WSUAG-Arduinoapp

Project Testing and Acceptance Plan

AgriData Intelligence

Members: Aidan Bachart, Forest Cook, Shreyas Jagalur

Oct. 26, 2023

TABLE OF CONTENTS

I. Introduction

I.1. Project Overview

This project is a sophisticated software system designed to enhance precision agriculture practices. It leverages advanced data processing and machine learning techniques to analyze crop-related images captured by Arduino sensors. The software is divided into two main components, Radiometric Correction and Plot Segmentation. The primary goal of this software is to democratize precision agriculture by providing local farmers with advanced tools for crop monitoring. It aims to improve crop productivity, reduce labor-intensive tasks, and enhance agricultural practices by delivering accurate and timely insights. Through the testing and acceptance plan outlined in this document, the software aims to ensure that it meets its intended requirements and functions as expected, providing a robust and dependable solution for precision agriculture.

I.2. Test Objectives and Schedule

Approach: The test plan objectives for the Project aim to ensure that our software meets its intended requirements, functions correctly, and aligns with the precision agriculture goals. The primary objectives include validating functional requirements, verifying performance under different conditions, and confirming user acceptance for operational use.

Required Resources: The successful execution of our testing activities requires a dedicated team of testers with expertise in image processing and machine learning. We will also need access to test environments, hardware and software resources, a comprehensive test data set, and a project agreement for user acceptance testing.

Schedule of Testing Activities:
Our testing activities will be organized into distinct phases, each with its specific focus and timeline. The schedule will be detailed as follows:

- Test Planning: Develop test cases, scenarios, and plans for functional and performance validation.
- Functional and Performance Testing: Execute automated tests to validate functional requirements and performance. Analyze performance data under different conditions.
- User Acceptance Testing: Engage with end-users to perform acceptance testing and gather feedback for real-world evaluation.
- Data Collection and Analysis: Collect and analyze data from various testing phases, identify issues, and report findings.
- Issue Resolution and Retesting: Address identified issues, document resolutions, and conduct retesting to ensure problem resolution.

Major Work Activities: Our major work activities encompass various critical aspects of our testing plan. These activities include test planning, including the development of detailed test cases and scenarios. Test execution is fundamental, with a focus on functional and performance validation. User interactions during user acceptance testing provide valuable feedback.

Products to be Delivered:

- Comprehensive test plans outlining the scope and objectives of each testing phase.
- Test cases and scenarios designed for functional and performance validation.
- Performance data and analysis reports.
- User acceptance testing feedback and observations.
- Issue reports and resolution documentation.

Major Milestones:

- Milestone 1: Completion of functional and performance testing, ensuring core features operate correctly.
- Milestone 2: Successful completion of user acceptance testing, confirming the system's readiness for operational use in precision agriculture.
- Milestone 3: Resolution of identified issues and successful retesting, preparing the software for deployment.

I.3. Scope

This document serves as a comprehensive guide to testing for this Project and encompasses the definition of major functionalities that need rigorous testing to ensure the software's reliability and functionality. It also outlines the approach, resource requirements, and schedule for testing activities, as well as the specific work activities including test planning, execution, and documentation. The document also clarifies the products to be delivered as outcomes of the testing process, such as test plans, test cases, and reports. Furthermore the document highlights the major milestones that will be tracked to ensure alignment with the project's development timeline and objectives.

II. Testing Strategy

Project testing will be set up in such a way that all models can be verified both automatically and by hand. The major models in this application are the reference panel detection model. The reference panel detection model is used to identify the reference panel in a given picture of several plots. The plot segmentation model is used to identify each individual plot within a given picture of several plots. We are also developing an algorithm to apply radiometric correction to photos using the identified reference panel. We intend on testing the two models and the radiometric correction individually as three separate pieces, as well as combined such that the entire application is tested as a whole unit. In other words, we will be testing using continuous integration. Continuous Integration (CI) will be employed to automate and streamline the testing process.

Testing Process

1. **Determine model requirements:** Determine specific model requirements for both the reference panel detection model as well as the plot segmentation model.
2. **Label images:** Label images using requirements outlined in step 1.
3. **Create model:** Create both models using labeled plot images.
4. **Determine accuracy requirements:** Figure out a realistic accuracy requirement that will provide the desired results.
5. **Refit model until accuracy is met:** Refit and retrain model as needed until desired accuracy is met.
6. **Develop radiometric correction algorithm:** We are not using Test-Driven Development. The code will be written to correct the image colors based on the colors identified in the reference panel.

7. **Develop radiometric correction tests:** Develop tests for the radiometric correction algorithm. These tests should be able to test all images produced by the reference panel detection model.
8. **Run radiometric correction tests:** The developers should run the tests on the code. If any tests fail, evaluate whether the issue is with this algorithm or with the reference panel detection model. If the issue is with the radiometric correction model, begin at step 4 again.
9. **Fix issues with radiometric correction:** If the issues are with the radiometric correction algorithm, fix said issues.
10. **Develop tests for the whole unit:** Once the entire process has been tested individually, develop tests for the entire pipeline.
11. **Run tests for the whole unit:** Run said tests. Be sure to identify where issues are caused.
12. **Fix issues with the whole unit:** Fix issues were caused, starting at applicable steps in the test plan.

III. Test Plans

III.1. Unit Testing

Due to the nature of our project, automated unit tests that ensure the system is 100% accurate will not be possible to produce. Our project relies on machine learning which is not 100% accurate. We can, however, gain statistics on how accurate the model is likely to be. For part 1 (plate identification), we will have a set of test images we run it against to check the accuracy. The test set will be images that the model has not seen before. This should give us a good idea of what results we can expect.

For part 2 (Image processing) we will run it on a series of test images, and compare the output files to existing outputs. Given that we are using a different method than was previously used and that different parameters may be used, we will allow for some variation. We will also run unit tests with a high zoom out parameter to allow the user to see that the individual boxes have been correctly identified.

For part 3 (Plot identification) we will likely be using a deep learning model, thus it will suffer from the same uncertainty as any other model. Given this we will use a test set of images to get statistical information about its accuracy. The test data will be a separate dataset that has not been seen before as with part 1.

III.2. Integration Testing

For integration testing we will have two phases, first, we will run it on a set of test images where it outlines the plot and boxes, allowing us to visually verify that all components are outputting the correct data. Then we will run the whole system and verify that it is producing output similar to the previous results. Given the different methods employed we expect some variation, but this should allow us to detect problems with the system.

III.3. System Testing

System testing is a type of black box testing that tests all the components together, seen as a single system to identify faults with respect to the scenarios from the overall requirements specifications. Entire system is tested as per the requirements.

During system testing, several activities are performed:

III.3.1. Functional testing:

In the context of our project, functional testing will validate that our software performs the functions specified in the project requirements. We will develop test cases based on the functional requirements outlined in our project documents, including radiometric correction and plot segmentation. These test cases will encompass various scenarios, such as image correction, plot identification, and plot normalization. The software will be executed with the defined test cases, simulating real-world use cases. The outcomes will be compared to the expected results to identify any discrepancies.

III.3.2. Performance testing:

Performance testing for our project is vital to ensure it meets nonfunctional requirements and design goals. We will focus on load testing, stress testing, and performance under typical operational conditions.

Specifics:

- Load testing will evaluate the software's performance under typical operating conditions, such as processing a substantial number of images.
- Stress testing will examine how the system behaves when pushed beyond its specifications, helping us identify its failure points.

We will monitor and record response times, throughput, and resource utilization as well and the performance data will be analyzed against predefined performance goals and nonfunctional requirements to ensure our system performs optimally under various conditions.

III.3.3. User Acceptance Testing:

In the context of our project, user acceptance testing (UAT) ensures that our system aligns with the project agreement and is ready for operational use. This phase involves end-users who will assess the system's functionality and performance.

Specifics:

- End-users, potentially with assistance from developers, will compare the system's functionality and behavior against the initial requirements and project agreement.
- We will design test scenarios to reflect real-world usage of our software, including radiometric correction and plot segmentation.
- End-users will actively interact with the system to validate its performance and functionality.
- Feedback and observations will be collected and analyzed to determine whether the system is ready for operational use.

IV. Environment Requirements

The test system will be able to run on a windows computer with python 3.8.0, tensorflow, numpy, and opencv2. The computer should have sufficient disk space for the tests. 10gb should be sufficient. For the best experience the computer should have modern hardware specification.

V. Glossary

Edge Computing - A method where computers do jobs right where they are needed, in order to make things faster and more efficient.

Image Correction - Correcting the images in order for them to all be based on the same base reference values.

IoT - Internet of Things (IoT) devices are everyday devices that are connected to the internet and to each other.

Phenotype - The observable characteristics of an organism.

Plot Segmentation - The division of images of crops into separate plots of different breeds of plant.

TensorFlow - A machine learning library that lends itself well to this kind of work

VI. References

[1] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," 2019. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[2] Gitter, "Building Online Communities: Keras.io," *We've moved to freeCodeCamp.org/news*, Oct. 11, 2016. https://medium.com/free-code-camp/building-online-communities-keras-io-be7256abe9c5 (accessed Sep. 22, 2023).

[3] "Image segmentation | TensorFlow Core," *TensorFlow*. https://www.tensorflow.org/tutorials/images/segmentation

[4] J. Terra, "Keras vs Tensorflow vs Pytorch: Popular Deep Learning Frameworks," *Simplilearn.com*, Jul. 27, 2020. https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article

[5] "Object Detection | TensorFlow Hub," *TensorFlow*. https://www.tensorflow.org/hub/tutorials/object_detection

[6] C. Hansen, "Winter wheat production in Northwest Region up 59 percent from last year," *Morning Ag Clips*, Oct. 04, 2022. https://www.morningagclips.com/winter-wheat-production-in-northwest-region-up-59-percent-from-last-year/#:~:text=acres%20from%202021.- (accessed Sep. 29, 2023).

[7] “Study: Climate change a growing threat to wheat | World Grain,” *www.world-grain.com*. https://www.world-grain.com/articles/18599-study-climate-change-a-growing-threat-to-wheat#:~:text=The%20study%20noted%20that%20high (accessed Sep. 29, 2023).

[8] W. Sangjan, A. H. Carter, M. O. Pumphrey, V. Jitkov, and S. Sankaran, “Development of a Raspberry Pi-Based Sensor System for Automated In-Field Monitoring to Support Crop Breeding Programs,” *Inventions*, vol. 6, no. 2, p. 42, Jun. 2021, doi: https://doi.org/10.3390/inventions6020042.

Appendix-A