IMU TK 2

Leandro de Souza Rosa
leandro.desouzarosa@iit.it

August 26, 2021

Contents

1 Introduction

The document summarizes the usage of the IMU calibration tool and all extra option developed in the IIT context. The objective of this report is to introduce the IMU calibration tool, explain its options, briefly describe the work done in our context, and to log the state and partial results and conclusions.

The original work [1] presents an IMU calibration tool which does not depend on specific equipment, not precise positioning of the IMU, which is compliant with our IMU system. As such, in order to compute the calibration parameters, an optimization must be performed using several different IMU static positions.

Other methods use specific equipment to reduce the number of positions required for the calibration [2], to analytically calculate the calibration parameters [3]. [4] proposes the same approach.

2 Method

The method for calibration considers the frames of the accelerometer and gyroscope as two non-orthogonal and misaligned frames, as illustrated in Figure 1, and estimates a linear transformation from each sensor's frame to a common orthogonal one.

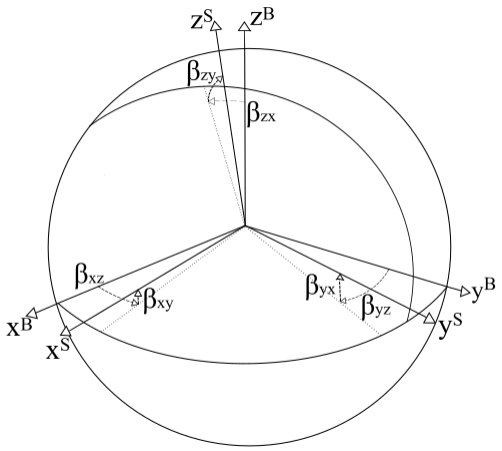

Figure 1: Accelerometer and Gyroscope axis are non orthogonal and misaligned.
Source: Reproduced from [1].

In other words, let *acc* and *gyr* be the accelerometer and gyroscope readings, the method estimates the scale $K_{3\times3}$, the skew $S_{3\times3}$, and the bias $B_{3\times1}$ for both sensors, and the calibrated measures acc^* and gyr^* are computed as follows:

$$\begin{aligned} acc^* &= K^a S^a (acc - B^a) \\ gyr^* &= K^g S^g (gyr - B^g) \end{aligned} \tag{1}$$

Important

1. The *acc* and *gyr* readings given by the IMU are usually in their own scale. The conversion to S.I. is made intrinsically through the scale matrices S^a and S^g.
2. The scale K is a diagonal matrix (only the 3 diagonal elements are non-zero), S^a is a triangle upper matrix (only the 3 elements on the upper triangle are non-zero), and S^g is a triangle upper and lower matrix (all the 6 non-diagonal elements are non-zero).

The first step is to calculate the calibration parameters for the accelerometer, which is done by optimizing:

$$\underset{K^a,S^a,B^a}{\operatorname{argmin}} \left(||g|| - ||K^a S^a (acc - B^a)||\right)^2 \tag{2}$$

Where g is the local gravity, and the acc measures used here belong to the static intervals of the data, which are automatically detected by the tool (more in Section 3).

After the accelerometer is calibrated, the gyroscope is calibrated in two steps.

First, B^g is estimated by averaging the gyroscope measures in the initial static interval (see Section 3). Second, K^g and S^g are estimated by optimizing:

$$\underset{K^g,S^g}{\operatorname{argmin}} \left\| (\tilde{g}_{t_{i+1}} - \tilde{g}_{t_i}) - \int_{t_i}^{t_{i+1}} K^g S^g (gyr - B^g) dt \right\| \tag{3}$$

Where $\tilde{g}_{t_i}$ is the gravity versor at time t_i, meaning that the term $||\tilde{g}_{t_{i+1}} - \tilde{g}_{t_i})||$ is the angular displacement between to consecutive static intervals, and the integral calculates the angular displacement from the gyroscope readings.

Important:

Important

Errors in the accelerometer's calibration will propagate to the gyroscope's calibration giver the later depend on the earlier.

3 Data Collection

The data must be collected in the pattern illustrated in Figure 2. First, the IMU must be static for t_{init} seconds. Then it must be placed in another aleatory position for t_{wait} seconds, which causes the accelerometer reading to flip. This process is repeated $n\times$, generating the accelerometer signals patterns illustrated in Figure 2.

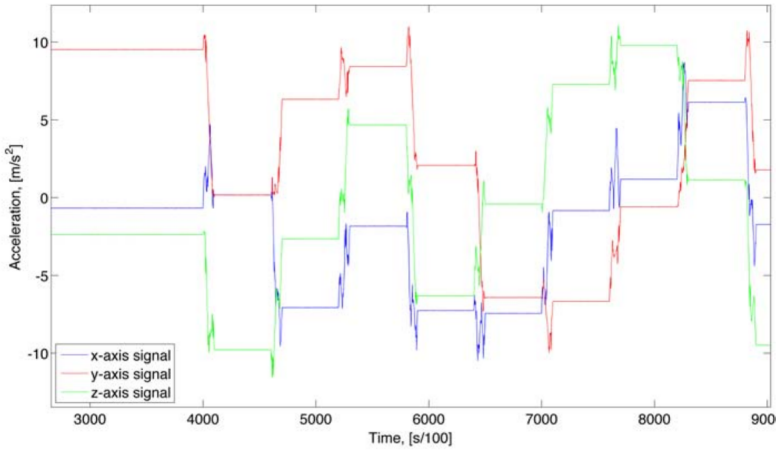

Figure 2: Data collection pattern.
Source: Reproduced from [5]

Important

The software will automatically detect the static intervals using the variance of the accelerometer readings as described in [1]. The first static interval needs to be long (recommended $t_{init} \approx 50s$), which the other intervals are short (recommended $t_{wait} \approx 3s$). The tool will not perform the calibration if it fails to identify static intervals satisfying these conditions.

We implemented a module to record and save the IMU readings in a data within the `event-driven` library. For installation follow:

1. Check out the branch: `https://github.com/robotology/event-driven/tree/IMU_port`
2. Compile with the `cmake` option `-DENABLE_IMUCalibDumper=ON`

For running, we recommend starting the module using the command line, which will prompt a timer to help the user to keep track of the t_{init} and t_{wait} times. Use the following command, indicating where the accelerometer and gyroscope files must be saved (choose an existing folder):

`IMUCalibDumper --accFileName <full path for acc file> --gyrFileName <full path for gyr file>`

Important

The integral to calculate the orientation displacement between two static intervals (Equation 3) is susceptible to integration bias, the angular displacements between two static intervals during data acquisition must be short (try to keep is less than 90°).

The data will be written in the specified file only during `IMUCalibDumper`'s closure.

4 IMU TK 2

In this section we describe the installation and usage of the calibration toll.

The code is available at: `https://github.com/event-driven-robotics/imu_tk2` Refer to the `README` file for dependencies and installation. Table 1 summarises the repository structure and its contents.

Table 1: Brief description of the repository contents.

Folder	Description
apps	c++ top level application code. `test_imu_calib.cpp` is the main file, which uses `lib boost` to receive the application options.
include/imu_tk	Contains the header files `.h`.
lib	Static library files.
python	Contains scripts to run the application over multiple calibration and test data files, evaluate, and plot the results.
report	The files of this report.
src	Contains the main c++ files. The `io_utils.cpp` implements functions to read the data collected with the `IMUCalibDumper` module described in Section 3. The `calibration.cpp` file implements the main calibration steps.
data.zip	Contains 10 calibration data and 10 test data used for evaluation.

Note that the structure, data types, and templated used in the implementation are legacy from the original `IMU_TK` software.

After installation, you can see the implemented options (described on Section 4.1) by running:

```
$ ./bin/test_imu_calib --help
```

4.1 Options

acc_file, gyr_file

Path for the file containing the accelerometer and gyroscope data for calibration. Those should be the file paths passed as arguments to the `IMUCalibDumper` module described in Section 3.

g_mag

Magnitude of the local gravity. The value is retrieved from public datasets. A list of tools and datasets for calculating the local data can be found in `https://www.isobudgets.com/how-to-calculate-local-gravity/`.

init_interval_duration

Duration of the initial static interval t_{init} (described in Section 3). If the tool will not perform the calibration if it fails to identify an initial interval longer than t_{init}.

gyr_dt

Gyroscope sampling period used in the integration to compute the angular displacement between to static interval (Equation 3). Leave this option unset, or set it to -1, and the toll will automatically calculate the dt from the samples timestamps.

nominal_1g_norm

Conversion value between the IMU scale and $1g$ given by the data sheet. This value is used as an initial guess for the accelerometer calibration.

alpha

Unused.

min_num_intervals

Minimal number of static intervals for performing the calibration. The toll will not perform the calibration if it is unable to detect at least this number of static intervals that last longer than t_{wait} s. The recommended number is > 36.

interval_n_samples

Minimal number of IMU samples in each static interval. This number can be estimated by dividing t_{wait} by the IMU sample rate. This value is used as a condition to count the minimal number os static intervals defined by the option `min_num_intervals`.

max_iter

Sets the maximum number of iterations for the Ceres solved to optimise Equations 2 and 3. In our tests, with the initial conditions set up properly, the solver is typically able to find a solution with tenths of iterations.

Important

It is worthy mentioning that the optimisations have no constraints, meaning that the solver has freedom to select arbitrary values for some calibration parameters, which is compensated by other parameters. Furthermore, the solver is sensitive to the initial conditions. As such, a large number of iterations indicate that the initial conditions might not be properly set.

acc_use_means

Instructs the tool to use the mean of each static interval to compute the residuals during the optimizations of Equations 2 and 3. Otherwise, it the tool will add the residual for each IMU reading during the static interval.

As such, with this option turned on, high-frequency noise is filtered, and the solver speed is improved (not a significant difference) Our tests indicate that the impact of this option is not significant.

opt_gyr_b

In our tests, we noticed that calculating the gyroscope biases B^g only from the initial static interval was inconsistent. As such, we implemented an optimisation for calculating B^g bases on all static intervals. This is done by optimising

$$\underset{B^g}{\operatorname{argmin}} \|var(gyr, B^g)\| \tag{4}$$

Where, $var(gyr)$ is the variance of B^g in the static intervals. However, due to restrictions imposed by the solver, the above optimisation cannot be implemented as it is. Thus we develop the optimisation as follows:

$$\begin{aligned} var(gyr, B^g) &= \sqrt{\sum(\tilde{gyr} - gyr)^2} && (5)\\ &= \sqrt{\sum(\tilde{B^g} - gyr)^2} && (6)\\ &= \sqrt{\sum(B^g - gyr)^2} && (7) \end{aligned}$$

Where $\tilde{gyr}$ is the average value of the gyroscope readings during the static interval. From Equations 5 to 6, we use the fact that the reading are composed solemn by the bias, given the data is static. From Equations 6 to 7, we use the fact that the bias should be constant, allowing to eliminate the mean.

As such, when the *opt_gyr_b* is turned on, B^g is estimated by optimising:

$$\underset{B^g}{\operatorname{argmin}} \|B^g - gyr\| \tag{8}$$

use_acc_G

Unused.

use_gyr_G

According to [6] the gyroscope bias has a component proportional to the accelerometer readings. We extended the optimisation to estimate the gyroscope calibration parameters (Equation 3) to also estimate this accelerometer-dependent factor as:

$$\underset{K^g, S^g, G^g}{\operatorname{argmin}} \left\| (\tilde{g}_{t_{i+1}} - \tilde{g}_{t_i}) - \int_{t_i}^{t_{i+1}} K^g S^g (gyr - B^g - G^g acc) dt \right\| \tag{9}$$

Where $G^g_{3\times3}$ is a matrix which encodes the relation between the accelerometer reading and the gyroscope biases. The calibrated gyroscope values is calculated according to:

$$gyr^* = K^g S^g (gyr - B^g - G^g acc) \tag{10}$$

Results indicate a non-significant reduction on the error during our integration tests (see Section 5.1).

verbose

Print debug, staging, and solver messages on the terminal.

init_acc_bias

Initial guess for the accelerometer bias.

init_gyr_scale

Analogous to `nominal_1g_norm`, this value encodes the conversion factor between the raw gyroscope reading and S.I. obtained from the IMU data-sheet. This value is used as an initial guess for K^g.

suffix

After a successful calibration, the parameters for the accelerometer and gyroscope are saved in two files in the same folder as the calibration data indicated by the `acc_file, gyr_file` parameters. The `suffix` is appended to the name of the resulting parameter file names. This is useful when performing many calibrations with different options.

5 Tests

The `python` folder contains scripts to calibrate, test, evaluate, and compare calibrations. Table 2 presents a summary of the scripts found in the folder.

Table 2: Brief description of the python scripts.

Script	Description
apply-Calib.py	Performs the calibration over a set of calibration data using different combinations of the options described in Section 4.1.
checkStatic-Data.py	Reads a set of data collected with the static camera in different orientations and plots the gyroscope readings distributions. This allows an visualisation of the gyroscope bias according to its orientation (i.e. depending on the accelerometer readings).
madgwick-ahrs.py	Madgwick filter used to remove the gravity from the accelerometer for the integration tests.
plotCalibBi-asAndInte-gration.py	Reads the calibration parameter for each calibration data and each calibration option, calculates the calibrated values for the accelerometer and gyroscope readings for a set of test data, and visualise the resulting magnitude of the accelerometer and gyroscope biases. It also performs and visualises the integration tests, in which the orientation and linear velocities are calculated from the gyroscope and accelerometer, respectively, given a metric for the impact of the calibration.
quater-nion.py	Basic library for handling quaternions. Useb by the Madgwick filter.
utils.py	Miscellaneous functions used by the other scripts.
test.py	Development board. do not use.

5.1 Integration Tests

The calibrations evaluation is two-fold. On one hand we perform several calibrations, with the same options, and evaluate which one gives the best results. On the other hand, we want to perform calibrations with different options, and evaluate which ones lead to the best results. Regardless, we test it over several test data, to have an idea of the overall precision. We collected 10 calibration data, using 5 combinations of calibration options, and tested over 10 test data, givin a total of 500 tests.

As a metric we perform a simple integration of the accelerometer into the linear velocities, and from the gyroscope into orientation. The test data always start and ends with the stationary

camera, and the initial and final orientation is approximately the same, which is obtained by aligning tow surfaces of the IMU box with two flat surfaces (wall and table).

Before calculating the linear velocities, the gravity effect on the accelerometer is removed using the Madgwick filter. And the first 20 s of the data is discarded, giving time for the Madgwick filter convergence, during this period the camera was stationary.

Figure 3 shows the comparison between calibration trials, showing that the accelerometer calibration usually improves the integration error (left). However, the out-layers show that, for some test cases, the calibration leads to larger error than the integration of the uncalibrated data. We suspect that this is a consequence that the accelerometer is calibrated by comparing the norm of the accelerometer readings against the norm of the gravity (Equation 2), making it possible to obtain a wrong biases for each axis, as long as their magnitude matches the optimisation criteria.

Still on Figure 3, the orientation error (right) shows significant variation between calibration trials.

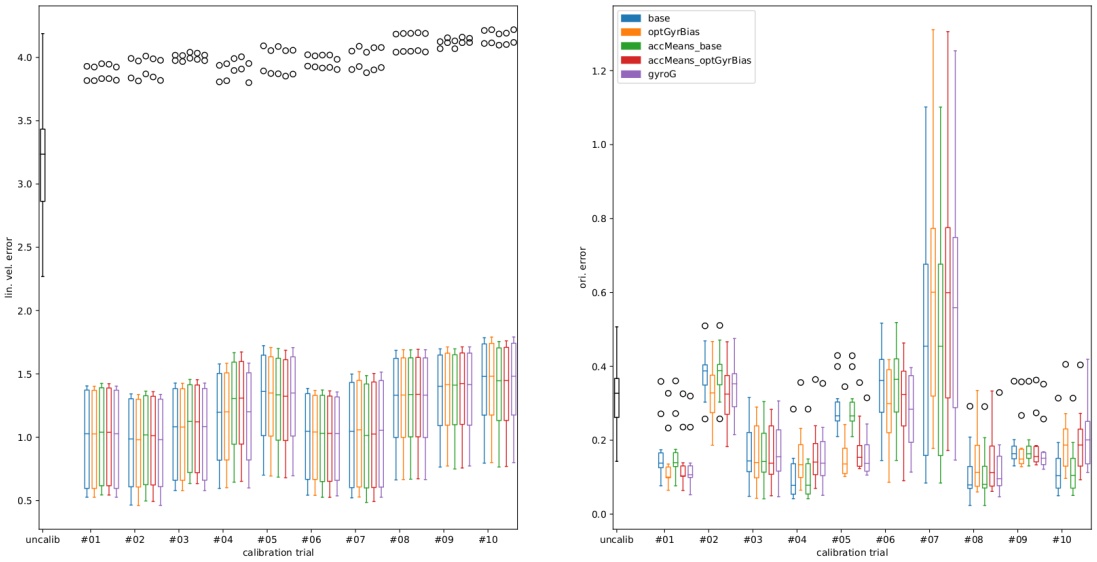

Figure 3: Comparison between calibrations trials. Each colour is a different combination of calibration options, black (on the left of each figure) refers to uncalibrated data. The linear velocity error (on the right) and the orientation (on the left) are calculated through the integration of the IMU readings.

Figure 4 compares different combinations of the calibration options, showing that the developed `opt_gyr_b` and `use_gyr_G` options slightly improve the integration error results.

From these tests, we conclude that more investigation can be done on the accelerometer calibration in order to pin-point what is causing the large integration error depending on the data.

Furthermore, it is worthy to remember that the accelerometer calibration impacts the gyroscope one, and the gyroscope data is used in the gravity compensation step to before integrating the linear velocities, making errors in the gyroscope calibration to propagate the other way around to the accelerometer-related results.

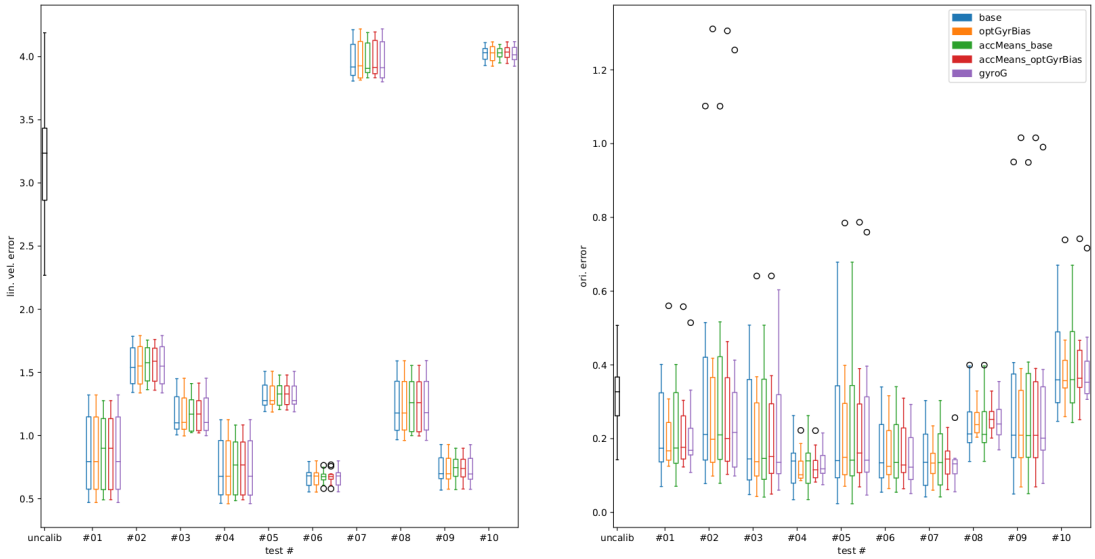

Figure 4: Comparison between calibration options. Each colour is a different combination of calibration options, black (on the left of each figure) refers to uncalibrated data. The linear velocity error (on the right) and the orientation (on the left) are calculated through the integration of the IMU readings.

References

[1] David Tedaldi, Alberto Pretto, and Emanuele Menegatti. A robust and easy to implement method for imu calibration without external equipments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3042–3049, 2014.

[2] Jixin Lv, Ankit A Ravankar, Yukinori Kobayashi, and Takanori Emaru. A method of low-cost imu calibration and alignment. In *2016 IEEE/SICE International Symposium on System Integration (SII)*, pages 373–378. IEEE, 2016.

[3] Sitank Bhatia, Hai Yang, Rui Zhang, Fabian Höflinger, and Leonhard Reindl. Development of an analytical method for imu calibration. In *2016 13th International Multi-Conference on Systems, Signals & Devices (SSD)*, pages 131–135. IEEE, 2016.

[4] Umar Qureshi and Farid Golnaraghi. An algorithm for the in-field calibration of a mems imu. *IEEE Sensors Journal*, 17(22):7479–7486, 2017.

[5] David Tedaldi. Imu calibration without mechanical equipment.(calibrazione di imu svincolata da apparati meccanici). 2013.

[6] Bagis Altinöz and Derya Ünsal. Determining efficient temperature test points for imu calibration. In *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 552–556, 2018.