VnV Presentation

Attitude Check:
An IMU-based Attitude Estimator

Adrian Sochaniwsky

Software Engineering MASc. Student
McMaster University

February 11, 2024

Quick Re-introduction

Attitude estimation: Given noisy sensor measurements, determine the orientation of a body relative to another frame (i.e. the Earth)

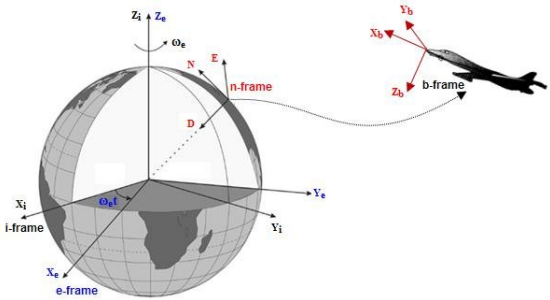

Figure 1: Diagram.

Objectives

In scope:

- Build confidence in the software correctness.
- Demonstrate the software's ability to accurately estimate orientation.

Not in scope:

- Demonstration of adequate usability.
- Verification of external libraries.

Table 1: Table of Abbreviations and Acronyms

symbol	description
accel	Accelerometer
gyro	Gyroscope
IMU	Inertial Measurement Unit
mag	Magnetometer
SRS	Software Requirements Specification
VnV	Verification and Validation

SRS Verification Plan

The SRS will be verified via feedback from a domain expert and secondary reviewers.

Checklist:

- ☐ Problem Statement: confirm it is a valid statement.
- ☐ Goal Statement: confirm it is a valid goal.
- ☐ Physical system: check for missing descriptions.
- ☐ Assumptions: confirm assumptions are specific and meaningful.
- ☐ Theory: confirm instance models and all required theory is logical and complete enough for development.
- ☐ Requirements: confirm that the requirements are reasonably complete for testers and clients.

Design Verification Plan

To verify the design, a domain expert and secondary reviewer will use the following checklist:

- ☐ Inspect the design document for the following items:
  - Complete internal cross-references.
  - External libraries and hardware interface definitions.
  - Consistent use of conventions and abbreviations.
  - Sufficient detail for implementation
- ☐ Create a high-level diagram that represents the relationships between modules. This is intended to find gaps in the design.

Verification and Validation Plan Verification Plan

Checklist for reviewers:

- ☐ Confirm all sections contain a verification checklist or description.
- ☐ Inspect system-level test cases for coverage of SRS requirements and goal statements.
- ☐ Inspect all test cases for complete definitions.

Implementation Verification Plan

Static Testing

Cppcheck is a static code checking tool that will be employed to analyze the code for undefined behaviour and poor design constructs.

Dynamic Testing

Dynamic system and unit level tests will be used to verify the implementation.

Automated Testing and Verification Tools

- **Cmake** is a build system tool that will be used to simplify building and testing.
- **GoogleTest** is a unit testing framework that support c++ code. It also includes a mocking framework.
- **GCOV, LCOV** are tools for calculating and displaying unit test coverage metrics.
- **ValGrind** is a suite of tools for memory and performance profiling. It will be used to check for memory leaks and efficient memory utilization.
- **Uncrustify** is a tool that applies code formatting rules based on a configuration file. It will be used before every commit to the repo.
- **GitHub CI** workflow will automate regression tests and checks that Attitude Check builds are passing before a PR is merged into a protected branch.
- **Docker** is a mechanism to containerize applications. It will be used to ensure installability and usability of the project.

Software Validation

- Software validation will consist of benchmarking against existing algorithms
- The Root Mean Squared Error (RMSE) of Attitude Check should lie within ϵ of the RMSE of attitude estimators implemented by TBD
- The data will come from the BROAD dataset `https://github.com/dlaidig/broad`.

T1: **Input/Output Test**
Control: Automatic
Initial State: Uninitialized

Step	Input	Output
1	Initialize with: $\Delta t = 1.0, {}^{E}\mathbf{b} = [1.0, 1.0, 1.0]$, $\gamma = 0.6$, outputType = Quat	Get params method matches the inputs.
2	One set of [accel, gyro, mag] measurements.	Orientation output should match output format from above
3	Initialize with: $\Delta t = 1.0, {}^{E}\mathbf{b} = [1.0, 1.0, 1.0]$, $\gamma = 0.6$, outputType = Rot	Get params method matches the inputs.
4	One set of [accel, gyro, mag] measurements.	Orientation output should match output format from above
5	Initialize with: $\Delta t = 1.0, {}^{E}\mathbf{b} = [1.0, 1.0, 1.0]$, $\gamma = 0.6$, outputType = Euler	Get params method matches the inputs.
6	One set of [accel, gyro, mag] measurements.	Orientation output should match output format from above

Test Case Derivation: N/A
How test will be performed: At each step, apply the inputs and assert the output.

T2: **Orientation with Magnetometer**
Control: Automatic
Initial State: Uninitialized

Step	Input	Output
1	One complete set of valid initialization inputs.	-
2	[accel, gyro, mag] measurements.	Calculate RMSE of output vs. ground truth
3	Repeat step 2 for all measurements in dataset	-
4	Calculate average RMSE	Assert average error is within the tolerance.

Test Case Derivation: Benchmark dataset with labelled ground truth orientation for each set of sensor measurements.
How test will be performed: At each step, apply the inputs and assert the output.

Accuracy

T3: **test-a1**
Type: Automatic
This test refers to Section XYZ.

Usability

T4: **test-u1**
Type: Manual
Initial State: N/A
Input/Condition: N/A
Output/Result: Successful integration with existing code.
How test will be performed: A manual inspection of the code, specifically the header file. Can a developer read the documention/header file and correctly interact with the functions/objects?This test passes if this is true.

Unit Testing

- 1 test per input, alternating which one is invalid
- All 0/Null input test
- If using pointers, test Null pointer check
- Test const pointers are not modified
- ...

The End

Questions?