Verification and Validation Report: 2D Localizer

Aliyah Jimoh

April 18, 2025

1 Revision History

Date	**Version**	**Notes**
2024/04/18	1.0	Initial Draft

2 Symbols, Abbreviations and Acronyms

symbol	description
FM	Fiducial Marker
T	Test
SRS	Software Requirement Specification
VnV	Verification and Validation

Contents

List of Tables

List of Figures

This document shows the results of the system and unit tests done on the 2D Localizer software. More detailed information of the tests cases performed are documented in this system's VnV Plan.

3 Functional Requirements Evaluation

3.1 Validate and Verify Inputs

- **Test Case(s):** `test_map_image`, `test_coordinates`, `test_range_measurements`, `test_invalid_input`, `test_nan_beacon_rejection`
- **Requirement(s) Met:**
  - R1: Accept a map/image as an input
  - R2: Accept sensor and fiducial marker (FM) coordinates as an input
  - R3: Accept sensor measurements as an input
  - R4: Verify that inputs provided are within their constraints
- **Type:** Automatic
- **Testing Method:** `pytest`
- **Result Summary:** All 5 successfully passed
- **Result Location:** Unit tests from `test_inputs_results.txt` and `test_input_file_results.txt` are located in the timestamp folder of test/results.

3.2 Verify Localization

- **Test Case(s):** `test_range_only`, `test_fiducials`, `test_sensor_fusion`
- **Requirement(s) Met:**
  - R5: Compute estimated pose from provided inputs
- **Type:** Automatic
- **Testing Method:** `pytest`

- **Result Summary:** All 3 successfully passed
- **Result Location:** Can all be found in `test_pose_estimation_results.txt` located in the timestamp folder of test/results.

3.3 Accuracy Evaluation

- **Test Case(s):** `test_accuracy`
- **Requirement(s) Met:**
  - R6: Evaluate the accuracy of the estimated pose with statistical models (found in SRS)
- **Type:** Automatic
- **Testing Method:** `pytest`
- **Result Summary:** Successfully passed
- **Result Location:** Can all be found in `test_accuracy_results.txt` located in the timestamp folder of the results folder.

3.4 Validate Visualization

- **Test Case(s):** `test_visualization`
- **Requirement(s) Met:**
  - R7: Display visual representation (plot) of pose estimate
  - R8: Display reat-time coordinate updates of the robot
  - R9: Display the robot's trajectory path
- **Type:** Automatic
- **Testing Method:** `pytest`
- **Result Summary:** Successfully passed. There are warnings that come up since the plot is being tested that it can display rather than actually displaying it.
- **Result Location:** Can all be found in `test_visuals_results.txt` located in the timestamp folder of test/results.

4 Nonfunctional Requirements Evaluation

4.1 Accuracy

- **Test Case(s):** `test_accuracy`
- **Requirement(s) Met:**
  - NFR1: Ensure that the accuracy metric output by the system is numerically sound.
- **Type:** Automatic
- **Testing Method:** `pytest`
- **Result Summary:** Successfully passed
- **Result Location:** Can all be found in `test_accuracy_results.txt` located in the timestamp folder of the results folder.

4.2 Maintainability

- **Requirement(s) Met:**
  - NRF2: Software should be simple to modify some components when needed.
- **Type:** Manual
- **Testing Method:** Modifying components in modules and rerunning the unit tests in pytest to see if 2D Localizer still works.
- **Result Summary:** Software still operational.

4.3 Usability

- **Requirement(s) Met:**
  - NFR3: Software should be able to run on Linux operating systems and macOS.
- **Type:** Manual

- **Testing Method:** Running the `make install` command from the Makefile found in the src folder using two laptops (one with Linux and the other with macOS)

- **Result Summary:** Both laptops were successfully able to install the virtual environment and run the software.

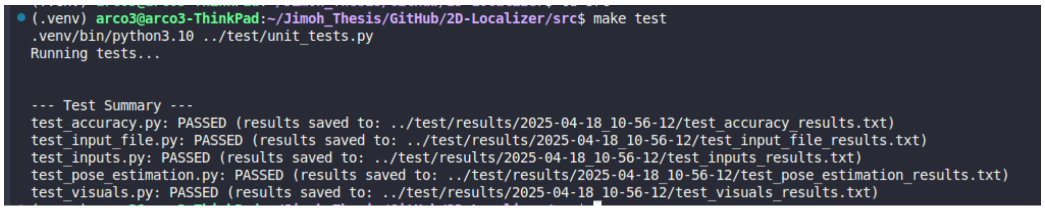
5 Comparison to Existing Implementation

This section is not applicable.

6 Unit Testing

The unit tests for 2D Localizer are run using the command `make test` which executes the program `unit_tests.py` located in the test folder. The script looks through all the programs that start with 'test_' and end with '.py' as that is the naming convention that was chosen for all test cases.

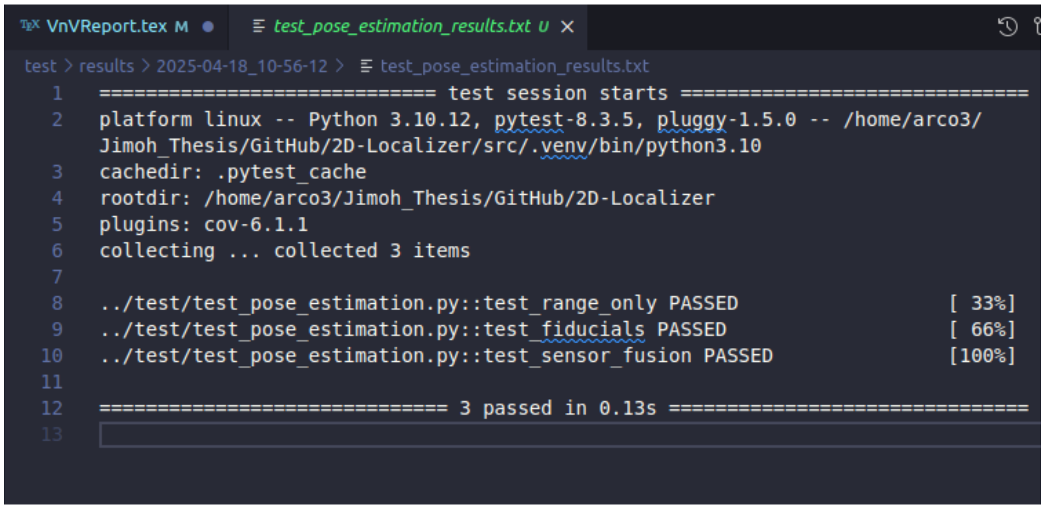
After all the tests finish running, the program generates a .txt summary file each module tested. This file records whether the test passed or failed and indicates the location where the result is stored. A combined terminal output also provides a quick summary of the results.



```
(.venv) arco3@arco3-ThinkPad:~/Jimoh_Thesis/GitHub/2D-Localizer/src$ make test
.venv/bin/python3.10 ../test/unit_tests.py
Running tests...


--- Test Summary ---
test_accuracy.py: PASSED (results saved to: ../test/results/2025-04-18_10-56-12/test_accuracy_results.txt)
test_input_file.py: PASSED (results saved to: ../test/results/2025-04-18_10-56-12/test_input_file_results.txt)
test_inputs.py: PASSED (results saved to: ../test/results/2025-04-18_10-56-12/test_inputs_results.txt)
test_pose_estimation.py: PASSED (results saved to: ../test/results/2025-04-18_10-56-12/test_pose_estimation_results.txt)
test_visuals.py: PASSED (results saved to: ../test/results/2025-04-18_10-56-12/test_visuals_results.txt)
```

Figure 1: Output of the make test command


VnVReport.tex M
test_pose_estimation_results.txt U
test > results > 2025-04-18_10-56-12 > test_pose_estimation_results.txt

```
============================ test session starts =============================
platform linux -- Python 3.10.12, pytest-8.3.5, pluggy-1.5.0 -- /home/arco3/
Jimoh_Thesis/GitHub/2D-Localizer/src/.venv/bin/python3.10
cachedir: .pytest_cache
rootdir: /home/arco3/Jimoh_Thesis/GitHub/2D-Localizer
plugins: cov-6.1.1
collecting ... collected 3 items

../test/test_pose_estimation.py::test_range_only PASSED                  [ 33%]
../test/test_pose_estimation.py::test_fiducials PASSED                   [ 66%]
../test/test_pose_estimation.py::test_sensor_fusion PASSED               [100%]

============================= 3 passed in 0.13s ==============================
```

Figure 2: Test Summary for test_pose_estimation.py

7 Changes Due to Testing

- **Improved Accuracy Validation:** The approach for evaluating pose accuracy was expanded. The final version includes explicit tests for Fisher Information Matrix (FIM) and Cramér-Rao Lower Bound (CRLB), validated in `test_accuracy.py`.
- **Expanded Unit Testing and Coverage:** Initial versions of the VnV Plan lacked specific testing for key modules such as plotting and input validation. These were added in `test_visuals.py` and `test_inputs.py`, respectively.
- **Integration with Automation Tools:** To support repeatable testing, automated testing was introduced through an extra command in the Makefile to output summaries. This ensures that changes to the codebase can be quickly validated after updates.

8 Automated Testing

All unit and system tests were written using the `pytest` framework. Test scripts followed a consistent `test_*.py` naming convention and were validated using both standalone runs and automated workflows. This custom Python script, `unit_tests.py`, automatically discovers all test files and records their output in corresponding `.txt` summary files the test/results folder in the GitHub Repository.

9 Trace to Requirements

Test Case	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**NFR1**
test_invalid_yaml				X						
test_map_image	X									
test_coordinates		X								
test_range_measurements			X							
test_range_only			X							
test_fiducials		X								
test_sensor_fusion		X	X		X					
test_visualization							X	X	X	
test_accuracy						X				X

Table 1: Tracibility Matrix Between the Test Cases & Requirements

10 Trace to Modules

Test Case	**localization.py**	**input_format.py**	**accuraccy.py**	**plot.py**
test_invalid_yaml		X		
test_map_image		X		
test_coordinates		X		
test_range_measurements		X		
test_range_only	X			
test_fiducials	X			
test_sensor_fusion	X			
test_visualization				X
test_accuracy			X	

Table 2: Tracibility Matrix Between the Test Cases & Modules

11 Code Coverage Metrics

As mentioned in the VnV Plan, there were some modules that were justified for not having any unit tests resulting in their coverage being 0%. Due to this, the main focus will be the coverage:

- Input Format Module (M3)
- Localization Module (M6)
- Plotting Module (M8)
- Accuracy Evaluation Module (M9)

Name	**Stmts**	**Miss**	**Coverage**
accuracy.py	15	0	100%
input_format.py	52	2	92%
localization.py	38	0	100%
plot.py	62	20	68%

Table 3: Code Coverage Metric for the Unit Tested Modules

From what is being prioritized, all modules have a with over 60%. The rationale for the Plotting Module being relatively low is due to the unit test not showing the plot but rather confirming that it can display and update itself.

References

Timothy D. Barfoot. *State Estimation for Robotics.* Cambridge University Press, 2017.

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

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. Which parts of this document stemmed from speaking to your client(s) or a proxy (e.g. your peers)? Which ones were not, and why?
4. In what ways was the Verification and Validation (VnV) Plan different from the activities that were actually conducted for VnV? If there were differences, what changes required the modification in the plan? Why did these changes occur? Would you be able to anticipate these changes in future projects? If there weren't any differences, how was your team able to clearly predict a feasible amount of effort and the right tasks needed to build the evidence that demonstrates the required quality? (It is expected that most teams will have had to deviate from their original VnV Plan.) Barfoot (2017)