Project Title: System Verification and Validation Plan for Mechatronics

Team #20, OpenASL
Robert Zhu zhul49
Zifan Meng mengz17
Jiahui Chen chenj194
Kelvin Huynh huynhk12
Runze Zhu zhur25
Mirza Nafi Hasan hasanm21

November 02, 2022

1 Revision History

Date	Version	Notes
November 02, 2022	1.0 Everyone	
Date 2	1.1	Notes

Contents

List of Tables

2 Symbols, Abbreviations and Acronyms

Term, Abbreviation, or Acronym	Description
ASL	Shorthand for American Sign Language. It is a form of sign language primarily used in the US and in parts of Canada
CFR	Shorthand for Camera Functional Requirement
CV	Shorthand for computer vision, computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos
FPS	Shorthand for frames per second. It is the measure of how many frames are displayed within a second. This is a camera performance metric.
MLFR	Shorthand for Machine Learning Functional Requirement
NFR	Shorthand for Non-Functional Requirement
OpenASL	This is the name of the project which is to create a sign language translator. The objective and purpose of this project can be found in the *Problem Statement* [3] and *SRS* [4] documentation of the project respectively
OpenCV	Shorthand for computer vision, computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos
RDP	Shorthand for Real-time Data Processing
SRS	Shorthand for System Requirement Specification
TC	Shorthand for Test Case

Table 1: Symbols, Abbreviations, and Acronyms

3 General Information

3.1 Summary

[Say what software is being tested. Give its name and a brief overview of its general functions. —SS]

3.2 Objectives

The objectives to be fulfilled by utilizing the VnV plan are as follows:

- Building confidence that the software was implemented correctly for the purpose of the project
- Ensuring that OpenASL displays adequate usability for its intended purpose. See *Problem Statement* [3] documentation

3.3 Relevant Documentation

The relevant documentation used to formulate the VnV plan include:

- Problem Statement [3]
- Development Plan [1]
- SRS [4]
- Hazard Analysis [2]

4 Plan

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

4.1 Verification and Validation Team

Name	Responsibility
Robert Zhu	White/Black Box Testing; Manual SRS Verification
Zifan Meng	OpenCV Verification; Manual code Verification
Jiahui Chen	End-to-End Testing; Manual SRS Verification
Kelvin Huynh	Machine Learning Verification; Manual code Verification
Runze Zhu	White/Black Box Testing; End-to-End Testing
Mirza Nafi Hasan	Performance Testing; Manual code Verification
Classmate Peer Review	Provide peer reviews for our project
Dr. Spencer Smith / TAs	Provide reviews and feedback for our project

Table 2: Verification and Validation Team Members and Roles

4.2 SRS Verification Plan

The approaches for the SRS verification plan can be peer reviews from other teams, reviews from our group and reviews from TAs.

4.3 Design Verification Plan

Similar to the SRS verification plan, the approaches involve peer reviews from teammates and other teams and the reviews from TAs.

4.4 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]

4.5 Automated Testing and Verification Tools

As stated in the Development Plan [1], unit testing is planned to be accomplished through the use of the Pytest Unit Testing framework. Code coverage can also be determined using this same framework using the pytest-cov plugin. The framework would enable us to utilize automated testing; however, unit tests have not been developed for the current phase of the project.

For performance testing, there are options available such as the utilization of OpenCV's getTickCount() and getTickFrequency() which can be used to calculate the FPS of the camera. In addition, there is also the use of Python's own time library function, time.perf_counter(), enabling us to measure the execution time of our code.

In terms of code verification, the code for the project will loosely follow the PEP8 Python coding standard. The linter that the project will use is the Flake8 linter for Python which will enable us to conform to this coding standard and catch syntactic errors that the code may have.

4.6 Software Validation Plan

The software will be validated through blackbox and white box testing. The whitebox testing is used to validate the inner working of the project such as coding. The only input for our device is the hand gestures from the users, so the blackbox testing can be adopted to ensure that the correct word is outputted. Various inputs are needed for the validation process and it can be achieved by having different users perform different hand gestures.

5 System Test Description

5.1 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. —SS]

5.1.1 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. 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 —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:

5.1.2 Area of Testing2

...

5.2 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. —SS]
[Tests related to usability could include conducting a usability test and survey. —SS]

5.2.1 Area of Testing1

Title for Test

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:

5.2.2 Area of Testing2

...

5.3 Traceability Between Test Cases and Requirements

Test Case ID	Requirement ID	Requirement Description
TC-CFR1	CFR1	The camera detects hand gestures and capture images
TC-MLFR1	CFR2, MLFR1, MLFR5	The system recognizes joints of the user's hand
TC-MLFR2	MLFR2, MLFR6	The system recognizes x, y, z coordinates of each joint relative to the camera
TC-MLFR3	MLFR3	The system identifies and separates two hands from each other
TC-MLFR4	MLFR1, MLFR3	The system identifies more than two hands and notifies users
TC-MLFR5	MLFR7	The model updates the database in learning mode
TC-RDP1	MLFR4, MLFR5	The model processes data in real-time according to user's continuous input
TC-RDP2	MLFR6	The system provides text-to-speech translation in real-time
TC-NFR1	TC-NFR1	The system provides results that have acceptable accuracy
TC-NFR3	TC-NFR3	New users quickly understands how to use the device
TC-NFR4	TC-NFR4	Instructions are easily understandable
TC-NFR6	TC-NFR6	The device is small and portable
TC-NFR7	TC-NFR7	The system has the ability of translating different forms of sign languages

Table 3: Traceability Between Test Cases and Requirements

References

[1] R. Zhu, Z. Meng, J. Chen, K. Huynh, R. Zhu, and M. N. Hasan. Development plan. https://github.com/kelhuynh/OpenASL, 2022.

[2] R. Zhu, Z. Meng, J. Chen, K. Huynh, R. Zhu, and M. N. Hasan. Hazard analysis. https://github.com/kelhuynh/OpenASL, 2022.

[3] R. Zhu, Z. Meng, J. Chen, K. Huynh, R. Zhu, and M. N. Hasan. Problem statement and goals. https://github.com/kelhuynh/OpenASL, 2022.

[4] R. Zhu, Z. Meng, J. Chen, K. Huynh, R. Zhu, and M. N. Hasan. System requirements specification. https://github.com/kelhuynh/OpenASL, 2022.

6 Appendix

6.1 User Experience Survey Questions

- On a scale of 1-10, how easy was it to learn the functions of the device (10 = very easy to learn, 1 = very difficult to learn)? Was there anything in particular that you found confusing about the device?
- Does the system translate your signs fast enough? Was there any delay in its processing?
- At any point, did you have to slow down your signing for the machine to correctly process your signing?
- On a scale of 1-10, how portable would you say the device is? (10 = very portable, 1 = not portable at all) Was there anything in particular that made the device less portable? (too large and bulky, too fragile, connections get loose, etc.)
- Does the hand tracking require you to reposition the camera repeatedly? Is this a major inconvenience?

6.2 Reflection

1. 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, etc. You should look to identify at least one item for each team member

    i) filler

2. 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?