Senior Design

Automatic Face Tracking Camera

System Test Report V2.1

08/18/22

Summer 2022

Team 12: All Stars

Nicholas Frazer

Dylan Peters

Hesham Alaidarous

Michael Ankony

Advisor: Paul Watta

Table of Contents

1. Introduction

1.1 Purpose
This Test Report document presents the system test of the Automatic Face Tracking Camera. This document's purpose is to describe the extent of which the System's Requirements, defined in the System Requirements Specification, were met.

1.2 Scope of the Test Plan
The system test report will include all recorder data collected from our system test. As described in the System Test Plan, raw data will be collected and presented for each defined test case. The test report will analyze the data and, based on the Requirements Specification, will make a determination on the quality of each test case.

1.3 Test Case Execution:

- Test Case 4: Camera Rotation Range
- Test Case 1: Camera FOV
- Test Case 2: Camera FPS
- Test Case 3: Camera Detection Accuracy
- Test Case 5: Camera Rotation Speed
- Test Case 6: Image Aspect Ratio
- Test Case 7: Displaying/Storing Images

2. Test Case Results

2.1 Test Case: Camera FOV (Requirement 3.1.1)

Started testing on 08/04/2022 at 12:30 pm. Completed testing on 08/04/2022 at 1:00 pm.

2.1.1 Test Case Summary:

- Test Initialization:
  - Connect motors to Arduino and the Arduino to the Raspberry Pi.
  - Initialize motor positions to be centered on the horizontal and vertical axis.

- Test Steps:
  1. Run Test Code to rotate the horizontal motor to its maximum clockwise direction.
  2. Measure degrees rotated from the starting position.
  3. Repeat the first two steps for the maximum counterclockwise direction.
  4. Repeat the steps for vertical rotation.

- Test Data Log:
  - The maximum degree difference from the starting position will be recorded for the clockwise and counterclockwise changes for both the horizontal and vertical motors. Each direction and motor will be run 2 times.

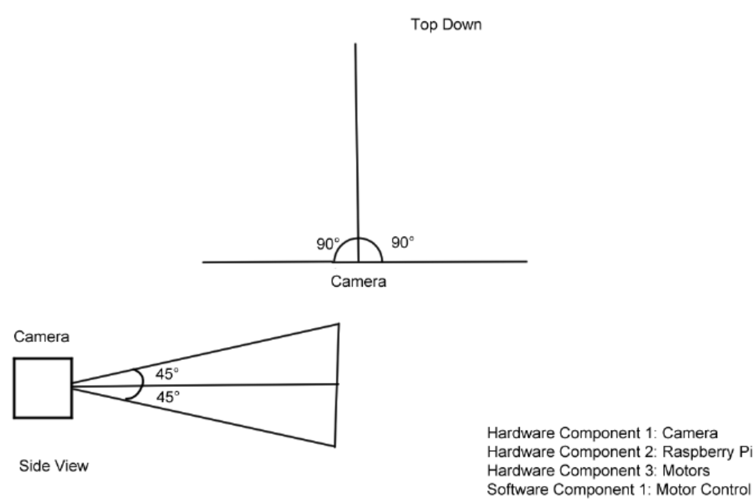


Figure 2.1.1: Camera FOV Test Case Setup

2.1.2 Test Environment and Conditions:

All tests will be conducted in a lab environment with ideal lighting. The background of all tests will be against a static background of a light color (either white or beige). The test environment will be room temperature and will not be introduced to any moisture.

2.1.3 Input Data Set:

The camera will have a specified FOV, which is determined by the model of the camera, and is measured in degrees.

2.1.4 Expected Data Values and Results:

The expected data will be the width and height of the FOV measured in centimeters and used to calculate degrees. The expected value of this will be at least 60 degrees horizontally and 40 vertically.

2.1.5 Test Case Summary Data Analysis:

- Data Logs:
    - The average horizontal FOV was found to be 61.93 degrees, with a range of 1.47.
    - The average vertical FOV was found to be 43.6 degrees, with a range of 1.29.
- Anomalies:
    - No anomalies were reported during this test case.
- Criteria for Test Suspension:
    - Testing will be suspended in the event that the field of view is not within the required range.

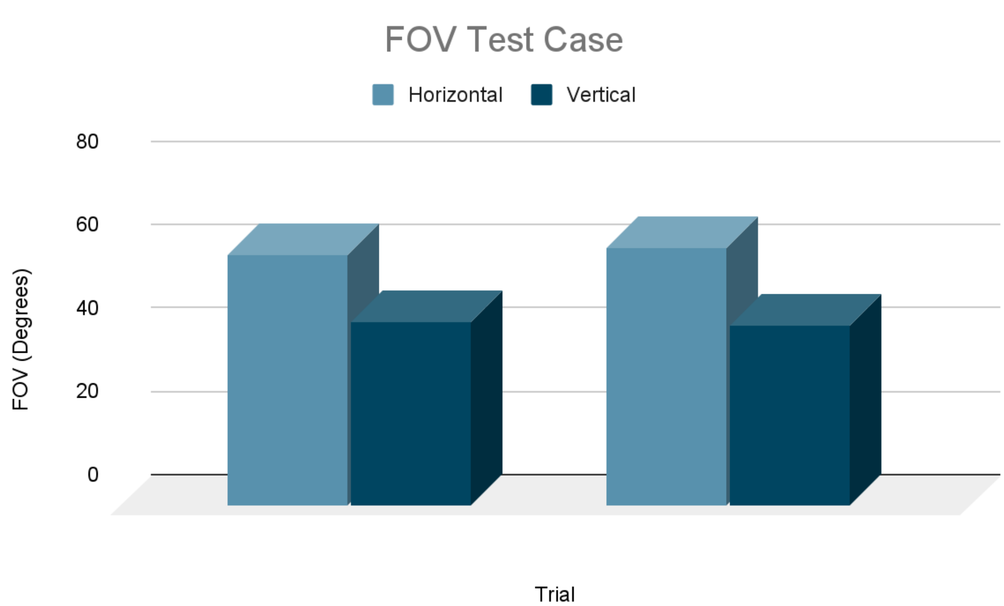


Figure 2.1.2: Graph of Results For FOV Test Case

2.1.6 Test Case Analysis:

Per requirement 3.1.1, these test results slightly exceeded the requirements.

Data Type	Test Criteria	Test Data Result	Rating
Horizontal FOV	Degrees	61.93 ± 0.5	Satisfactory
Vertical FOV	Degrees	43.60 ± 0.5	Satisfactory

2.2 Test Case: Camera FPS (Requirement 3.1.2)

Started testing on 08/04/2022 at 1 pm. Completed testing on 08/04/2022 at 1:30 pm.

2.2.1 Test Case Summary:

The FPS of a camera can be measured directly through software. The frames per will be monitored for a period of at least 10 min and sampled every second. The average FPS will then be calculated. This will be measured in average frames per second.

- Test Initialization:
  - Raspberry Pi will be connected to a temporary test display.
  - Device will first be tested with no user present.

- Test Steps:
  1. Run the main program with test code included, measuring the frames captured in one second by incrementing a counter for every frame captured from the camera and displaying and resetting the counter after 1 second has elapsed.
  2. Record one minute of FPS readings and find the average.
  3. Introduce the user in the center of the camera's field of view.
  4. Repeat steps one and two.

- Test Data Log:
  - Each run will record one minute of the FPS every second. There will be two runs, one including a user and one without.

2.2.2 Test Environment and Conditions:

All tests will be conducted in a lab environment with ideal lighting. The background of all tests will be against a static background of a light color (either white or beige). The test environment will be room temperature and will not be introduced to any moisture.

2.2.3 Input Data Set:

For the first part of this test, we will not input anything, just let the camera run. The second part of this test will input a tester's face to test the load on the system.

2.2.4 Expected Data Values and Results:

The expected data for this test will be at least 10 frames per second and will be considered successful if it remains above 10 frames per second on average for the entire process of the camera starting to track a user to it taking a picture. Our tape measure will have an error of ± 1 centimeter.

2.2.5 Test Case Summary Data Analysis:

- Data Logs:
    - For this test case, we found our system operated at an average of 10 FPS, with a range of 2.
- Anomalies:
    - No anomalies were reported during this test case.
- Criteria for Test Suspension:
    - Suspension will occur if FPS numbers are drastically under required values (4 FPS or less).

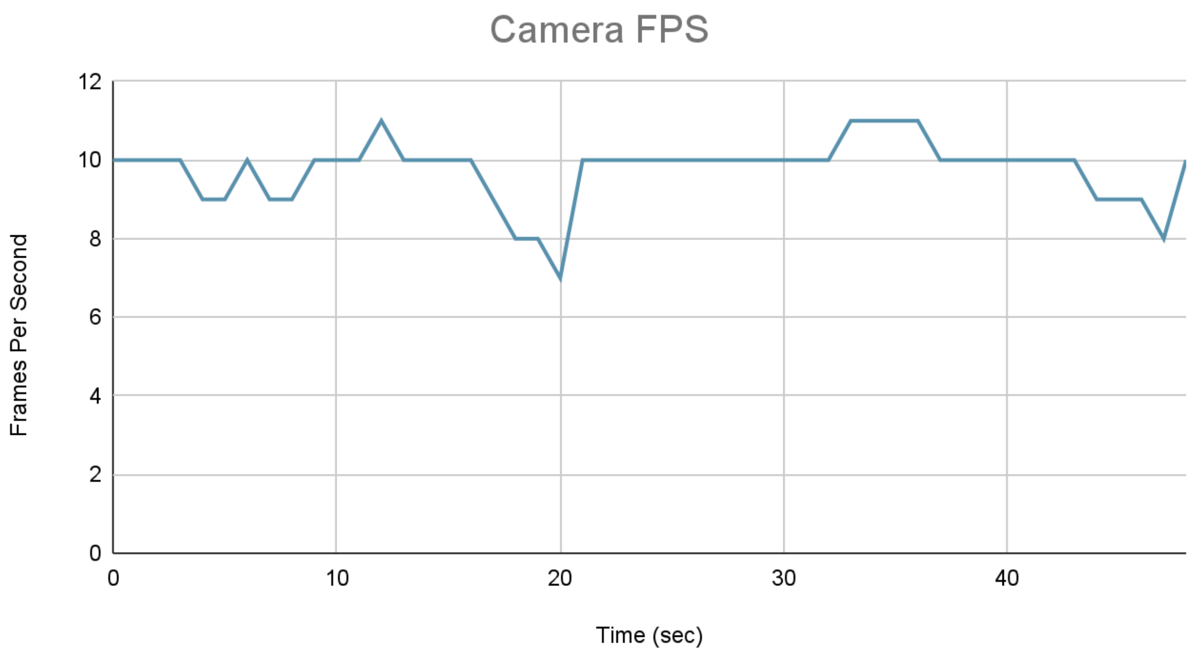


Figure 2.2.1: Graph of Results for Camera FPS Test Case

2.2.6 Test Case Analysis:

During our testing, we have satisfied the requirement 3.1.2. Our horizontal and vertical FOV is within specification.

Data Type	Test Criteria	Test Data Result	Rating
Frames Per Second	FPS	10 ± 0.5	Satisfactory

2.3 Test Case: Camera Detection Accuracy (Requirement 3.1.3-3.1.7, 3.4.1-3.4.3)

Started testing on 08/04/2022 at 1:30 pm. Completed testing on 08/04/2022 at 3:30 pm.

2.3.1 Test Case Summary:

- Test Initialization:
  - Raspberry Pi will be connected to a temporary display.
  - User will be placed within the center of the camera's FOV and will remain still.

- Test Steps:
  1. The program will be run along with test code to measure frames with detected faces and those without.
  2. Frames that detected a face and frames that did not will be totalled and recorded.
  3. The user will then begin moving at a speed of 0.5 m/s horizontally across the center of FOV.
  4. Steps one and two are repeated for new conditions.
  5. The user will then begin moving at a speed of 0.5 m/s vertically across the center of FOV.
  6. Steps one and two are repeated for new conditions.

- Test Data Log:
  - This test will be measured in the number of successes and number of failures.

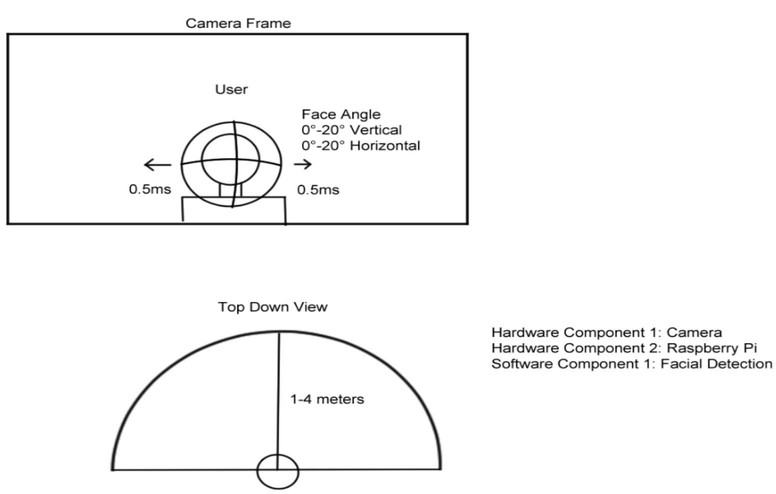


Figure 2.3.1: Camera Detection Accuracy Test Case Setup

2.3.2 Test Environment and Conditions:

All tests will be conducted in a lab environment with ideal lighting. The background of all tests will be against a static background of a light color (either white or beige). The test environment will be room temperature and will not be introduced to any moisture.

2.3.3 Input Data Set:

The range of usable radius of the device is within 1 to 3 meters of the device. The user will be able to move from 0 to 0.8 meters per second within the device's range. The user can tilt their face within 45 degrees horizontally and vertically looking towards the camera.

2.3.4 Expected Data Values and Results:

This test will be measured in the percent of successes. The expected value of this test will be an average of an 80% success rate.

2.3.5 Test Case Summary Data Analysis:

- Data Logs:
  - We found the average accuracy from a variety of different angles and distances.
- Anomalies:
  - There were multiple anomalies when the system was facing to the left of the system.
  - There were many anomalies when the distance of the tester exceeded 1.5 meters.
    - These results could be biased due to a lack of variety in test subjects (hair length and varying accessories).
- Criteria for Test Suspension:
  - If the number of successes are not within the specified numbers the test will be suspended.

Direction of User’s Face	Distance from Tester	Degree of Tilt	Average % Success	Rating
Straight	1 m	0	99.08 %	Excellent
Right	1 m	45	81.92 %	Satisfactory
Left	1 m	45	75.48 %	Failure
Down	1 m	45	99.69 %	Excellent
Up	1 m	45	99.16 %	Excellent
Moving within the Frame	1 m	0 - 45	88.12 %	Satisfactory
Straight	1.5 m	0	19.77 %	Failure
Right	1.5 m	45	0 %	Failure
Left	1.5 m	45	0 %	Failure
Down	1.5 m	45	0 %	Failure
Up	1.5 m	45	0 %	Failure
Moving within the Frame	1.5 m	0 - 45	0 %	Failure
Straight	3 m	0	0 %	Failure
Right	3 m	45	0 %	Failure
Left	3 m	45	0 %	Failure
Down	3 m	45	0 %	Failure
Up	3 m	45	0 %	Failure
Moving within the Frame	3 m	0 - 45	0 %	Failure

2.3.6 Test Case Analysis:

While this test case performed well by meeting requirements 3.1.4-3.1.7 and 3.4.1-3.4.3, this system failed to meet requirement 3.1.3. This system meets all tracking and detection requirements while in a short range of one meter. If you attempt to use the system at a greater range, it fails to meet the specification.

2.4 Test Case: Camera Rotation Range (Requirement 3.2.1)

Started testing on 08/04/2022 at 3:30 pm. Completed testing on 08/04/2022 at 4:00 pm.

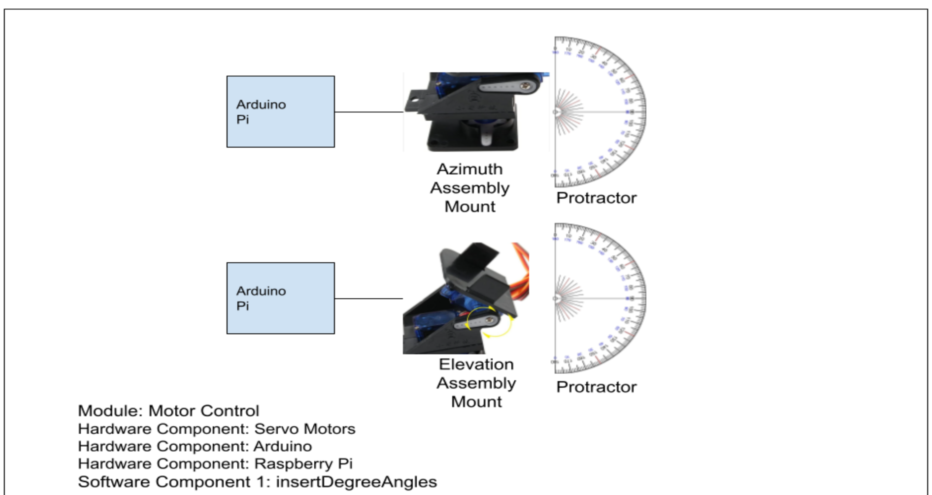
2.4.1 Test Case Summary:

- Test Initialization
  - Connect Arduino to laptop
  - In the Arduino program, set a starting position of 0 degrees for azimuth and 45 degrees for elevation.
  - Line a protractor up to the zero location for azimuth.

- Test Steps:

  1. Insert a command for azimuth to move to 105 degrees. Record actual degree traveled using protractor.
  2. Insert a command for azimuth to move to -105 degrees. Record actual degree traveled using protractor. Return azimuth to zero.
  3. Set elevation to 45 degrees. Line up protractor (0 degree) with elevation assembly.
  4. Repeat steps 1 and 2.

- Test Data Log
  - The maximum degree difference from the starting position will be recorded for the clockwise and counterclockwise changes for both the horizontal and vertical motors. Each direction and motor will be run 2 times.



Test Diagram 1.4.4: Component and Modules used in Test Case 4 to validate Requirement 3.2.1

Figure 2.4.1: Camera Rotation Range Test Case Setup

2.4.2 Test Environment and Conditions:

All tests will be conducted in a lab environment with ideal lighting. The background of all tests will be against a static background of a light color (either white or beige). The test environment will be room temperature and will not be introduced to any moisture.

2.4.3 Input Data Set:

We will input the range of the camera rotation, which will be 90 degrees, horizontally left and right, and 45 degrees, vertically up and down, of its set starting position.

2.4.4 Expected Data Values and Results:

This test will measure the range in degrees. The expected value of the camera's range is 180 degrees horizontally and 45 degrees vertically. We will be using a protractor to measure the range, so our measurement error will be ± 1 degree.

2.4.5 Test Case Summary Data Analysis:

- Data Logs:
  - The horizontal moving servo motor could rotate to 90 degrees, on average, right or left (or 180 degrees of full motion), with a range of 1.5.
  - The vertical moving servo motor could rotate greater than 50 degrees, on average, up or down (or 100 degrees of full motion), with a range of 4.
- Anomalies:
  - No anomalies were reported during this test case.
- Criteria for Test Suspension:
  - Test suspension will occur if motors do not move if commanded. Will be resumed if they respond to commands

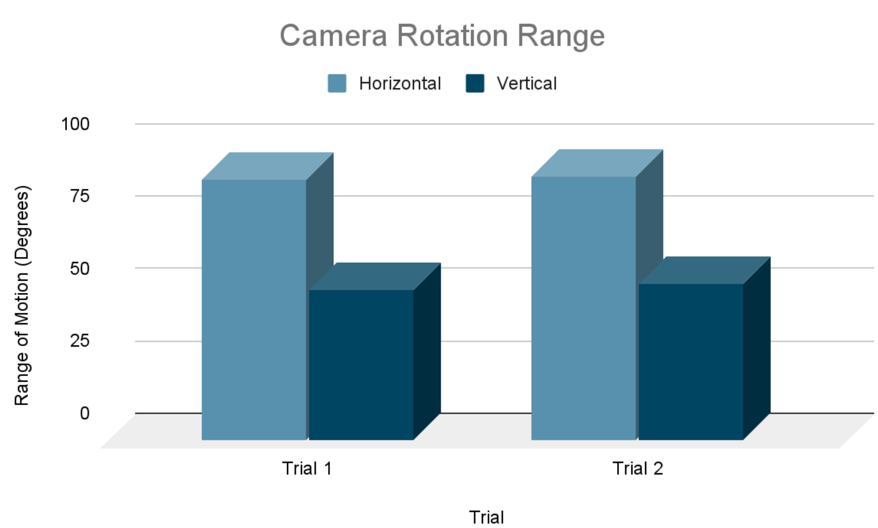


Figure 2.4.2: Graph of Camera Rotation Range Test Case Results

2.4.6 Test Case Analysis:

Per our requirements for 3.2.1, we met the requirement for the horizontal servo motor and exceeded the requirement for the vertical motor.

Data Type	Test Criteria	Test Data Result	Rating
Horizontal Range	Degrees	90 ± 0.5	Satisfactory
Vertical Range	Degrees	50 ± 0.5	Excellent

2.5 Test Case: Camera Rotation Speed (Requirement 3.2.2)

Started testing on 08/05/2022 at 12:30 pm. Completed testing on 08/05/2022 at 2:30 pm.

2.5.1 Test Case Summary:

- Test Initialization
    - Connect Arduino to laptop.
    - One tester will sit 1.5 meters away from the camera.
    - Mark 1 meter right and left of the seated position of the tester with tape.
    - Find 1 meter above and below the top of the tester's head with a meter stick.
    - Tester will sit still during camera power on and through initialization.
    - Set protractor starting azimuth at 0.

- Test Steps:

  1. Set azimuth to 0 and center the camera on the still tester. Align protractor to azimuth assembly (see test procedure 1.4.4).

  2. At a speed of 0.1 meters per second, move to the tester's left, making sure that the tester's image is within the center of the camera image. If the camera has reached the maximum range (90 degrees), record a success, otherwise a failure. Repeat this step four times.

  3. Repeat camera initialization steps. Repeat step 1, except for moving to the tester's right, making sure that the tester's image is within the center of the camera image. If the camera has reached the maximum range (90 degrees), record a success, otherwise a failure. Repeat this step four times.

  4. Return azimuth to 0 degrees. Then, set elevation to 45 degrees. Align protractor with elevation assembly (see test procedure for 1.4.4).

  5. Repeat camera initialization steps. At a speed of 0.1 meters per second, the tester will stand up, making sure that the tester's image is within the center of the camera image. If the camera has reached the maximum range (45 degrees), record a success, otherwise a failure. Repeat this step four times.

  6. Repeat camera initialization steps. Repeat step 3, except the tester will be moving down. If the camera has reached the maximum range (45 degrees), record a success, otherwise a failure. Repeat this step four times.

  7. Repeat steps 1-4, with the tester moving at 0.8 meters per second.

- Test Data Log
    - This test will be run at least 4 times per direction and minimum/maximum speed (32 times total). Each trial will record the direction, speed, and success of track targets.

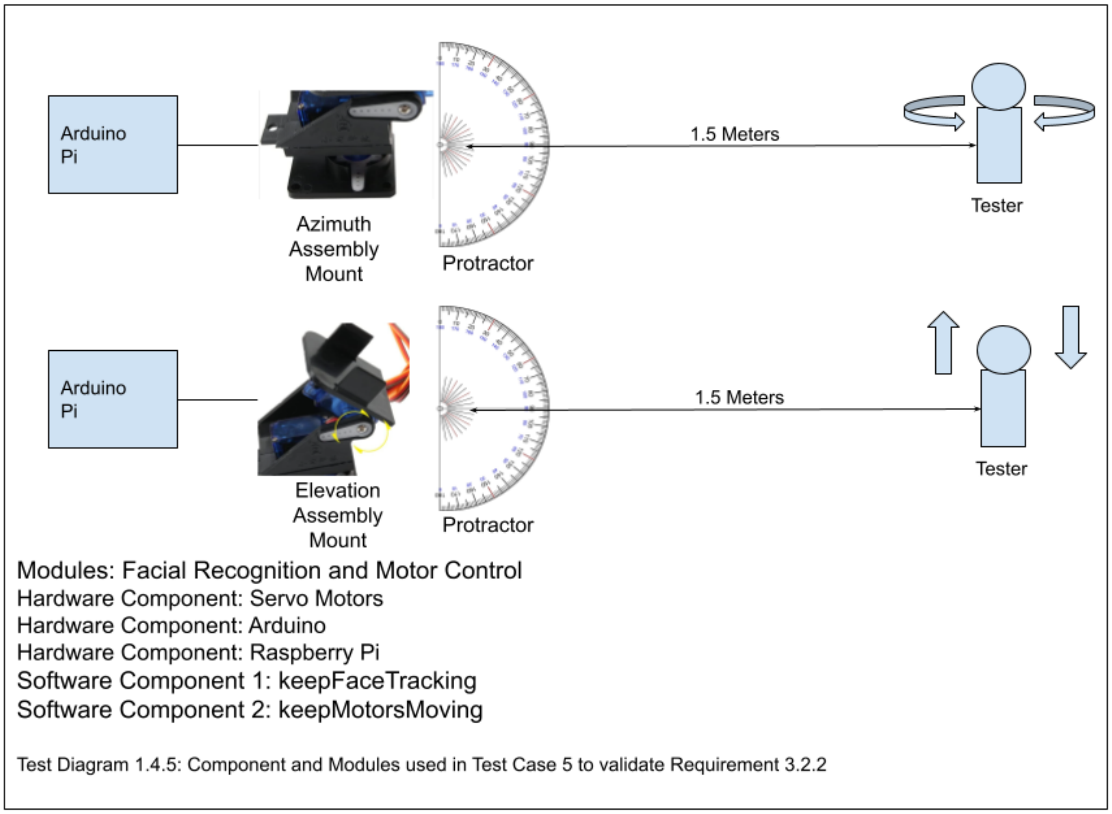


Figure 2.5.1: Camera Rotation Speed Test Case Setup

2.5.2 Test Environment and Conditions:

All tests will be conducted in a lab environment with ideal lighting. The background of all tests will be against a static background of a light color (either white or beige). The test environment will be room temperature and will not be introduced to any moisture.

2.5.3 Input Data Set:

The data inputted for this test will be reliant on the test subjects speed and position which will remain within the specified range. The test subject will move at various speeds within the range.

2.5.4 Expected Data Values and Results:

The expected value of this test will be a maximum of 0.8 meters per second. The expected result of this test is to have it match the user's speed. Due to the test setup (moving a measured difference, over a specific time), there is ± 0.2 meters per second error.

2.5.5 Test Case Summary Data Analysis:

- Data Logs:
    - The system could not track a target at 0.5 meters per second.
    - Test was suspended and we reduced our speed to 0.25 meters per second.
        - We were able to track 82.40% of the time at reduced speed.
- Anomalies:
    - No anomalies were reported during this test case.
- Criteria for Test Suspension:
    - Test suspension will occur if the motors do not respond to moving faces. Test will resume if the issue is fixed

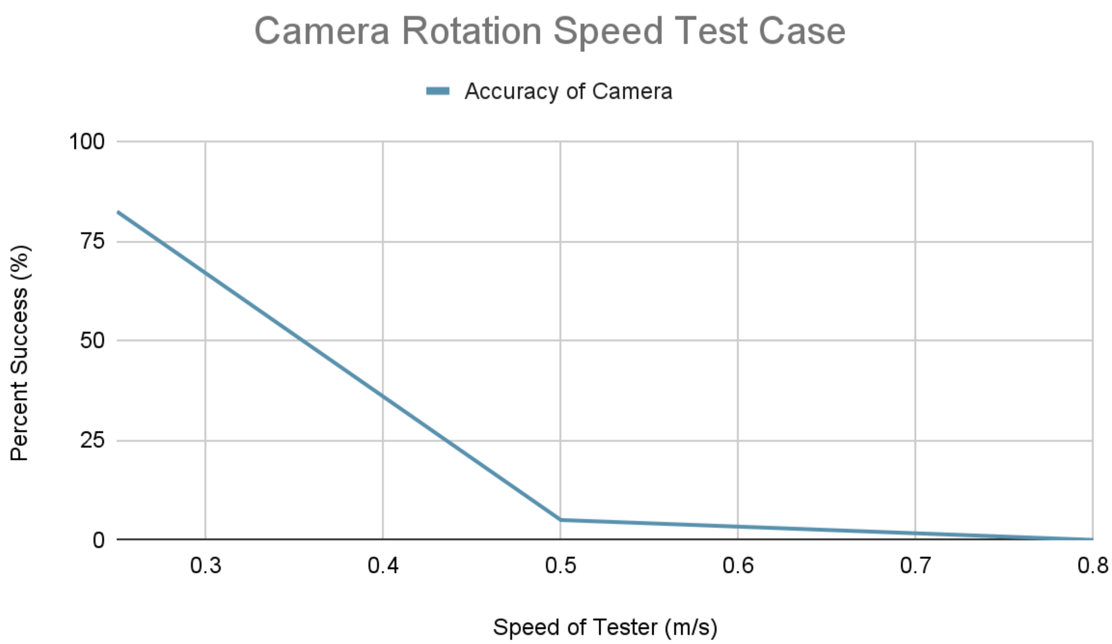


Figure 2.5.2: Camera Rotation Speed Test Case Results

2.5.6 Test Case Analysis:

Per our requirement, we failed to meet requirement 3.2.2. The system could not keep pace with the tester. At slower speeds, the system is able to detect and track with the user. However, it is not at the maximum speed set in the requirements.

Data Type	Test Criteria	Test Data Result	Rating
User’s Max Speed	Successful Tracking	0 %	Failure

2.6 Test Case: Image Aspect Ratio (Requirement 3.3.1)

Started testing on 08/05/2022 at 2:30 pm. Completed testing on 08/04/2022 at 3:00 pm.

2.6.1 Test Case Summary:

- Test Initialization
  - Raspberry Pi will be connected to a temporary display.

- Test Steps:
  1. Images will be captured using a test program.
  2. Images will be taken until 20 images are saved.
  3. Record the image aspect ratio of each trial.

- Test Data Log
  - This test will be measured in the successes of the aspect ratios of resulting images.

2.6.2 Test Environment and Conditions:

All tests will be conducted in a lab environment with ideal lighting. The background of all tests will be against a static background of a light color (either white or beige). The test environment will be room temperature and will not be introduced to any moisture.

2.6.3 Input Data Set:

There will be no input data for this test

2.6.4 Expected Data Values and Results:

The aspect ratio of images is measured in the number of pixels height and width. The expected value of this test is 1080x1080.

2.6.5 Test Case Summary Data Analysis:

- Data Logs:
  - The aspect ratio of the images captured was 800x800.
    - Testing at a 1080x1080 aspect ratio the images appear distorted and incomprehensible.
- Anomalies:
  - No anomalies were reported during this test case.
- Criteria for Test Suspension:
  - Tests will be terminated if images are not being saved.

2.6.6 Test Case Analysis:

Per our requirement, we failed to meet requirement 3.3.1. While we can output a 1000 x 1000 image, it is distorted and of extremely poor quality. We can only output a quality image with an aspect ratio of 800 x 800.

Data Type	Test Criteria	Test Data Result	Rating
Image Aspect Ratio	Pixels	800 x 800	Failure

2.7 Test Case: Displaying/Storing Images (Requirement 3.3.2-3.3.4)

Started testing on 08/04/2022 at 3:30 pm. Completed testing on 08/04/2022 at 4:00 pm.

2.7.1 Test Case Summary:

The system will show images within an accessible display. Once 20 images are taken displayed the device will be activated again, if the device saves more images then it will have failed this test.

- Test Initialization
    - Raspberry Pi will be connected to a temporary display.

- Test Steps:
    1. Images will be captured using a test program.
    2. Attempt to take 25 images.
    3. Note how many images were actually taken
    4. Power cycle Raspberry Pi.
    5. Check PI to see how many images were successfully stored

- Test Data Log
    - This test will be measured in the successful saving of images. Test will be conducted three times.

2.7.2 Test Environment and Conditions:

All tests will be conducted in a lab environment with ideal lighting. The background of all tests will be against a static background of a light color (either white or beige). The test environment will be room temperature and will not be introduced to any moisture.

2.7.3 Input Data Set:

Commands will be input into the Raspberry Pi.

2.7.4 Expected Data Values and Results:

This test will be measured in the success rate. The expected result is that once the memory has reached its designated limit of images it will cease capturing images.

2.7.5 Test Case Summary Data Analysis:

- Data Logs:
    - The system captured 20 images.
    - The system saved 86.66% of those images.
- Anomalies:
    - No anomalies were reported during this test case.
- Criteria for Test Suspension:
    - Test will be terminated if no images are being saved.

2.7.6 Test Case Analysis:

Per the requirements, we met the requirement of 3.3.2-3.3.4. The system captured exactly 20 images and saved them successfully a majority of the time.

Data Type	Test Criteria	Test Data Result	Rating
Capturing Images	Successfully Captured	20	Satisfactory
Saving Images	Successfully Saving	86.66 %	Satisfactory

3. Test Report Summary

3.1 Summarize Overall Test Results

Test Case	Test Name	Test Result	Error	Rating
1.1	FOV Horizontal	61.93 Degrees	± 0.5 Degree	Satisfactory
1.2	FOV Vertical	43.6 Degrees	± 0.5 Degree	Satisfactory
2	FPS	10 FPS	± 0.5 Frame	Satisfactory
3.1	Accuracy Straight - 1m	99.08%	± 0.2 m/s	Excellent
3.2	Acc. Right - 1m	81.92%	± 0.2 m/s	Satisfactory
3.3	Acc. Left - 1m	75.48%	± 0.2 m/s	Failure
3.4	Acc. Down - 1m	99.69%	± 0.2 m/s	Excellent
3.5	Acc. Up - 1m	99.16%	± 0.2 m/s	Excellent
3.6	Acc. Moving - 1m	88.12%	± 0.2 m/s	Satisfactory
3.7	Acc. Straight - 1.5m	19.77%	± 0.2 m/s	Failure
3.8	Acc. Right - 1.5m	0 %	± 0.2 m/s	Failure
3.9	Acc. Left - 1.5m	0 %	± 0.2 m/s	Failure
3.10	Acc. Down - 1.5m	0 %	± 0.2 m/s	Failure
3.11	Acc. Up - 1.5m	0 %	± 0.2 m/s	Failure
3.12	Acc. Moving - 1.5m	0 %	± 0.2 m/s	Failure
3.13	Acc. Straight - 3m	0 %	± 0.2 m/s	Failure
3.14	Acc. Right - 3m	0 %	± 0.2 m/s	Failure
3.15	Acc. Left - 3m	0 %	± 0.2 m/s	Failure
3.16	Acc. Down - 3m	0 %	± 0.2 m/s	Failure
3.17	Acc. Up - 3m	0 %	± 0.2 m/s	Failure
3.18	Acc. Moving - 1.5m	0 %	± 0.2 m/s	Failure
4.1	Rotation Horizontal	90 Degrees	± 0.5 Degree	Satisfactory
4.2	Rotation Vertical	50 Degrees	± 0.5 Degree	Excellent
5.1	Rotation Speed 0.5m/s	0%	± 0.2 m/s	Failure
5.2	Rotation Speed 0.25m/s	82.40%	± 0.2 m/s	Satisfactory
6	Aspect Ratio	800x800	N/A	Failure
7.1	Taking Image	20 Images	N/A	Satisfactory
7.2	Storing Images	86.66	N/A	Excellent

3.2 Path Forward

To resolve the issues with the camera rotation speed, we plan to continue researching new python libraries and cascade classifiers to enhance our ability to focus on moving targets. We also will consider the use of a new lens to give higher definition and ability to track farther moving targets. These issues could also be alleviated by using a more powerful processor that is more suited for image processing. This would allow us to increase the aspect ratio of the images that the system is trying to detect faces in, which will increase distance and overall accuracy. The system's ability to track users could also be improved by using motors with feedback control to correct for errors.

To resolve our image aspect ratio, we would consider a new camera lens with the ability to capture more pixels without distorting the image.

To improve our FPS, we will test new coding methods and libraries to speed up the performance of the Raspberry Pi. A more specialized image processor could also greatly improve the FPS.

Most servo motors that fit within the budget of our project only rotate 180 degrees. To exceed our requirement, we would have to pick a different type of motor with the same speed as the servo motors.

3.3 System Test Summary

This system was promising, yet unsuccessful. We failed to meet some of our core requirements, such as 3.2.2 and 3.3.1. The facial tracking is accurate enough for the requirements, however the speed of the user has to be severely limited. Also, the user can only be a short distance away from the camera.

The project still has some ways to go in improving the quality of the camera. We have proved the core fundamentals are possible, but fell short in meeting our requirements.

4. References:

[1] System Requirements Specification
[2] System High Level Design Specification
[3] System Low Level Design Specification
[4] System Test Plan
[5] System Test Result Master Excel File