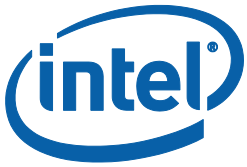

Intel® RealSense™ Depth Camera D435i IMU Calibration

Revision 001

January 2019

By Daniel J. Mirota, Ph.D.

Senior Machine Learning Engineer, Intel Corporation

James Scaife Jr

Senior Customer Engineer, Intel Corporation

Contents

Tables

List of Figures

Revision History

Revision Number	Description	Revision Date
001	Initial Release	January 2019

1 Introduction

1.1 Purpose and Scope of This Document

In order to best utilize the inerial measurement unit (IMU) within Intel® RealSense™ Depth Camera D435i device efficiently and accurately, it is recommended for users to perform an IMU calibration. This document serves as a guide for users to use a provided Python script which computes the calibration.

It is not in the scope of this document to discuss details of calibration algorithm or accuracy.

1.2 Organization

This document is organized into four main parts: overview, setup, calibrating a device with the Python script, and writing the calibration back to the camera:

- **Overview** – brief overview of the calibration parameters.
- **Setup** – hardware and software setup for running the calibration Python script to calibrate a device.
- **Calibrating and Writing Calibration Parameters with the calibration Python script** – describes the necessary hardware and software setup required running the calibration Python script and details steps to calibrate device.

2 Overview

2.1 IMU Overview

Inerial Measurement Units (IMUs) are often composed of an accelerometer to measure acceleration usually output in International System (SI) units of meters per seconds squared (m/s^2) and a gyroscope which measures angular velocity usually in SI units of radians per second (rad/s). The IMU within the D435i is no different and contains both an accelerometer and gyroscope with configurable output frequencies.

2.2 IMU Calibration Parameters

The IMU calibration parameters includes intrinsic and extrinsic paramters. While there are many possible IMU calibration parameters, for the sake of simplicity we consider the following parameters:

The intrinsic parameters include:

- For the accelerometer:
  - Scale factor (sensitivity) – which are terms used to multiply the raw measurements to ensure the output is metric scale. Mathmatically written as s_x, s_y, s_z
  - Bias (zero offset) – which are terms used to cancel any non-zero values when the sensor should be reading zero. Mathmatically written as $\vec{a}_{bais}$
  - Off-axis terms – these terms are used to correct if the axes of the accelormeter are not orthogonal. Mathmatically written as $c_{xy}, c_{yx}, c_{xz}, c_{zx}, c_{zy}, c_{yz}$
- For the gyroscope:
  - Bias (zero offset) – which are terms used to cancel any non-zero values when then sensor should be reading zero. Mathmatically written as omega bais, $\vec{\omega}_{bais}$

The instrinic parameter attempt to transform the raw sensor data into real measurements by modeling the inaccuracies of the sersor. Mathimatically for the accelerometer this transformation is write as follows:

$$\vec{a}_{true} = \begin{bmatrix} s_x & c_{xy} & c_{xz} \\ c_{yx} & s_y & c_{yz} \\ c_{zx} & c_{zy} & s_z \end{bmatrix} \vec{a}_{raw} - \vec{a}_{bais}$$

In this case the gyroscope follows a simplier model as follow:

$$\vec{\omega}_{true} = \vec{\omega}_{raw} - \vec{\omega}_{bais}$$

For further reading on basics of IMU instrinic paramters please refer to STMicroelectronics AN4508 "Parameters and calibration of a low-g 3-axis accelerometer" available at https://www.st.com/content/ccc/resource/technical/document/application_note/a0/f0/a0/62/3b/69/47/66/DM00119044.pdf/files/DM00119044.pdf/jcr:content/translations/en.DM00119044.pdf

The extrinsic parameters include:

- Rotation - rotation from the left infrared (IR) camera (IR1) to IMU, specified as a 3x3 rotation matrix
- Translation - translation from the left IR camera to IMU, specified as a 3x1 vector in millimeters

The scope of this document only includes the intrinsic parameter calibration process. The extrinsic parameters are available using librealsense.

2.3 Accuracy

The calibration will not be accurate if all steps are not followed. To improve the accuracy of each pose a 3 axis level is recommended to ensure the closest alignment to gravity.

3 Setup

This section describes the required hardware and software setup for running the calibration Python script to calibrate a device.

3.1 Hardware

The hardware required includes the D435i device to be calibrated, a USB cable, and a computer running Windows* 10 or Ubuntu* 16.04.

Figure 3-1 Hardware Setup

3.1.1 Device

Intel® RealSense™ Depth Camera D435i device as shown below is used to show the calibration process.

Figure 3-2 D435i Device

3.1.2 USB

A USB type C cable to connect the device to the host computer.

3.1.3 PC

A computer running Windows* 10 or Ubuntu* 16.04 for a full list of supported operating system please refer to the librealsense README.md.

3.2 Software

Install the following on the host computer.

3.2.1 Python 2.7 or 3

On Windows:

Download and install Python: https://www.python.org/downloads/windows/

On Ubuntu:

- Python 2.7
    - sudo apt-get install python
- Python 3
    - sudo apt-get install python3

3.2.2 Python Calibration Script

The calibration script (part of Librealsense SDK version 2.18 and newer), rs-imu-calibration.py, is included in the Intel® RealSense™ SDK available at https://github.com/IntelRealSense/librealsense. The file is located in the tools/rs-imu-calibration directory of the source tree.

3.2.3 Intel® RealSense™ SDK 2.0

Install the latest release of the Intel® RealSense™ SDK. ***Table 3-1*** contains pointers to the SDK homepage, GitHub* repository where you can download the latest release, and the SDK documentation.

Table 3-1. Intel® RealSense™ SDK Resources

Resource	URL
Intel® RealSense™ SDK Home Page	https://software.intel.com/en-us/realsense/sdk
LibRealSense GitHub*	https://github.com/IntelRealSense/librealsense
SDK Documentation	https://github.com/IntelRealSense/librealsense/tree/master/doc
Python Wrapper	https://github.com/IntelRealSense/librealsense/tree/master/wrappers/python

3.2.4 Pip / Numpy / Enum

On Windows:

Pip:

- Download "get-pip.py" script: https://bootstrap.pypa.io/get-pip.py
- Run command: python get-pip.py

Numpy:

- Run command: pip install numpy

Enum:

- Python 2.7
  - Run command: pip install enum34

On Ubuntu (Pip/Numpy/Enum):

Pip:

- Python 2.7
  - Run command: sudo apt-get install python-pip
- Python 3
  - Run command: sudo apt-get install –y python3-pip

Numpy:

- Python 2.7
    - Run command: sudo pip install numpy
- Python 3
    - Run command: sudo pip3 install numpy

Enum

- Python 2.7
    - Run command: sudo pip install enum34

Note: Some of the packages in this section are already included in Python3, please refer to the location in which Python 3 was obtained for details.

3.2.5 Intel® RealSense™ SDK pyrealsense2 wrapper

On Windows from a command prompt window with python pip in the path:

pip install pyrealsense2

Note: pip install on Windows* 10 only works with Python 2.7. Users with Python 3 will need to compile pyrealsense2 from source. Follow directions in Windows section: https://github.com/IntelRealSense/librealsense/blob/master/wrappers/python/readme.md

On Ubuntu:

- Python 2.7
    - Run command: sudo pip install pyrealsense2
- Python 3
    - Run command: sudo pip3 install pyrealsense2

4 Calibrating Device with the Python Calibration Script

4.1 Process Overview

The general process to calibrate a device with the Python Calibration Script starting the script to capture IMU data in 6 different positions, then computing the parameters and writing the results to the camera. It is important to read this entire section before performing the calibration process.

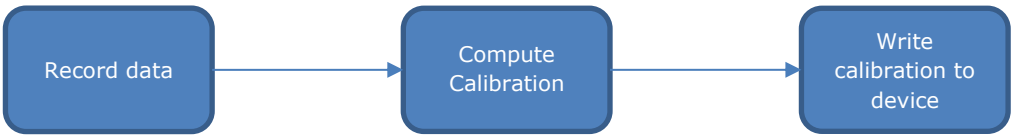

Figure 4-1 Calibration Process

4.2 Connect Device to Computer

Connect the device using the USB cable to the PC where Intel® RealSense™ SDK 2.0 has been installed.

4.3 Running the rs-imu-calibration.py

4.3.1 Starting the process

Use a bash terminal on Ubuntu or a command prompt in Windows to navigate to where rs-imu-calibration.py is installed.

From a command prompt:

python rs-imu-calibration.py

Note: Recommend user to verify setup of Python Wrapper as outlined at https://github.com/IntelRealSense/librealsense/tree/master/wrappers/python prior to running python script

4.3.2 Capturing IMU data from 6 positions

The calibration algorithm in the Calibration Python script requires 6 different position of the device to compute the calibration. The device should be still in each of the 6 positions for 3 to 4 seconds. Be sure to hold the camera as steady as possible in each position.

Example output of the start of the script:

Start interactive mode:

FOUND GYRO with fps=400

FOUND ACCEL with fps=250

4.3.2.1 Position #1 – Upright facing out

Leave the camera place in the camera's natural position with the ¼-20 threaded tripod mount to the ground as shown in Figure 4-2

Figure 4-2 Upright facing out position

After starting the recording as previously described in section 4.3.1, leave the camera still in this position for 3 to 4 seconds. The script will provide a prompt and guide through each of the positions.

Example script output:

Align to direction: [0. -1. 0.] Upright facing out

Status.collect_data[.....]]

Direction data collected.

4.3.2.2 Position #2 – USB cable up facing out

With the camera facing the same direction as described in section 4.3.2.1, rotation the camera 90 degree about the camera's viewing directions such that the USB cable is now pointed to the ceiling.

Figure 4-3 USB cable up facing out

Leave the camera still in this position for 3 to 4 seconds.

4.3.2.3 Position #3 – Upside down facing out

From Position #2 rotation the camera an additional 90 degrees about the viewing direction of the camera such that now the camera is upside down with the ¼ - 20 threaded tripod mount facing up.

Figure 4-4 Upside down facing out

Leave the camera still in this position for 3 to 4 seconds.

4.3.2.4 Position #4 – USB cable pointed down

From Position #3 rotate the camera an additional 90 degrees such that the USB cable is now pointed down.

Figure 4-5 USB cable pointed down

Once again, leave the camera still in this position for 3 to 4 seconds.

4.3.2.5 Position #5 – Viewing direction facing down

Place the camera viewing direction facing down such that the Intel® RealSense™ logo is facing up.

Figure 4-6 Viewing direction facing down

Leave the camera still in this position for 3 to 4 seconds.

4.3.2.6 Position #6 – Viewing direction facing up

From Position #5 rotate the camera 180 degrees about the USB cable such that the Intel® RealSense™ logo is face down.

Figure 4-7 Viewing direction facing up

Leave the camera still in this position for 3 to 4 seconds. Let the camera remain in this final position until the recording stops.

4.4 Computing the calibration

Once all 6 positions have been captured the script will provide a prompt asking if the raw data is to be saved.

Example output:

Would you like to save the raw data? Enter footer for saving files (accel_<footer>.txt and gyro_<footer>.txt)

Enter nothing to not save raw data to disk. >

Next the the calibration is computed and output the results of the optimization to the screen.

Example output:

[-0.0063249 0.00059934 -0.00040395]

[1000 1000 1000 1000 1000 1000]

using 6000 measurements.

[[1.02087445e+00 2.37974585e-02 7.15352304e-03]

[-8.14349108e-03 1.01582282e+00 1.00026542e-02]

[3.99463242e-05 1.56443124e-02 1.00332357e+00]

[-6.91525434e-02 4.49215851e-02 -1.96254675e-02]]

residuals: [22.17221649 61.83331734 207.34662776]

rank: 4

singular: [440.37743294 432.58894714 425.28730317 77.45069689]

norm (raw data): 9.675980

norm (fixed data): 9.802881 A good calibration will be near 9.806650

4.5 Updating Results to Device

After computing the calibration the script presents the option of writing the results to the camera's eeprom.

Example output:

Would you like to write the results to the camera's eeprom? (Y/N)y

Writing calibration to device.

Device PID: 0B3A

Device name: Intel RealSense D435I

Serial number: 831612073544

Firmware version: 05.10.15.00

255.255.255.255

SUCCESS: saved calibration to camera.

Done.