

System Verification and Validation Plan for 2D Localizer

Aliyah Jimoh

February 23, 2025

Revision History

| Date | Version | Notes |
|------------|---------|-------|
| 24/02/2025 | 1.0 | Notes |
| Date 2 | 1.1 | Notes |

[The intention of the VnV plan is to increase confidence in the software. However, this does not mean listing every verification and validation technique that has ever been devised. The VnV plan should also be a **feasible** plan. Execution of the plan should be possible with the time and team available. If the full plan cannot be completed during the time available, it can either be modified to “fake it”, or a better solution is to add a section describing what work has been completed and what work is still planned for the future. —SS]

[The VnV plan is typically started after the requirements stage, but before the design stage. This means that the sections related to unit testing cannot initially be completed. The sections will be filled in after the design stage is complete. the final version of the VnV plan should have all sections filled in. —SS]

Contents

| | | |
|----------|--|-----------|
| 1 | Symbols, Abbreviations, and Acronyms | iv |
| 2 | General Information | 1 |
| 2.1 | Summary | 1 |
| 2.2 | Objectives | 1 |
| 2.3 | Challenge Level and Extras | 1 |
| 2.4 | Relevant Documentation | 1 |
| 3 | Plan | 2 |
| 3.1 | Verification and Validation Team | 2 |
| 3.2 | SRS Verification Plan | 2 |
| 3.3 | Design Verification Plan | 3 |
| 3.4 | Verification and Validation Plan Verification Plan | 3 |
| 3.5 | Implementation Verification Plan | 3 |
| 3.6 | Automated Testing and Verification Tools | 3 |
| 3.7 | Software Validation Plan | 4 |
| 4 | System Tests | 4 |
| 4.1 | Tests for Functional Requirements | 4 |
| 4.1.1 | Area of Testing1 | 5 |
| 4.1.2 | Area of Testing2 | 5 |
| 4.2 | Tests for Nonfunctional Requirements | 6 |
| 4.2.1 | Area of Testing1 | 6 |
| 4.2.2 | Area of Testing2 | 7 |
| 4.3 | Traceability Between Test Cases and Requirements | 7 |
| 5 | Unit Test Description | 7 |
| 5.1 | Unit Testing Scope | 7 |
| 5.2 | Tests for Functional Requirements | 8 |
| 5.2.1 | Module 1 | 8 |
| 5.2.2 | Module 2 | 9 |
| 5.3 | Tests for Nonfunctional Requirements | 9 |
| 5.3.1 | Module ? | 9 |
| 5.3.2 | Module ? | 9 |
| 5.4 | Traceability Between Test Cases and Modules | 10 |

| | | |
|----------|---------------------------------------|-----------|
| 6 | Appendix | 11 |
| 6.1 | Symbolic Parameters | 11 |
| 6.2 | Usability Survey Questions? | 11 |

List of Tables

| | | |
|---|--|---|
| 1 | Verification and Validation Team | 2 |
| | [Remove this section if it isn't needed —SS] | |

List of Figures

[Remove this section if it isn't needed —SS]

1 Symbols, Abbreviations, and Acronyms

| Symbol | Description |
|--------------|-------------------------------------|
| 2D | Two-Dimensional |
| 2D Localizer | 2D Localization Solution |
| A | Assumption |
| CRLB | Cramér-Rao Lower Bound |
| DD | Data Definition |
| FIM | Fisher Information Matrix |
| FM | Fiducial Marker |
| GD | General Definition |
| GS | Goal Statement |
| MG | Module Guide |
| MIS | Module Interface Specification |
| MLE | Maximum Likelihood Estimation |
| PDF | Probability Density Function |
| PS | Physical System Description |
| R | Requirement |
| SE(2) | Special Euclidean Group in 2D |
| SRS | Software Requirements Specification |
| T | Test |
| TM | Theoretical Model |
| UC | Unlikely Change |
| VnV | Verification and Validation |

For a full list of symbols, abbreviations, and acronyms used, refer to section 1 in the [SRS](#) document.

This document shows the verification and validation plan of the 2D Localizer program. This plan starts with general information that talks about 2D Localizer in section 2. The plan and system tests involved with this software is explained in sections 3 and 4.

2 General Information

2.1 Summary

This document examines the verification and validation plan of 2D Localizer. This software is used to help accurately locate mobile robots in a provided map given the measurements and coordinates of the sensors and fiducial markers (FMs).

2.2 Objectives

The objective of this plan is to validate the accuracy of this program to build confidence in the software correctness. This plan also aims to satisfy all the requirements the System Requirements Specification ([SRS](#)) document outlined.

2.3 Challenge Level and Extras

This system has an advanced research level which can be seen from the implementation and the topic. Setting up the robot's movement, accurately displaying the trajectory, coordinating the sensors' measurements, and finding a way to animate the output would definitely add difficulty to this system, however, this is not a niche topic meaning that there are papers or libraries available to draw inspiration from.

2.4 Relevant Documentation

The documentation relevant to the 2D Localizer includes the Problem Statement since it explains the proposed software, the [SRS](#) which talks about the requirements needed to properly use the system, the Verification and Validation ([VnV](#)) report that goes through the tests and plans for the system, and the Module Guide ([MG](#)) and Module Interface Specification ([MIS](#)) for the design.

3 Plan

This section describes the test made for the 2D Localizer system. This begins by mentioning the VnV team in section 3.1, then followed by the SRS verification plan in 3.2, the design verification plan in section 3.3, the VnV plan verification plan in section 3.4, the implementation verification plan in section 3.5, the automated testing and verification tools in section 3.6, and the software validation plan in section 3.7.

3.1 Verification and Validation Team

This section shows the members of the VnV team. They are shown in Table 1 along with what document they contributed in and their roles.

| Name | Document | Role |
|-------------------|------------|------------------------|
| Aliyah Jimoh | All | Author |
| Dr. Spencer Smith | All | Instructor Reviewer |
| Kiran Singh | SRS VnV | Domain Expert |

Table 1: Verification and Validation Team

[Your teammates. Maybe your supervisor. You should do more than list names. You should say what each person’s role is for the project’s verification. A table is a good way to summarize this information. —SS]

3.2 SRS Verification Plan

[List any approaches you intend to use for SRS verification. This may include ad hoc feedback from reviewers, like your classmates (like your primary reviewer), or you may plan for something more rigorous/systematic. —SS]

[If you have a supervisor for the project, you shouldn’t just say they will read over the SRS. You should explain your structured approach to the review. Will you have a meeting? What will you present? What questions

will you ask? Will you give them instructions for a task-based inspection?
Will you use your issue tracker? —SS]
[Maybe create an SRS checklist? —SS]

3.3 Design Verification Plan

[Plans for design verification —SS]
[The review will include reviews by your classmates —SS]
[Create a checklists? —SS]

3.4 Verification and Validation Plan Verification Plan

[The verification and validation plan is an artifact that should also be verified.
Techniques for this include review and mutation testing. —SS]
[The review will include reviews by your classmates —SS]
[Create a checklists? —SS]

3.5 Implementation Verification Plan

[You should at least point to the tests listed in this document and the unit testing plan. —SS]
[In this section you would also give any details of any plans for static verification of the implementation. Potential techniques include code walk-throughs, code inspection, static analyzers, etc. —SS]
[The final class presentation in CAS 741 could be used as a code walk-through. There is also a possibility of using the final presentation (in CAS741) for a partial usability survey. —SS]

3.6 Automated Testing and Verification Tools

[What tools are you using for automated testing. Likely a unit testing framework and maybe a profiling tool, like ValGrind. Other possible tools include a static analyzer, make, continuous integration tools, test coverage tools, etc. Explain your plans for summarizing code coverage metrics. Linters are another important class of tools. For the programming language you select, you should look at the available linters. There may also be tools that verify that coding standards have been respected, like flake9 for Python. —SS]

[If you have already done this in the development plan, you can point to that document. —SS]

[The details of this section will likely evolve as you get closer to the implementation. —SS]

3.7 Software Validation Plan

[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS]

[You might want to use review sessions with the stakeholder to check that the requirements document captures the right requirements. Maybe task based inspection? —SS]

[For those capstone teams with an external supervisor, the Rev 0 demo should be used as an opportunity to validate the requirements. You should plan on demonstrating your project to your supervisor shortly after the scheduled Rev 0 demo. The feedback from your supervisor will be very useful for improving your project. —SS]

[For teams without an external supervisor, user testing can serve the same purpose as a Rev 0 demo for the supervisor. —SS]

[This section might reference back to the SRS verification section. —SS]

4 System Tests

[There should be text between all headings, even if it is just a roadmap of the contents of the subsections. —SS]

4.1 Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

[Include a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. —SS]

4.1.1 Area of Testing1

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

Title for Test

1. test-id1

Control: Manual versus Automatic

Initial State:

Input:

Output: [The expected result for the given inputs. Output is not how you are going to return the results of the test. The output is the expected result. —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

2. test-id2

Control: Manual versus Automatic

Initial State:

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

4.1.2 Area of Testing2

...

4.2 Tests for Nonfunctional Requirements

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. Not all projects will necessarily have nonfunctional requirements related to accuracy. —SS]

[For some nonfunctional tests, you won't be setting a target threshold for passing the test, but rather describing the experiment you will do to measure the quality for different inputs. For instance, you could measure speed versus the problem size. The output of the test isn't pass/fail, but rather a summary table or graph. —SS]

[Tests related to usability could include conducting a usability test and survey. The survey will be in the Appendix. —SS]

[Static tests, review, inspections, and walkthroughs, will not follow the format for the tests given below. —SS]

[If you introduce static tests in your plan, you need to provide details. How will they be done? In cases like code (or document) walkthroughs, who will be involved? Be specific. —SS]

4.2.1 Area of Testing¹

Title for Test

1. test-id1

Type: Functional, Dynamic, Manual, Static etc.

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:

4.2.2 Area of Testing2

...

4.3 Traceability Between Test Cases and Requirements

[Provide a table that shows which test cases are supporting which requirements. —SS]

5 Unit Test Description

[This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS]

[To save space and time, it may be an option to provide less detail in this section. For the unit tests you can potentially layout your testing strategy here. That is, you can explain how tests will be selected for each module. For instance, your test building approach could be test cases for each access program, including one test for normal behaviour and as many tests as needed for edge cases. Rather than create the details of the input and output here, you could point to the unit testing code. For this to work, your code needs to be well-documented, with meaningful names for all of the tests. —SS]

5.1 Unit Testing Scope

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

5.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

5.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. test-id1

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

2. test-id2

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

3. ...

5.2.2 Module 2

...

5.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

5.3.1 Module ?

1. test-id1

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

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:

5.3.2 Module ?

...

5.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

References

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

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

6.2 Usability Survey Questions?

[This is a section that would be appropriate for some projects. —SS]