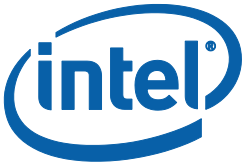

IMU Calibration Tool for Intel® RealSense™ Depth Camera

Intel® RealSense™ Depth Camera D435i, Intel® RealSense™ Depth Camera D455, Intel® RealSense™ LiDAR Camera L515

Revision 1.4

July 2020

Contents

Tables

List of Figures

Revision History

Revision Number	Description	Revision Date
1.0	Initial Release	January 2019
1.1	Updated calibration tool description and limitations	March 2020
1.4	Intel® RealSense™ LiDAR Camera L515 Intel® RealSense™ Depth Camera D455	July 2020

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 and D455 cameras and Intel® RealSense™ LiDAR Camera L515 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 sample 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 Intel® RealSense™ Depth Camera D435i and D455 cameras and Intel® RealSense™ LiDAR Camera L515 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 IMU Calibration Tool

A calibration python script, rs-imu-calibration.py, is included in the Intel® RealSense™ SDK 2.0. Please see details later in this document.

This tool is provided as a convenience as well as an example to demonstrate the process to calibrate the IMU on the D435i, D455, and L515 cameras.

2.4 Limitations and Accuracy

While the provided tool achieves good overall calibration results, the calibration will not be accurate if all steps are not followed. Factors outlined below may also affect calibration performance:

- Calibration requires positioning the camera device in 6 designated orientations and alignments (horizontally or vertically depends on orientation). Better alignment minimizes error and improves accuracy. A 3-axis level is recommended to ensure the closest alignment to gravity.
- The tool optimizes all directions, while the average acceleration result is close to gravity acceleration (9.8 m/s^2), sometimes acceleration in individual direction may be slightly under or over perfect gravity acceleration.

2.5 User Custom Calibration

In case users prefer to use their own tool to calibrate the device to meet specific application requirement, results can be written to device as outlined by methods demonstrated in rs-imu-calibration.py.

IMU calibration table format is specified in https://user-images.githubusercontent.com/6958867/50902974-20507500-1425-11e9-8ca5-8bd2ac2d0ea1.png

The table format is same on all devices, except the version and table type fields in the table header.

	Table version	Table Type
D435i and D455	2.1	0x20
L515	5.1	0x243

2.6 FW Requirement

Intel® RealSense™ LiDAR Camera L515 FW 1.4.1.0 and later is required to support IMU calibration.

Intel® RealSense™ Depth Camera D435i/D455 FW released with camera and later is supported for IMU calibration.

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, D455 or L515 device to be calibrated, a USB cable, and a computer running Windows* 10, Ubuntu* 16.04, and Ubuntu 18.04.

Figure 3-1 Hardware Setup

3.1.1 Device

Intel® RealSense™ Depth Camera D435i, Intel® RealSense™ Depth Camera D455, or Intel® RealSense™ LiDAR Camera L515 device as shown below is used to show the calibration process.

Figure 3-2 D435i Camera

Figure 3 D455 Camera

Figure 3-4 L515 Camera

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 PythonOn Windows:

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

On Ubuntu:

- Python 2.7
    - sudo apt-get install python

3.2.2 Python Calibration Script

The calibration script (part of Intel® RealSense™ SDK), rs-imu-calibration.py, is available at https://github.com/IntelRealSense/librealsense. Please download the latest version to ensure all calibration features are supported. 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

Numpy:

- Python 2.7
    - Run command: sudo pip install numpy

Enum

- Python 2.7

3.2.5 Run command: sudo pip install enum34Intel® 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. 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

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.

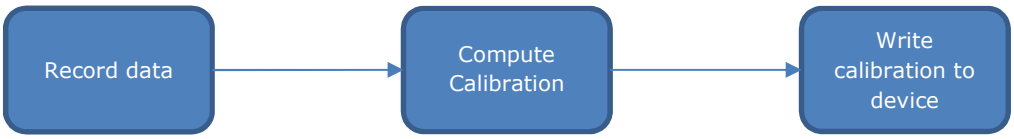

Figure 4-1 Calibration Process

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

Example output of the start of the script:

```
waiting for realsense device...
 Device PID:  0B64
```

```
 Device name:  Intel RealSense L515
 Serial number:  00000000f9340895
 Product Line:  L500
 Firmware version:  01.04.01.00
checking minimum firmware requirement ...
 firmware 01.04.01.00 passed check.
Start interactive mode:
FOUND GYRO with fps=100
FOUND ACCEL with fps=100
------------------------
*** Press ESC to Quit ***
------------------------
```

Note: On L515, the script will check FW to ensure it meet the minimum required FW 1.4.1.0. If not, it stops and prints a warning message out and asks user to upgrade firmware.

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.

The positions vary between D435i, D455 and L515 due to product design and IMU physical configuration in the product, but the goal in each position is to align the axis of the IMU with direction of gravity, and in the order described below. Each of these products is equipped with ¼-20 threaded tripod mount screw thread at the bottom of the device, so the calibration script and instruction below use it as assistance to the device physical orientation.

Real life pictures are used to better illustrate the exact position and orientation.

4.3.2.1 D435i and D455

Following instruction from calibration script to capture data with D435i device at positions below. D455 physical shape and USB port location is slightly different, but overall calibration procedure is similar.

4.3.2.1.1 Position #1 – Mounting screw pointing down, device 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 D45i and D455 Position #1

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.] Mounting screw pointing down device facing out

Status.collect_data[.....]]

Direction data collected.

4.3.2.1.2 Position #2 – Mounting screw pointing left, device facing out

With the camera facing the same direction as described in section 4.3.2.1.1, rotate the camera 90 degree about the camera's viewing directions such that ¼ - 20 threaded tripod mount pointing to the left.

Figure 4-3 D45i and D455 Position #2

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

4.3.2.1.3 Position #3 – Mounting screw pointing up, device facing out

From Position #2 rotate 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 D45i and D455 Position #3

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

4.3.2.1.4 Position #4 – Mounting screw pointing right, device facing out

From Position #3 rotate the camera an additional 90 degrees such that the ¼ - 20 threaded tripod mount pointing right.

Figure 4-5 D45i and D455 Position #4

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

4.3.2.1.5 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 D45i and D455 Position #5 viewing direction facing down

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

4.3.2.1.6 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 D45i and D455 Position #6 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.

4.3.2.2 L515

Following instruction from calibration script to capture data with L515 camera at positions below.

4.3.2.2.1 Position #1 – Mounting screw pointing down, device facing out

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

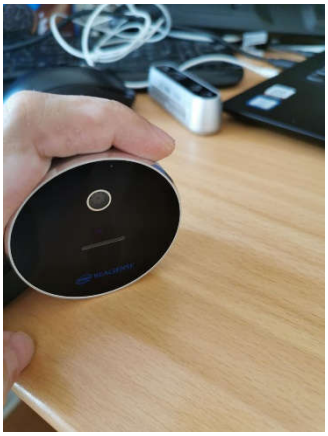

Figure 4-8 L515 Position #1 (Upright facing out position)

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.

4.3.2.2.2 Position #2 – Mounting screw pointing left, device facing out

With the camera facing the same direction as described in section 4.3.2.1.1, rotate the camera 90 degree about the camera's viewing directions such that device is oriented as below with the ¼ - 20 threaded tripod mount is now pointed to left direction.

Figure 4-9 L515 Position #2

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

4.3.2.2.3 Position #3 – Mounting screw pointing up, device facing out

From Position #2 rotate 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.

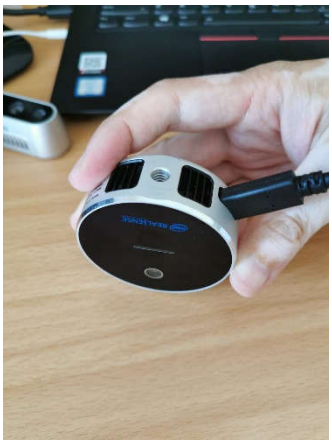

Figure 4-10 L515 Position #3 (Upside down facing out)

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

4.3.2.2.4 Position #4 – Mounting screw pointing right, device facing out

From Position #3 rotate the camera an additional 90 degrees such that the device is oriented as below with ¼ - 20 threaded tripod mount is now pointed upper left direction.

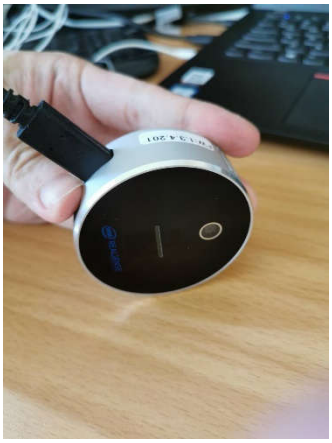

Figure 4-11 L515 Position #4

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

4.3.2.2.5 Position #5 – Viewing direction facing down

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

Figure 4-12 L515 Position #5 Viewing direction facing down

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

4.3.2.2.6 Position #6 – Viewing direction facing up

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

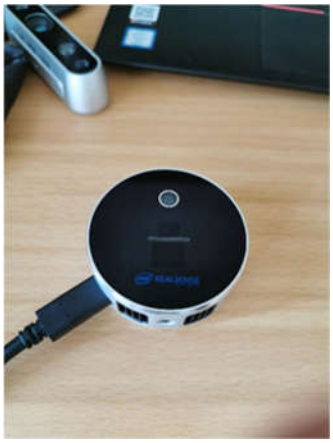

Figure 4-13 Position #6 Viewing direction facing down

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:

[0.0034026 0.00059176 0.00353117]
[1000 1000 1000 1000 1000 1000]
using 6000 measurements.
[[1.00538003 0.03177483 0.01375647]
[-0.02447432 1.0064362 0.00409543]
[-0.00509237 -0.00333411 1.00595617]
[0.00534571 -0.09900499 -0.01434132]]
residuals: [73.43844817 52.70612129 5.35963101]
rank: 4
singular: [438.25439734 435.97173412 433.29013118 77.44714542]
norm (raw data): 9.745653
norm (fixed data): 9.805321 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? (Y/N)Y
Writing calibration to device.

Done.