Verification and Validation Report: Mechatronics Engineering

Team # 34, ParkingLotHawk
Fady Zekry Hanna
Winnie Trandinh
Muhammad Ali
Muhammad Khan

April 6, 2023

1 Revision History

Date	Version	Notes
March 8, 2023	1.0	Initial Revision

2 Definitions, Abbreviations, and Acronyms

symbol	**description**
Fail	None of the test case's purpose was verified.
FR	Functional Requirement
FSM	Finite State Machine
NFR	Non Functional Requirement
Partially Pass	A part of the test case's purpose was verified, but another part was not.
Pass	Expected and actual results of the test case match.
Principally Pass	The purpose of the test case is verified in a similar/modified test.
SITL	Software in The Loop
SRS	Software Requirement Specifications
VnV	Verification and Validation

Contents

List of Tables

List of Figures

3 Design Verification

As outlined within the Plan, multiple methods are utilized to verify the SRS and design to ensure that the design is meeting the requirements. This is done by verifying the SRS, the design, and then the VnV Plan itself.

3.1 SRS Verification

The Finite State Machine (FSM) outlined within the Software Requirement Specifications (SRS) has been formally verified through the maintenance of a decision table, located within Appendix B: Formal Transition Table. Through informal verification, the team has conducted a guided review with a technical external party, using the process outlined within the SRS Verification Plan. All Nonfunctional Requirements (NFRs) mentioned by the external party are present within the SRS, with additional NFRs not mentioned by the external party. These NFRs were discussed to determine if they are required, and no changes were made to the SRS as a result.

The checklist outlined within the SRS Verification Plan has also been conducted internally within the team and all points have been met. Furthermore, a code review has been conducted within the team to recreate the FSM from just the code. The resulting FSM matches the one located within Appendix B: Formal Transition Table, thus ensuring that the FSM is correctly implemented.

3.2 Design Verification

The Design Verification consists of a guided review with and external party, using the checklist provided within the Design Verification Plan as a reference. The main purpose is to ensure that all Functional Requirements (FRs) have been met. The results of the review can be found in External Design Verification. For conciseness, points that have been met and has no additional feedback are excluded in the results.

Table 1: External Design Verification

Criteria	Feedback	Team Response
The condition i_MinHoverHeight <= i_DesiredHoverHeight <= i_MaxHoverHeight shall always be true.	This requirement is not explicitly associated with a specific module or state, whether it's the GUI module or the Configure State. Thus, no module is currently verifying this condition.	The SRS was updated to specify that the condition will be verified within the Configure State.

A more detailed internal review has also been conducted using Design Verification Plan, with the results in Internal Design Verification.

Table 2: Internal Design Verification

Criteria	Feedback
Does the design implement all the performance requirements listed below?	By satisfying the relevant system test cases, the product meets performance requirements.
The product shall be mechanically waterproof, to the point it can sustain a light drizzle for 1 minute of operation while still performing within the requirements.	The product has not been explicitly tested within the rain for operation, but design considerations have been made to mechanically isolate the electronic components. The product is capable of aerial flight during light snow flurries and takeoff/landing in 5cm+ of frozen snow.
The product shall be able to operate and provide data to the operator for > 5 minutes without the need to recharge.	The product is currently capable of only 3.5 minutes of flight outdoors. However, this degradation in battery capacity is due to the cold weather during testing (-5°C and below) and its negative impact on LiPo battery performance.

3.3 Verification and Validation Plan Verification

The VnV Plan has been verified through a review process with an external Capstone group, using the Verification and Validation Plan Verification Plan. All points have been met. However, the external party has noted that there are too many test cases given the flight time of 5 minutes, thus making the tests difficult to execute and would require multiple batteries to be swapped and recharged. Therefore, the team will consider using a larger battery capacity for future iterations.

4 System Testing

This section contains the results of the System Tests outlined within System Test Description. Detailed test procedures can be found within the Test Case Reference provided within the results. Each test case is then either evaluated as a Pass, Principally Pass, Partially Pass, or Fail, as defined within Definitions, Abbreviations, and Acronyms. Some test cases use the SITL environment as defined within Software in The Loop Testing.

4.1 Flight Dynamics

Table 3: STCR_001

Test Case Result ID	STCR_001
Test Case Reference	STC_001
Requirements Tested	GEN_003, GEN_004, STA_000, STA_001, STA_004, STA_005, STA_006, TRANS_002, TRANS_003, TRANS_009, PERF_002
Procedure	Fly the drone indoors in its hovering state for a minute.
Expected Result	The drone should reach its hover height within 25 seconds, and hover within 0.5m of the requested hovering height. The drone should also stay within 1.5m of the launch location.
Actual Result	The drone was flown outdoors using an RC Controller in a guided control mode(i.e. the Parking Lot Hawk which autonomously flies the drone was not used). Within this mode, altitude and location are maintained when no user inputs are given. It successfully flew for 60 seconds and took off in less than 25 seconds. No additional movement commands were sent after takeoff, and the drone maintained a 1.5m radius around the launch location, and height was maintained within 0.5m.
Result Analysis	Pass. The purpose of this test was to verify that the drone is mechanically and electrically able to take off and maintain flight in a simple no-wind indoor scenario. By hovering outdoors (with wind gusts) for 60 seconds, the essence of the test was verified on a more difficult scenario.

Table 4: STCR_002

Test Case Result ID	STCR_002
Test Case Reference	STC_002
Requirements Tested	SAFE_002, GEN_004
Procedure	Attempt to set the height parameters of the drone to invalid values. First, try to set them to be below the minimum height requirement (7m), then try to set the maximum hover height to be less than the minimum hover height.
Expected Result	The drone should prevent the user from entering invalid height parameters both times, and return an error that the height parameters are invalid.
Actual Result	The drone does not fulfill the user's request to set the height parameters, and an error message is logged in the GUI log box.
Result Analysis	Pass.

Table 5: STCR_003

Test Case Result ID	STCR_003
Test Case Reference	STC_003
Requirements Tested	GEN_003, GEN_004, STA_000, STA_001, STA_004, STA_005, STA_006, TRANS_002, TRANS_003, TRANS_009, PERF_002
Procedure	Attempt to take off and hover the drone at the minimum safe hover height (7m) for 60 seconds. After the test, land the drone by asserting m_land.
Expected Result	The drone should launch to 7m within 25 seconds, hover within 0.5m of the 7m height, and laterally stay within 1.5m of the launch location.
Actual Result	This test was performed with the physical drone in an outdoor environment, the drone successfully takes off to 7m. It successfully remains within the height and lateral requirements for the time period specified.
Result Analysis	Pass.

Table 6: STCR_004

Test Case Result ID	STCR_004
Test Case Reference	STC_004
Requirements Tested	TRANS_005, PERF_008, PERF_006, USE_001, PERF_003
Procedure	While in a flight state, set the desired location to be 20m diagonally forward-right, and enter the compulsive move state (by asserting m_CompulsiveMove).
Expected Result	The drone should fly toward the desired location, traveling at least 4 m/sec. A live display of the drone's location and a visual trace of the drone's past location should be visible in the GUI. After reaching the desired location, the drone should remain within 1.5m of the desired location.
Actual Result	The drone successfully flies in a straight line path to the desired location, while satisfying the height, speed, and lateral position requirements. The location trace was visible.
Result Analysis	Pass.

4.2 Battery

Table 7: STCR_005

Test Case Result ID	STCR_005
Test Case Reference	STC_005
Requirements Tested	SR_003
Procedure	Compare the battery capacity displayed on the GUI with the actual battery capacity (obtained by connecting the battery to its charger).
Expected Result	The battery capacity displayed on the GUI should match the true battery capacity displayed on the charger.
Actual Result	The GUI's battery capacity is 2% less than the true battery capacity.
Result Analysis	Pass, the battery capacity error is negligible.

Table 8: STCR_006

Test Case Result ID	STCR_006
Test Case Reference	STC_006
Requirements Tested	SR_011, STA_009 USE_003, SR_003
Procedure	Fly the drone in any flying state.
Expected Result	After a sufficiently long time (at least 3.5 minutes), when the battery capacity remaining is less than 1.5 minutes, the drone will enter the Malfunction state. Upon entering the malfunction state, the drone shall log an error message and land itself at its original launch location.
Actual Result	During physical flight, the drone enters the Malfunction state after about 4 minutes of flight. The battery capacity remaining at his point is 20% (1.5 minutes). The drone logs a low battery message and lands at its launch location.
Result Analysis	Pass.

Table 9: STCR_007

Test Case Result ID	STCR_007
Test Case Reference	STC_007
Requirements Tested	SR_012
Procedure	Attempt to launch the drone when the battery remaining is less than 3 minutes of flight.
Expected Result	The drone does not launch and instead returns a descriptive error message.
Actual Result	The drone remains in the Idle state (does not launch). A low battery error message is seen in the GUI's log box.
Result Analysis	Pass.

4.3 Communication

Table 10: STCR_008

Test Case Result ID	STCR_008
Test Case Reference	STC_008
Requirements Tested	STA_010, TRANS_010
Procedure	During any of the flight states, disconnect the Operator's PC from the Wifi communication network.
Expected Result	The drone should log an error message, return an unhealthy status, and enter the communication lost state.
Actual Result	The drone correctly enters the Communication Lost state and lands upon the communication network being disconnected. A log message and unhealthy status is seen on the Operator's PC.
Result Analysis	Pass.

Table 11: STCR_009

Test Case Result ID	STCR_009
Test Case Reference	STC_009
Requirements Tested	STA_010, TRANS_010
Procedure	During any of the flight states, close the Parking Lot Hawk application running on the Operator's PC.
Expected Result	The drone should log an error message, return an unhealthy status, and enter the communication lost state.
Actual Result	Within SITL, the drone enters the Communication Lost state and lands upon the application being closed. A log message and unhealthy status were seen on the Operator's PC. While testing on the drone but with propellers disconnected and in a flight state, it is observed that the drone enters the Communication Lost state.
Result Analysis	Principally Pass. The purpose of this test was to verify that the drone correctly transitions to the Communication Lost state when the Parking Lot Hawk application is closed, and verify the operation of the Communication Lost state. The operation of the communication lost state was verified in SITL, and the transition into the state was verified using the stationary drone test. Thus the purpose of this test was accomplished. The purpose of this test did not mention any flight dynamics testing, as there are other test cases to verify landing dynamics (STCR_003), thus a real-life flight is not necessary.

Table 12: STCR_010

Test Case Result ID	STCR_010
Test Case Reference	STC_010
Requirements Tested	SR_007, STA_010, TRANS_010
Procedure	Fly the drone forward until Communication is lost, which can be done by choosing a far away desired location and using Compulsive Move.
Expected Result	Upon communication being lost, the drone should enter the Communication Lost state and fly back toward the launch location. While flying back the drone should eventually reconnect to the Operator's PC, and enter the Hover state.
Actual Result	The drone correctly entered and exited the communication lost state. The drone lost communication at about 50m. While flying back, the drone gained communication at about 20m.
Result Analysis	Pass.

4.4 User Error States

Table 13: STCR_011

Test Case Result ID	STCR_011
Test Case Reference	STC_011
Requirements Tested	STA_013, TRANS_005
Procedure	Change m_DesiredUserLocation to a location outside of the parking lot. Assert m_CompulsiveMove to make the drone move to the desired location. A prompt will open asking if the user is sure they want the drone to move a location outside of a parking lot, respond with "OK".
Expected Result	When the user attempts to move to an invalid location, the ParkingLotHawk application running on the Operator's PC should display a prompt, indicating that the desired location is invalid. When the user presses "OK", indicating they still want to fly to the location, the drone should fly toward the desired location of the user.
Actual Result	When attempting to move to a location outside of the parking lot, a prompt appears on the Operator's PC Application. Upon selecting "Yes", the drone flew towards the requested location.
Result Analysis	Pass.

Table 14: STCR_012

Test Case Result ID	STCR_012
Test Case Reference	STC_012
Requirements Tested	STA_003, TRANS_004, STA_011, TRANS_012, STA_008, TRANS_008
Procedure	While flying in a non-parking lot area, attempt to enter the drone Autonomous Explore state. Then fly the drone to a location within the parking lot (using the Compulsive move feature) and attempt to enter the Autonomous Explore state.
Expected Result	The first attempt to Autonomously Explore the parking lot should fail and the drone should enter the No Parking Lot Detected Error state. During this state, the drone should continuously hover, a descriptive error message should appear in c_Log, and c_UserError should change to No_Lot_Detected_State. During the Compulsive Move state, c_UserError should be None. The second attempt to enter the Autonomous Explore state should succeed.
Actual Result	Within SITL, the first attempt to enter the Autonomous Explore state failed. The drone entered the No Parking Lot Error state when the desired location was not within a parking lot. During the No Parking Lot Detected state, the drone was hovering and an error message is seen in c_Log, and c_UserError is No_Lot_Detected_-State. During the Compulsive Move state, c_UserError changed to None. The second attempt to enter the Autonomous Explore succeeded. The test was not performed in real life.
Result Analysis	This test verifies behavior needed for a stretch goal of this project, thus passing it is not necessary for a minimum viable product. However, the test can be declared as principally pass. The purpose of the test was to verify the transition to the Autonomous Explore state, verify the behavior of the compulsive move state moving to a location, and verify the behavior of the No Parking Lot Error state. These activities were verified by the SITL test. The ability of the drone to detect parking boundaries was not a motivation for this test, and rather is the motivation for STC_014.

4.5 Visual Perception and Path Planning

Table 15: STCR_013

Test Case Result ID	STCR_013
Test Case Reference	STC_013
Requirements Tested	GEN_005, GEN_006, STA_003, GEN_001, GEN_002, STA_003, TRANS_004, PERF_004, PERF_001
Procedure	Place a fully charged drone within a small (30 spots) rectangular shaped parking lot. Launch the drone.
Expected Result	The drone should automatically detect the parking lot upon launch and enter the Autonomous Explore state. The drone should fully explore the parking lot within battery capacity. c_CameraView should contain the live view of the drone's camera and should update at 0.5 FPS. The occupancy map should be completed with reasonable accuracy.
Actual Result	The test was not performed.
Result Analysis	This test verifies behavior needed for a stretch goal of this project, thus passing it is not necessary for a minimum viable product. Fail. The autonomous explore feature has not been implemented, it was a stretch goal.

Table 16: STCR_014

Test Case Result ID	STCR_014
Test Case Reference	STC_014
Requirements Tested	GEN_001, SR_009
Procedure	Hold and carry the drone to an area that has grass, then to the middle of a roadway with lane markings, and finally move with the drone to a parking lot.
Expected Result	The Operator's PC should display c_ParkLotDetected as false in all places except when the drone is inside a parking lot.
Actual Result	The Operator's PC displayed c_ParkLotDetected as false in all places except when the drone is inside a parking lot, where it was true.
Result Analysis	Pass.

Table 17: STCR_015

Test Case Result ID	STCR_015
Test Case Reference	STC_015
Requirements Tested	GEN_005, SR_009
Procedure	Hold the drone over a printed picture of a parking lot.
Expected Result	c_CurrentView should contain the printed image in its view.The drone should also identify if the location it is hovering over is occupied by an object or not.
Actual Result	c_CurrentView contains the current image. It correctly detected if it was hovering over a vehicle or empty parking lot space.
Result Analysis	Pass.

Table 18: STCR_016

Test Case Result ID	STCR_016
Test Case Reference	STC_016
Requirements Tested	GEN_001, GEN_002, GEN_005, GEN_006, STA_003, TRANS_-003
Procedure	Launch the drone within a custom-made 3D SITL environment with a small (30 spaces) rectangular parking lot.
Expected Result	The drone should automatically detect the parking lot upon launch, and enter the Autonomous Explore state. The drone should fully explore the parking lot. c_CameraView should contain the live view of the drone's camera, and should update at 0.5 FPS. The occupancy map should be completed with reasonable accuracy.
Actual Result	The test was not performed.
Result Analysis	This test verifies behavior needed for a stretch goal of this project, thus passing it is not necessary for a minimum viable product. Fail. The path planning feature was a stretch goal for this project and was not implemented, thus the Autonomous Explore state is empty (it simply hovers the drone).

4.6 User Manual

The following tests were conducted with the User Manual that is included with the product.

Table 19: STCR_017

Test Case Result ID	STCR_017
Test Case Reference	STC_017
Requirements Tested	STC_012, USE_004
Procedure	Two volunteers who have never used Parking Lot Hawk before reads the user manual, attempt to install the Parking Lot Hawk application, and conduct test case STC_012 independently.
Expected Result	Both volunteers successfully install the application and complete STC_012 without any help from the developers within 2 hours.
Actual Result	The volunteers independently installed the Parking Lot Hawk Application and conducted the STC_012 within 2 hours.
Result Analysis	Pass.

Table 20: STCR_018

Test Case Result ID	STCR_018
Test Case Reference	STC_018
Requirements Tested	SR_002, SR_006, SR_010, SAFE_003
Procedure	Have two participants read through the user manual to verify that various safety requirements are mentioned in the user manual (e.g. weather conditions, drone inspections, holding the drone, password, etc.)
Expected Result	Both volunteers successfully highlighted the sentences where each of the requirements are mentioned in the User Manual.
Actual Result	Both volunteers successfully highlighted the sentences where each of the requirements are mentioned in the User Manual. The highlighted sections with the desired info are in Drone Flight Checklist and Safety.
Result Analysis	Pass.

4.7 Usability

Table 21: STCR_019

Test Case Result ID	STCR_019
Test Case Reference	STC_019
Requirements Tested	STD_001
Procedure	Place the drone on a weighing scale.
Expected Result	The drone weighs less then 25 kg.
Actual Result	The drone weighs less then 25 kg.
Result Analysis	Pass.

Table 22: STCR_020

Test Case Result ID	STCR_020
Test Case Reference	STC_020
Requirements Tested	MTNC_001
Procedure	Discharge the drone's battery completely. Then charge the battery fully.
Expected Result	Recharging should take less then 1 hour.
Actual Result	Recharging took less then 1 hour.
Result Analysis	Pass.

Table 23: STCR_021

Test Case Result ID	STCR_021
Test Case Reference	STC_021
Requirements Tested	SAFE_004, SAFE_001
Procedure	Fly the drone in a parking lot. Survey 5 pedestrians as to whether the flying drone would harm their driving in a serious way.
Expected Result	No participant indicates that the flying drone negatively influences their driving.
Actual Result	No participant indicates that the flying drone negatively influences their driving.
Result Analysis	Pass. The test was completed by flying the drone 20m manually using an RC Controller.

Table 24: STCR_022

Test Case Result ID	STCR_022
Test Case Reference	STC_022
Requirements Tested	USE_002
Procedure	Use the save button on the GUI to save the current camera image (c_CurrentView) and occupancy map (c_OccupancyMap) into a folder of their choosing.
Expected Result	A file explorer opens to let the user pick a folder. An image of the c_CurrentView and c_OccupancyMap are saved into the folder.
Actual Result	The user interface does not contain a save button.
Result Analysis	The save feature was a stretch goal for this project, and its implementation is not necessary for a minimum viable product. Fail.

Table 25: STCR_002

Test Case Result ID	??
Test Case Reference	??
Requirements Tested	??
Procedure	Attempt to arm the drone in a location without pre-downloaded satellite imagery.
Expected Result	The drone should not arm, a descriptive error message should be logged to c_Log.
Actual Result	The drone does not arm, a descriptive error message is logged to c_Log.
Result Analysis	Pass.

5 Unit Testing

This section contains the results of the Unit Tests outlined within Unit Test Description. Detailed test procedures can be found within the Test Case Reference provided within the results. As with the System tests, each test case is either evaluated as a Pass, Principally Pass, Partially Pass, or Fail, as defined within Definitions, Abbreviations, and Acronyms. Some test cases use the SITL environment as defined within Software in The Loop Testing.

5.1 Hardware Hiding

Table 26: UTCR_001

Test Case Result ID	UTCR_001
Test Case Reference	UTC_001
Requirements Tested	–
Procedure	With the propellors removed, connect the battery and slowly raise the throttle on the RC Controller.
Expected Result	Each motor should spin incrementally faster with time. Each motor should also spin in the prescribed direction.
Actual Result	Each motor should spin incrementally faster with time. Each motor should also spin in the prescribed direction.
Result Analysis	Pass.

Table 27: UTCR_002

Test Case Result ID	UTCR_002
Test Case Reference	UTC_002
Requirements Tested	–
Procedure	Connect the battery. Use the RC controller to test takeoff, lateral (forward and backwards), and longitudinal (left and right) movement.
Expected Result	The drone flies up, move forward, backwards, left, then right.
Actual Result	The drone flies up, move forward, backwards, left, then right.
Result Analysis	Pass.

Table 28: UTCR_003

Test Case Result ID	UTCR_003
Test Case Reference	UTC_003
Requirements Tested	–
Procedure	Connect the battery. SSH into the drone and utilize MavROS to read Compass, Barometer and global GPS values.
Expected Result	The heading from the compass, height from the barometer, and position from the GPS matches, within a tolerance specified by the manufacturer, that of external sensors, such as from a smartphone.
Actual Result	The heading from the compass, height from the barometer, and position from the GPS matches, within a tolerance specified by the manufacturer, that of external sensors, such as from a smartphone.
Result Analysis	Pass. Absolute values of height from the barometer are not accurate, but the relative height differences over time are accurate. Therefore, set starting height = 0m before takeoff.

Table 29: UTCR_004

Test Case Result ID	UTCR_004
Test Case Reference	UTC_004
Requirements Tested	–
Procedure	Power the drone through a USB C connector. SSH into the drone and utilize MavROS to read IMU Acceleration values. Move the drone in the x, y, and z directions.
Expected Result	The sign of the acceleration values match that of the movement conducted.
Actual Result	The sign of the acceleration values match that of the movement conducted.
Result Analysis	Pass.

5.2 Operator Camera

Table 30: UTCR_005

Test Case Result ID	UTCR_005
Test Case Reference	UTC_005
Requirements Tested	GEN_002, PERF_004
Procedure	Power the drone through a USB C connector. Pick up the drone so that one's face is within the camera view. Make various facial and hand gestures.
Expected Result	The various gestures are visible on the live camera video feed window created on the Operator's PC. Furthermore, the video feed has a speed of at least 0.5 FPS.
Actual Result	The various gestures are be visible on the live camera video feed window created on the Operator's PC. Furthermore, the video feed has a speed of at least 0.5 FPS.
Result Analysis	Pass.

5.3 Drone Camera

Table 31: UTCR_006

Test Case Result ID	UTCR_006
Test Case Reference	UTC_006
Requirements Tested	GEN_002, PERF_004
Procedure	Power the drone through a USB C connector. Pick up the drone so that one's face is within the camera view. Make various facial and hand gestures.
Expected Result	The various gestures are visible as JPEGs in a folder on the Raspberry Pi. Furthermore, the images has a speed of least 0.5 FPS.
Actual Result	The various gestures are visible as JPEGs in a folder on the Raspberry Pi. Furthermore, the images has a speed of least 0.5 FPS.
Result Analysis	Pass.

5.4 Message Socket

Table 32: UTCR_007

Test Case Result ID	UTCR_007
Test Case Reference	UTC_007
Requirements Tested	PERF_004
Procedure	Power the drone through a USB C connector. Using the Message Socket on the drone, send an empty heartbeat message to the Message Socket on the Operator's PC. Using the Message Socket on the Operator's PC, send the string "{'Type': 'Command', 'Action':'None'}" to the Message Socket on the Drone.
Expected Result	Empty heartbeat messages is printed on the console of the drone, and "{'Type': 'Command', 'Action':'None'}" is printed on the console of the Operator's PC. At least 1 heartbeat message is printed every 2 seconds, and at least 1 "{'Type': 'Command', 'Action':'None'}" message is printed every 2 seconds.
Actual Result	Empty heartbeat messages is printed on the console of the drone, and "{'Type': 'Command', 'Action':'None'}" is printed on the console of the Operator's PC. At least 1 heartbeat message is printed every 2 seconds, and at least 1 "{'Type': 'Command', 'Action':'None'}" message is printed every 2 seconds.
Result Analysis	Pass.

5.5 Interface Hiding

Table 33: UTCR_008

Test Case Result ID	UTCR_008
Test Case Reference	UTC_008
Requirements Tested	USE_005, SR_002
Procedure	Hardcode a blank red image into the Operator's Camera, and hardcode a heartbeat message from the drone.
Expected Result	The live camera window features a blank red image. The User Interface should visually match what is sent in the heartbeat message (location, health, battery capacity, altitude, user error, and drone state).
Actual Result	The live camera window features a blank red image. The User Interface should visually match what is sent in the heartbeat message (location, health, battery capacity, altitude, user error, and drone state).
Result Analysis	Pass.

Table 34: UTCR_009

Test Case Result ID	UTCR_009
Test Case Reference	UTC_009
Requirements Tested	SR_013, SEC_001
Procedure	The user enters an incorrect username and/or password. Then the user enters the correct username and password.
Expected Result	The user can only access the user interface once the correct username and password is entered.
Actual Result	Not login page appears when the GUI is opened.
Result Analysis	Adding the login page was a stretch goal of this project, and implementing it is not necessary for a minimum viable product. Fail.

Table 35: UTCR_010

Test Case Result ID	UTCR_010
Test Case Reference	UTC_010
Requirements Tested	–
Procedure	A volunteer goes through the list of inputs and outputs in the SRS, as well as the specifications specified in the Design Document (MIS), and ensures that there is a means for capturing each input, display each output, and implements the specifications in the MIS.
Expected Result	All inputs, outputs, and specifications are accounted for.
Actual Result	All inputs, outputs, and specifications are accounted for.
Result Analysis	Pass.

5.6 Vision App

Table 36: UTCR_011

Test Case Result ID	UTCR_011
Test Case Reference	UTC_011
Requirements Tested	GEN_005
Procedure	Curate a dataset of at least 50 images of varying weather, lighting, and height conditions (details in the full test case specification). Use the dataset to profile the performance of the algorithm (FPS) on the Raspberry Pi hardware. Also, use the dataset to estimate the classifier's accuracy.
Expected Result	The algorithm processes at least 0.5 frames per second on the Raspberry Pi Hardware. Given the simplicity of the classification task, an accuracy of 70% for assigning the center pixel correctly to either parking lot, nature, or occupied is required.
Actual Result	Among over 50 samples, the classifier takes on average 0.478 seconds with a standard deviation 0.048 sec, to process one frame. The maximum time to process a single frame was 0.68 sec. The histogram of time is shown in Figure 1. The overall accuracy of the classifier was 77%. The classifier performed well on the all classification tasks except for classifying parking lots in sunny conditions. Table 37 breaks down the results by object and weather type. Samples of the images tested are shown in Figure 2.
Result Analysis	Pass. The classifier performs at a sufficient speed to satisfy the 0.5 FPS requirement. However, the accuracy of the classifier was well above the expected 70%.

Figure 1: Occupancy Classifier Performance

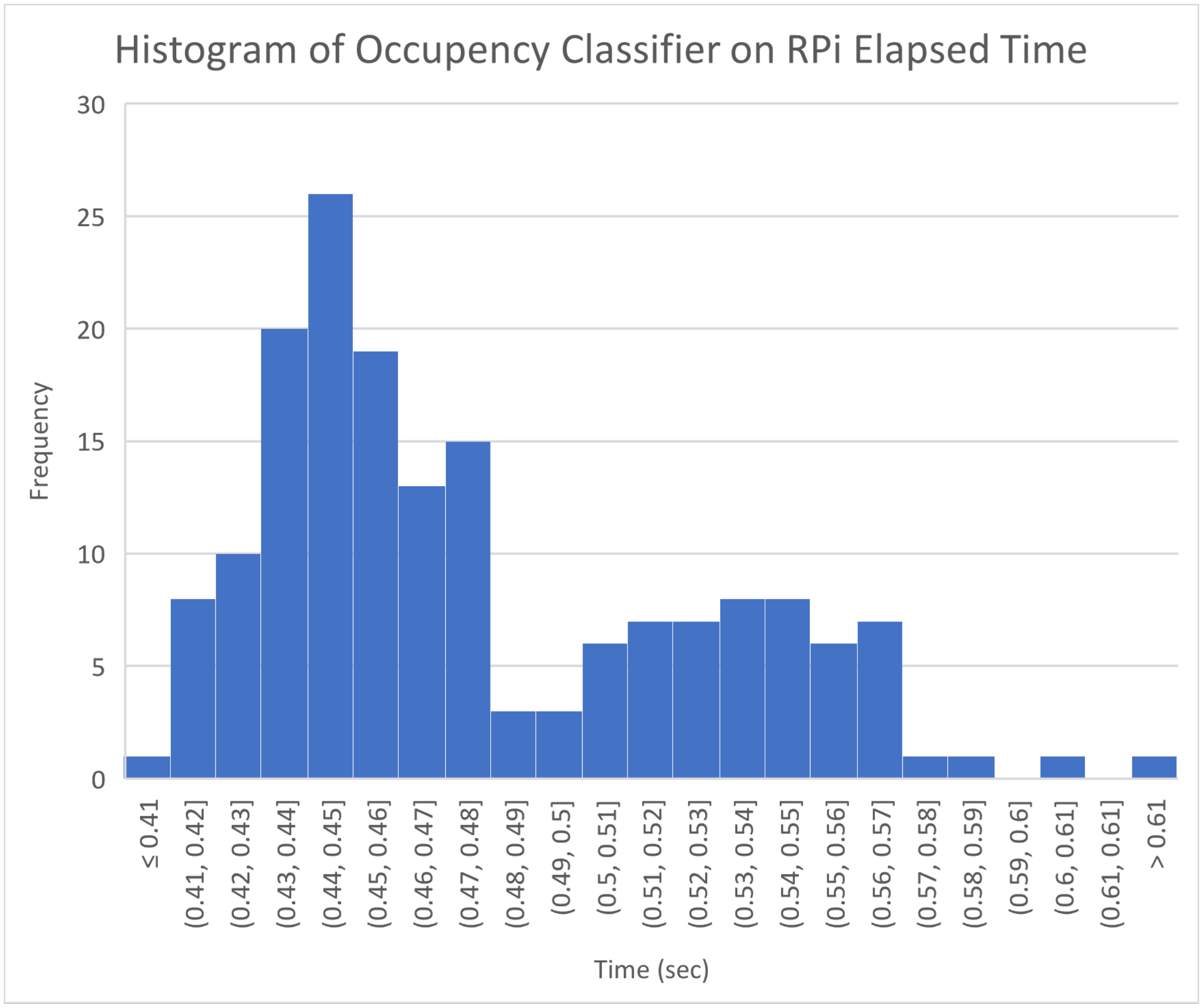


Table 37: Occupancy Classifier Summary

Object Type	Nature	Parking Lot	Occupied (e.g. car, bin, etc.)
Total Accuracy	16/18	12/18	16/21
Accuracy on Sunny Conditions	8/9	4/10	8/10
Accuracy on Overcast Conditions	8/9	8/8	8/11

Figure 2: Sample Images for Sunny Occupied (Left), Sunny Nature (Middle), and Overcast Parking Lot (Right)

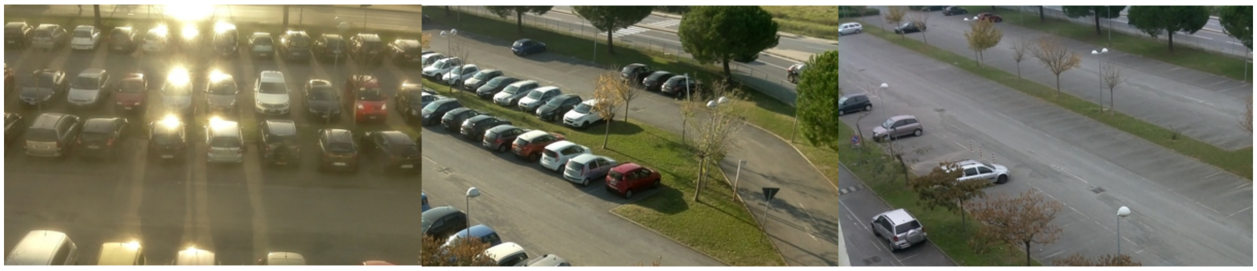


Table 38: UTCR_012

Test Case Result ID	UTCR_012
Test Case Reference	UTC_012
Requirements Tested	GEN_001
Procedure	Measure the accuracy of the segmentation algorithm on 21 satellite images dataset of.
Expected Result	The segmentation algorithm has an average accuracy of above 70%. Accuracy is calculated as pixels correctly classified divided by the total number of pixels.
Actual Result	The segmentation algorithm has an average accuracy of 74% with a standard deviation of 0.18. Individual results per image are shown in Figure 4.
Result Analysis	Pass. From Figure 4, it can be seen that certain images have much poorer accuracy, such as image 8. The segmented result of image 9 is shown in Figure 5. The parking lot is abnormally dark.

Figure 3: Segmentation Sample: Raw image on left, algorithm segmentation on right (white is parking lot, black is non-parking lot)

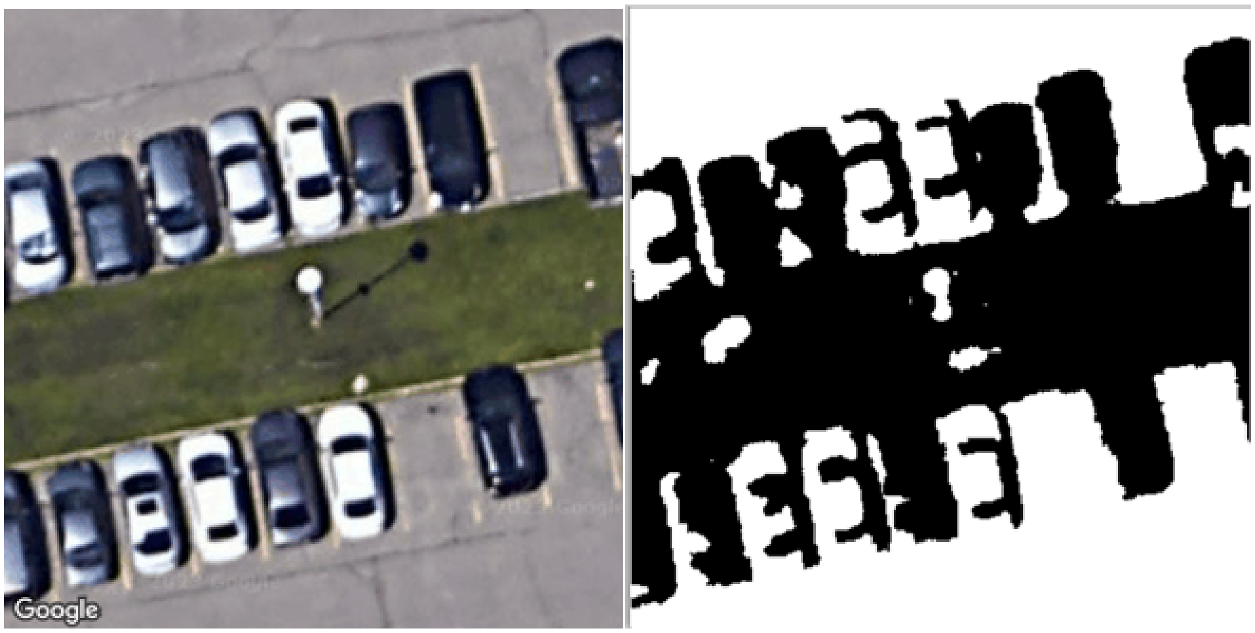


Figure 4: Segmentation Accuracy per Image

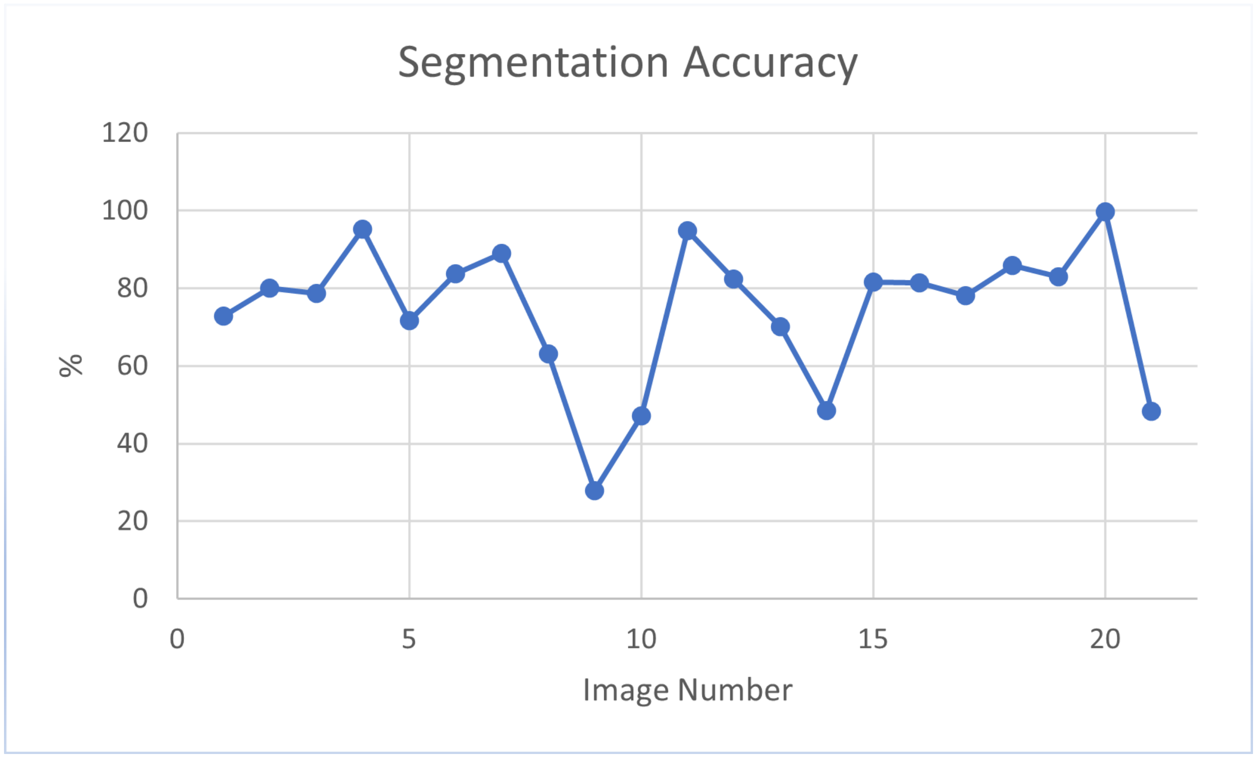
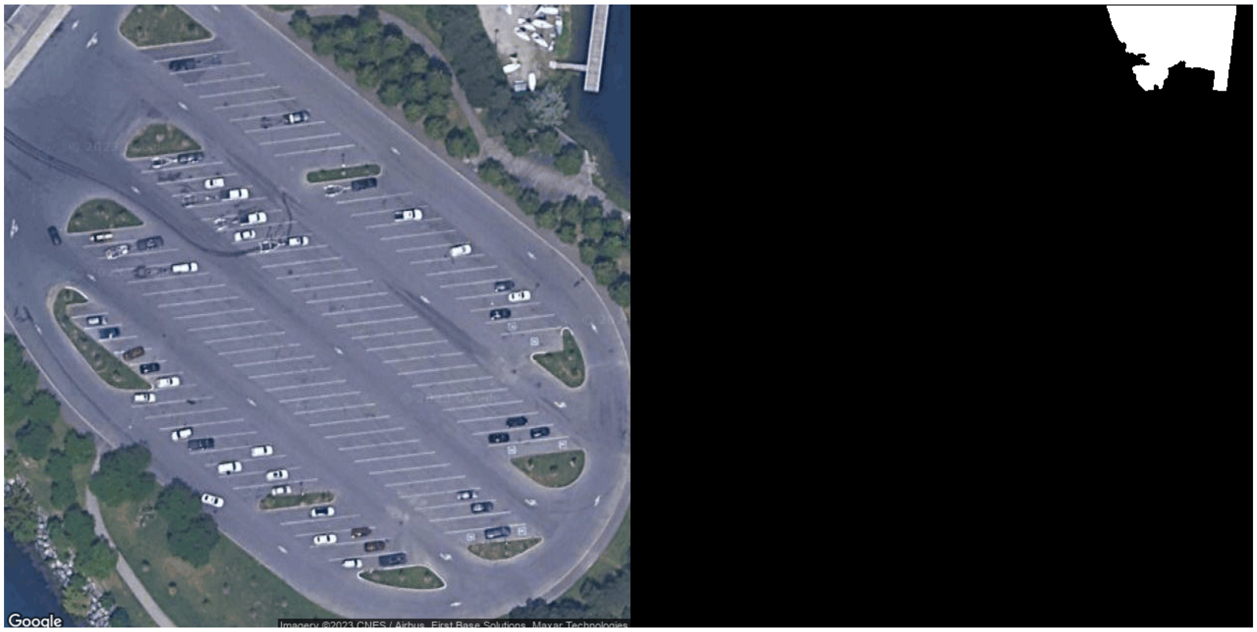


Figure 5: Segmentation Result of Image 8, an abnormally dark road


Google
Imagery ©2023 CNES / Airbus, First Base Solutions, Maxar Technologies

5.7 Mapper App

Table 39: UTCR_013

Test Case Result ID	UTCR_013
Test Case Reference	UTC_013
Requirements Tested	GEN_002
Procedure	Feed a pre-recorded video of the drone flying over a parking lot into the Vision App. Compare the final occupancy map and parking slot map to the actual occupancy and parking slot map of the parking lot in the video.
Expected Result	The estimated occupancy map should be similar to the actual occupancy in the video.
Actual Result	The test was not attempted.
Result Analysis	Occupancy map and parking slot creation was a stretch goal for this project. Fail. This test required recording a video and annotating it, for which there was not enough time.

5.8 Path Plan App

No unit tests. See Path Plan App for details, and System Testing for system tests involving this module.

5.9 Drone Decision and Control Hiding

Table 40: UTCR_014

Test Case Result ID	UTCR_014
Test Case Reference	UTC_014
Requirements Tested	STA_004, GEN_003, TRANS_003, TRANS_015
Procedure	While in the Idle state, send a user command to configure the height parameters to Min_Hover_Params.
Expected Result	The height params within "Params.txt" and the drone is equal to Min_Hover_Params.
Actual Result	The height params within 'Params.txt' and the drone is equal to Min_Hover_Params.
Result Analysis	Pass.

Table 41: UTCR_015

Test Case Result ID	UTCR_015
Test Case Reference	UTC_015
Requirements Tested	STA_000, STA_001, STA_004, STA_011, STA_012, STA_-013, TRANS_003, TRANS_009, TRANS_012, TRANS_013, TRANS_014
Procedure	While in the Idle state, send a user command to arm the drone, takeoff, move 5m to the left, and finally land at the original launch location.
Expected Result	The drone arms, takeoffs to the maximum hover height, moves 5m left, and then lands at the original location.
Actual Result	The drone arms, takeoffs to the maximum hover height, moves 5m left, and then lands at the original location.
Result Analysis	Pass.

Table 42: UTCR_016

Test Case Result ID	UTCR_016
Test Case Reference	UTC_016
Requirements Tested	STA_009, SR_007, SR_011
Procedure	While in the Idle state, send a user command to arm and takeoff. After a sufficient amount of time, the battery level will drop to 20% and enter the Malfunction state and lands.
Expected Result	The drone takeoffs and hovers until battery capacity reaches 20%, at which point it enters the Malfunction state and lands.
Actual Result	This test was conducted within the SITL environment. The drone takeoffs and hovers until battery capacity reaches 20%, at which point it enters the Malfunction state and lands.
Result Analysis	Pass.

Table 43: UTCR_017

Test Case Result ID	UTCR_017
Test Case Reference	UTC_017
Requirements Tested	STA_010, TRANS_010
Procedure	While in the Idle state, send a user command to arm and takeoff. Once max hover height is reached, disconnect the Message Socket.
Expected Result	The drone takeoffs and hovers until the Message Socket is disconnected, at which point it enters the Communication Lost state and lands.
Actual Result	Within SITL, the drone takes off and hovers until the Message Socket is disconnected, at which point it enters the Communication Lost state and lands.
Result Analysis	Pass.

Table 44: UTCR_018

Test Case Result ID	UTCR_018
Test Case Reference	UTC_018
Requirements Tested	STA_008, TRANS_008
Procedure	While in the Idle state, send a user command to arm and takeoff. Then send a user command to enter the Autonomous Explore state while no parking lot is detected.
Expected Result	After the drone takeoffs, it enters the No Parking Lot Detected Error state.
Actual Result	Within SITL, after the drone takes off and receives the Autonomous Explore request, it enters the No Parking Lot Detected Error state.
Result Analysis	Pass.

Table 45: UTCR_019

Test Case Result ID	UTCR_019
Test Case Reference	UTC_019
Requirements Tested	STA_003, TRANS_004
Procedure	While in the Idle state, send a user command to arm, takeoff and finally autonomously explore. Hardcode a suggested path of left 5m from the Path Plan App.
Expected Result	After the drone takeoffs, it enters the Autonomous Explore state and begins to move left 5m.
Actual Result	Within SITL, after the drone takes off within a parking lot, it enters the Autonomous Explore state. It does not move left 5m.
Result Analysis	Fail. The autonomous explore state was a stretch goal of this project, and is not required for a minimum viable product.

5.10 DDC Topic Interface

Table 46: UTCR_020

Test Case Result ID	UTCR_020
Test Case Reference	UTC_020
Requirements Tested	–
Procedure	Provide power to the drone. Hardcode the State Publisher to publish "Unit test 101". Rotate the drone counterclockwise.
Expected Result	"Unit test 101" is published on the currStatePub topic. The DDC Topic Interface's readings of the compass values should match that of the readings from a smartphone when rotated.
Actual Result	"Unit test 101" is published on the currStatePub topic. The DDC Topic Interface's readings of the compass values should match that of the readings from a smartphone when rotated.
Result Analysis	Pass.

5.11 Algorithm Topic Interface

Table 47: UTCR_021

Test Case Result ID	UTCR_021
Test Case Reference	UTC_021
Requirements Tested	–
Procedure	Provide power to the drone. Use the vision app health publisher (visionAppHealth) to publish True. Publish "Unit test 101" on the "current state" topic, and print the drone state being read by the Topic Interface to the console.
Expected Result	"Unit test 101" is printed to the console. The visionAppHealth topic is publishing True.
Actual Result	"Unit test 101" is printed to the console. The visionAppHealth topic is publishing True.
Result Analysis	Pass.

5.12 DDC Service Interface

Table 48: UTCR_022

Test Case Result ID	UTCR_022
Test Case Reference	UTC_022
Requirements Tested	_
Procedure	While in the Idle state, call each of the service routines to set the RtlAlt to 10m, call the mode service to set the mode to "Guided", call the arm service to arm the drone, call the takeoff service to takeoff the drone, and then call the landing service (using the callService_-TypeCommand routine).
Expected Result	The drone takeoffs from the ground to a height of 10m, and then lands.
Actual Result	Within the SITL environment, the drone takeoffs from the ground to a height of 10m, and then lands. Using the physical drone, the test is changed to set RtlAlt to 1m. The drone then takeoffs to 1m and then lands. However, the landing is rough and the drone flips over during landing.
Result Analysis	Principally pass. The test within SITL verifies that the module is working according to its specifications. However, the physical drone performance regarding landing is lacking, and may be due to the poorly tuned control parameters of the drone.

6 Reflection

6.1 Project Assessment

At Revision 0, the quality of the project was inadequate, as the MVP still has yet to be developed. In particular, the autonomous flight software not being tuned to fly autonomously was preventing the project from reaching the MVP milestone. However, there had been significant development in meeting some stretch goals, especially those concerned with visual perception. These are shown within Vision App. The visual perception features however performed below the 70% accuracy requirement.

As of Revision 1, the minimum viable product, having a completed aerial drone that can autonomously fly to a location upon a user click, is complete. Some progress has been made on the stretch goals. The occupancy map feature has been implemented, although its accuracy has never been profiled during a live physical drone test. The visual perception stretch goal was fully implemented, and the unit tests showed their accuracy was now sufficient. The "No Parking Lot Detected" error state was implemented. The Autonomous Explore state was implemented although, functionally it simply hovered at its location (path planning was not implemented).

6.2 Changes Due to Testing

The documentation review yielded several important changes that need to be made. Improvements to the SRS included specifying which module implements the height requirement, and requirements need to be rewritten for the design decision to use predownloaded satellite imagery. This includes specifying where satellite imagery is stored on the Operator's PC, specifying how the user can specify a location to download satellite imagery, and what happens when the drone flies in an area without predownloaded satalite imagery. Furthermore, due to changes in design, the definition of c_CurrentView needs to be changed to be just the live camera view without annotations as opposed to an annotated camera stream. Changes to the FSM are also required as the state machine did not account for when software algorithm modules fail to execute, in addition to the changes made to the compulsive move state. The VnV Plan survey also revealed that multiple spare batteries, or larger capacity batteries, should be purchased to facilitate the testing process.

The test cases also revealed issues with the implementation at that time. For example a visual trace of the drone's previous path in the past 10 seconds was not visible in the GUI, and the accuracy of the visual perception algorithm (both segmentation and classification) is poor and not robust, thus should be improved.

7 Automated Testing

Many automated testing tools were used to automate the testing and verification process, as outlined within the Automated Testing and Verification Tools. For consistency within the team, VS Code is used as the Integrated Development Environment, with Pylint and clang-tidy for Python and C++ linters respectively. As outlined within System Test Description and Unit Test Description, many of the test cases are automatic and are conducted through the UnitTest framework. A list of these automatic test cases are included in Table 49. In addition to unit testing frameworks, a custom evaluation tool is used to automatically execute and generate metrics for the Vision App test cases, such as within Occupancy Classifier Summary.

Table 49: Automated Test Cases

Visual Perception and Path Planning	STC_015, STC_016
Message Socket	UTC_007
Interface Hiding	UTC_008, UTC_009, UTC_010
Vision App	UTC_011, UTC_012
Mapper App	UTC_013
Drone Decision and Control Hiding	UTC_014, UTC_015, UTC_016, UTC_017, UTC_018, UTC_019
DDC Topic Interface	UTC_020
Algorithm Topic Interface	UTC_021
DDC Service Interface	UTC_022

8 Trace to Requirements

Table 50: FR Traceability Table

Functional Requirement	Test Case to Verify
GEN_001	STCR_014, STCR_016, UTCR_012
GEN_002	STCR_013, STCR_016, UTCR_005, UTCR_006, UTCR_013
GEN_003	STCR_001, STCR_003, UTCR_014
GEN_004	STCR_001, STCR_002
GEN_005	STCR_013, STCR_016, STCR_015, UTCR_011
GEN_006	STCR_013, STCR_016
STA_000	STCR_001, STCR_002, STCR_003, UTCR_015
STA_001	STCR_001, STCR_002, STCR_003, UTCR_015
STA_003	STCR_016, STCR_012, STCR_013, UTCR_019
STA_004	STCR_001, STCR_002, STCR_003, UTCR_014, UTCR_015
STA_005	STCR_001, STCR_002, STCR_003
STA_006	STCR_001, STCR_002, STCR_003
STA_008	STCR_012, UTCR_018
STA_009	STCR_006, UTCR_016
STA_010	STCR_008, STCR_009, STCR_010, UTCR_017
STA_011	STCR_012, UTCR_015
STA_012	STCR_001, STCR_003 UTCR_015
STA_013	STCR_001, STCR_003, UTCR_015

Functional Requirement	Test Case to Verify
TRANS_001	STCR_002, STCR_003
TRANS_002	STCR_001, STCR_002, STCR_003
TRANS_003	STCR_001, STCR_002, STCR_003, UTCR_014, UTCR_015
TRANS_004	STCR_013, STCR_012, UTCR_019
TRANS_005	STCR_004, STCR_011
TRANS_006	STCR_011
TRANS_007	STCR_012, STCR_013
TRANS_008	STCR_011, UTCR_018
TRANS_009	STCR_001, STCR_002, STCR_003, UTCR_015
TRANS_010	STCR_008, STCR_009, STCR_010, UTCR_017
TRANS_011	STCR_008, STCR_009, STCR_010
TRANS_012	STCR_012, STCR_010, UTCR_015
TRANS_013	STCR_001, STCR_003, UTCR_015
TRANS_014	STCR_001, STCR_003, UTCR_015
TRANS_015	STCR_003, UTCR_014

Table 51: NFR Traceability Table

Non-Functional Requirement	Test Case to Verify
PERF_001	STCR_013
PERF_002	STCR_001, STCR_002, STCR_003
PERF_003	STCR_004
PERF_004	STCR_013, UTCR_007
PERF_005	STCR_002, STCR_003, UTCR_005, UTCR_006
PERF_006	STCR_004
PERF_007	STCR_015
PERF_008	STCR_004
DES_001	None, it is a pass or fail depending on components already bought.
STD_001	STCR_019
STD_002	None, it is a pass or fail depending on components already bought.
SEC_001	UTCR_009
SEC_002	None, pass or fail.
MTNC_001	STCR_020
MTNC_002	None, may damage expensive parts. In the interest of cost, this is not tested.
MTNC_003	None, may damage expensive parts. In the interest of cost, this is not tested.
SAFE_001	STCR_021
SAFE_002	STCR_002
SAFE_003	STCR_018
SAFE_004	STCR_021
SAFE_005	None, pass or fail dependent on components bought.
USE_001	STCR_004
USE_002	STCR_022
USE_003	STCR_006
USE_004	STCR_017
USE_005	UTCR_008

Non-Functional Requirement	Test Case to Verify
SR_002	STCR_018, UTCR_008
SR_003	STCR_005, STCR_006
SR_004	None, pass or fail determined by the components bought.
SR_005	None, pass or fail determined by the components bought.
SR_006	STCR_018
SR_007	STCR_010, UTCR_016
SR_008	None, redundant localization without gps
SR_009	STCR_015, STCR_014
SR_010	STCR_018
SR_011	STCR_006, UTCR_016
SR_012	STCR_007
SR_013	UTCR_009

9 Trace to Modules

This section summarizes the traceability between Test Cases and Modules, see Table 52.

Table 52: Traceability Between Test Cases and Modules

Hardware Hiding	STCR_001, STCR_002, STCR_003, STCR_004, STCR_005, STCR_006, STCR_007, UTCR_001, UTCR_002, UTCR_003, UTCR_004
Operator Camera	UTCR_005
Drone Camera	UTCR_006
Message Socket	STCR_001, STCR_002, STCR_003, STCR_004, STCR_008, STCR_009, STCR_010, UTCR_007
Interface Hiding	STCR_022, UTCR_008, UTCR_009, UTCR_010
Vision App	UTCR_011, UTCR_012
Mapper App	UTCR_013
Path Plan App	UTCR_016
Drone Decision and Control Hiding	STCR_001, STCR_002, STCR_003, STCR_004, UTCR_014, UTCR_015, UTCR_016, UTCR_017, UTCR_018, UTCR_019
DDC Topic Interface	STCR_011, STCR_012, UTCR_020
Algorithm Topic Interface	STCR_001, STCR_002, STCR_003, STCR_004, STCR_013, STCR_014, STCR_015, STCR_016, UTCR_021
DDC Service Interface	UTCR_022

10 Symbolic Constants

See Symbolic Constants.

Appendix — Reflection

Verification and Validation (VnV) is a critical process in software development which involves assessing the quality and correctness of the software. The VnV plan outlines the strategies and methodologies that will be employed to verify and validate the software. However, in practice, there may be differences between the VnV plan and the actual VnV activities conducted. These differences may result from unforeseen circumstances, changes in requirements, or other factors.

Many of the test cases involving flight were done within the SITL environment as opposed to with the physical drone. This is due to the incomplete tuning process of the drone, thus preventing autonomous flight. The main cause for the incomplete tuning is due to the extensive damage incurred during testing, and the additional time required to repair and rebuild the drone. However, parallel development continued on the software allowing for nearly all test cases to be tested at least within the SITL environment. Although the extent of damage incurred could not be predicted, additional time could be allocated within the project scheduling for testing to allow a larger buffer for unexpected circumstances.

Another source of change in the VnV Plan is due to the changing requirements, as mentioned within Changes Due to Testing. An example would be the change in definition of c_CurrentView from being an annotated camera stream to an unoannotated one, which was a result of a design change made to simplify the communication infrastructure. Although such changes to the requirements could not be predicted precisely, certain requirements are labeled as requirements likely to change. Thus, test cases involving these requirements could be made with flexibility in mind such that changing the requirement would only require a small change within the test case procedure or expected result.

In conclusion, deviations from the VnV plan are common in software development projects. These deviations may result from changes in requirements, unexpected issues, or other factors. To anticipate changes and minimize the impact of these deviations, project managers must engage in continuous monitoring and evaluation of the development process, regularly review the VnV plan, and remain open to feedback from team members, stakeholders, and end-users.