Team 1:

Indoor UAV Tracking System

Test Review

Supervisor: Prof. Costas Armenakis
Industry Advisor: P.Eng. Lui Tai

March 19, 2018

Kevin Arindaeng (213094016)
Ariel Laboriante (212951984)
Zhuolin (Jack) Lu (212848834)
Varsha Ragavendran (213193065)

Table of Contents

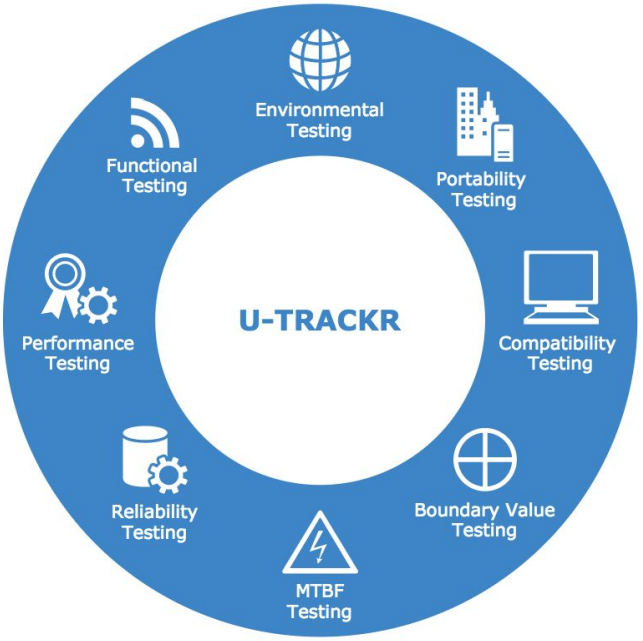
1. Introduction 3
2.1. Test Summary 5
Table 1. Summary of Functional Tests (FUNC) 6
Table 2. Summary of Performance Tests (PER) 7
Table 3. Summary of Reliability Tests (REL) 7
Table 4. Summary of Mean Time Before Failures Test (MTBF) 8
Table 5. Summary of Boundary Value Tests (BVT) 8
Table 6. Summary of Compatibility Tests (COM) 9
Table 7. Summary of Portability Tests (POR) 9
Table 8. Summary of Environmental Test (ENV) 9
2.2. Test Results 10
2.3. Test Analysis and Discussion 11
2.3.1. Functional 11
2.3.2. Performance 12
2.3.3. Mean Time Before Failure 12
2.3.4. Environment 13
2.3.5. Reliability 13
2.3.6. Compatibility 14
2.3.7. Portability 14
2.3.8. Boundary Value 15
2.4. Omitted Test Cases 15
2.4.1. PER-02 and PER-03 15
2.4.3. MTBF-02 15
2.4.4. COM-02 16

3. Self-Assessment of the Test Review **16**

4. Appendix **17**
4.1 Appendix A - Functional Testing 17
4.2 Appendix B - Performance Testing 20
4.3 Appendix C - Reliability Testing 22
4.4 Appendix D - Mean Time Before Failure Testing 23
4.5 Appendix E - Boundary Value Testing 25
4.6 Appendix F - Compatibility Testing 26
4.7 Appendix G - Portability Testing 27
4.8 Appendix H - Environment Testing 28

1. Introduction

The goal of this project was to successfully track the trajectory of UAVs in an indoor setting using image-processing and photogrammetry. Our system improves the efficiency and cost-effectiveness of UAVs by providing positioning technology and indoor navigation. To verify whether the U-TRACKR system is fit for its intended purpose, our group conducted verification tests which evaluated our system in eight areas that we established in our TRR:



Functional Testing	Establishes that our system's basic functions such as WiFi, SSH connection, and video stream are working
Performance Testing	Ensures that our data is transmitted at real time by checking the quality of the video stream in terms of FPS, CPU usage, camera resolution, and stream bitrate
Reliability Testing	Checks the system's accuracy and precision when locating objects in a defined space
MTBF Testing	Finds the mean (average) time between failures of the system during normal operation
Boundary Value Testing	Tests the system's accuracy in locating objects at the maximum and minimum distances
Compatibility Testing	Assesses the system's ability to detect different subjects and different colours
Portability Testing	Evaluates the system's ability to function if the hardware components were changed
Environmental Testing	Tests system functions when environment settings such as lux levels are changed

Figure 1. Summary of tests that validates the U-TRACKR system design

In this deliverable, we describe the test procedures in detail, analyze the test results, and justify any failures in the system. After having tested the system, we plan to improve the frames of the U-TRACKR to improve stability, and extend the system to track multiple objects. Given the time, we also plan on customizing the U-TRACKR so that it can be applied in two other fields:

- People counter
  - The U-TRACKR can keep track of how many people enter and leave a facility
  - Statistical information can be used to optimize store hours or display location
  - Cameras can be mounted at the entrances of malls and museums to see where people tend to visit first
- Animals in captivity
  - The U-TRACKR system can track animal lifestyle and behaviour
  - The system minimizes human interaction and human disturbances

2. Test Verification and Validation

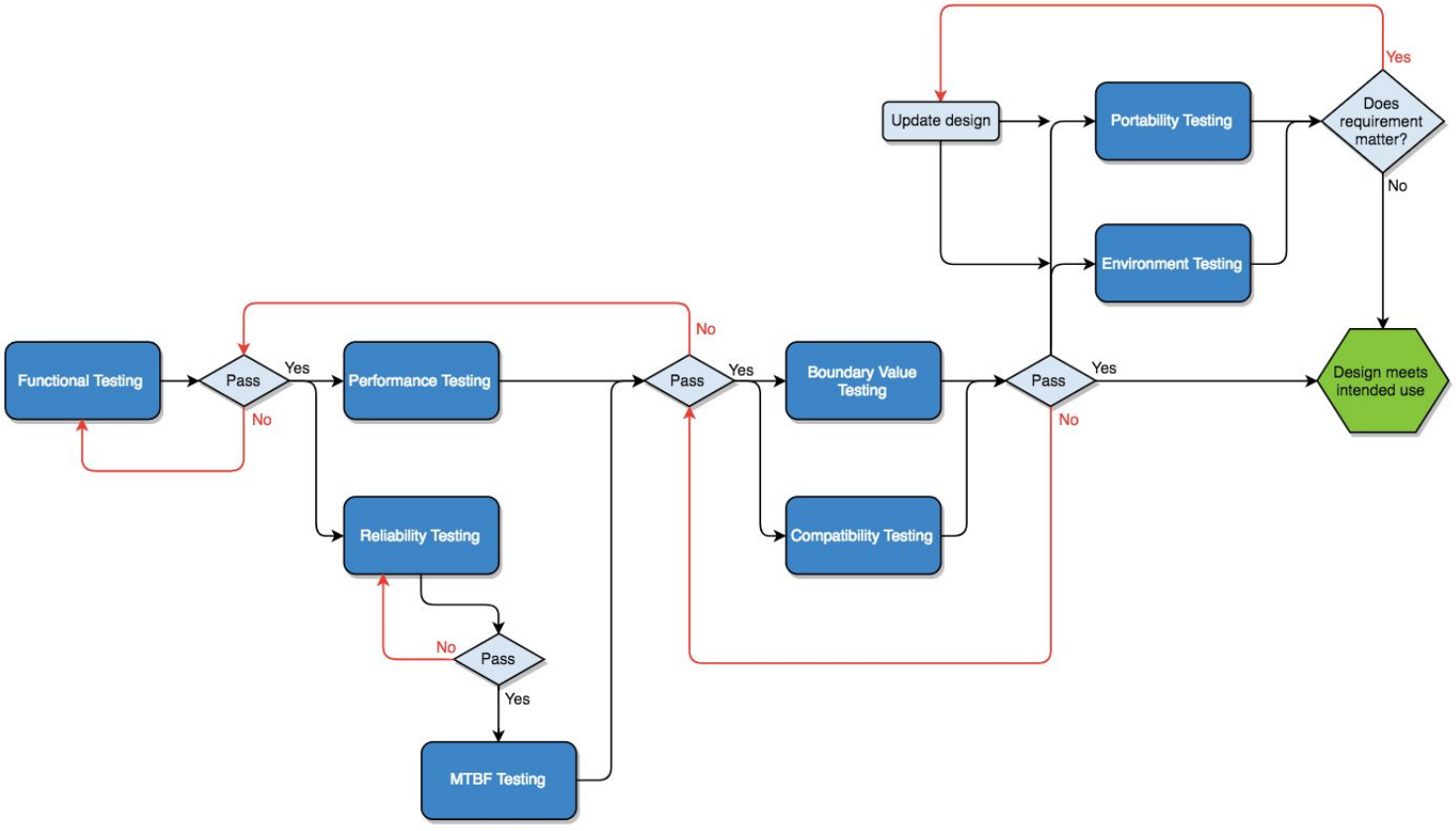


Figure 2. Sequence of tests carried out during system verification and validation

The tests were conducted in a specific sequence shown in Figure 2. Design updates were made throughout the testing process whenever the U-TRACKR design failed to meet minimum system requirements.

First, it was necessary that the system passes the Functional Testing phase since other tests rely on the system's basic functions. For example, the connection to the Raspberry Pi microcontroller must be established before the position of a test object can be calculated. The Performance and Reliability Testing phases were done in parallel since it was possible to check FPS, CPU usage, camera resolution, stream bitrate, and the accuracy of position calculations independently from each other. The MTBF Testing phase depended on the success of the Reliability Test since the system must begin in a working state before a state of failure can be determined. Furthermore, the Boundary Value and Compatibility Testing phases can only be carried out if the accuracy of the system's position calculations have been verified. The Portability and Environmental Testing phases were carried out near the end of the entire testing process since they were not critical requirements in validating the U-TRACKR's design.

2.1. Test Summary

For a more detailed description of the procedures in each test, refer to the following sections in the Appendix:

Appendix A - Functional Testing (FUNC)

Appendix B - Performance Testing (PER)

Appendix C - Reliability Testing (REL)

Appendix D - Mean Time Before Failures Testing (MTBF)

Appendix E - Boundary Value Testing (BVT)

Appendix F - Compatibility Testing (COM)

Appendix G - Portability Testing (POR)

Appendix H - Environmental Testing (ENV)

Table 1. Summary of Functional Tests (FUNC)

Test ID	Procedure	Test Type/ Tools Used	Comments	Limitations and Constraints
FUNC-01	Verify that each Raspberry Pi can connect to the same WiFi network as the main controller (laptop)	Automated Python Unit Test Paramiko Framework	Realistic test since the same process applies to different WiFi networks. Repeatable test since connection can be disabled and enabled. Test yields the same result every time.	Limitations: None Constraints: Connection was only tested at WiFi speeds over 500 mbps
FUNC-02	Verify the ability to establish an SSH connection from the main controller to each Raspberry Pi.	Automated Python Unit Test Paramiko Framework	Realistic test since the main controller will invoke and process the video feed in the same process, regardless of the type of WiFi network and environment setting (whether it be the scaled prototype or a real-life setting such as a warehouse). Repeatable test since connection can be enabled and disabled. Test yields the same result every time.	Limitations: None Constraints: Connection was only tested at WiFi speeds over 500 mbps
FUNC-03	Verify the ability to stream the camera feed from the Raspberry Pi's to the main controller from each Raspberry Pi's at the same time.	Automated Python Unit Test Paramiko Framework	Realistic test since the main controller may receive the video feed coming in from any of the system's four cameras, and at different times due to disruptions in the SSH connections. Repeatable test since the main controller can invoke the ability to stream the video feeds from the cameras at one at a time, or all at once. Test yields the same result every time.	Limitations: None Constraints: Quality of camera feed depends on the WiFi network quality and room lighting
FUNC-04	Verify that OpenCV software runs with no issues on main controller and identifies objects within the frame.	Automated Python Unit Test OpenCV software	Realistic test as objects of various colours and shapes were used as camera subjects to test OpenCV. The U-TRACKR program will not be able to identify and track objects without the use of OpenCV. Repeatable test since we used a python script to verify if OpenCV successfully identified and tracked the object within the frame. The test successfully ran with every target object.	Limitations: OpenCV software can run only if successful SSH connection is established and camera feed is streamed to the main controller. Constraints: WiFi network quality and room lighting
FUNC-05	Verify that the U-TRACKR program executes the Python script that determines the X, Y, Z coordinates of the object within in the frame.	Automated Python Unit Test OpenCV software	Realistic test as objects can be anywhere within our scaled version of the project, as this is the defined space for tracking. Therefore, this depicts objects being anywhere within an indoor space. The U-TRACKR program should be able to track and provide X, Y, Z coordinates of the object. Repeatable test since we used a python script to verify if the output coordinates were produced by the system when the target object was placed within the frame. The test was successful at every run.	Limitations: System can only track objects within the frame. OpenCV must be running, in order to identify and determine the X, Y, Z coordinates. Constraints: WiFi network quality and room lighting

Table 2. Summary of Performance Tests (PER)

Test ID	Procedure	Test Type/ Tools Used	Comments	Limitations and Constraints
PER-01	Check if each Raspberry Pi CPU usage meets the specified requirements	Automated Python Unit Test	Realistic test to ensure that our system operates properly and close to real-time since higher CPU usage indicates bugs or a slow system. Repeatable test since a python script was created to verify CPU usage. Test meets minimum requirements and yields the same result almost every time.	Limitations: Raspberry Pi clock speed Constraints: WiFi network quality
PER-04	Check if each Raspberry Pi stream bitrate meets the specifications	Automated Python Unit Test	Realistic test as the bitrate of the Raspberry Pi is a dynamic variable that can differ at certain instances of time, depending on the configurations and environment setting. Repeatable test since the bitrate configurations can be modified for every test run. The preferred configuration is chosen to be tested multiple times. Test yields the same result every time.	Limitations: Raspberry Pi clock speed Constraints: WiFi network quality

Table 3. Summary of Reliability Tests (REL)

Test ID	Procedure	Test Type/ Tools Used	Comments	Limitations and Constraints
REL-01	Check if the U-TRACKR program will not crash if one or more of the Raspberry Pi camera breaks down.	Manual Integration Test Test object, target markers	Realistic test since malfunctions in one of the four cameras will deem that camera useless. It will be as if the system has only three cameras. Repeatable test since connections from each camera can be disabled one at a time. Test yields the same result during multiple tries.	Limitations: At least two working cameras are required to produce accurate position calculations. Current program requires all cameras to be working. An exception will be thrown if one camera fails. Constraints: None
REL-02	Verify the accuracy of the position calculated by the U-TRACKR program.	Manual Integration Test Test object, ruler, target markers	Realistic since the actual position of the test object was measured using the ruler and test markers. Output of the program was compared to actual measurements with a 0.5 mm uncertainty. Repeatable test since the object's position in the frame was known in each test. The U-TRACKR program successfully calculated the coordinates.	Limitations: System can only track objects within the frame. Only one object can be tracked at a time. Constraints: WiFi network quality, room lighting

Table 4. Summary of Mean Time Before Failures Test (MTBF)

Test ID	Procedure	Test Type/ Tools Used	Comments	Limitations and Constraints
MTBF-01	Determine the mean time before failures of the U-TRACKR program while running for at least 8 hours.	Manual Integration Test Floor markers, test subject, ruler	Realistic test as the system would need to operate during working hours in a real-life application setting. Not repeatable since our program was unable to run longer than an hour. Test did not produce the expected results.	Limitations: X, Y, Z calculation we inconsistent. Unexpected exceptions were thrown while program is running. System overheats and becomes too slow. Constraints: WiFi network quality

Table 5. Summary of Boundary Value Tests (BVT)

Test ID	Procedure	Test Type/ Tools Used	Comments	Limitations and Constraints
BVT-01	Determine the U-TRACKR program is functional when tracking an object at the farthest distance specified by the requirements.	Manual Integration Test Floor markers, test subject, measuring tape	Realistic test since objects must be successfully tracked up to the edges of our defined 3D space within the system frame. Repeatable test since the subject's position was defined using floor markers. Test yields the same results every time.	Limitations: Maximum distance of subject from the camera was limited to the frame space. Constraints: A measuring tape was used so measurements had a 0.5 cm uncertainty.
BVT-02	Determine the U-TRACKR program is functional when tracking an object at the shortest distance specified by the requirements.	Manual Integration Test Floor markers, test subject, ruler	Realistic test as the camera subject can be very close to the camera within the defined space for tracking. Since the Raspberry Pi is a fixed focus module and gives sharper images at 60 cm, it was important to ensure that the tracking feature is still functional at our minimum threshold. Repeatable test since the subject can be brought to a specific distance to the camera each time using a ruler. Test yields similar results every time.	Limitations: Depending on the size of the subject, the camera's angular view can be entirely covered by the subject at a minimum distance so it was impossible to know if the system was actually still tracking. Constraints: Camera subject can physically only be brought up to 1 cm in front of the camera before light is blocked. A ruler was used instead of the vernier caliper so measurements had a 0.5 mm uncertainty.

Table 6. Summary of Compatibility Tests (COM)

Test ID	Procedure	Test Type/ Tools Used	Comments	Limitations and Constraints
COM-01	Determine that the U-TRACKR program can identify objects using the full range of values (ex. HSV colours)	Manual Integration Test Floor markers, Different colour test subjects, ruler	Realistic test since system may be required to track objects of various colours within the indoor space. The U-TRACKR program was tested over a range of HSV colours. Repeatable since each test subject tracked by the system has a certain HSV color space value that can be replicated.	Limitations: Subject within the frame can only be tracked by color, and multiple subjects of same color cannot be distinguished from one another. Constraints: WiFi network quality, room lighting, limited time to manually test all possible colours

Table 7. Summary of Portability Tests (POR)

Test ID	Procedure	Test Type/ Tools Used	Comments	Limitations and Constraints
POR-01	Determine the level of ease to apply the U-TRACKR program to other hardware applications.	Manual Integration Test Laptop webcam, floor markers, test subject, ruler	Realistic test since different cameras may be better suited for different needs, like longer focus or larger angular view. The U-TRACKR program should operate and detect the coordinates of the objects, regardless of the type of hardware, provided a video stream can be fed to the main controller. Repeatable test since our program was able to operate and track the object using the webcam of a laptop.	Limitations: U-TRACKR program would have to be modified slightly to invoke different commands depending on the hardware. Constraints: Hardware should have the ability to allow main controller to establish an SSH connection and invoke commands to retrieve video feed.

Table 8. Summary of Environmental Test (ENV)

Test ID	Procedure	Test Type/ Tools Used	Comments	Limitations and Constraints
ENV-01	Determine the U-TRACKR program is functional within a specific lux threshold	Manual Integration Test Lux Light Meter app, test subject (tennis ball), external light source	Realistic test since we tested from 0-500 lux, where 500 lux is the normal illumination for labs and office work. System should be able to track objects between 100-500 lux. Repeatable as settings can be reset to test the full range of light that comes into the frame.	Limitations: System breaks down at 30 lux since there is not enough light entering the camera lens. Constraints: Test was limited to 300 lux emitted by external light source.

2.2. Test Results

Table 9. Test Results

Test ID	Results (with screenshot)	Expected Results	Actual Results

FUNC-01	PASS	`Ran 1 test in XX.XXXs`
		
		`OK`
			`Ran 1 test in 2.498s`
			
			`OK`

FUNC-02	PASS	`Ran 1 test in XX.XXXs`
		
		`OK`
			`Ran 1 test in 2.116s`
			
			`OK`

FUNC-03	PASS	`RPi Timestamp 1: AAAA-BB-CC XX:YY:ZZ`
		`RPi Timestamp 2: AAAA-BB-CC XX:YY:ZZ`
		`RPi Timestamp 3: AAAA-BB-CC XX:YY:ZZ`
		`RPi Timestamp 4: AAAA-BB-CC XX:YY:ZZ`
		
		`Ran 1 test in XX.XXXs`
		
		`OK`
			`RPi Timestamp 1: 2018-03-17 18:23:21`
			`RPi Timestamp 2: 2018-03-17 18:23:21`
			`RPi Timestamp 3: 2018-03-17 18:23:21`
			`RPi Timestamp 4: 2018-03-17 18:23:21`
			
			`Ran 1 test in 21.868s`
			
			`OK`

FUNC-04	PASS	`RPi 1 (x,y): (x1,y1)`
		`RPi 2 (x,y): (x2,y2)`
		`RPi 3 (x,y): (x3,y3)`
		`RPi 4 (x,y): (x4,y4)`
		
		`Ran 1 test in X.XXXs`
		
		`OK`
			`RPi 1 (x,y): (253.959457397,224.121627808)`
			`RPi 2 (x,y): (222.5,273.0)`
			`RPi 3 (x,y): (255.812332153,226.267349243)`
			`RPi 4 (x,y): (257.5,208.5)`
			
			`Ran 1 test in 5.002s`
			
			`OK`

FUNC-05	PASS	`Position Calc (X,Y,Z): [x],[y],[z]`
		
		`Ran 1 test in X.XXXs`
		
		`OK`
			`Position Calc (X,Y,Z): [-144.66784341], [211.91764875],[0.72982254]`
			
			`Ran 1 test in 5.192s`
			
			`OK`

PER-01	PASS	`RPi 1 CPU Usage: AA`
		`RPi 2 CPU Usage: BB`
		`RPi 3 CPU Usage: CC`
		`RPi 4 CPU Usage: DD`
		
		`Ran 1 test in X.XXXs`
		
		`OK`
			`RPi 1 CPU Usage: 16`
			`RPi 2 CPU Usage: 16`
			`RPi 3 CPU Usage: 21`
			`RPi 4 CPU Usage: 16`
			
			`Ran 1 test in 2.998s`
			
			`OK`

PER-04	PASS	`RPi 1 Bit Rate: 25000000`
		`RPi 2 Bit Rate: 25000000`
		`RPi 3 Bit Rate: 25000000`
		`RPi 4 Bit Rate: 25000000`
		
		`Ran 1 test in X.XXXs`
		
		`OK`
			`RPi 1 Bit Rate: 25000000`
			`RPi 2 Bit Rate: 25000000`
			`RPi 3 Bit Rate: 25000000`
			`RPi 4 Bit Rate: 25000000`
			
			`Ran 1 test in 2.066s`
			
			`OK`

MTBF-01	FAIL	The U-TRACKR system runs for at least 8 hours.	The U-TRACKR system throws an exception and stops tracking within a minute.
ENV-01	FAIL	The U-TRACKR system can run from dim indoor lighting (~5 lux) to bright indoor lighting (~100 lux).	The U-TRACKR system fails to track in dim light, as it incorrectly detects the whole screen as the object. The U-TRACKR system fails to track often in bright indoor lighting, as it sometimes fails to detect the object.
REL-01	PARTIAL PASS	The U-TRACKR system will keep running if one or more Raspberry Pi Cameras stops functioning.	The U-TRACKR system continues to run after one or more of the cameras becomes unoperational. However, it throws an exception.
REL-02	FAIL	The U-TRACKR system will provide accurate data (within ~1cm) on the position calculation of an object relative to the frame.	The U-TRACKR system provides data, however it is widely inaccurate, even for a stationary object located at (0,0,0).
COM-01	PASS	The U-TRACKR system can detect various objects with a full range of HSV values in standard lighting.	The U-TRACKR system can detect an array of colours, with respect their HSV lower and upper bounds.
POR-01	PASS	The U-TRACKR system can be used regardless of the specifications of a camera.	The U-TRACKR system is able to run on other camera systems such as laptop webcams.
BVT-01	PASS	The U-TRACKR system can function (objects can be detected) when extended beyond the frame.	The U-TRACKR system is able to track objects up until 201.526cm (around 6.8 feet) compared to the default frame distance of 99.62cm
BVT-02	PASS	The U-TRACKR system can function (objects can be detected) when the object is within 5cm of the cameras.	The U-TRACKR system is able to track objects as close as 5cm.

2.3. Test Analysis and Discussion

2.3.1. Functional

Overall, the test cases that were performed for the functional testing portion of the system passed successfully. These test cases consisted of fundamental components that have to be operating successfully in order for the U-TRACKR program to operate as intended to and also as specified in the requirements. These test cases are in place to ensure that the U-TRACKR program will be able to establish an SSH connection to all four raspberry pi's on the same WiFi network and invoke commands to retrieve the video feed, in order to process these feeds to identify and perform position calculations.

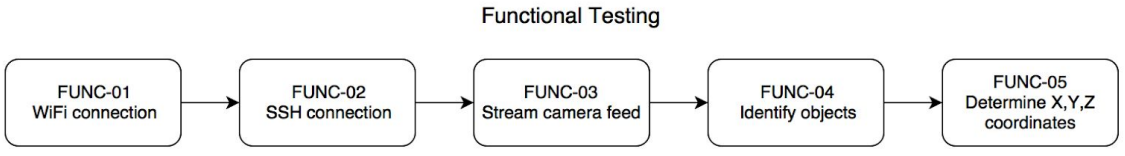


Figure 3. Sequence of tests in the Functional Testing phase

The prior test cases had to be successful before the next test case could be performed. For example, **FUNC-05** test case required **FUNC-01**, **FUNC-02**, **FUNC-03**, and **FUNC-04** test cases to have been performed with a successful outcome. This allowed FUNC-05 test case to run with no issues and produce the X, Y, Z coordinate of the object identified within the frame.

One of the requirements specified earlier, was to ensure that the U-TRACKR program would avoid/minimize any significant impact in terms of time delays. To address this requirement, the functional testing phase allows us to verify that the main controller retrieves the video feed from all four cameras at the same time, keeping other factors consistent (i.e. WiFi network, etc). As the fundamental components of the system are working as expected and has passed the developed test cases, this portion of testing is marked complete.

2.3.2. Performance

Out of the two test cases performed in this testing phase, all of these tests were successful. These tests were written as automated black-box unit tests, and they returned the expected output when run programmatically. These tests ensure the validity of two dynamic variables, which are CPU usage and bitrate of the Raspberry Pi and camera respectively. These tests attempt to connect to the Raspberry Pi, and run various Linux commands as a script to determine the CPU usage and bitrate. This test design is valid, as it can be run multiple times to determine the CPU usage and bitrate at certain instances of time in different configurations and environments.

In the requirements from the Critical Design Review, we stated that the “microcontroller’s processing power must ensure little to no delay when feeding the video recordings to the server and carrying out multiple tasks on the same controller.” These performance test designs validate these requirements since the CPU usage and bitrate are the two most important variables when streaming videos on a server. Overall the test results are as expected, and should not regress in the future. Since these are automated black-box tests, they can be run while the U-TRACKR system is operating, allowing us to vary the system’s configurations accordingly.

2.3.3. Mean Time Before Failure

The test case developed to test the mean time before failure is to account for the long periods of time this system would operate in a real life application such as a warehouse setting. This test case does not address any requirements stated previously, and was derived while implementing the system. There are possibilities of the U-TRACKR producing incorrect position coordinates due to various factors such as disconnected WiFi network, broken pipeline of an established SSH connection, environment settings, etc. This test case is to identify the outliers in the calculated X, Y, Z coordinates, and determine if these outliers are caused at regular intervals while the system operates for long periods of time.

The incorrect calculations of the X,Y, Z coordinates of an object within the defined area causes the system to breakdown and throws an exception. Therefore, we consider this test case to have failed. Since this step did not produce the expected results, the next steps are as follows:

- Ensure that the U-TRACKR system can provide accurate X, Y, Z coordinates with an error tolerance of +/- 1 cm.
- Account for unexpected exceptions that can be thrown, and modify the U-TRACKR system to throw warnings rather than causing a system failure.
- Run this test again, and verifying the mean time before the U-TRACKR system provides inaccurate X, Y, Z coordinates.

2.3.4. Environment

Only one test case was performed in this testing phase, and this test failed. This test is a manual integration test. Verification of this test case was done by running the U-TRACKR system and manually verifying if it is operational at certain lux environments. This test ensures the validity of the system's object detection algorithm by testing it in dim (~5 lux) and bright (~100 lux) environments. These two environments would test the minimum and maximum values of lux. The assumption is that if our system is able to track objects on these two values, then it can track objects in any light environments within these two extreme lux values. Due to this, this test design is valid.

One of the requirements from the critical design review was to "perform position calculation on images to determine the precise location of objects". Since image recognition algorithms are highly dependent on the lighting of the environment, this environmental test design validates these requirements since it tests various lux values. The test results are not as expected, but remedations will be made to ensure this test passes. Our system's object detection will be changed to utilize a deep learning framework in order to detect objects of various colours and sizes, which is accurate even in low and high light environments.

2.3.5. Reliability

The test cases that were performed for the reliability testing portion of the system marginally passed.

REL-01 was mainly developed to account for any malfunctions that could possibly occur in the raspberry pi cameras. In a real life setting, certain factors such as environment could cause malfunctioning of components within the system. The U-TRACKR program should be able to continue operating, rather than causing system failure if one or more components are malfunctioning. This test was not based on any requirements specified earlier. While developing the system, we decided to add in contingency measures should a component cause failure. The U-TRACKR program was able to continue operating and produce X,Y ,Z coordinates, although an exception was thrown to warn that one or more of the cameras are malfunctioning. In this case, we considered this test case to be a partial pass as the system did not breakdown and continued operating. Currently, the exception that gets thrown does not provide much information regarding the camera that is malfunctioning. Therefore, the next steps for this test case, would be to modify the U-TRACKR program to throw meaningful exceptions that will help a user to identify which camera is malfunctioning.

REL-02 was developed to verify if the U-TRACKR program provides accurate X,Y, Z coordinates of an object placed within the frame. This test encompases the project's objectives, and accuracy of the calculated X,Y, Z coordinates is crucial for the success of this project. As previously stated in the CDR document, one of our requirements was to ensure that the U-TRACKR program is able to track an object and identify the precise location as the object is moved around within the area of coverage of the cameras. This reliability test is to verify and validate this requirement by testing to see if an accurate coordinate is produced while moving the tennis ball within the frame.

The U-TRACKR system is able to identify the objects within the defined area and calculates the X, Y, Z coordinate of the object, however it is widely inaccurate, even for a stationary object located at (0,0,0). Therefore, this test case is considered as failed. The next steps for this test case includes the following:

- Ensuring that the U-TRACKR system can provide accurate X, Y, Z coordinates with an error tolerance of +/- 1 cm.
- Retesting this test, and verifying the accuracy by measuring the coordinates manually.

2.3.6. Compatibility

Only one test case was performed in this testing phase, and this test passed. This test is a manual integration test. Verification of this test case was done by running the U-TRACKR system and manually verifying if the system can detect various objects with different color. This test ensures the validity of the system's object detection algorithm by testing the system with various objects in different color spaces. The objects being tested are adjusted in hue, by testing the three major colours of red (orange/pink), green, and blue. The assumption is that if our system can track these objects, it can track objects with any combination of these hue values. Due to this, this test design is valid.

One of the requirements from the critical design review was to "perform position calculation on images to determine the precise location of objects". Since image recognition algorithms are dependent on the color of the object it is tracking, this compatibility test design validates these requirements. Overall the test results are as expected.

2.3.7. Portability

The test case developed for portability is to mainly address that different cameras may be suited for different needs within an indoor area. As our project is a scaled version of the actual project, the U-TRACKR program should have the flexibility to accommodate various hardware components. The raspberry pi camera will cover the area of our frame (88x88x88 cm). However it will not cover all area in a warehouse setting, for example. Therefore, to be more flexible when implementing the actual project, the U-TRACKR program must be able to accommodate different hardware components with a few modifications in the code.

Previously, one of our requirements was to incorporate the use of frames that can be assembled/disassembled allowing for portability. This test was derived from this requirement, and further enhancing this requirement to allow for hardware changes that

would allow the system to be more portable. Therefore we tweaked our program to operate with a video stream fed from a webcam. This test case passed, as the program is able to track a object from the video stream fed from a webcam and only a few modifications had to be made to the U-TRACKR program. As the actual results of this test case matched with the expected results, this test case is marked complete. However, due to the incorrect X,Y, Z coordinates calculated by the system, this test case will be tested again once the calculations are deemed accurate.

2.3.8. Boundary Value

Two test cases were performed in this testing phase, and these tests passed. Both of these tests are manual integration tests. Verification of this test case was done by running the U-TRACKR system and manually verifying if the system can detect an object when the cameras are at the farthest and shortest possible distance. This test ensures the validity of the system's object detection algorithm by testing the cameras at a very short distance (~1cm) and a very far distance (>~100cm). These two values would test the minimum and maximum camera distance environments. The assumption is that if the system is able to track these two values, then it can track objects in any objects within these two extremes. Due to this, this test design is valid.

One of the requirements from the critical design review was to "perform position calculation on images to determine the precise location of objects". Since image recognition algorithms are dependent on the distance of the cameras, this boundary value test design validates these requirements. Overall the test results are as expected, and should not regress in the future.

2.4. Omitted Test Cases

These test cases were omitted due to various changes in the requirements. These test cases can be viewed in their related section in the Appendix.

2.4.1. PER-02 and PER-03

The PER-01 and PER-04 tests are testing dynamic variables such as CPU usage and bit rate, which change over time based on various factors such as the network speed of the WiFi, and the number of processes running on the Raspberry Pi. Static variables such as the ones being tested for PER-02 and PER-03 (camera FPS and resolution) do not change over time, and are set when the system is run. Due to this, these test cases were omitted because the verification of these tests would not provide us new information on the validity of the U-TRACKR system.

2.4.3. MTBF-02

This test case was omitted because our project is a scaled version. Our main objective is to build a working small scale prototype before investing towards a bigger version. We were also constrained by our budget. As of right now, we are aware that this test case will most definitely fail since our system is not waterproof, and is currently built on unstable frames that may cause the cameras to move if the frames were suddenly moved.

2.4.4. COM-02

The COM-02 test determines whether the U-TRACKR system is able to track multiple objects. Initially, our goal for the U-TRACKR system was to prevent collisions between two objects. Due to various constraints, this requirement was removed and a new application domain related to animal tracking was introduced. For now, this test is omitted since the system successfully functions without the added capability of tracking multiple objects.

3. Self-Assessment of the Test Review

Table 10. Self Evaluation for TR Rubric Criteria

RUBRIC CRITERION FOR TEST REVIEW	SELF-EVALUATION RANKING	OUR JUSTIFICATIONS
Rubric 1: **Analyze essential results of solutions and test for validity.**	Level 4: Critically analyzes essential results of solutions and chooses the most effective test for validity.	We analyzed each test area, and answered all questions with regards to the system test verification and validation. We provided clear reasoning on how each test area meets the requirements specified in our preliminary and critical design review.
Rubric 2: **Clearly present information in professional engineering charts, tables, graphs, and diagrams within a report or design document.**	Level 3: Figures and tables are legible, convincing, and consistent with accepted standards, all items correctly labelled; items referred to in document.	All figures related to test case successes or failures are linked within the test results table, and greater detail is provided in the Appendix. Each figure and table is labelled.

4. Appendix

4.1 Appendix A - Functional Testing

Table 11. FUNC-01

Test ID	FUNC-01
Test Case	Verify that each Raspberry Pi can connect to the same WiFi network as the main controller (laptop)
Test Case Type	Automated Python Unit Test
Test Case Location (GitHub)	/U-TRACKR/beta/testing/FUNC-01.py
Preconditions	1. The Raspberry Pi is turned on. 2. The Raspberry Pi networking module is enabled.
Test Steps	1. Ensure the Raspberry Pi Raspbian OS loads up. 2. Login to Raspberry Pi. 3. Attempt to connect to the same WiFi network as the main controller (laptop).
Test Data	N/A
Expected Results	Each Raspberry Pi must be able to connect to the same WiFi network successfully.
Test Failure Plan	1. Troubleshoot the networking module of the Raspberry Pi 2. Re-image/Reinstall the Raspbian OS to the memory card.
Health and Safety Considerations	No relevant considerations.

Table 12. FUNC-02

Test ID	FUNC-02
Test Case	Verify the ability to establish an SSH connection from the main controller to each Raspberry Pi.
Test Case Type	Automated Python Unit Test
Test Case Location (GitHub)	/U-TRACKR/beta/testing/FUNC-02.py
Preconditions	1. The Raspberry Pi's are running and functional 2. The Raspberry Pi's are connected to the same WiFi network as the main controller (laptop)
Test Steps	1. Create an SSH connection from the main controller to the Raspberry Pi's. 2. Run a command on the SSH connection and retrieve the system output
Test Data	• IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	Verify that a proper connection has been made by obtaining the system output to the main microcontroller (laptop) console
Test Failure Plan	1. Verify the IP addresses of the Raspberry Pi's. 2. Debug the SSH connection Python module to find the cause of failure.
Health and Safety Considerations	No relevant considerations.

Table 13. FUNC-03

Test ID	FUNC-03
Test Case	Verify the ability to stream the camera feed from the Raspberry Pi's to the main controller (laptop) from each Raspberry Pi's at the same time.
Test Case Type	Automated Python Unit Test
Test Case Location (GitHub)	/U-TRACKR/beta/testing/FUNC-03.py
Preconditions	1. The Raspberry Pi's are running and functional 2. The Raspberry Pi's are connected to the same WiFi network as the main controller (laptop)
Test Steps	1. Establish an SSH connection from the main controller to the Raspberry Pi's. 2. Execute the Python script written to execute the camera video streaming command on all the Raspberry Pi's, at the same time. 3. Determine if all cameras started at the same time by verifying their timestamps.
Test Data	• IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	All Raspberry Pi cameras must have started the camera at the same time, and must be able to stream the video feed to the main controller.
Test Failure Plan	1. Debug the Python script to find the cause of failure to start all 4 cameras at the same time. 2. Verify on each Raspberry Pi that no other processes are running.
Health and Safety Considerations	No relevant considerations.

Table 14. FUNC-04

Test ID	FUNC-04
Test Case	Verify that OpenCV software runs with no issues on main controller, and identifies objects within the frame.
Test Case Type	Automated Python Unit Test
Test Case Location (GitHub)	/U-TRACKR/beta/testing/FUNC-04.py
Preconditions	1. The Raspberry Pi's are running and functional 2. The Raspberry Pi's are connected to the same WiFi network as the main controller (laptop) 3. Establish an SSH connection from the main controller to the Raspberry Pi's.
Test Steps	1. Execute the U-TRACKR program to run OpenCV on the video feed coming in from all 4 Raspberry Pi's. 2. Verify that the objects within the frame are identified.
Test Data	• IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	All 4 Raspberry Pi's cameras must have started the camera at the same time, and must be able to stream the video feed to the main controller.
Test Failure Plan	1. Debug the Python script to find the cause of failure to start all 4 cameras at the same time. 2. Verify on each Raspberry Pi that no other processes are running.
Health and Safety Considerations	Ensure that the objects being tracked do not collide with the frame. Depending on its velocity and mass, it may cause damage to the frame.

Table 15. FUNC-05

Test ID	FUNC-05
Test Case	Verify that the U-TRACKR program executes the Python script that determines the X, Y, Z coordinates of the object within in the frame.
Test Case Type	Automated Python Unit Test
Test Case Location (GitHub)	/U-TRACKR/beta/testing/FUNC-05.py
Preconditions	1. The Raspberry Pi's are running and functional 2. The Raspberry Pi's are connected to the same WiFi network as the main controller (laptop) 3. SSH connection established between the main controller to the Raspberry Pi's. 4. The U-TRACKR program is running OpenCV on the video feed coming in from all 4 Raspberry Pi's.
Test Steps	1. The U-TRACKR program is to execute the Python script which should determine the X, Y, Z coordinates of the object and output to console. 2. Verify that the coordinates are outputted to the console.
Test Data	• IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	The X, Y, Z coordinates of the object must be outputted to the console.
Test Failure Plan	1. Debug the Python script to find the cause of failure to output information to the console. This could be due to multiple processes running, causing a lag, or if the script is corrupted.
Health and Safety Considerations	Ensure that the objects being tracked do not collide with the frame. Depending on its velocity and mass, it may cause damage to the frame.

4.2 Appendix B - Performance Testing

Table 16. PER-01

Test ID	PER-01
Test Case	Check if each Raspberry Pi CPU usage meets the specified requirements
Test Case Type	Automated Python Unit Test
Test Case Location (GitHub)	/U-TRACKR/beta/testing/PER-01.py
Preconditions	1. The Raspberry Pi and camera are running and functional 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop)
Test Steps	1. Create an SSH connection from the main controller to the Raspberry Pi 2. Run the U-TRACKR program 3. While the U-TRACKR program is running, continuously monitor the CPU usage 4. Identify if the CPU usage is appropriate, and not consistently at 100%, which indicates more processing power is required.
Test Data	• Raspberry Pi CPU usage • IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	CPU usage of the Raspberry Pi is within a stable threshold
Test Failure Plan	1. Overclock the Raspberry Pi if it requires more processing power 2. Debug the U-TRACKR program if its below a threshold to find the cause of failure
Health and Safety Considerations	The failure where the Raspberry Pi CPU usage is at 100% for long periods of time, or the solution of overclocking will produce excess heat, which may be greater than normal operating levels. This combination of heat through the processing chip causes electromigration, which could be dangerous to the parts in the long term.

Table 17. PER-02

Test ID	PER-02
Test Case	Check if each Raspberry Pi Camera FPS meets the specifications
Test Case Type	Automated Python Unit Test
Preconditions	1. The Raspberry Pi and camera are running and functional 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop)
Test Steps	1. Create an SSH connection from the main controller to the Raspberry Pi 2. Run the U-TRACKR program 3. While the U-TRACKR program is running, monitor the FPS of the Raspberry Pi Camera for 10 seconds 4. If the obtained FPS meets the specified requirements during the 10 second period, the test passes. If it does not, then it fails
Test Data	• Raspberry Pi Camera FPS • IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	FPS of the Raspberry Pi Camera is meets the requirements
Test Failure Plan	1. Modify the arguments of the Raspberry Pi Camera FPS based on the API and ensure it is equal to the one specified in the requirements. 2. Replace the Raspberry Pi Camera with a similar alternative
Health and Safety Considerations	No relevant considerations.

Table 18. PER-03

Test ID	PER-03
Test Case	Check if each Raspberry Pi Camera resolution meets the specifications
Test Case Type	Automated Python Unit Test
Preconditions	1. The Raspberry Pi and camera are running and functional 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop)
Test Steps	1. Create an SSH connection from the main controller to the Raspberry Pi 2. Run the U-TRACKR program 3. While the U-TRACKR program is running, monitor the camera resolution of the Raspberry Pi Camera for 10 seconds 4. If the obtained camera resolution meets the specified requirements during the 10 second period, the test passes. If it does not, then it fails
Test Data	• Raspberry Pi Camera resolution • IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	Resolution of the Raspberry Pi Camera meets the requirements
Test Failure Plan	1. Modify the arguments of the Raspberry Pi Camera resolution based on the API and ensure it is equal to the one specified in the requirements. 2. Replace the Raspberry Pi Camera with a similar alternative
Health and Safety Considerations	No relevant considerations.

Table 19. PER-04

Test ID	PER-04
Test Case	Check if each Raspberry Pi stream bitrate meets the specifications
Test Case Type	Automated Python Unit Test
Test Case Location (GitHub)	/U-TRACKR/beta/testing/PER-04.py
Preconditions	1. The Raspberry Pi and camera are running and functional 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop)
Test Steps	1. Create an SSH connection from the main controller to the Raspberry Pi 2. Run the U-TRACKR program 3. While the U-TRACKR program is running, monitor the stream bitrate from the Raspberry Pi. 4. If the obtained stream bitrate meets the specified requirements, the test passes. If it does not, then it fails
Test Data	• Raspberry Pi video stream bitrate • IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	Raspberry Pi stream bitrate meets the requirements
Test Failure Plan	1. Modify the arguments of the Raspberry Pi stream bitrate based on the API and ensure it is equal to the one specified in the requirements. 2. Find an alternative network utility used to stream the video produced from the Raspberry Pi Camera
Health and Safety Considerations	No relevant considerations.

4.3 Appendix C - Reliability Testing

Table 20. REL-01

Test ID	REL-01
Test Case	Check if the U-TRACKR program will not crash if one or more of the Raspberry Pi camera breaks down.
Test Case Type	Manual Integration Test
Preconditions	1. The Raspberry Pi and camera are running and functional 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop) 3. Create an SSH connection from the main controller to the Raspberry Pi 4. Run the U-TRACKR program 5. While the U-TRACKR program is running, place a trackable object within the frame's tracking area
Test Steps	1. Remove the power supplied to one of the Raspberry Pi's first, then test again by removing the power supplied to another Raspberry Pi. 2. Verify if the U-TRACKR program is still functioning correctly, and also still able to track the location of the object within the frame's tracking area.
Test Data	• Object identification values used for tracking • IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	The U-TRACKR Program must be able to continue functioning correctly even if one of the Raspberry Pi's breaks down. If more than one Raspberry Pi's breaks down, the accuracy of the computed object coordinates may be compromised.
Test Failure Plan	1. Debug the U-TRACKR program and modify the program to ensure the system will be able to support one or more Raspberry Pi breakdowns.
Health and Safety Considerations	No relevant considerations.

Table 21 . REL-02

Test ID	REL-02
Test Case	Test the accuracy of the U-TRACKR program
Test Case Type	Manual Integration Test
Preconditions	1. The Raspberry Pi and camera are running and functional 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop) 3. There is an SSH connection from the main controller to the Raspberry Pi 4. The U-TRACKR program is being run 5. A trackable object is within the frame's area.
Test Steps	1. Monitor the U-TRACKR program for five time instances, and collect the X, Y, Z data of the object. 2. Manually measure the object's position relative to the frame, and collect the X, Y, Z data of the object.
Test Data	• Object identification values used for tracking • IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi • Object position at five time instances from U-TRACKR • Object position at five time instances from manual calculation
Expected Results	The object's position recorded from U-TRACKR is similar to the position recorded from manual calculations within ~1cm.
Test Failure Plan	Review the space resection and space intersection implementation.
Health and Safety Considerations	No relevant considerations.

4.4 Appendix D - Mean Time Before Failure Testing

Table 22. MTBF-01

Test ID	MTBF-01
Test Case	Determine the mean time before failure of the U-TRACKR program while running for a long period of time.
Test Case Type	Manual Integration Test
Preconditions	1. The Raspberry Pi and camera are running and functional 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop) 3. Create an SSH connection from the main controller to the Raspberry Pi 4. Run the U-TRACKR program 5. While the U-TRACKR program is running, place a trackable object within the frame's tracking area
Test Steps	1. Determine the mean time before failure of the U-TRACKR program while running for a long period of time. 2. If the system operates successfully and correctly for more than the time specified by the requirements, the test is considered successful.
Test Data	• Raspberry Pi video stream bitrate • IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi • Position coordinates of object used for tracking
Expected Results	The U-TRACKR program should be able to operate successfully and correctly for approximately 8 hours.
Test Failure Plan	1. Determine the cause of failure (i.e. CPU usage is 100% and system breakdown) and remediate the problem (i.e. Stop applications which are not currently in use to lessen the CPU usage, or if possible add more RAM, etc).
Health and Safety Considerations	While testing for a long period of time, the Raspberry Pi may become overheated, which could be dangerous to the components in the long term.

Table 23. MTBF-02

Test ID	MTBF-02
Test Case	Determine the mean time before failure of the U-TRACKR program while the system is exposed to various environmental conditions faced indoors.
Test Case Type	Manual Integration Test
Preconditions	1. The Raspberry Pi and camera are running and functional 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop) 3. Create an SSH connection from the main controller to the Raspberry Pi 4. Run the U-TRACKR program 5. While the U-TRACKR program is running, place a trackable object within the frame's tracking area
Test Steps	1. Expose the environment of the system to conditions dealt on a day to day basis indoors such as : a. A rush of wind b. Vibrations of floor due to heavy traffic c. Various temperature settings ranging from hot to cold d. Sprinkler system in case of fire indoors. (This condition will NOT be tested as the system currently is not waterproof. However, this is a condition to be tested if the system is further expanded to a bigger area). 2. Verify under which conditions the system still operates correctly, else measure the mean time before the system fails to operate correctly.
Test Data	● Raspberry Pi video stream bitrate ● IP address of the Raspberry Pi ● SSH username of the Raspberry Pi ● SSH password of the Raspberry Pi ● SSH port of the Raspberry Pi ● Position coordinates of object used for tracking
Expected Results	The U-TRACKR program can function correctly for most conditions mentioned above except for probably the Sprinkler system in case of fire indoors. The system is currently not waterproof, and therefore this may cause the system to operate incorrectly and/or crash.
Test Failure Plan	1. Replace the Raspberry Pi Camera that stops operating, with a similar alternative
Health and Safety Considerations	Some considerations: ● While testing under various temperature settings, the Raspberry Pi may become overheated, which could be dangerous to the components in the long term. ● While testing the system with a sprinkler like object, it is important to stay a few steps back as the system may come in contact with water and breakdown.

4.5 Appendix E - Boundary Value Testing

Table 24. BVT-01

Test ID	BVT-01
Test Case	Determine the U-TRACKR program is functional when tracking an object at the farthest distance specified by the requirements.
Test Case Type	Manual Integration Test
Preconditions	1. The Raspberry Pi and camera are running and functional 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop)
Test Steps	1. Create an SSH connection from the main controller to the Raspberry Pi 2. Run the U-TRACKR program 3. While the U-TRACKR program is running, place a trackable object within the frame's tracking area 4. Vary the distance of the Raspberry Pi and camera such that it reaches the farthest distance it can track the object. Measure this distance.
Test Data	• Object identification values used for tracking • IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	The farthest distance that the U-TRACKR program can track matches the farthest distance specified by the requirements.
Test Failure Plan	1. Lower the farthest distance specified by the requirements 2. Fine-tune the object identification values used for tracking 3. Use an alternate method of tracking
Health and Safety Considerations	No relevant considerations.

Table 25. BVT-02

Test ID	BVT-02
Test Case	Determine the U-TRACKR program is functional when tracking an object at the shortest distance specified by the requirements.
Test Case Type	Manual Integration Test
Preconditions	1. The Raspberry Pi and camera are running and functional 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop)
Test Steps	1. Create an SSH connection from the main controller to the Raspberry Pi 2. Run the U-TRACKR program 3. While the U-TRACKR program is running, place a trackable object within the frame's tracking area 4. Vary the distance of the Raspberry Pi and camera such that it reaches the shortest distance it can track the object. Measure this distance.
Test Data	• Object identification values used for tracking • IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	The shortest distance that the U-TRACKR program can track matches the shortest distance specified by the requirements.
Test Failure Plan	1. Increase the shortest distance specified by the requirements 2. Fine-tune the object identification values used for tracking 3. Use an alternate method of tracking
Health and Safety Considerations	• No relevant considerations.

4.6 Appendix F - Compatibility Testing

Table 26. COM-01

Test ID	COM-01
Test Case	Determine that the U-TRACKR program can identify objects using the full range of values (ex. HSV colours)
Test Case Type	Manual Integration Test
Preconditions	1. The Raspberry Pi and camera are running and functional 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop)
Test Steps	1. Create an SSH connection from the main controller to the Raspberry Pi 2. Run the U-TRACKR program 3. While the U-TRACKR program is running, place a trackable object with the lowest values to identify it within the frame's tracking area. (Ex. Black color) 4. Remove the previous object from the frame's test area. 5. While the U-TRACKR program is running, place a trackable object with the highest values to identify it within the frame's tracking area. (Ex. White color) 6. Remove the previous object from the frame's test area. 7. While the U-TRACKR program is running, place a trackable object with the mean average values to identify it within the frame's tracking area. (Ex. Green color)
Test Data	• Object identification values used for tracking • IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	On each video stream, OpenCV can track all 3 objects and output each of their positions on the frame.
Test Failure Plan	1. Modify the object identification values in closer precision 2. Use another method of tracking
Health and Safety Considerations	Some considerations: • Ensure that the objects being tracked do not collide with the frame. Depending on its velocity and mass, it may cause damage to the frame.

Table 27. COM-02

Test ID	COM-02
Test Case	Determine that the U-TRACKR program can track multiple objects
Test Case Type	Manual Integration Test
Preconditions	1. The Raspberry Pi and camera are running and functional 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop)
Test Steps	1. Create an SSH connection from the main controller to the Raspberry Pi 2. Run the U-TRACKR program 3. While the U-TRACKR program is running, place a trackable object within the frame's tracking area 4. Place another copy of the trackable object within the frame's tracking area
Test Data	• Object identification values used for tracking • IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	On each video stream, OpenCV can track both objects simultaneously and output each position on the frame.
Test Failure Plan	1. Check if FUNC-04 or FUNC-05 is failing for one object. Make appropriate fixes based on those test cases. 2. Check to see if the two objects return the same object identification values in OpenCV. If not, then replace the objects with
Health and Safety Considerations	Some considerations: • Ensure that the two objects being tracked does not collide with the frame. Depending on its velocity and mass, it may cause damage to the frame. • Ensure that the two objects being tracked does not collide with each other. Depending on its velocity and mass, it may cause damage to each other.

4.7 Appendix G - Portability Testing

Table 28. POR-01

Test ID	POR-01
Test Case	Determine the level of ease to apply the U-TRACKR program to other hardware applications.
Test Case Type	Manual Integration Test with similar cameras (webcam)
Preconditions	1. The Raspberry Pi must be connected to a similar camera(i.e webcam), and must be running and functional. 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop)
Test Steps	1. Create an SSH connection from the main controller to the Raspberry Pi 2. Run the U-TRACKR program 3. While the U-TRACKR program is running, place a trackable object within the frame's tracking area. 4. Ensure that the requirements are met.
Test Data	• Object identification values used for tracking • IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	The U-TRACKR program must detect the coordinates of the objects, regardless of the type of hardware application, provided a live video stream is fed to the main controller.
Test Failure Plan	1. Check compatibility of the similar camera(i.e webcam) to the Raspberry Pi's. 2. First try running a live feed from the Raspberry Pi itself, before running an SSH connection to execute a command on the Raspberry Pi to do the same. This will help in identifying where exactly the problem lies.
Health and Safety Considerations	No relevant considerations.

4.8 Appendix H - Environment Testing

Table 29. ENV-01

Test ID	ENV-01
Test Case	Determine the U-TRACKR program is functional within a specific lux threshold
Test Case Type	Manual Integration Test
Preconditions	1. The Raspberry Pi and camera are running and functional 2. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop)
Test Steps	1. Create an SSH connection from the main controller to the Raspberry Pi 2. Run the U-TRACKR program 3. While the U-TRACKR program is running, place a trackable object within the frame's tracking area 4. Find the lowest lux (lumens per square meter) such that the object can be trackable. Ensure it is consistent with the requirements 5. Find the highest lux (lumens per square meter) such that the object can be trackable. Ensure it is consistent with the requirements.
Test Data	• Lux of the tracked object • Object identification values used for tracking • IP address of the Raspberry Pi • SSH username of the Raspberry Pi • SSH password of the Raspberry Pi • SSH port of the Raspberry Pi
Expected Results	The U-TRACKR program can detect the given object within a lux level specified by the requirements
Test Failure Plan	1. Modify the lux level specified in the requirements to the threshold found within this test case. 1. Replace the Raspberry Pi Camera with a similar alternative
Health and Safety Considerations	Ensure that the objects being tracked do not collide with the frame. Depending on its velocity and mass, it may cause damage to the frame.