System Verification and Validation Plan for PCD: Partially Covered Detection of Obscured People using Point Cloud Data

Team #14, PCD
Tarnveer Takhtar
Matthew Bradbury
Harman Bassi
Kyen So

November 5, 2024

Revision History

Date	Version	Notes
Nov 11, 2024	Rev 0	Initial Draft

Contents

List of Tables

1 Symbols, Abbreviations, and Acronyms

All symbols, abbreviations, and acronyms can be found within section E.1 (Glossary) of the SRS Report.

This document outlines the Verification and Validation plan for our software project. The purpose of this plan is to increase confidence in our software by ensuring it meets its requirements and performs as expected. This plan will list our objectives and processes related to verification and validation of our system and our roadmap for doing so.

2 General Information

2.1 Summary

The software being tested is our Partially Covered Detection (PCD) system. The system leverages depth and RGB layers to form a combined coloured point cloud, which is then analyzed to accurately detect individuals even when they are not fully visible. Detection can occur in a live setting through a Kinect, or using offline file captures.

2.2 Objectives

The primary objective of this VnV plan is to ensure the system's correctness and performance, verifying both its functional and non-functional requirements. This involves testing the system's ability to accurately identify partially obscured individuals and demonstrate that it operates efficiently within its environment. Additionally, the plan aims to ensure that the system implementation matches the project specifications. Testing within dark environments is considered out of scope, as RGB data cannot be captured in such settings. Furthermore, the project assumes that any external libraries used have already been verified by their respective implementation teams.

2.3 Challenge Level and Extras

The challenge level for this project is advanced, as agreed upon with our assigned TA. A User Manual and Design Thinking additions are our included extras.

2.4 Relevant Documentation

The following documents are relevant to the Verification and Validation efforts for this project. Each document listed below provides information supporting the VnV process, ensuring that the system meets its requirements and operates as intended.

1. Problem Statement and Goals: This document outlines the primary objectives and challenges of the project. It provides a clear understanding of the problem being addressed and the goals to be achieved, and is vital for defining the scope and focus of the VnV proceedings.

2. Development Plan: This plan includes risks related to the project prior to conducting a formal hazard analysis. It is important to verify the risks outlined in the document relating to the software system have been mitigated.

3. SRS Report: The Software Requirements Specification (SRS) report defines the functional and non-functional requirements of the system. It includes a brief baseline for VnV efforts, providing the criteria against which the system's correctness and performance will be evaluated.

4. Hazard Analysis: This document identifies potential hazards and risks associated with the system. It is essential for the VnV process as it helps in prioritizing the testing efforts to address the most critical risks and ensure the system's safety and reliability.

5. Module Interface Specification: The Module Interface Specification (MIS) describes the interfaces between different modules of the system. It is relevant to the VnV efforts as it ensures that the interactions between modules are correctly implemented and function as intended.

6. Module Guide: The Module Guide (MG) provides detailed descriptions of each module's design and implementation. It is used in the VnV process to verify that each module meets its design specifications and integrates seamlessly with other modules.

7. VnV Report: This report documents the results of the VnV activities, including the tests performed, issues identified, and their resolutions. It provides a comprehensive evaluation of the system's compliance with its requirements and serves as a record of the VnV efforts.

3 Plan

This section describes the overall plan for the verification and validation of our system. It includes the work breakdown of each member of the verification and validation team. This section also outlines the plans for the verification of our SRS, Design, and VnV. Furthermore, it details the plans for the implementation of these verification strategies as well as the implementation of the testing tools and the software validation plan.

3.1 Verification and Validation Team

Table 1: Verification and Validation Team Members Table

Name	**Role(s)**	**Responsibilities**
Harman Bassi	Lead test developer, Test developer, Manual tester	Lead the test development process. Create automated tests for backend code. Main verification and reviewer of system/unit tests.
Matthew Bradbury	Test developer, Manual tester, Code Verifier	Create automated tests for backend code. Manually test human detection algorithm functionality. Ensure source code follows project coding standard. Verification reviewer for the Hazard Analysis and SRS.
Kyen So	Test developer, Manual tester, Code Verifier	Create automated tests for backend code. Manually test human outline manager functionality. Ensure source code follows project coding standards. Main verification reviewer for the Verification and Validation document.
Tarnveer Takhtar	Test developer, Manual tester	Create automated tests for backend code. Manually test Kinect manager and ensure proper functionality. Verification Reviewer of Hazard Analysis and SRS
Dr. Gary Bone	Supervisor, SRS validator, Final reviewer	Make sure SRS meets requirements of the project, Validate code functionality. Because Dr. Bone is the supervisor of this project, he can verify that the project is functioning as expected.

3.2 SRS Verification Plan

The current plan to verify our SRS involves incorporating both self-review and peer-review feedback, used in tandem with notes from our TA and a final read-over with our supervisor. Our team will first do a quick read-through of each other's sections and provide feedback for changes in the form of comments on the issue. We will then incorporate the feedback we receive from our peers in another group, delivered to us via separate Github issues. These issues will be assigned to a single member of the team, who will have the responsibility of finishing and closing it. Additionally, we will create issues related to the feedback we received from our TA and work on adding the corresponding changes to our SRS. Finally, after incorporating all the feedback received, we will host a meeting with Dr. Bone. In this meeting, the team will walk Dr. Bone through our SRS and get his opinion on any final changes that need to be made to our requirements. Issues will be created to address the requested changes.

3.3 Design Verification Plan

Like the SRS, the plan for design verification includes a team review, a peer review, and a TA review. Like the SRS, the team review will involve team members reviewing another team member's section and commenting on changes that should be made. The peer review will similarly consist of another team adding Github issues to our repository and getting assigned to a team member. At that time, the specified team member will complete and close the issue. Additionally, we will incorporate issues raised by our TA.

3.4 Verification and Validation Plan Verification Plan

Similarly to the previous plans, this one will consist of a team review, a peer review, and a TA review, as well as a review from our supervisor. Just as the previous plans, the team, peer, and TA review will be conducted in the same way as previously mentioned. Additionally, for this verification plan, we will also be including our supervisor, Dr. Bone. The team will schedule a meeting with him and go through specific parts of the document. These parts will be our plan for what automated tests we are implementing, and Dr. Bone will have the final word on any improvements or changes we should make before proceeding.

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 walkthroughs, code inspection, static analyzers, etc. —SS]

[The final class presentation in CAS 741 could be used as a code walkthrough. 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

For the automated testing, cppunit will be used as it provides a simple and portable way to unit test the system. When it comes to doing coverage testing it would be best to use GCov because it is compatible with VScode and easier to set up compared to other applications. The main coverage focus for the project would be MC/DC coverage because it is a good coverage test for complicated decisions which the PCD system will have to make. Linters are also a good tool to help ensure all the code meets a certain standard that is respected within the field. For this project, Clang-Tidy will be used because it is compatible with VS code and meets the standards within the industry for C++.

3.7 Software Validation Plan

For software validation, we will be providing Dr. Bone with bi-weekly written updates on the progress of the project. These updates will serve as a brief validation that our software matches the aforementioned requirements outlined in the SRS. These smaller written checkups serve as an iterative way to validate the software against the requirements by constantly getting input from Dr. Bone about new code. Larger releases, such as code milestones or demos, will be accompanied by a meeting with Dr. Bone instead of a written update. These meetings will be lead by the main developers of the corresponding section and serve as the main form of software validation. The team will also consider peer feedback and TA/prof feedback from the demo to determine if the software accomplishes its goals.

4 System Tests

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

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.

Table 2: Test and Requirements Traceability Matrix - See Section G.4 In SRS Report.

Tests	Requirement
(FT11)	[F411]
(FT12)	[F411]
(FT13)	[F411]
(FT14)	[F411]
(FT15)	[F411]
(FT16)	[F411]
(FT17)	[F411]
(FT18)	[F411]
(FT21)	[F412]
(FT22)	[F412]
(FT23)	[F412]
(FT24)	[F412]
(FT25)	[F412]
(FT26)	[F412]
(FT27)	[F412]
(FT28)	[F412]
(FT31)	[F413]
(FT32)	[F413]
(FT41)	[F414]
(FT42)	[F414]
(FT43)	[F414]
(FT51)	[F415]
(FT52)	[F415]
(NFT11)	[NF431]
(NFT21)	[NF432]
(NFT31)	[NF433]

Appendix — Reflection

[This section is not required for CAS 741 —SS]

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning.

1. Why is it important to create a development plan prior to starting the project?

   It is important to create a development plan prior to starting the project as it allows most of the heavy lifting behind project planning to be done before any work has started. It allows for expectations and workflow to be clearly defined before any issues arise. It also creates a document that can be referenced at other times to avoid confusion.

2. In your opinion, what are the advantages and disadvantages of using CI/CD?

   Employing CI/CD allows for better issue tracking and rollbacks. Utilizing CI/CD gives the opportunity for teams to better track individual issues and commits, leading to increased awareness and visibility on workflow issues. It also allows for easier time rolling back to a previous version in case something goes wrong. Some disadvantages are with the conceptual depth and speed. Ensuring a specific workflow and constant PR reviews can slow things down as contributors have to make sure that they are following the workflow properly and have to wait for PR reviews (when necessary). Furthermore, it is more effort to set up, both in the codebase and conceptually. The process has to be talked through and understood by all team members.

3. What disagreements did your group have in this deliverable, if any, and how did you resolve them?

   Our group mainly debated how to set up our GitHub workflows. Initially, we considered using individual branches for each issue, but we ultimately decided on a more streamlined approach with two revision branches and separate forks. This allows us to effectively manage pull requests for merging feature changes into the codebase. We reached

this agreement after discussing the benefits of clarity and collaboration in our development process.

4. What went well while writing this deliverable?

   This deliverable allowed for the group to be able to discuss standard goals vs stretched goals. Everyone contributed their ideas and we were able to come to a clear conclusion when it came to the goals and problem statement of the project. It also allowed us all to see where the project is headed and how we should properly prepare ourselves so that we can achieve our goals.

5. What pain points did you experience during this deliverable, and how did you resolve them?

   The biggest pain point during this deliverable was being able to decide which goals were too ambitious and outside our design scope. Some goals such as the aspect of outlining the human in the environment were broken up. There was a deep discussion on how to properly decide the goals, but at the end the group got a better understanding of the project.

6. How did you and your team adjust the scope of your goals to ensure they are suitable for a Capstone project (not overly ambitious but also of appropriate complexity for a senior design project)?

   Because our project is presented by a professor, the team already had a pretty clear understanding of the goals that needed to be achieved for our project to be considered a success. The only issue was trying to ensure that the goals can be properly broken down so that a goal that seemed a bit ambitious could be broken into something that seems doable. For example, the human detection is broken into the Minimal and viable product goal and then also extended into the stretch goal. This was an important discussion that the group had to ensure that we deemed our goals to be doable.

7. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possi-

ble knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

Every member of the group needs to acquire knowledge on Computer Vision and understanding how pcd files operate. Overall these are big topics and so the basics should be acquired by every member, but some specifics would be broken down to different parts for each member. Tarnveer and Matthew would be assigned to understanding how to track people on screen and map them on the screen. Harman and Kyen would be working on reading in the pcd files and understanding the PCL. The PCL will explore on aspects of boxing the points on screen. Tarnveer would also be responsible for understanding the real time coding aspect in c++. Matthew would also be assigned to acquire knowledge on improving the human outline based on better data. Harman will be assigned to understanding how to cancel out noise from the data set. Kyen will have to understand how to find the person based off the pcd and understand how to properly read the files.

8. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

   One approach for acquiring the knowledge would be to use the provided documentation for the libraries (PCL and OpenCV). This would provide a good fundamental understanding for the important aspects of the project.

   Another approach would be to just watch videos on the specific topics and try to understand from there. This would be able to provide a more visual explanations for the topics.

   Tarnveer: use documentation and videos Matthew: use documentation and videos Kyen: use documentation Harman: use documentation and videos

9. What went well while writing this deliverable?

The deliverable was straightforward. We had a rough idea of the main hazards within our project and tried to make sure that we covered the main scope. The document writing was split between all of us. The document is pretty straightforward and we as a group were able to talk over the different sections and divide up the work.

10. What pain points did you experience during this deliverable, and how did you resolve them?

    The biggest pain point was probably discussing what would be some assumptions we had to make, but we were able to come to an agreement by communicating our points of why or why not.

11. Which of your listed risks had your team thought of before this deliverable, and which did you think of while doing this deliverable? For the latter ones (ones you thought of while doing the Hazard Analysis), how did they come about?

    All the risks were mainly thought of before the deliverable. We knew that we needed to ensure that the offline file is the correct format and that the system needs to make sure it is working with a Kinect sensor and not something else.

    The privacy risk was something we thought of at the informal interview.

12. Other than the risk of physical harm (some projects may not have any appreciable risks of this form), list at least 2 other types of risk in software products. Why are they important to consider?

    Could be some performance risks and making sure that the performance of the software meets the goals/requirements for the project. Another risk could be in terms of privacy. The application is capturing sensitive data and so its important on how the application handles this data.

1. What went well while writing this deliverable?

2. What pain points did you experience during this deliverable, and how did you resolve them?

3. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing

knowledge, static testing knowledge, specific tool usage, Valgrind etc. You should look to identify at least one item for each team member.

4. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?