Project Title: System Verification and Validation Plan for Attitude Check

Adrian Sochaniwsky

February 9, 2024

Revision History

Date	Version	Notes
2024/02/13	1.0	Initial draft

Contents

List of Tables

List of Figures

1 Symbols, Abbreviations, and Acronyms

For symbols and units see the Section 1 of the SRS (Sochaniwsky, 2024). Table 1 defines the abbreviations and acronyms used in this document.

Table 1: Table of Abbreviations and Acronyms

symbol	**description**
IMU	Inertial Measurement Unit
SRS	Software Requirements Specification
VnV	Verification and Validation

2 Introduction

A verification and validation (VnV) plan is a document that describes the objectives, scope, methods, and criteria for verifying and validating a software product. Verification is the process of checking whether the software meets the specified requirements and design specifications. Validation is the process of checking whether the software meets the user's needs and expectations. A VnV plan helps to ensure the quality, reliability, and functionality of the software, as well as to identify and correct any defects or errors before the software is released or deployed.

This document details the plan for verification and validation of Attitude Check. Section 3 provides an overview and objectives of this document. Section 4 discusses the methods of achieving the objectives. Section 5 covers the test descriptions.

3 General Information

3.1 Summary

Attitude Check is an IMU-based attitude estimation algorithm. It consumes sensor data and produces and estimate of the current orientation of the sensor relative to the Earth.

3.2 Objectives

The objectives of this Verification and Validation plan are the following:

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

Objectives that are not included in the scope of this document:

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

It will be assumed that the documentation of Attitude Check is adequate to facilitate adequate usability for developers wishing to use it. Furthermore, it is assumed that external libraries have been independently verified and validated.

3.3 Relevant Documentation

See the Software Requirements Specification (Sochaniwsky, 2024) for Attitude Check, it details the goals, requirements, assumptions, and theory of the software.

4 Plan

This section will detail the plan for the verification of the documentation and software for Attitude Check. The primary items that will be verified are: SRS, design, VnV, implementation.

4.1 SRS Verification Plan

The SRS (Sochaniwsky, 2024) will be verified via feedback from a domain expert and secondary reviewers. The following is a checklist for SRS review:

- ☐ 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.

4.2 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:
  -
- ☐ Create a high-level diagram that represents the relationships between modules. This is intended to find gaps in the design.

4.3 Verification and Validation Plan Verification Plan

This VnV plan will be verified via inspection from a domain expert and a secondary review. The inspection will look for the following items:

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

4.4 Implementation Verification Plan

4.5 Automated Testing and Verification Tools

- GoogleTest https://github.com/google/googletest
- Gcov, lCov for calculating and displaying test coverage
- Valgring/memcheck for RAM usage and checking for memory leaks
- Cmake for building
- Uncrustify for automatic formatting
- CppCheck for static analysis
- Github PR checklists
- Github CI workflow for automated regression test and checking thats builds pass before entering the main branch
- Docker

4.6 Software Validation Plan

For the scope of this project, 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 Laidig (2023). The data will come from the BROAD dataset (Laidig et al., 2021).

5 System Test Description

5.1 Tests for Functional Requirements

5.1.1 Inputs and Outputs

This section covers requirements R1 and R4 of the SRS. This includes the input and output requirements for Attitude Check. For input constraints see Section 4.2.8 of the SRS.

1. **Input/Output Test**

   Control: Automatic

   Initial State: Uninitialized

   Input: One complete valid set of initialization inputs, one set of measurements from: [Accelerometer, Gyroscope, Magnetometer].

   Output: Attitude Estimator (AE) object is initialized. Get parameters method returns the same parameters that were given. Orientation output is in the expected format.

   Test Case Derivation: N/A

   How test will be performed: Test fixture will create and initialize the AE.

2. Periodic Input Test

   Initial State: Initialized

   Input:

   Output:

   Test Case Derivation:

   How test will be performed:

5.1.2 Calculations

1. Periodic Input Test

Initial State:

Input:

Output:

Test Case Derivation:

How test will be performed:

Note: See Section 4.2.8 of the SRS for input constraints.

5.2 Tests for Nonfunctional Requirements

5.2.1 Accuracy

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

5.2.2 Portability

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

5.3 Traceability Between Test Cases and Requirements

6 Unit Test Description

6.1 Unit Testing Scope

6.2 Tests for Functional Requirements

6.2.1 Module 1

1. **Input Test**

   Initial State: Uninitialized

   Input:

   $$\Delta t = 1.0$$
   $${}^{E}\mathbf{b} = [1.0, 1.0, 1.0]$$
   $$\gamma = 0.6$$
   $$\text{outputTypeEnum} = \text{Quaternion}$$

   One set of measurements from: [Accelerometer, Gyroscope, Magnetometer]

   Output: Attitude Estimator (AE) object is initialized. Get parameters method returns the same parameters that were given.

   Test Case Derivation: N/A

   How test will be performed: Test fixture will create and initialize the AE.

   Note: See Section 4.2.8 of the SRS for input constraints.

2. Periodic Input Test

   Initial State: Initialized

   Input:

   Output:

   Test Case Derivation:

How test will be performed:

Note: See Section 4.2.8 of the SRS for input constraints.

3. test-id1

   Type:

   Initial State:

   Input:

   Output:

   Test Case Derivation:

   How test will be performed:

4. test-id2

   Type:

   Initial State:

   Input:

   Output:

   Test Case Derivation:

   How test will be performed:

5. ...

6.2.2 Module 2

...

6.3 Tests for Nonfunctional Requirements

6.3.1 Module ?

1. test-id1

   Type:

Initial State:

Input/Condition:

Output/Result:

How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

6.3.2 Module ?

...

6.4 Traceability Between Test Cases and Modules

References

Daniel Laidig. Berlin Robust Orientation Estimation Assessment Dataset (BROAD). https://github.com/dlaidig/broad/tree/main, 2023.

Daniel Laidig, Marco Caruso, Andrea Cereatti, and Thomas Seel. Broad—a benchmark for robust inertial orientation estimation. *Data*, 6(7), 2021. ISSN 2306-5729. doi: 10.3390/data6070072. URL https://www.mdpi.com/2306-5729/6/7/72.

Adrian Sochaniwsky. System requirements specification for attitude check. https://github.com/adrian-soch/attitude_check/blob/7c8cd7387ab120dd1267b56902499314c696e42b/docs/SRS/SRS.pdf, 2024.

7 Appendix

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.