displayIMU User Guide

by
Simeon Symeonidis

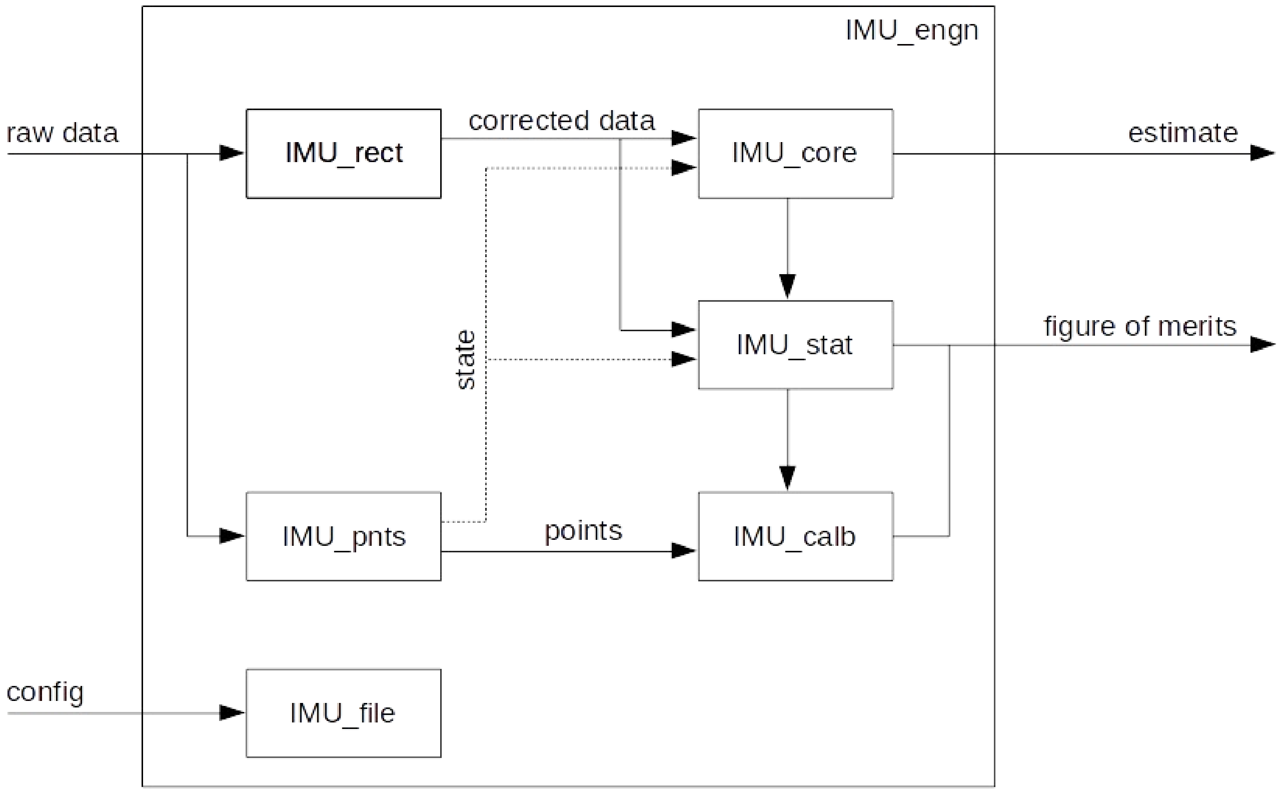
11/13/2018
revision 1.0

Overview

The displayIMU project contains the algorithm, software library, and tools necessary for the processing of Microelectromechanical Systems (MEMS) accelerometer, magnetometer, and gyroscope sensor data to estimate system orientation and translational acceleration. At its core is a optimized C library (Linux shared object file), which can be embedded into a senor or hosted by a server for off-line processing. A QT display tool is available to display incoming sensor data, tune the algorithm, manage config files, and view results. The QT display tool works with streaming UDP data or CSV files. Also available is a parser, which generates a CSV report given sensor data and config files (command line arguments). To aid future development, the Matlab source is included within the project. A common code directory contains functions that can aid in system integration and applications development. Last, testing code and guide lines are available for maintenance purposes. This user guide will provide information on all these project components.

IMU Core

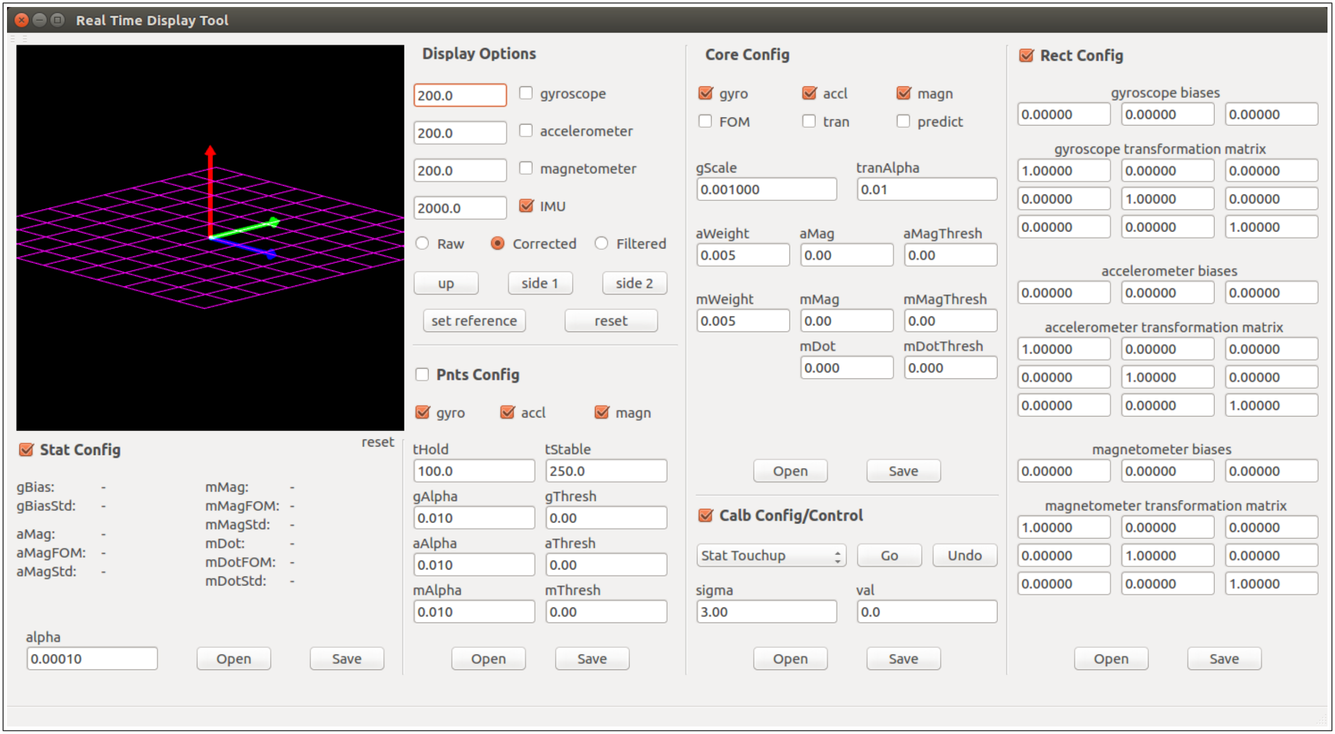
The core consists of the modules needed to calibrate, correct, and process sensor data. All the modules are integrated into the engine (IMU_engn), which provides asynchronous, non-blocking sensor/output interfaces, manages module configurations, and provides command and control. The algorithm guts, which performs filtering, generates estimates, and creates figure of merits data is done by IMU_core. Sensor correction, used to compensate for subtle response differences, is applied via IMU_rect. These coefficients can be generated by IMU_calb, which takes user-directed, multi-point calibration inputs, created by IMU_pnts, or continuous, background data generated by IMU_stat. Module configuration can by done via IMU_file, which reads/writes json files. This overview is capture in the block diagram below.



The top-level IMU core API is captured in IMU_engn.h header file. For more information on how to configure or general library use, see the IMU Core Reference Manual. Default configuration json files, which is necessary to properly use IMU_engn, are provided within the displayIMU/config directory.

QT Display Tool

A display tool is available for displaying data, parameter tuning, managing config files, and evaluating system performance. After compilation (see top-level README), the displayIMU binary can be found in the displayIMU/bin folder. To execute, type ./displayIMU <IMU_engn.conf> <sensor_data.csv>. The second command line argument is used to populate configuration parameter fields for each module (see below). These fields can be modified and saved within the tool (be sure that the filename matches files listed in IMU_engn config). If not specified, config/default_engn.json will be assumed. The third command line argument is used to specify a Comma Separated Value (CSV) source file. The sensor format (one datum per line) is: “sensor type, time stamp (10us LSB), val x, val y, val z”, where type 1 is gyroscope data, type 2 is accelerometer data, and type 3 is magnetometer data. A synchronous format is also available and its format is: “0, time stamp (10us LSV), gyro x, gyro y, gyro z, accl x, accl y, accl z, magn x, magn y, magn z”. displayIMU/stim contains synthetic data for reference and test purposes. If this command line argument is not specified, a UDP port is opened and streaming data, whose format matches the CSV file, can be processed. It is recommend to use this streaming interface to evaluate this algorithm on live data before integration into the final system. The port number, along with other display tool parameters are captured in config/displayIMU.json.



Display Options

200.0 ☐ gyroscope
200.0 ☐ accelerometer
200.0 ☐ magnetometer
2000.0 ✓ IMU
Raw [✓ Corrected] Filtered

☐ Pnts Config

✓ gyro ✓ accl ✓ magn

tHold	tStable
100.0	250.0
gAlpha	gThresh
0.010	0.00
aAlpha	aThresh
0.010	0.00
mAlpha	mThresh
0.010	0.00

✓ Stat Config reset

gBias:	-	mMag:	-
gBiasStd:	-	mMagFOM:	-
		mMagStd:	-
aMag:	-	mDot:	-
aMagFOM:	-	mDotFOM:	-
aMagStd:	-	mDotStd:	-

alpha 0.00010

Core Config

✓ gyro ✓ accl ✓ magn
☐ FOM ☐ tran ☐ predict

gScale	tranAlpha	
0.001000	0.01	
aWeight	aMag	aMagThresh
0.005	0.00	0.00
mWeight	mMag	mMagThresh
0.005	0.00	0.00
	mDot	mDotThresh
	0.000	0.000

✓ Calb Config/Control

Stat Touchup

sigma 3.00 val 0.0

✓ Rect Config

gyroscope biases

0.00000	0.00000	0.00000

gyroscope transformation matrix

1.00000	0.00000	0.00000
0.00000	1.00000	0.00000
0.00000	0.00000	1.00000

accelerometer biases

0.00000	0.00000	0.00000

accelerometer transformation matrix

1.00000	0.00000	0.00000
0.00000	1.00000	0.00000
0.00000	0.00000	1.00000

magnetometer biases

0.00000	0.00000	0.00000

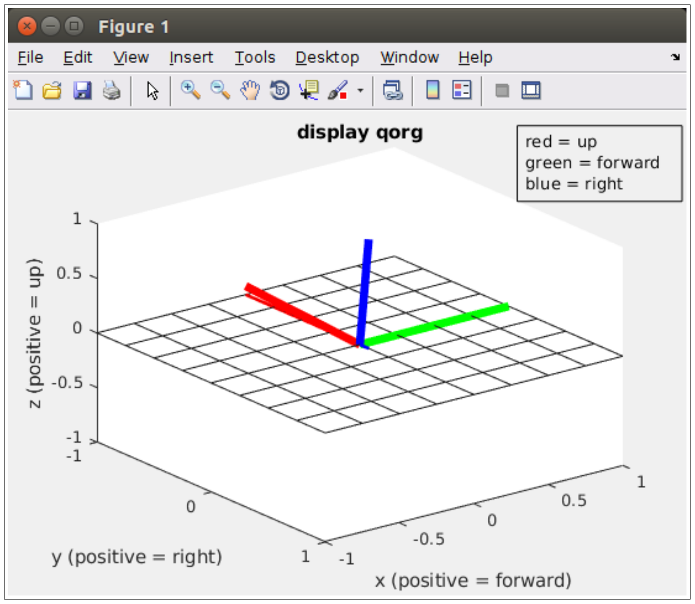
magnetometer transformation matrix

1.00000	0.00000	0.00000
0.00000	1.00000	0.00000
0.00000	0.00000	1.00000

CSV Parser

To support server-hosted batch processing, a CSV parsing tool is available. After compilation (see top-level README), the binary can be found in the displayIMU/bin folder. To execute, type ./csvProcess <IMU_engn.conf> <sensor_data.csv>. See the previous section (displayIMU) for specifics on each of these command line arguments. By default, the CSV estimation report will be directed to the console. To save, redirect this stream using the ">" operator to an output file. The display/results directory was created for storing these results. The output file format is determined by <IMU_engn.conf>. Estimates will be synchronized with input data (each datum will generate a corresponding line in the CSV report). The first four values will contain the system orientation (represented as a quaternion). If "isTran" is enabled, the next three values will capture the translational acceleration estimate. If "isRef" is enabled, the next four values will be the rotated system orientation (represented as a quaternion). The rotational reference is "qRef". If "isAng" is enabled, the last three values will be the system orientation in Euler angles, i.e. yaw, pitch, and roll.

To view the csv data within Matlab, the display_qorg.m function is available. For general use of both display_qorg.m and csvParse, see displayIMU/csv/run_stim.m and display/csv/run_stim.sh repectively. This Matlab and shell script generates a report using synthetic data and creates the display shown. For the initial release, display_qorg.m only supports formats with "isTran", "isRef", and "isAng" disabled.



Matlab

The displayIMU/matlab directory hosts the algorithm core source. Many of these functions have been ported to C and can be found in either displayIMU/core/IMU_math.h or displayIMU/core/IMU_math.c. Also contained in this folder is top-level test scripts, which include applyGyroTest.m, applyAcclTest.m, and applyMagnTest.m scripts. In addition to producing visuals and figure-of-merits, these top-level scripts can generate synthetic sensor data in the form of "displayIMU" and "csvProcess" compatible csv files. To create them, ensure that the "csv_enable" variable is set to true and run. The location and name of the output file is specified in the variable "csv_filename" and is usually set to create a file somewhere in displayIMU/stim.

Common

The common directory hosts functions that can aid in system integration and applications development. dataIF.h and dataIF.c has CSV and UDP parsing data, and is used by “displayIMU” and “csvProcess”. It is recommended that new sensor data interfaces to be included here to insure maximum compatibility between applications and tools. IMU_util.h contains utility functions associated with the IMU library core, i.e. error trapping and status reporting. As the displayIMU project evolves, this directory should capture reusable code to aid application develop and system integration.

Testing

The displayIMU/test folder contains the unit test suite. It is recommended to run them (see top-level Makefile) as part of regression testing. After invoking “make test”, the results for each test need to be visually inspected for “pass: <test_name>”. In the case of “error” or “fail” message, it is recommended to rerun the test with a larger msg_delay time (see test_utils.h). If problems persist, change directories to displayIMU/test/bin and execute the same test. Compare that verbose output against the source code (where the checks are being made). In some cases, unit test pass/fail comparisons were created from running Matlab mode and in the case of algorithm updates need be modified.