Team 1:

Indoor UAV Tracking System

Test Readiness Review

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

February 05, 2018

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

Link to Video:
https://www.youtube.com/watch?v=ivF6iSyullQ

Table of Contents

1. System State

1.1 Introduction

The ultimate goal of this project is to be able to track and model the trajectory of multiple autonomous UAVs using image-processing and photogrammetry. We are focused on indoor UAV applications because for outdoor applications, UAVs are hindered by flying regulations and noise pollution control. Moreover, UAVs have great potential when it comes to applications specific to warehouses because of their flexible flight path, automation, and ability to enter environments that are dangerous to human life. The U-TRACKR system will improve the efficiency and cost-effectiveness of UAVs in this context by providing positioning technology and indoor navigation. Our camera system will locate the UAV in an indoor space, where GPS is unreliable, calculate the trajectory using the image sequences, and then model the path. Currently, we have accomplished time synchronization between our cameras, as well as space resectioning and position calculations. We will also improve our system to locate surrounding obstacles, and predict the path of any collisions.

1.2 Engineering Tools and Processes

At this stage, we have completed our project's main features to ensure the readiness of the system for testing. The main goal of our project is to obtain an accurate position calculation of an object within the test frame boundaries. Below are the three areas identified to achieve this, as specified in the requirements section of the Critical Design Review:

- Time Synchronization
- Space Resection
- Position Calculation

1.2.1 Measurement Model

This section presents the measurement models that are used for computing the positional coordinates in the post-processing and post-estimation stages. To calculate the 3D position of an object, first we must identify the coordinates of the camera used. Second, the camera must have the same time reference.

1.2.1a Time Synchronization

One of our main requirements, which we identified in the Critical Design Review as a bottleneck, was the following: "*Synchronizing all four cameras to determine the accurate location of the objects*". This was achieved using Python's threading module to concurrently output each camera's timestamp, x-image position, and y-image position relative to the frame. This will synchronize the frames on one camera to the other, allowing for consistent data.

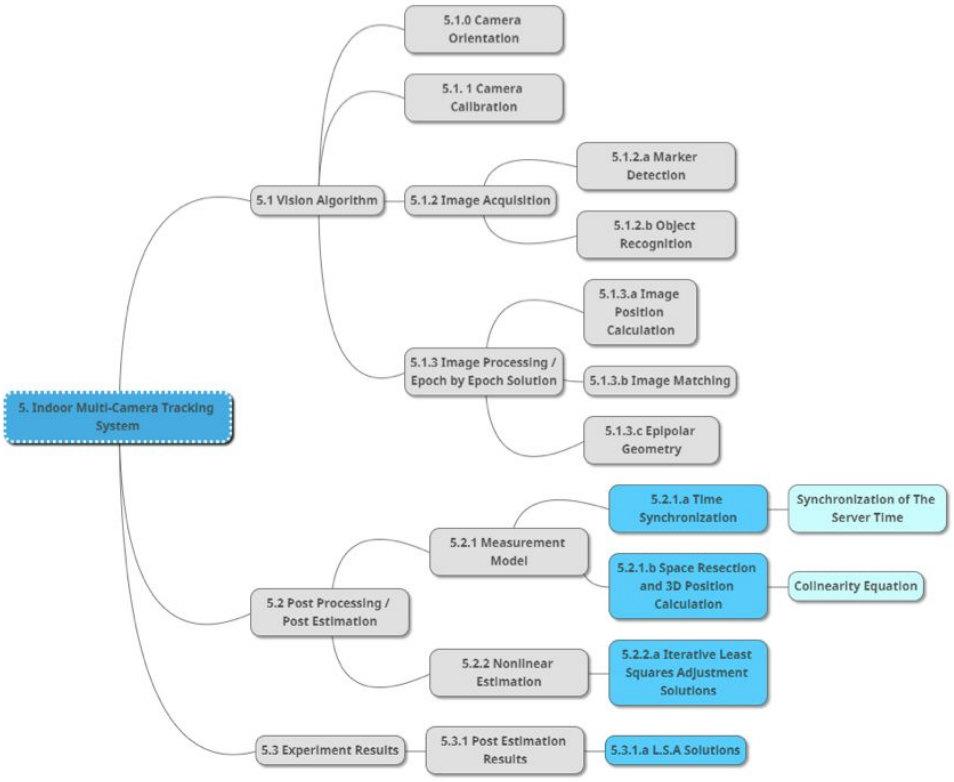

Figure 1. Multi-camera Tracking System: Post processing and post estimation of LSA solutions

1.2.1b Space Resection

Figure 2. Components of the space resection process

1.2.1b.i Pixel Coordinates

The pixel coordinate calculations utilizes the pre-existing algorithm to calculate the pixel coordinates defined by the marker detection and recognition software. The pixel coordinates are taken per second and the data is inputted into the pixel-coordinate-to-image-coordinate converter.

1.2.1b.ii Image Coordinates

From pixel coordinates, the image coordinates can be found using transformation derived from pixel size, resolution, and pixel coordinates. This process is done using the Python code, with the output matrix set aside for manipulation later. The pixel-to-image coordinate conversion is present by the following equations:

Table 1. Pixel Coordinates to Image Coordinates

Pixel Coordinates to Image Coordinates	
$SizeOfPixel = 1.12\ mm$ $Width_{ofImage} = 1280/2$ $Height_{ofImage} = 720/2$	Where, $SizeOfPixel$ is pixel size of the image predefined by the Pi cameras $Width_{ofImage}$ is predefined by the image resolution of 1280 X 720 $Height_{ofImage}$ is predefined by the image resolution of 1280 X 720
$x_{ImageCoords} = (x_{PixelCoords} - Width_{ofImage} - 0.5) * SizeOfPixel$ $y_{ImageCoords} = (Height_{ofImage} - y_{PixelCoords} + 0.5) * SizeOfPixel$	$x_{PixelCoords}$ is recorded from positional tracking algorithm presented in CDR $y_{PixelCoords}$ is recorded from positional tracking algorithm presented in CDR $x_{ImageCoords}$ is the converted X image coordinates matrix $y_{ImageCoords}$ is the converted Y image coordinates matrix
$ImageCoordinates\ (x_{ImageCoords}, y_{ImageCoords})$	$ImageCoordinates\ (x_{ImageCoords}, y_{ImageCoords})$ is the converted (x, y) image coordinates matrix $[n\ X\ 2]$
- Generic calculation of the image coordinates.	

The computed image coordinates take the parameter of the pixel coordinates, where the pixel coordinates have the same image dimension for both length and width. The resultant output is used to define the parameters of the space resection calculation.

1.2.1b.iii Space Resection

Space resection is the mathematical computation of the camera coordinates used to define the frame of reference in a three-dimensional system. This process calculates the exterior orientation derived from collinearity equation, conformal transformation, linearization by means of Taylor series of expansion and least square adjustment to eliminate the blunders. The output result gives each camera a 3-dimensional coordinate position and angular orientation with respect to the camera frame system. The following equations represent the procedural relationship of the space resection process:

Table 2. Space Resection (1/4)

Space Resection (1/4)	
Exterior Orientation $(\omega, \phi, \kappa, X_L, Y_L, Z_L)$ *Image Coordinates* (x, y) *Ground Coorinates* (X, Y, Z)	Where, $(\omega, \phi, \kappa, X_L, Y_L, Z_L)$ are the exterior orientation to be calculated by space resection (x, y) are the image coordinates denoted by the lower case (X, Y, Z) are the ground coordinates denoted by the upper case
$\omega = 0, \phi = 0$	$\omega = 0, \phi = 0$ are the pre-set angular orientation to be iterated
$(\overline{AB})^2 = (X_A - X_B)^2 + (Y_A - Y_B)^2$	$(\overline{AB})^2$ is the distance calculation between one ground coordinate and the other in a planimetric coordinate
$(\overline{AB})^2 = \left(x_a\left(\frac{H - Z_A}{f}\right) - x_b\left(\frac{H - Z_B}{f}\right)\right)^2 + \left(y_a\left(\frac{H - Z_A}{f}\right) - y_b\left(\frac{H - Z_B}{f}\right)\right)^2$	H is the height to be rearranged and solved
$H = Z_L$	Z_L The height of the exterior orientation
$X'_N = x_n\left(\frac{H - Z_N}{f}\right)$ $Y'_N = x_n\left(\frac{H - Z_N}{f}\right)$	(X'_N, Y'_N) are the ground coordinates of the ground control points from the assumed vertical photo; N and n denotes index of matrix from 1 to n
$X = aX' - bY' + T_x$ $Y = bX' - aY' + T_y$	(a, b) are the coefficients of polynomial of conformal coordinate transformation using least square principle T_x, T_y are the translation of a conformal coordinate transformation using least square principle
- Space resection by iteration of initial parameters. Ground coords. [m], Image Coords. [mm]	

Table 3. Space Resection (2/4)

Space Resection (2/4)	
And the LSA solution $L = \begin{bmatrix} X_1 \\ Y_1 \\ \dots \\ X_N \\ Y_N \end{bmatrix}$	Where, L is the ground coordinate (X, Y) matrix
$A = \begin{bmatrix} X_A' & Y_A' & 1 & 0 \\ X_B' & Y_B' & 0 & 1 \\ \dots & \dots & \dots & \dots \\ X_N' & Y_N' & 0 & 1 \end{bmatrix}$	A is the designed matrix of the calculated ground control points
$\hat{X} = \begin{bmatrix} \hat{a} \\ \hat{b} \\ \widehat{T_X} \\ \widehat{T_Y} \end{bmatrix}$	$\hat{X}$ is the adjusted matrix of the least square adjustment (LSA) solution
$\hat{X} = (A^T A)^{-1} A^T L$	
$\kappa = \theta = \tan^{-1}(\frac{\hat{b}}{\hat{a}})$	κ is one of the calculated angular orientation
The rotation matrix $m = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$	m is rotation matrix used to calculate the conformal transformation of the coordinate system
$m_{11} = cos\phi cos\kappa$ $m_{12} = sin\omega sin\phi cos\kappa + cos\omega sin\kappa$ $m_{13} = -cos\omega sin\phi cos\kappa + sin\omega sin\kappa$ $m_{21} = -cos\phi sin\kappa$ $m_{22} = -sin\omega sin\phi sin\kappa + cos\omega cos\kappa$ $m_{23} = cos\omega sin\phi sin\kappa + sin\omega cos\kappa$ $m_{31} = sin\phi$ $m_{32} = -sin\omega cos\phi$ $m_{33} = cos\omega cos\phi$	m_{nn} is the individual matrix calculation of the conformal rotation
Linearization using Taylor series of Expansion Solving for $(d\omega, d\phi, d\kappa, dX_L, dY_L, dZ_L)$	$(d\omega, d\phi, d\kappa, dX_L, dY_L, dZ_L)$ are the estimated and linearized parameters using Taylor series of Expansion
$J_N = b_{n1}^N d\omega + b_{n2}^N d\phi + b_{n3}^N d\kappa - b_{n4}^N dX_L - b_{n5}^N dY_L - b_{n6}^N dZ_L$ $K_P = b_{n1}^P d\omega + b_{n2}^P d\phi + b_{n3}^P d\kappa - b_{n4}^P dX_L - b_{n5}^P dY_L - b_{n6}^P dZ_L$	J_N, K_N linearized parameter for ground coordinates (X, Y); N and P are the indices for ground coordinates from X to Y from A to D $(i.e. b^A \dots b^D)$ and n is numerical index from 1-2 row matrix $i.e. b_{11}^A \dots b_{21}^D)$
- Space resection by iteration of initial parameters	

Table 4. Space Resection (3/4)

Space Resection (3/4)	
Elements for the Designed B Matrix $r = m_{11}(X_A - X_L) + m_{12}(Y_A - Y_L) + m_{13}(Z_A - Z_L)$ $s = m_{21}(X_A - X_L) + m_{22}(Y_A - Y_L) + m_{23}(Z_A - Z_L)$ $q = m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L)$ $b_{11}^N = \frac{f}{q^2}[r(-m_{33}\Delta Y + m_{32}\Delta Z) - q(-m_{13}\Delta Y + m_{12}\Delta Z)]$ $b_{12}^N = \frac{f}{q^2}[r(cos\phi\Delta X + sin\omega sin\phi\Delta Y - cos\omega sin\phi\Delta Z) - q(-sin\phi cos\kappa\Delta X + sin\omega cos\phi cos\kappa\Delta Y - cos\omega cos\phi cos\kappa\Delta Z)]$ $b_{13}^N = -\frac{f}{q}(m_{21}\Delta X + m_{22}\Delta Y + m_{23}\Delta Z)$ $b_{14}^N = \frac{f}{q^2}(rm_{31} - qm_{11})$ $b_{15}^N = \frac{f}{q^2}(rm_{32} - qm_{12})$ $b_{16}^N = \frac{f}{q^2}(rm_{33} - qm_{13})$ $b_{21}^P = \frac{f}{q^2}[s(-m_{33}\Delta Y + m_{32}\Delta Z) - q(-m_{13}\Delta Y + m_{12}\Delta Z)]$ $b_{12}^P = \frac{f}{q^2}[s(cos\phi\Delta X + sin\omega sin\phi\Delta Y - cos\omega sin\phi\Delta Z) - q(-sin\phi cos\kappa\Delta X + sin\omega cos\phi cos\kappa\Delta Y - cos\omega cos\phi cos\kappa\Delta Z)]$ $b_{23}^P = \frac{f}{q}(m_{11}\Delta X + m_{12}\Delta Y + m_{13}\Delta Z)$ $b_{24}^P = \frac{f}{q^2}(sm_{31} - qm_{11})$	Where, r, s, q are the calculated coefficients corresponding to A matrix $b_{nn}^{N/P}$ is the element of the calculated B matrix derived from r, s, q matrix, rotation matrix, initial assignment and calculation of (ω, ϕ, κ)
- $b_{nn}^{N/P}$ are the individual elements assigned in the B matrix.	

Table 5. Space Resection (4/4)

Space Resection (4/4)	
$b_{25}^{P} = \frac{f}{q^2}(sm_{32} - qm_{12})$ $b_{26}^{P} = \frac{f}{q^2}(rm_{33} - qm_{13})$	Where,
Least Square Adjustment of B matrix $B = \begin{bmatrix} b_{11}^{A} & b_{12}^{A} & b_{13}^{A} & -b_{14}^{A} - b_{15}^{A} - b_{16}^{A} \\ b_{21}^{A} & b_{22}^{A} & b_{23}^{A} & -b_{24}^{A} - b_{25}^{A} - b_{26}^{A} \\ \dots & \dots & \dots & \dots \\ b_{23}^{D} & b_{23}^{D} & b_{23}^{D} & -b_{24}^{D} - b_{25}^{D} - b_{26}^{D} \end{bmatrix}$	B matrix is the design matrix to solve the calculate exterior orientation matrix; derived from the combined elements of b matrix
$L = \begin{bmatrix} X_1 \\ Y_1 \\ \dots \\ X_N \\ Y_N \end{bmatrix}$	L is the ground coordinate (X, Y) matrix
$\Delta = \begin{bmatrix} d\omega \\ d\phi \\ d\kappa \\ dX_L \\ dY_L \\ dZ_L \end{bmatrix}$ $\Delta = (B^T B)^{-1}(B^T L)$	Δ solution of the LSA of B matrix
Calculation of Exterior Orientation (EO) $(\omega, \phi, \kappa, X_L, Y_L, Z_L)$ $\omega = d\omega\left(\frac{180^o}{\pi}\right) \pm 360^o$ $\phi = d\phi\left(\frac{180^o}{\pi}\right) \pm 360^o$ $\kappa = \theta + d\kappa$ $X_L = T_X + dX_L$ $Y_L = T_Y + dY_L$ $Z_L = T_Z + dZ_L$	$(\omega, \phi, \kappa, X_L, Y_L, Z_L)$ calculated E.O
- Calculated E.O. in $[m]$ and $[^o]$	

1.2.1b.iv 3D Positional Calculation

The process of calculating the final solution is the same as the methodology used to calculate space resection, where the initial image coordinate and ground coordinates are changed from the calculated position of the four cameras and the image coordinates are measured by each camera. By looping through the system multiple times, the solution of any object within the referenced coordinate system can be calculated.

1.3 Equipment and Hardware

Our project consists of a customized system frame, four microcontrollers, four cameras, four power cords, a personal laptop, and two (or more) drones for testing. We purchased the Raspberry Pi Zero W kit that comes with the Raspberry Pi cameras and microcontroller cases. Compared to other microcontrollers available, the Raspberry Pi has proven to be the best option for our project in terms of its specifications and cost. Table 6 shows the specifications of the top microcontrollers that best suited our project. The Raspberry Pi Zero W is superior to the other microcontrollers in terms of CPU speed, size, weight, RAM, and capability for its low price. With the Raspberry Pi, we have the added benefit of using the OpenCV library for real-time image processing and computer vision. The kit we purchased also included the camera, and all of the accessories we needed for the project. Furthermore, two of our group members already own Raspberry Pi microcontrollers, so it was advantageous to use equipment that they were already experienced with.

Table 6. Comparison of the top microcontrollers that best suited for the U-TRACKR system

Name	Processor	Operating Voltage	CPU Speed	RAM	Size	Weight	Wireless	Bluetooth	Price
Raspberry Pi Zero W	ARM11	5 V	1 GHz	512 MB	65 X 30 mm	9 g	✓	✓	$49.99 (with camera and case)
Arduino Uno Rev3	ATmega 328P	5 V	16 MHz	2 KB	68.6 X 53.4 mm	25 g			$33.95
Beagle Bone Black Rev C Element 14	OSD3358	5 V	1 GHz	512 MB	89 X 55 mm	40.55 g	✓	✓	$87.00
Thunderboard React	ARM Cortex-M4	5 V	25 MHz	32 KB	44 X 25 mm			✓	$60.32
NodeMCU & DEVKIT board	ESP8266	5 V	80 MHz	20 KB	48 X 23 mm	9.1 g		✓	$23.49
S7 Synergy Starter Kit	ARM Cortex-M4	5 V	25 MHz	640 KB	145 X 120 mm			✓	$84.00
NRF52-DK	ARM Cortex-M4F	5 V	2.4 GHz	64 KB	44 X 25 mm		✓	✓	$45.02
Freedom Board	ARM Cortex-M0	5 V	32 MHz	128 KB	85 X 55 mm		✓	✓	$147.22

1.4 Evaluation of Design Requirements

In order to ensure readiness, an evaluation of our previous design requirements was made. These are the steps taken before the testing phase:

1. Verified that functional requirements were met

   Initially, our strategy was to incorporate the use of a server to process the video captured by the Raspberry Pi camera to identify the objects within the tracking area. However, after further development activities, we realized that reading and writing to the server caused a lag in our system. This defeated our purpose of identifying the trajectory of objects in real-time. Therefore, we redesigned our strategy to first run the U-TRACKR program to establish an SSH connection with all four Raspberry Pi Zeros and execute a command on each Raspberry Pi Zero to start and transfer the live video feed to the main controller (laptop) at the same time. The U-TRACKR program was then successfully able to run the OpenCV software on the live feed frame by frame to identify objects within the defined area. The U-TRACKR program then executes a Python script which calculates the coordinates of the object within the frame's tracking area and outputs these coordinates to the console successfully.

2. Confirmed that performance requirements were fulfilled

   Initially, our strategy was to run the OpenCV software separately on all four Raspberry Pi Zeros. However this resulted in the CPU usage of all Raspberry Pi's to skyrocket to 100%, causing critical lag issues. Therefore we shifted our strategy to incorporate the use of a main controller (laptop), which will receive the live video feed from all four Raspberry Pis and run the OpenCV software. While programming the Python script to calculate the coordinates of the object within the tracking area, we noticed a small discrepancy between the Python script and the MATLAB code, due to data type and rounding. We plan to improve upon these issues during the testing phase.

3. Verified that interface requirements were met

   We are currently working on an alarm system for the U-TRACKR which will set off if the UAV was to collide with another object. We are also working on a functional user interface so that our customers can view the UAV model and manage their drones using our system. However, the alarm system and the user interface are not crucial to the system's test readiness. For the initial testing phase, we will be experimenting with multiple objects (UAVs and ground moving objects) to identify their coordinates within the tracking area.

4. Checked that regulatory requirements were followed

   Since our system applies to indoor UAV applications in private warehouse buildings, there are no flying regulations or noise level requirements. The system complies with the Workplace Safety and Insurance Board (WSIB) policies, the Worker Health and Safety - Ontario Ministry of Labour policies, and the York University's policy with regards to Temporary Use of University Space. Since the system will operate in a company's private building, the system complies with the privacy policies in place due to recording a defined area.

5. Ensured that the process requirements are met

   We have successfully met our work milestones and so far, our project is on schedule so the system is ready for the first phase of testing.

1.5 Remaining Development Activities

We are currently working on several features, all which are not necessary in order to begin our initial testing phases. These features include:

Tracking Simulation

We have currently narrowed down two available softwares, Gazebo(ROS) and Unity 3D, which best fit our needs for modelling the trajectory of our UAV in low latency time. Our initial strategy for modelling is to establish a serial port connection with the chosen software and send the X, Y, Z coordinates calculated by the U-TRACKR program using this connection. Based on the coordinates received, the software will shift the position of the 3D model of the UAV to the new coordinate. The previous coordinates will also be modelled as a continuously drawn path.

Power Usage Improvement

We need to improve how we power our microcontrollers and eliminate (or hide) all of the visible wires. Our prototype needs to be portable and presentable for the demonstration of our project.

Microcontroller Angle Flexibility

We are in the process of 3D-printing customized Raspberry Pi holders which will give us the flexibility to change the angles of our cameras. The cameras and the microcontrollers will be attached to our static, equidistant frame using the ring clamp of the customized holders.

Detection of Surrounding Obstacles

Our original scope of the system was to detect objects in motion, and track their position within the frame area. However, we would like to extend the capability of our system to detect the static obstacles around the moving object. To implement this, we plan on attaching colored markers on the obstacles for identification and ensure that the system is able to predict any collision that may occur between the mobile UAVs and the static obstacles.

Prediction of Collisions

The system is currently able to calculate the coordinates of objects within the tracking area in low latency time. This information will be used to detect if a collision may occur and immediately sound an alarm to notify. The predicted trajectory of the UAVs will be calculated based on its current location, previous trajectory, speed, and orientation.

Drone Demonstration

We are currently waiting for our drones to be shipped and we are looking to obtain the programmable AR Parrot Drone 2.0. Moreover, we are working on an automated flight path for the drones in order to test our system on a faster, moving object. It is assumed that our customers would be utilizing our system for their own UAVs. Therefore, the control of the UAVs' trajectories is technically beyond the scope of our project. However, we need to use drones for the U-TRACKR testing phases and for the project demonstration. Hence, we will need to have a separate testing phase for the drones to ensure that the flight path is controlled. We need to follow the regulations in the *Interim Order Respecting the Use of Model Aircraft* [3] which apply to recreational drones, as well as the drone safety regulations established by Transport Canada [4]. Since our drones are less than 35 kg, we do not need special permission from Transport Canada. However, we will still need to obey rules that govern restricted areas and "No drone zones" [5].

2. Testing Plan

This section specifies the testing plan for the current state of our system. This testing plan is loosely based off of known design verification test areas issued by the FDA for medical devices [2]. Here we plan to provide multiple tests per area to ensure the system will be in a state where it will function under minimal requirements.

2.1 Testing Process

Our testing process will utilize GitHub's issue tracker on our repository. Below is our process of reporting a test failure from the test cases below.

1. Open an issue, with the title describing the cause of the test failure.
2. Add a label to the issue based on the 8 test areas described below.

Table 7. Test areas and their associated label

Test Area	Associated Label
Functional Testing	Test Failure: FUNC
Performance Testing	Test Failure: PER
Mean Time Before Failure Testing	Test Failure: MTBF
Environment Testing	Test Failure: ENV
Reliability Testing	Test Failure: REL
Compatibility Testing	Test Failure: COM
Portability Testing	Test Failure: PORT
Boundary Value Testing	Test Failure: BVT

3. Within the issue description, add a point indicating the specific failed Test ID.

For example, say the test id “PER-02” failed since there was a problem returning the FPS from the microcontroller. A sample issue would look similar to Figure 1 and 2.

Figure 3. Sample Issue Title

As more features are added based on Section 1.5, and as issues are fixed, previously run test cases must be run again on the latest version to ensure the system functionality has not regressed.

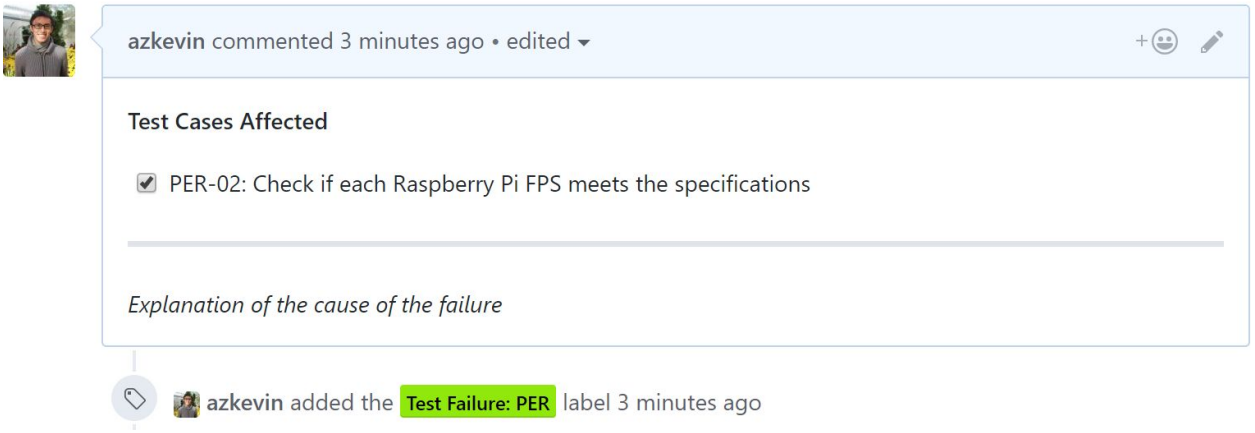

Figure 4. Sample Issue Description

New features and fixed issues may also require new test cases to arise. As the development of the project progresses, this list of test cases will also be updated.

2.2 Functional Testing

Table 8. 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
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 9. 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
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 10. 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
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 11. 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
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 by ensuring there's a tracker in red surrounding the objects in the video. This ensures that OpenCV is tracking the object and producing the results required.
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 12. 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
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.

2.3 Performance Testing

Table 13. 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
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 for 10 seconds a. If the CPU usage is consistently at 100%, fail the test as this indicates more processing power is needed b. If the CPU usage is consistently below a certain threshold, fail the test as it indicates the U-TRACKR program is not functioning correctly
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 14. 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 15. 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 16. 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
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 for 10 seconds 4. If the obtained stream bitrate meets the specified requirements during the 10 second period, 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.

2.4 Mean Time Before Failure Testing (MTBF)

Table 17. 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 testing
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 5 days, if not more, if no harsh environmental conditions such as water leaking onto the system, occurs.
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 18. 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	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) 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.

2.5 Environment Testing

Table 19. 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. 2. 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.

2.6 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 Rapsberry 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.

2.7 Compatibility Testing

Table 21. 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 colors)
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 22. COM-02

Test ID	COM-02
Test Case	Determine that the U-TRACKR program can track multiple objects

Test Case Type	Automated Python Unit Test
Preconditions	3. The Raspberry Pi and camera are running and functional 4. The Raspberry Pi is connected to the same WiFi network as the main controller (laptop)
Test Steps	8. Create an SSH connection from the main controller to the Raspberry Pi 9. Run the U-TRACKR program 10. While the U-TRACKR program is running, place a trackable object within the frame's tracking area 11. 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	3. Check to see if FUNC-04 or FUNC-05 is failing for one object. Make appropriate fixes based on those test cases. 4. 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.

2.8 Portability Testing

Table 23. 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.

2.9 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.

3. Experimental Results

3.1 Pixel Coordinates

The calculate pixel coordinates are recorded in the process of marker detection and object recognition. The recorded raw data for all four cameras are given saved in a .txt file. Refer to Appendix for the raw data output.

3.2 Image Coordinates

The calculated image coordinates are the values of running average for a stationary position at an interval of five seconds. This is then used to calculate the camera position in space resection process. The output of the image coordinates returned in the form of (x,y), where a matrix of [4x2] is taken for image resection process for the four measured predefined points. The following is a sample output of the recorded data.

Table 26. Calculated Image Coordinates for Cameras 1-4.

(X,Y) Camera 1		(X,Y) Camera 2		(X,Y) Camera 3		(X,Y) Camera 4	
-381.3037	82.3402	-508.1522	68.5532	-380.2979	83.2602	-507.9316	69.3017
-364.7935	257.7761	-301.3971	51.3185	-364.4184	258.1870	-301.3209	51.0668
-546.5606	279.4319	-296.5760	223.7760	-545.5149	279.9576	-292.6560	224.6720
-576.0361	106.2174	-472.8272	228.6221	-576.0028	105.8876	-473.1475	228.1277

3.3 Space Resection & Position Calculation

The calculation of position for space resection takes the input of [4X2] image coordinate matrix, the predefined coordinate reference system and at least four known control points. The known control points are established using the coordinate reference system which is a 3D cartesian planar coordinate system. The frame has a dimensions 0.88m X 0.88 m X 0.88 m, and it holds up same volume in the imaginary coordinate reference system.

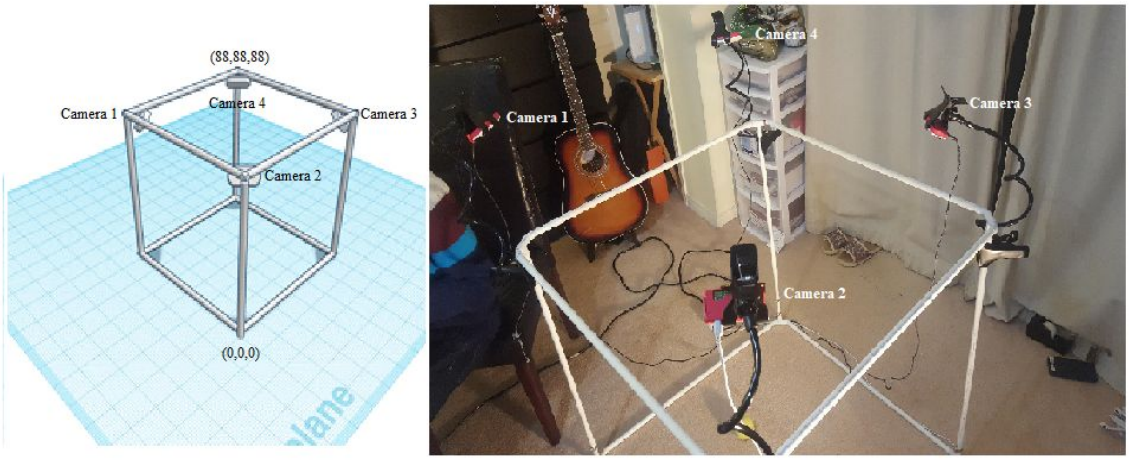

Figure 6. Designed Reference Camera Coordinate System : The cameras placed on top of the frame where the positioning of four cameras can be found using space resection. The reference system is set up to be (0,0,0) at the base of camera 2 and (88,88,88) to be the top of the frame close to camera 4. Note that the some dimensions of the camera are outside of the (88,88,88) frame.

Table 27. Predefined Ground Coordinates Control Point : For the calculation of image resection.

Ground Coordinates Control Points (X,Y,Z)			
Control Points	X [cm]	Y [cm]	Z [cm]
1	22	44	6.35
2	44	22	6.35
3	66	44	6.35
4	44	66	6.35

Using the implement equations provided in *Engineerings Tools and Process*, the code is compiled in the *Python*. The following output are generated for each of the position of each camera and the outputs were graphed using *MATLAB*.

Table 28. Calculated Exterior Orientations (6).

Camera	X [cm]	Y [cm]	Z [cm]	Omega [o]	Phi [o]	Kappa [o]
1	6.681700	84.91290	106.9060	0.0127	359.9920	232.3567
2	0.964900	0.372900	106.8558	-0.0211	359.9642	139.9983
3	77.36350	3.270700	106.9076	0.0129	359.9918	232.2930
4	83.81370	87.84900	106.8561	-0.0213	359.9637	140.2008

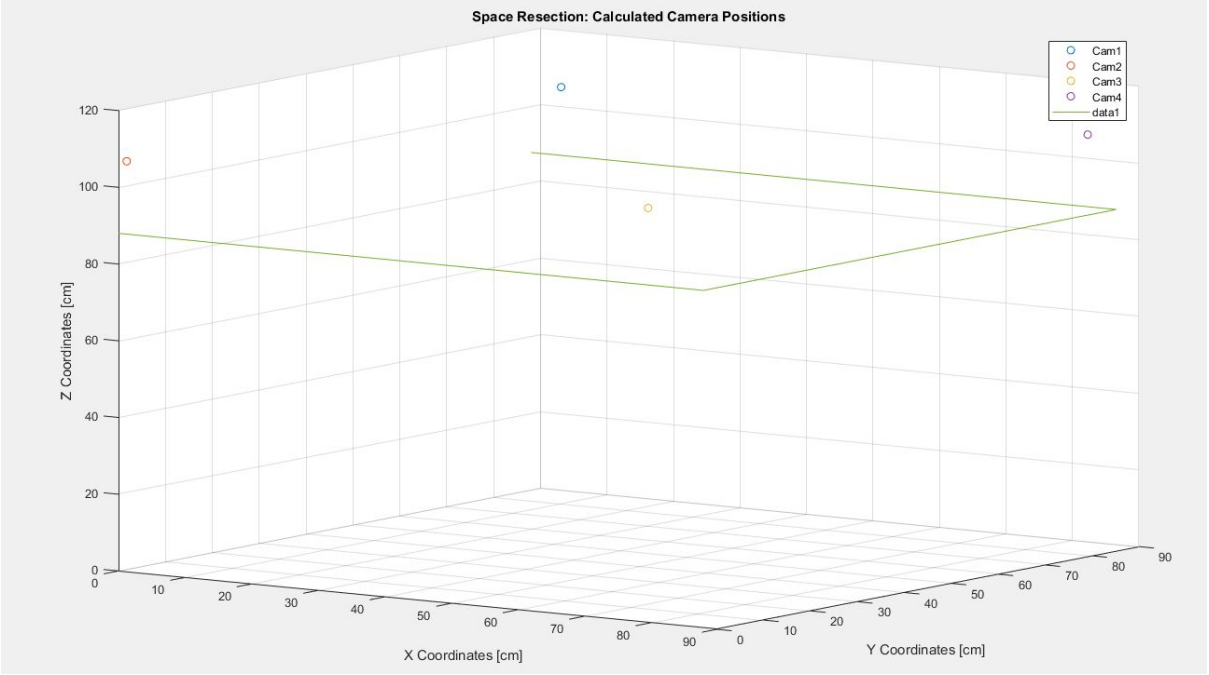

Figure 7. Space Resection and Position Calculation of Cameras 1-4 : The cameras placed on top of the frame has a value that is greater than the frame maximum values of (88,88,88). Note that green is the frame.

4. Self-Evaluation of the Test Readiness Review (Rubric)

Table 28. Self-Evaluation of the Test Readiness Review

RUBRIC CRITERION FOR TRR	SELF-EVALUATION RANKING	OUR JUSTIFICATIONS
Did you Build the Right System with Right Tools? Use appropriate engineering tools to produce a complete system that matches its requirements	Level 4: Effectively uses appropriate tools to produce a complete system that matches its requirements	A clear overview of the complete system required for testing is described. Major engineering processes and tools specified by the requirements are described in more detail.
Clear Instructions Provide clear instructions for engineering tasks	Level 4: Instructions are very clear and timely; checks that message has been correctly understood	An overview of the testing plan was provided. Each requirement and testing procedure was reviewed. We explained the procedure and future actions to take for failure for each test.
Health & Safety Incorporate health and safety practices appropriate to the context.	Level 3: Incorporates health and safety practices appropriate to the context	We have complied with the regulations that govern the design phase. We have also reviewed all of the health and safety considerations that are applicable and relevant to the testing phase.

5. References

[1] Costas , Armenakis. “Photogrammetry Lecture 8.” 1 Dec. 2017, Toronto. Slides 30-36

[2] Fda.gov. (2018). *Design Control Guidance - For Medical Device Manufacturers*. [online] Available at: http://www.fda.gov/RegulatoryInformation/Guidances/ucm070627.htm#_Toc382720788. [Accessed 5 Feb. 2018].

[3] Gazette.gc.ca. (2018). *Canada Gazette – GOVERNMENT NOTICES*. [online] Available at: http://www.gazette.gc.ca/rp-pr/p1/2017/2017-07-01/html/notice-avis-eng.html#ne6 [Accessed 5 Feb. 2018].

[4] Tc.gc.ca. (2018). *Drone safety - Transport Canada*. [online] Available at: https://www.tc.gc.ca/eng/civilaviation/drone-safety.html [Accessed 5 Feb. 2018].

[5] Tc.gc.ca. (2018). *No drone zones - Transport Canada*. [online] Available at: https://www.tc.gc.ca/eng/civilaviation/opssvs/no-drone-zones.html [Accessed 5 Feb. 2018].

6. Appendix

Raw data for camera 1-4 (pixel coordinates)

cam1: 2018-02-05 18:54:21 x: 0 y: 0
cam2: 2018-02-05 18:54:21 x: 0 y: 0
cam3: 2018-02-05 18:54:21 x: 0 y: 0
cam4: 2018-02-05 18:54:21 x: 0 y: 0
cam3: 2018-02-05 18:54:22 x: 300.832336426 y: 286.844421387
cam2: 2018-02-05 18:54:22 x: 188.700942993 y: 300.16506958
cam1: 2018-02-05 18:54:22 x: 300.832336426 y: 286.844421387
cam4: 2018-02-05 18:54:22 x: 188.700942993 y: 300.16506958
cam3: 2018-02-05 18:54:23 x: 300.5 y: 288.3387146
cam1: 2018-02-05 18:54:23 x: 300.5 y: 288.3387146
cam4: 2018-02-05 18:54:23 x: 188.508209229 y: 300.428863525
cam2: 2018-02-05 18:54:23 x: 188.508209229 y: 300.428863525
cam3: 2018-02-05 18:54:24 x: 300.0 y: 288.0
cam1: 2018-02-05 18:54:24 x: 300.0 y: 288.0
cam2: 2018-02-05 18:54:24 x: 187.224731445 y: 299.827514648
cam4: 2018-02-05 18:54:24 x: 187.224731445 y: 299.827514648
cam2: 2018-02-05 18:54:25 x: 187.0 y: 299.0
cam3: 2018-02-05 18:54:25 x: 302.0 y: 287.600006104
cam1: 2018-02-05 18:54:25 x: 302.0 y: 287.600006104
cam4: 2018-02-05 18:54:25 x: 187.0 y: 299.0
cam3: 2018-02-05 18:54:26 x: 301.362304688 y: 287.067321777
cam1: 2018-02-05 18:54:26 x: 301.362304688 y: 287.067321777
cam2: 2018-02-05 18:54:26 x: 187.484558105 y: 299.901550293
cam4: 2018-02-05 18:54:27 x: 188.942611694 y: 299.909820557
cam1: 2018-02-05 18:54:27 x: 300.5 y: 286.5
cam3: 2018-02-05 18:54:27 x: 300.5 y: 286.5
cam2: 2018-02-05 18:54:28 x: 188.658370972 y: 299.984161377
cam4: 2018-02-05 18:54:28 x: 188.06060791 y: 299.909088135
cam1: 2018-02-05 18:54:29 x: 301.546386719 y: 286.72164917
cam3: 2018-02-05 18:54:29 x: 301.261901855 y: 287.214294434
cam2: 2018-02-05 18:54:29 x: 188.572036743 y: 300.416412354
cam4: 2018-02-05 18:54:29 x: 191.0 y: 295.0
cam1: 2018-02-05 18:54:30 x: 303.0 y: 287.5
cam3: 2018-02-05 18:54:30 x: 301.0 y: 287.0
cam2: 2018-02-05 18:54:30 x: 189.0 y: 297.0
cam4: 2018-02-05 18:54:30 x: 188.201843262 y: 299.949554443
cam1: 2018-02-05 18:54:31 x: 300.0 y: 287.5
cam2: 2018-02-05 18:54:31 x: 187.192306519 y: 300.089752197
cam3: 2018-02-05 18:54:31 x: 300.683441162 y: 287.541412354
cam4: 2018-02-05 18:54:31 x: 188.605270386 y: 299.118408203
cam1: 2018-02-05 18:54:32 x: 299.502075195 y: 287.468536377
cam2: 2018-02-05 18:54:32 x: 188.5 y: 300.5
cam3: 2018-02-05 18:54:32 x: 300.5 y: 287.5
cam4: 2018-02-05 18:54:33 x: 188.5 y: 300.5
cam1: 2018-02-05 18:54:33 x: 299.769470215 y: 286.116760254
cam2: 2018-02-05 18:54:33 x: 187.745285034 y: 299.254730225
cam3: 2018-02-05 18:54:33 x: 301.0 y: 287.0

cam4: 2018-02-05 18:54:34 x: 187.437576294 y: 300.643218994
cam2: 2018-02-05 18:54:34 x: 187.5 y: 300.5
cam1: 2018-02-05 18:54:34 x: 301.540740967 y: 287.907531738
cam3: 2018-02-05 18:54:34 x: 301.5 y: 287.5
cam4: 2018-02-05 18:54:35 x: 187.892852783 y: 299.339294434
cam1: 2018-02-05 18:54:35 x: 300.0 y: 286.709686279
cam2: 2018-02-05 18:54:35 x: 187.244247437 y: 299.56137085
cam3: 2018-02-05 18:54:36 x: 302.082427979 y: 287.752746582
cam4: 2018-02-05 18:54:36 x: 188.694442749 y: 299.861114502
cam2: 2018-02-05 18:54:37 x: 188.232818604 y: 300.520751953
cam1: 2018-02-05 18:54:37 x: 301.290161133 y: 287.118438721
cam3: 2018-02-05 18:54:37 x: 301.290161133 y: 287.118438721
cam4: 2018-02-05 18:54:37 x: 187.30078125 y: 299.124847412
cam2: 2018-02-05 18:54:38 x: 187.857147217 y: 299.5
cam1: 2018-02-05 18:54:38 x: 300.5 y: 287.125
cam3: 2018-02-05 18:54:38 x: 300.264099121 y: 286.331115723
cam4: 2018-02-05 18:54:38 x: 188.10345459 y: 300.0
cam1: 2018-02-05 18:54:39 x: 300.0 y: 287.0
cam2: 2018-02-05 18:54:39 x: 187.5 y: 299.5
cam3: 2018-02-05 18:54:39 x: 301.5 y: 287.62902832
cam4: 2018-02-05 18:54:40 x: 188.0 y: 300.111114502
cam2: 2018-02-05 18:54:40 x: 187.598800659 y: 299.871185303
cam1: 2018-02-05 18:54:40 x: 300.5 y: 287.37097168
cam3: 2018-02-05 18:54:40 x: 302.0 y: 287.5
cam4: 2018-02-05 18:54:41 x: 187.655303955 y: 300.050415039
cam2: 2018-02-05 18:54:41 x: 188.075210571 y: 299.626739502
cam1: 2018-02-05 18:54:41 x: 301.708343506 y: 285.875
cam3: 2018-02-05 18:54:41 x: 302.0 y: 286.0
cam4: 2018-02-05 18:54:42 x: 187.566879272 y: 299.045898438
cam2: 2018-02-05 18:54:42 x: 188.061386108 y: 299.511505127
cam1: 2018-02-05 18:54:42 x: 302.5 y: 287.5
cam3: 2018-02-05 18:54:43 x: 304.081634521 y: 286.172058105
cam4: 2018-02-05 18:54:43 x: 187.759109497 y: 300.597167969
cam2: 2018-02-05 18:54:43 x: 187.066818237 y: 299.555297852
cam1: 2018-02-05 18:54:44 x: 302.013427734 y: 286.322143555
cam3: 2018-02-05 18:54:44 x: 302.411773682 y: 286.764709473
cam4: 2018-02-05 18:54:44 x: 189.0 y: 301.1612854
cam2: 2018-02-05 18:54:45 x: 189.0 y: 301.0
cam1: 2018-02-05 18:54:45 x: 302.5 y: 286.625
cam3: 2018-02-05 18:54:45 x: 302.332214355 y: 286.848999023
cam4: 2018-02-05 18:54:46 x: 187.574310303 y: 300.419616699
cam2: 2018-02-05 18:54:46 x: 188.154205322 y: 300.06539917
cam1: 2018-02-05 18:54:46 x: 300.681030273 y: 286.76361084
cam3: 2018-02-05 18:54:46 x: 301.264099121 y: 286.331115723
cam4: 2018-02-05 18:54:47 x: 188.182250977 y: 298.733642578
cam2: 2018-02-05 18:54:47 x: 188.491256714 y: 299.897979736
cam1: 2018-02-05 18:54:47 x: 300.0 y: 286.5
cam3: 2018-02-05 18:54:47 x: 301.5 y: 286.0
cam4: 2018-02-05 18:54:48 x: 189.307769775 y: 299.37802124
cam2: 2018-02-05 18:54:48 x: 187.590530396 y: 299.224273682

cam1: 2018-02-05 18:54:48 x: 300.673187256 y: 285.544677734
cam3: 2018-02-05 18:54:48 x: 300.680847168 y: 285.787231445
cam4: 2018-02-05 18:54:49 x: 187.019241333 y: 298.980773926
cam2: 2018-02-05 18:54:49 x: 190.0 y: 296.0
cam1: 2018-02-05 18:54:49 x: 301.475219727 y: 286.82232666
cam3: 2018-02-05 18:54:50 x: 302.0 y: 287.0
cam4: 2018-02-05 18:54:50 x: 187.837219238 y: 299.279083252
cam2: 2018-02-05 18:54:50 x: 188.636352539 y: 299.545440674
cam1: 2018-02-05 18:54:51 x: 301.5 y: 287.0
cam3: 2018-02-05 18:54:51 x: 300.0 y: 286.612884521
cam4: 2018-02-05 18:54:52 x: 187.0 y: 300.5
cam1: 2018-02-05 18:54:52 x: 301.454101562 y: 287.054901123
cam2: 2018-02-05 18:54:52 x: 187.0 y: 300.5
cam3: 2018-02-05 18:54:52 x: 301.0 y: 285.386352539
cam4: 2018-02-05 18:54:53 x: 186.059631348 y: 300.821075439
cam1: 2018-02-05 18:54:53 x: 301.507751465 y: 284.779083252
cam2: 2018-02-05 18:54:53 x: 186.463302612 y: 300.958740234
cam3: 2018-02-05 18:54:53 x: 302.5 y: 284.847808838
cam1: 2018-02-05 18:54:54 x: 299.5 y: 286.5
cam4: 2018-02-05 18:54:54 x: 187.459991455 y: 300.459991455
cam3: 2018-02-05 18:54:54 x: 301.123840332 y: 286.215576172
cam2: 2018-02-05 18:54:54 x: 186.5 y: 300.5
cam1: 2018-02-05 18:54:55 x: 302.0 y: 286.810333252
cam4: 2018-02-05 18:54:55 x: 185.5 y: 300.69354248
cam2: 2018-02-05 18:54:56 x: 186.5 y: 300.0
cam3: 2018-02-05 18:54:56 x: 300.5 y: 287.5
cam1: 2018-02-05 18:54:57 x: 299.0 y: 288.5
cam4: 2018-02-05 18:54:57 x: 188.0 y: 295.5
cam2: 2018-02-05 18:54:57 x: 187.5 y: 294.5
cam3: 2018-02-05 18:54:57 x: 299.617645264 y: 286.852935791
cam1: 2018-02-05 18:54:58 x: 298.243591309 y: 288.320526123
cam4: 2018-02-05 18:54:58 x: 187.928573608 y: 295.642883301