->1.5702；序号5:: 1.0859->0.47532；但是在数据Lobo81上，标定效果比不标定直接用矩阵 （公式(12)）效果差：序号9:: 0.12836->3.4656；序号13:: 1.02->2.1765

## 参考文献

[1] <http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_ROTATION_VECTOR>

[2] R. Liang, J. Mao Hand-Eye Calibration with a New Linear Decomposition Algorithm. In Journal of Zhejiang University, 9(10):1363-1368, 2008.

[3] F. Park, B. Martin Robot Sensor Calibration: Solving AX = XB on the Euclidean Group. In IEEE Transactions on Robotics and Automation, 10(5): 717-721, 1994.

[4] 《使用补偿滤波对IMU传感器数据融合测试报告》，张琛.

[5] 《时序数据文件格式》，杜宇.