Getting started with MotionGT gyroscope temperature calibration library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionGT is a middleware library part of X-CUBE-MEMS1 software and runs on STM32. It provides real-time temperature-compensated gyroscope calibration parameters across the temperature (angular zero-rate level coefficients offset) used to correct gyroscope data.

This library is intended to work with STMicroelectronics MEMS sensors only.

The algorithm is provided in static library format and is designed to be used on STM32 microcontrollers based on the ARM® Cortex®-M0+, ARM® Cortex®-M3, or ARM® Cortex®-M4 architecture.

It is built on top of STM32Cube software technology that eases portability across different STM32 microcontrollers.

The software comes with a sample implementation running on X-NUCLEO-IKS01A3 expansion board on a NUCLEO-F401RE, NUCLEO-L476RG, NUCLEO-L152RE , or NUCLEO-L073RZ development board.

1 Acronyms and abbreviations

Table 1. List of acronyms

Acronym	Description
API	Application programming interface
BSP	Board support package
GUI	Graphical user interface
HAL	Hardware abstraction layer
IDE	Integrated development environment

2 MotionGT middleware library for X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionGT overview

The MotionGT library expands the functionality of the X-CUBE-MEMS1 software.

The gyroscope sensor can have a temperature-dependent offset, which can cause problems when using the gyroscope output data after one-point calibration. The MotionGT library is able to minimize the offset drift due to temperature and solve this issue.

The library acquires temperature and offset/calibration data from the gyroscope calibration library (MotionGC) and calculates the angular zero-rate level coefficient (offset) at a given temperature for each axis. The angular zero-rate level coefficients are subsequently used to compensate for raw data coming from the gyroscope with different temperature. The library assumes the linear or zero order relation between temperature and gyroscope bias with some hysteresis.

The library is designed for ST MEMS sensors only. Functionality and performance when using other MEMS sensors are not analyzed and can be significantly different from what described in the document.

Sample implementation is available for X-NUCLEO-IKS01A3 expansion boards, mounted on a NUCLEO-F401RE, NUCLEO-L476RG, NUCLEO-L152RE, or NUCLEO-L073RZ development boards.

2.2 MotionGT library

Technical information fully describing the functions and parameters of the MotionGT APIs can be found in the MotionGT_Package.chm compiled HTML file located in the documentation folder.

2.2.1 MotionGT library description

The MotionGT gyroscope calibration library manages data acquired from temperature and bias/offset from the gyroscope calibration library (MotionGC); it features:

- The temperature dependent angular zero-rate level (offset) compensation
- Handle hysteresis drift up to 1 deg/s
- Handle temperature slope up to 40 mdps/^{o}C
- Able to calibrate with 8-10 ^{o}C in the case of low numbers of outlier.
- Rejection of outlier.
- Resources requirements:
  - Cortex-M0+: 5.9 KB of code and 3.5 KB of data memory
  - Cortex-M3: 4.0 KB of code and 3.4 KB of data memory
  - Cortex-M4: 4.4 KB of code and 3.4 KB of data memory
  - Cortex-M7: 3.9 KB of code and 3.4 KB of data memory
- Available for Arm Cortex-M0+, Cortex-M3, Cortex-M4, and Cortex-M7 architectures.

2.2.2 MotionGT APIs

The MotionGT library APIs are:

- `uint8_t MotionGT_GetLibVersion(char *version)`
  - Retrieves the version of the library
  - `*version` is a pointer to an array of 35 characters
  - Returns the number of characters in the version string
- `void MotionGT_Initialize(void)`
  - Performs MotionGT library initialization and setup of the internal mechanism

Note: This function must be called before using the motionGT library

  - The CRC module in STM32 microcontroller (in RCC peripheral clock enable register) has to be enabled before using the library for STM32

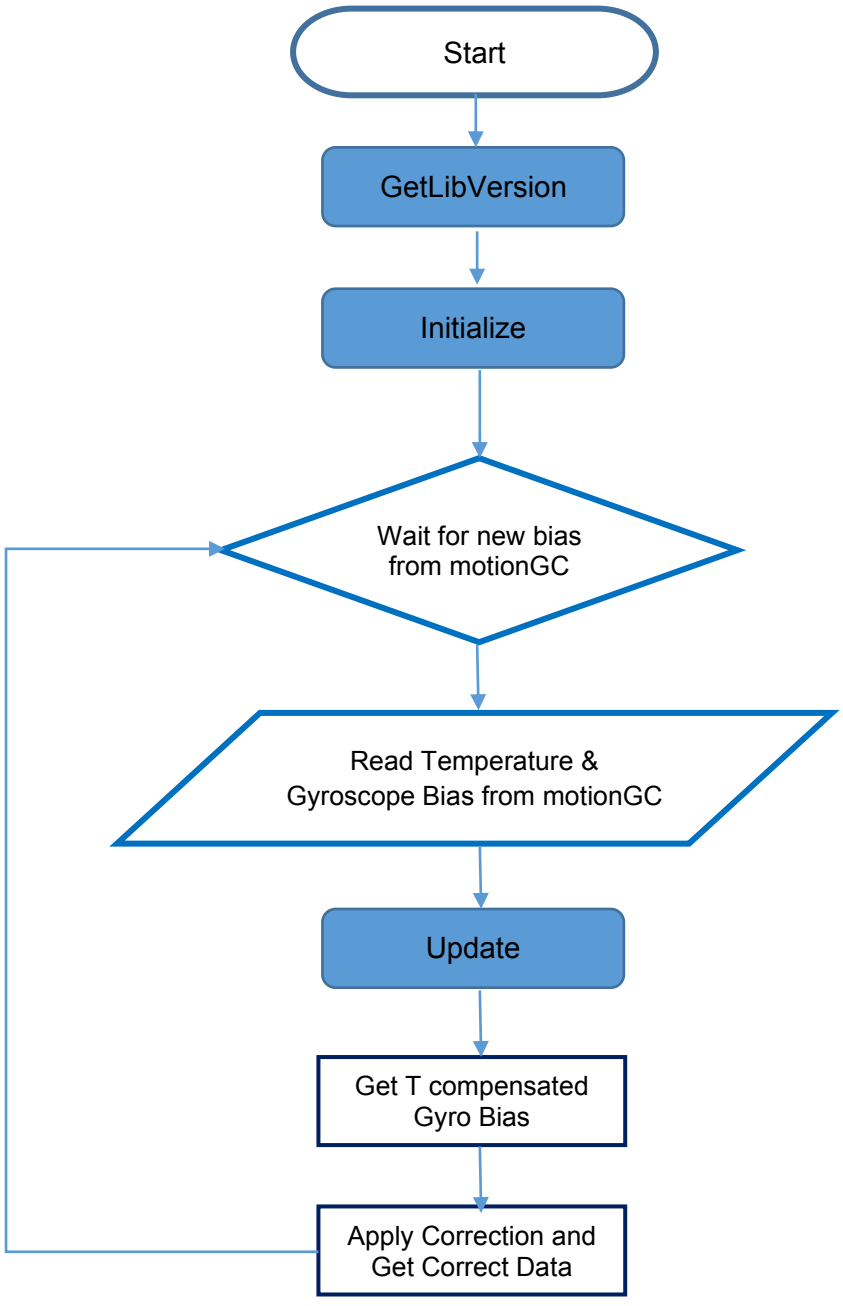
- `void MotionGT_Update(MGT_input_t *data_in, int *bias_update):`
  - Runs the temperature compensated gyroscope bias calibration algorithm and returns acknowledgment of model estimation
  - `*data_in` parameter is a pointer to a structure with input data
  - The parameters for the structure type `MGT_input_t` are the following:
    - `GyroBiasX` is gyroscope bias/offset along X axis in dps
    - `GyroBiasY` is gyroscope bias/offset along Y axis in dps
    - `GyroBiasZ` is gyroscope bias/offset along Z axis in dps
    - `Temp` is temperature of gyroscope in ^{o}C
    - `Timestamp` is time of current sample in ms
  - `*bias_update` is a pointer to an integer set to 1 if the temperature model is estimated, 0 otherwise. The value will remain 1 if model is estimated in past.

Note: This function has to be called when new bias is available by motionGC library with temperature and time.

- `void MotionGT_GetCompensatedBias(float temp, MGT_output_t *gyro_bias):`
  - Retrieves the gyroscope bias with temperature compensation parameters
  - `*gyro_bias structure` parameter is a pointer to a structure with gyroscope bias
  - The parameters for the structure type MGT_output_t are the following:
    - `GyroBiasX` is gyroscope bias/offset along X axis in dps
    - `GyroBiasY` is gyroscope bias/offset along Y axis in dps
    - `GyroBiasZ` is gyroscope bias/offset along Z axis in dps
- `void MotionGT_GetCalState(MGT_state_t *state)`
  - Gets the gyroscope compensation parameters with temperature slope, quality and last updated time, the output can be used to reinitialize the model next time for faster convergence. The API to reinitialize the model is MotionGT_SetCalState()
  - `*state` parameter is a pointer to a structure with temperature model
  - The parameters for the structure type `MGT_input_t` are the following:
    - `GyroBias[3]` is an array of gyroscope bias/offset at 0°C in dps
    - `Scale[3]` is an array of gyroscope bias/offset slope wrt temperature in dps/°C
    - `Quality` is the quality of model. The parameters cannot be used directly outside library
    - `ModelEstimated` is a flag to indicated if model is estimated or not
    - `LastUpdatedTime` is time of last update of model in ms
- `char MotionGT_SetCalState(const MGT_state_t *state)`
  - Sets the gyroscope compensation model for faster convergence. The API should be called just after MotionGT_Initialization()
  - `*state` parameter is a pointer to a structure with temperature model
  - `char` the return parameter and output will be 1 is model is accepted by library and 0 otherwise

2.2.3 API flow chart

Figure 1. **API flow chart diagram**



2.2.4 Demo code

The following is an example of the demonstration code that reads data from the gyroscope sensor and from the accelerometer sensor and calculates compensated data.

```
[…]

#define VERSION_STR_LENG               35

/*** Initialization ***/
char lib_version[VERSION_STR_LENG];
float sample_freq = SAMPLE_FREQUENCY;
MGT_output output_data;
int tModelEstimated=0;

/* T Compensated Gyroscope calibration API

initialization function */
MotionGC_Initialize(MGC_MCU_STM32, &sample_freq);
MotionGT_Initialize();
/* Optional: Get version */
MotionGT_GetLibVersion(lib_version);
/* Optional: Set sample frequency */
sample_freq = SAMPLE_FREQUENCY;
MotionGC_SetFrequency(&sample_freq);
[…]

/*** Using gyroscope calibration algorithm ***/
Timer_OR_DataRate_Interrupt_Handler()
{
  MGC_input_t data_in;
  MGC_output_t data_out;
  int bias_update;
  float gyro_cal_x, gyro_cal_y, gyro_cal_z;
  MGT_output_t tData_out;

/* Get acceleration X/Y/Z in g */
  MEMS_Read_AccValue(data_in.Acc[0], data_in.Acc[1], data_in.Acc[2]);
  /* Get angular rate X/Y/Z in dps */
  MEMS_Read_GyroValue(data_in.Gyro[0], data_in.Gyro[1], data_in.Gyro[2]);
/* Gyroscope calibration algorithm update */
  MotionGC_Update(&data_in, &data_out,
  &bias_update); /* Apply correction */
  if(bias_update)
{
  MGT_input inputData;
  inputData.TimeStamp = t_ms;
  inputData.GyroBiasX|Y|Z=data_out.GyroBiasX|Y|Z;
  inputData.Temp = temp_DegC;
  MotionGT_Update(&inputData, & tModelEstimated);
  }
  if(tModelEstimated)
  MotionGT_GetCompensatedBias(temp_DegC, &tData_out);
  gyro_cal_x =(data_in.Gyro[0]- tDdata_out.GyroBiasX);
  gyro_cal_y =(data_in.Gyro[1]- tData_out.GyroBiasY);
  gyro_cal_z =(data_in.Gyro[2]- tData_out.GyroBiasZ);
}
```

2.2.5 The calibration process

The calibration process does not need any sensor motion as the offset coefficient is calculated and updated when the temperature varies more than 8-10°C.

2.2.6 Algorithm performance

Figure 2 is showing fitting results on gyro calibration with reasonable accuracy vs temperature variation. The model is able to handle small jitter, outlier (removed) and able to produce reliable temperature slope.

Figure 2. **Temperature vs gyro bias**

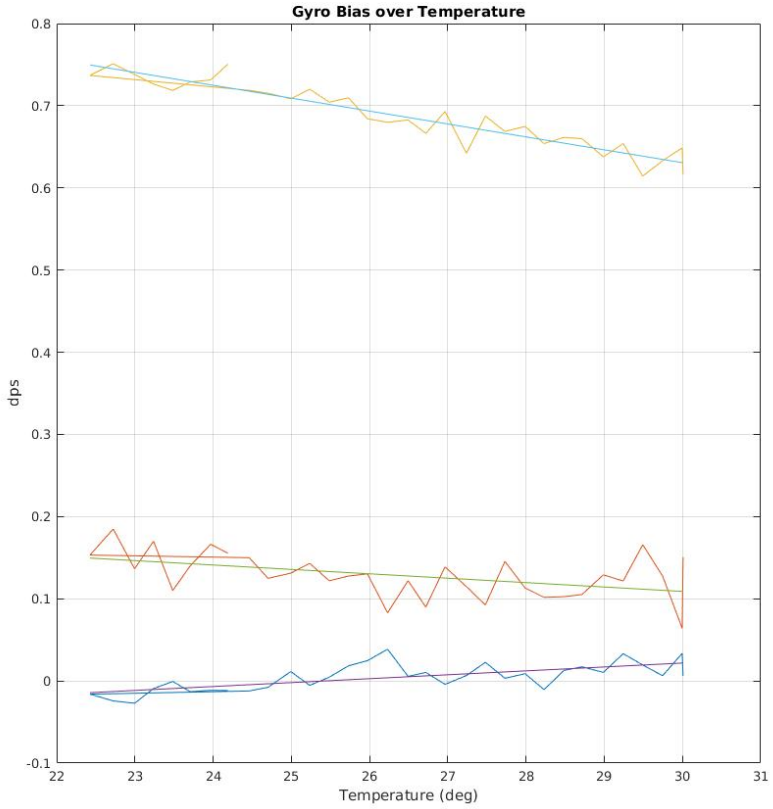
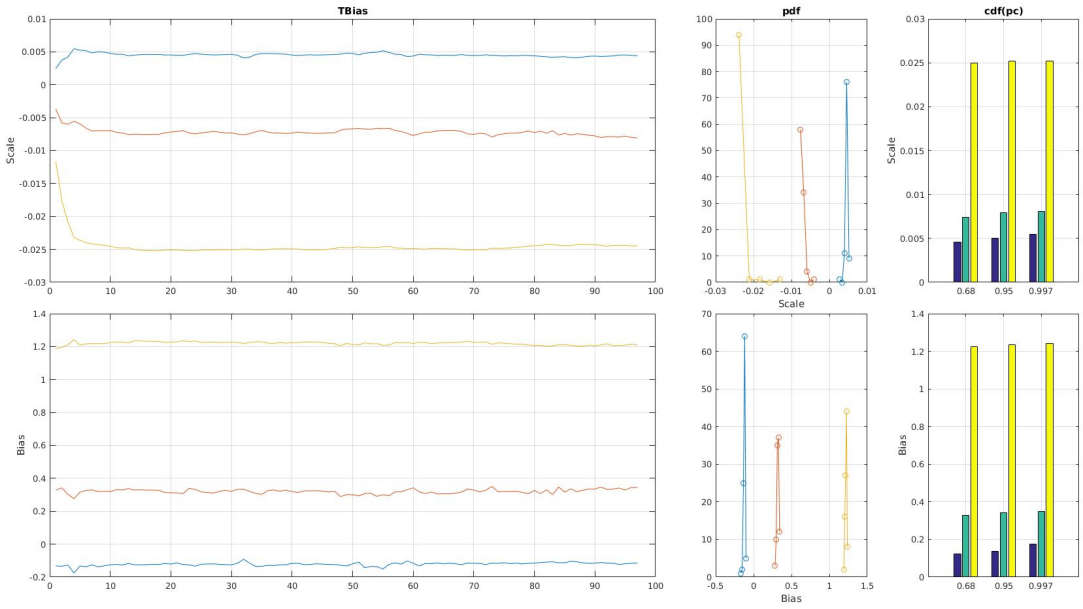


Figure 3. Overall statistics summarize the overall performance of algorithm on a normal test run. The estimation of slope and bias improve overall multiple run.

Figure 3. **Overall statistics**



3 References

All of the following resources are freely available on www.st.com.

1. UM1859: Getting started with the X-CUBE-MEMS1 motion MEMS and environmental sensor software expansion for STM32Cube
2. UM1724: STM32 Nucleo-64 boards (MB1136)
3. UM2128: Getting started with Unicleo-GUI for motion MEMS and environmental sensor software expansion for STM32Cube

Revision history

Table 2. **Document revision history**

Date	Version	Changes
24-Aug-2023	1	Initial release.

Contents

List of tables

List of figures