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

Contents

List of Tables

1 Revision History

Date	Version	Notes
November 02,2022	1.0	Everyone
March 22, 2023	1.1	Updating Test Cases
April 5, 2023	1.2	Corrected Traceability Matrix

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

The VnV plan is designed to provide a detailed plan, including the approach and timeline, for the testing of the ASL translator.

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

This section goes over the verification and validation team for OpenASL with their responsibilities over the project. Additionally, the following sections are tested such as SRS, design, and implementation, along with the automation tools and software validation used to perform these tests.

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. An SRS checklist will be used to verify all the requirements for the project. The requirements will be verified using the system tests that are listed in detail in the SystemTest Description section.

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. Their feedback can be utilized to verify our design. In addition, the checklists for MG and MIS are also approaches for the design verification to ensure that the actual device is implemented under the project's goal and scope.

4.4 Implementation Verification Plan

Implementation verification is the process of reviewing, testing and verifying that the device is working correctly under the desired performance. Both dynamic and static techniques will be utilized for the verification. The dynamic techniques include the system tests that are listed in the next section for the implementation of the functional and nonfunctional requirements for the device. Unit tests should also be performed during the implementation verification plan. In terms of the static techniques, code inspection and code walkthrough should be done for every single line of code that will be added to the main script by at least two team members before uploading. And for any new changes that need to be fixed in the program, the whole team should review and agree with the

new implementation before uploading the new changes to avoid the potential risk and problem for the project.

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. These different users can include, but are not limited to, ASL users, sign language interpreters, people interested in learning sign language, among others. These people can help us validate if our device is easy to use and how accurate our machine learning model is.

5 System Test Description

5.1 Tests for Functional Requirements

5.1.1 Hardware

1. TC-CFR1: Camera Detection
   Control: Manual, Dynamic, Functional

   Initial State: Camera is placed in front of the user

Input: User performs hand gestures

Output: Send the image to the algorithm and it is recognized

Test Case Derivation: The image will be processed and translated into its corresponding hand gesture.

How test will be performed: The tester uses a variety of different gestures in sequence that the algorithm has learned.

2. TC-CFR2: Raspberry Pi
Control: Manual, Static, Functional

Initial State: Raspberry Pi is placed with the camera

Input: Translated text from the algorithm

Output: The Raspberry Pi will act as a speaker to play the translation in English

Test Case Derivation: After the algorithm is input to the Raspberry, it should process it correctly and "readout" the translated text in English.

How test will be performed: Some English sentences will be chosen for the text, the users will perform the sign languages for the selected sentences and listen to the outputs provided by Raspberry Pi and compare them with the input sentences.

5.1.2 Machine Learning

1. TC-MLFR1: Joint recognition
Control: Manual, Dynamic, Functional

Initial State: Camera is placed in front of the user

Input: User performs hand gestures

Output: User's hand gesture with joint lines overlaid on top

Test Case Derivation: The system is able to capture the camera's vision by recognizing the joints on the user's hand.

How test will be performed: The tester will raise their hand to the camera in any orientation following ASL phrases/words.

2. TC-MLFR2: Display coordinates in space
Control: Manual, Dynamic, Functional

Initial State: Camera is placed in front of the user

Input: User performs hand gestures

Output: Displays the x,y,z coordinates of each joint relative to the camera

Test Case Derivation: These system uses these coordinates to calibrate and enhance predictive accuracy.

How test will be performed: The tester will use any type of hand motion with both hands and compare the results of each joint to the displayed coordinates over each iteration of the algorithm.

3. TC-MLFR3: Detecting multiple pairs of hands
   Control: Manual, Dynamic, Functional

   Initial State: Camera is placed in front of the user

   Input: User will place two hands in front of the camera

   Output: Both hands will have joints overlaid on top and following movement

   Test Case Derivation: The system will separate and identify both hands from each other.

   How test will be performed: The user/tester will raise both their hands in front of the camera.

4. TC-MLFR4: Detecting multiple pairs of hands
   Control: Manual, Dynamic, Functional

   Initial State: Multiple users are standing in front of the camera

   Input: Multiple pairs of hands will be placed for detection

   Output: The system will tell the user that it is detecting more than one pair of hands and stop tracking

   Case Derivation: The system is created for a single user, thus multiple users are not supported for translation at the same time.

   How test will be performed: More than 2 hands will perform multiple signs at the same time in front of the camera performing any type of gesture.

5. TC-MLFR5: Trainable Model
   Control: Manual, Dynamic, Functional

   Initial State: User input is labeled incorrectly

   Input: User will press a key to enter learning mode, a second key to identify the classifier to be retrained, a third key repeatedly while changing hand positions to store the coordinates of the correct gesture into the dataset, and a fourth key to exit learning mode and start the model retraining process with the new data points

   Output: When the user signs in front of the camera again, the model should be updated and classify the previously unknown sign correctly

   Test Case Derivation: The system should be enabled for expandability to increase its vocabulary and adapt to users' habits.

   How test will be performed: Sign a word that is not in the dataset to create a new classifier.

5.1.3 Real-time Data Processing

1. TC-RDP1: Process data in real-time
   Control: Manual, Dynamic, Performance

   Initial State: Nothing displayed on graphic output

   Input: User will perform a gesture that the system recognizes

   Output: ML model will classify the gesture based on training data used and label the gesture accordingly

   Test Case Derivation: The system should react within proper reason and provide a label within an acceptable time frame.

   How test will be performed: Users will perform multiple signs in front of the camera to simulate the changing of input in real-time.

2. TC-RDP2: Text-to-Speech in real-time
   Control: Manual, Dynamic, Performance

   Initial State: Nothing displayed on graphic output

   Input: User will perform a gesture that the system recognizes

   Output: ML model will classify the gesture and label it. Then the system will put that label through a text-to-speech system to say the word out loud

   Test Case Derivation: The system should say the respective word associated to the gesture within an acceptable time frame.

   How test will be performed: Users will perform signs in front of the camera to simulate the changing of input in real-time.

5.2 Tests for Nonfunctional Requirements

5.2.1 Accuracy Requirement

1. TC-NFR1: Translation Accuracy
   Control: Manual

   Initial State: Camera is placed in front of the user

   Input: A sequence of user's hand gestures

   Output: The screen displays words/phrases/sentences corresponding to the user's hand gestures, results should have acceptable accuracy.

   How test will be performed: Users will sign in front of the device for a short period of time. After that, users will look at the screen and verify that the results are accurate enough for people to understand.

5.2.2 Usability Requirement

1. TC-NFR2: Understandability
   Control: Manual

   Initial State: Users do not know how to use the device

   Input: Users will read through instructions

   Output: Users will understand how to use the device

   How test will be performed: Users will read the instructions before using the device. They will then try all functions of the device, rate the understandability of the instructions on a scale of 1 to 10 and provide feedback through a survey.

2. TC-NFR3: Ease of use
   Control: Manual

   Initial State: User is working with the device for the first time

   Input: Users will read the instructions and learn how to use the device

   Output: Users will learn to work with the translator on their own without any difficulties
   How the test will be performed: Multiple new users will be given the device along with its instructions, and we will compare how long it takes each user to learn the device and use its main functionalities.

5.2.3 Portability Requirement

1. TC-NFR4: Portability
   Control: Manual

   How test will be performed: Users will carry the device and perform some everyday activities. Users can then rate the experience on a scale of 1 to 10 and provide feedback on the portability of the device through a survey.

5.2.4 Cultural Requirements (Future Implementation)

1. TC-NFR5: Cultural Requirement
   Control: Manual

   Initial State: Users are working with the ASL translator

   Input: Users perform different styles of sign language

   Output: The translator will output the translation in English

   How test will be performed: The test will be performed by letting users perform different forms of sign languages (other than ASL) in front of the camera, and check if the corresponding output is the correct translation.

5.3 Traceability Between Test Cases and Requirements

Requirement IDs GFR1-8, MLFR1-5, UIFR1-2, CRF1-8, NFR1-6 can be found in the *SRS* [4] documentation

Test Case ID	**Requirement ID**	**Requirement Description**
TC-CFR1	GFR1, GFR3 CFR2	The camera detects hand gestures and capture images
TC-CFR2	UIFR2	The Raspberry Pi will receive the translated text from the algorithm and play out the translation in English
TC-MLFR1	GFR4,	The system recognizes joints of the user's hand
TC-MLFR2	GFR2, MLFR2, MLFR3	The system recognizes x, y, z coordinates of each joint relative to the camera
TC-MLFR3	GFR5, MLFR3	The system identifies and separates two hands from each other
TC-MLFR4	GFR5,	The system identifies more than two hands and notifies users
TC-MLFR5	GFR8, MLFR1, MLFR4, NFR4	The model updates the database in learning mode
TC-RDP1	GFR6, GFR7, UIFR1, RDP1, NFR1	The model processes data in real-time according to user's continuous input
TC-RDP2	RDP2, UIFR2	The system provides text-to-speech translation in real-time
TC-NFR1	MLFR5	The system provides results that have acceptable accuracy
TC-NFR2	NFR2	New users quickly understands how to use the device
TC-NFR3	NFR3	Instructions are easily understandable
TC-NFR4	NFR5	The device is small and portable
TC-NFR5	NFR6	The system has the ability of translating different forms of sign languages

Table 3: Traceability Between Test Cases and Requirements

6 References

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

[2] R. Zhu, Z. Meng, J. Chen, K. Huynh, R. Zhu, and M. N. Hasan. Hazard analysis. https://github.com/kelhuynh/OpenASL/blob/main/docs/HazardAnalysis/HazardAnalysis.pdf, 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/blob/main/docs/ProblemStatementAndGoals/ProblemStatement.pdf, 2022.

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

7 Appendix

7.1 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?

7.2 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) Jiahui Chen: TestProject
    In order to successfully complete the verification and validation plans for the project, knowledge of TestProject is needed for end-to-end testing. End-to-end testing is a software testing method to test the workflow of the application from the beginning to the end to ensure the application performs as expected. TestProject, a Python testing framework. will be used for the end-to-end testing in our project. TestProject can develop automation framework easily for generic purposes with the Python open source SDK, it can also support PyTest which will be used for unit testing.

    ii) Kelvin Huynh: Pytest Framework
    One aspect of the VnV plan is unit testing, although there are no unit tests planned for the current phase of the project, it is still important to understand how to automate and conduct unit tests. Unit testing by definition is testing of individual components of the system to ensure each component is working as intended. We can achieve proper automated unit testing through the use of the

PyTest unit testing framework and definition of test cases. The framework itself is simple to use, it follows three core principles for most unit tests: Arrange, Act and Assert. So learning the syntax for the framework itself should not be a challenge; the challenge comes from developing potentially complex unit tests.

iii) Nafi Hasan: Confusion Matrix
One major component of our project is the machine learning model. Our device will rely on this model to accurately determine which sign language gestures are being performed in order to translate it correctly. In order to accomplish this, we will need to ensure that the machine learning model we are using is able to correctly identify various sign language gestures. This is where a confusion matrix will be useful for testing and validation. A confusion matrix is a table that is used to assess errors within machine learning models that are used for classification, specifically hand gestures in our case. The confusion matrix will provide various metrics such as accuracy, precision, how well the model can predict positives and negatives, among others.

iv) Robert Zhu: Black Box Testing
As this project involves the usage of a unique type of language that is not fluently performed by any member of our team, focusing on the user's perspective and seeing if the system delivers on its promises for quick and accurate translations becomes paramount. Black box testing is a method that helps in this regard as it tests a system with no prior knowledge of its internal workings and exercises a system end-to-end. Our stakeholders do not need to learn any implementation and can test non-functional requirements such as user intuitiveness.

v) Zifan Meng: Static Testing
The correctness and robustness of the code is the key to the success of this project. Whether it's the machine learning model or coding for Raspberry Pi, if the code is unstable or performs poorly, we won't be able to achieve our intended goals. Static testing is a software testing method that examines the program and associated documents before the program is executed. In comparison to dynamic testing, static testing examines the structure of the program, analyzes the performance, maintains code specification for better readability, modularity and extensibility. Furthermore, static testing is usually more efficient than dynamic testing, especially in the early phase of the project, or for dynamic language like Python.

vi) Runze Zhu: White Box Testing
In this project, in order to ensure that the program can be implemented properly and reduce the possible drawbacks of possible bugs, it's necessary to analyze the code structure rather than just test the functionality. Therefore, white box testing should be performed during the design process. It can help us keep the code understandable through the whole design process as it can be performed from an early stage of the program without the need of any interface,

it can also keep us optimizing the code throughout removing errors and extra lines and make the test cases easy to automate.

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?

   i) Jiahui Chen: TestProject
   Some approaches to acquire the knowledge of TestProject are self-study online tutorials such as watching the tutorials on Test Automation, take courses on udemy or ask other teams who are proficient on TestProject for help. The approach that our team will pursue is watching online tutorials on our own. Because we can find all the knowledge we need for the project and there are many tutorial websites and videos available on the websites. And online tutorials offer a more flexible learning time.

   ii) Kelvin Huynh: Pytest Framework
   In order to successfully develop unit tests for the system, it is necessary to understand the usual types of tests that one would conduct on systems such as cameras utilizing OpenCV or a machine learning model. In order to acquire this knowledge it is possible to look into online tutorials utilizing sample machine learning code to generate test cases. Another approach could be to read and understand the fundamentals of machine learning as sometimes static inputs may produce the wrong results due to error. This is especially crucial because unit testing for machine learning depends on the accuracy of the model. The approach that I would prefer involves a combination of the two, this is because understanding common unit test cases will help us to develop unit tests and understanding how to deal with the complex nature of ML models during testing will help us improve our model's accuracy.

   iii) Nafi Hasan: Confusion Matrix
   In order to learn more and utilize the confusion matrix, some approaches we can use include finding tutorials online. There are several guides and explanations online that cover what exactly a confusion matrix is, and how to go about implementing it. Another potential approach could be to ask others who are knowledgeable with machine learning about confusion matrices. While online tutorials are helpful, they can be quite confusing, so having someone who we can ask more specific questions to can be a great asset in learning more about this verification method. Of these two approaches, I think the preferred method would be to use online tutorials, although we will likely use a combination of the two. Online tutorials are very easy to find and there are plenty of them available. Many of them also provide examples that make it easier to see exactly how to generate the matrix and explain how to interpret the output.

iv) Robert Zhu: Black Box Testing
One of the approaches for learning more about black box testing and its techniques, such as decision table testing, is through the many software courses that the university provides that deal with software development and documentation. Another approach is through online videos and websites that provide tutorials on the importance of Black box testing. In our case, reviewing previous courses will provide the most valuable method in acquiring skills as previous projects give context and real experience to mitigating failure in the system at the user's end.

v) Zifan Meng: Static Testing
For our project, Python is the main language, therefore one approach to apply static testing is to learn how to utilize the built in static testing function (Inspect Code) of Pycharm. This function automatically inspects code according to PEP8 Python coding standard, it can find problems like bad structure, unlocalized string, memory leak, etc. There are many tutorials and documents about configuring and using Pycharm, it can be learnt quickly and applied easily. Another approach for static testing is an analyzer called Cppcheck, this analyzer is designed specifically for C or C++, if we use C or C++ for the Raspberry Pi, then this tool can be used. Cppcheck is very user friendly, there aren't many configurations needed for using it, therefore, looking through documents provided by the designer is sufficient for learning features of this analyzer.

vi) Runze Zhu: White Box Testing
One of the potential approaches that we can use to learn about white box testing is going through the software tutorials that can be found online. There are many online classes and introductions that provides detailed explanation on the definition of white box testing and how to perform it while coding. Another possible approach is ask software developers tha are experienced with testing. Since white box testing is commonly used through design process, most programers should be familiar with it and can offer constructive comments to us. At this stage, we can try to start with reading online materials since the resources are rich, if we have problem in understanding a specific part of the testing, we can also go to ask people who are knowledgeable.