System Verification and Validation Plan for ProgName

Team #, Team Name
Student 1 name
Student 2 name
Student 3 name
Student 4 name

February 25, 2025

Revision History

Date	Version	Notes
February 23	1.0	Initial version of the VnV Plan

[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

List of Tables

[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**
T	Test

[symbols, abbreviations, or acronyms — you can simply reference the SRS (Grandhi, 2025) tables, if appropriate —SS]

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

This document ... [provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

2 General Information

2.1 Summary

The ESF project will involve constructing a library that enables users to both train and test multi-sensor 3D object detection networks on public autonomous driving datasets. Expanding upon the capabilities of the OpenPCDet repository, it will serve as a foundation for further research into the field by presenting a method for LiDAR-Camera fusion-based object detection that better addresses alignment issues faced by prior methods.

The training and inference (testing) portions of ESF correspond to two different use-cases that require different considerations. During training, the library is intended to allow an experienced machine learning and perception engineer to train a LiDAR-Camera fusion network on an autonomous driving dataset of their choosing. During inference, the library is intended to allow a user or a greater system to predict the set of dynamic objects in an autonomous vehicle's surroundings using a trained model and sensor input.

2.2 Objectives

One of the two primary objectives is to build confidence in the correctness of the software. As a routinely used and validated open-source repository, the base functionality of the OpenPCDet repository is considered to be error-free and thus its correctness is outside the scope of this document/project. The installation of the software and the extensions made to it are the components for which correctness will be verified.

Alongside correctness, verifying the accuracy of the solution relative to existing baselines implemented in OpenPCDet is the second primary objective. Since the software is intended to serve as a foundation for future research, its relative accuracy compared to state-of-the-art baseline methods will serve as a metric to judge how much it contributes to research in the field.

A secondary objective is ensuring the usability of the software. Since the library will be extended and used for research purposes, it is important to minimize the effort needed to understand and modify the code as needed.

2.3 Relevant Documentation

The other relevant design documents as well as their relations to this document are listed as follows:

- SRS (Grandhi (2025)): Describes the problem the software solves, the software's goals, the relevant theoretical models, and the requirements for potential solutions. This document thereby establishes the setting and mathematical tools necessary for the validation and verification methods described in this document.
- More documents will be added as they are completed.

3 Plan

[Introduce this section. You can provide a roadmap of the sections to come. —SS]

3.1 Verification and Validation Team

Name	**Role**	**Description**
Alaap Grandhi	Author	Write the VnV plan, carry out the testing detailed in it, fill in the VnV report, and validate the software and documents against the requirements during development.
Spencer Smith	Project Supervisor/Course Instructor	Review and provide feedback on the VnV plan and report. Additionally, provide insight into better testing methods.
Matthew Giamou	Masters Supervisor	Provide advice on the project in general.
Bo Liang	Domain Expert Reviewer	Review and provide feedback on the VnV plan and report.

3.2 SRS Verification Plan

The SRS will be reviewed in two stages. First, the Project Supervisor and the Domain Expert Reviewer will evaluate the document using the SRS checklist from lecture (cite here). Since they are familiar with the document structure, they will be able to provide feedback on how well the document adheres to the template and the required structure (in addition to providing feedback on document content).

Following the creation of a series of github issues corresponding to feedback on the document, the second stage of SRS reviewing will begin. At that point, a meeting will be set up with my Masters supervisor to go over the scientific principles discussed in the SRS document alongside the suggested changes and questions from the issues on github. Since he is familiar with my research he can provide additional feedback on any technical inconsistencies or ambiguities in the document.

3.3 Design Verification Plan

To verify the design, I have constructed the following checklist detailing the aspects of the design that the Domain Expert Reviewer and optionally the Masters and Project Supervisors (if they have time) will evaluate during regular discussions:

- Does the design explicitly or implicitly address all requirements described in the SRS document?
- Are the interfaces between modules of the design clear and in-line with the modular structure of the OpenPCDet repository?
- Given the overall design description, can the Domain Expert Reviewer accurately roughly predict the downstream code structure?
- Are any aspects of the design ambiguous?
- Does the design leverage existing functionality from the OpenPCDet repository where applicable?
- Do all of the design components contribute towards meeting a requirement?
  - If not, is this due to a redundancy in the design or something missing in the SRS document?

3.4 Verification and Validation Plan Verification Plan

The VnV plan will be verified in two parts corresponding to the two different types of tests it contains.

Where possible, tests governed by automated test suites like pytest will be tested using mutation testing to ensure that these tests sufficiently cover coding mistakes that are expected to happen. The specific 'mutations' to be inserted into the code will be decided after the implementation is done.

Manual tests and the overall structure of the VnV plan will be reviewed by the Project Supervisor and the Domain Expert Reviewer. This will be done to both ensure that the document adheres to the template and to ensure that the tests presented sufficiently cover the requirements described in the SRS document.

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

There are a few different automated tools that will be used for testing. Since the library will be written in Python (the OpenPCDet library is written in Python), PyFlakes will be used as an error linter to discover simple logical errors. Linters like flake8 that enforce style convention will not be used as the base OpenPCDet repository does not adhere to any particular style convention (it is designed with research and quick development as the primary objective) and it would be out of scope to refactor that. Additionally, the automated unit tests below will be implemented in pytest and will be set to automatically run on push using github actions. Code coverage will not be considered for this project since by nature different components are meant to be called according to the training configuration passed in and it would overly tedious to routinely test over all possible input configurations.

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, you 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

Alaap Grandhi. System requirements specification. https://github.com/alaapgrandhi/equivariant-sensor-fusion/tree/main/docs/SRS/SRS.pdf, 2025.

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]