**Flight test 1.5**

**Why are we testing and what is the purpose of the planned test?**

The reason for this flight test is to be able to pass an independent flight test: the ability to fly a single drone with the CV implemented. The software should be able to read in GPS points and the drone will fly to predetermined points by the software and identify a human target by the implemented CV algorithm.

The current configuration of the software is mature enough to be tested at this level because at this point, we believe that the software should be able to handle flying one and two drones with video and target detection. By completing this test, we will be able to verify what parts of the software are compatible with the drone, and see what parts need to be fixed before the next test. Before the drone was taken off the string, all pre-mission tests were flown with a string attached to the drone. In addition, all code used in the flight test was unit tested, and if possible, the Parrot simulator. The CV algorithm was tested by running through pre-collected photos from previous test events.

The first test event was scheduled on 3/14/2021, however, we did not fly because we did not pass the pre-flight checklist stating that we will fly in wind under 30 mph; the gusts that day were 35 mph. We flew on our alternative day 3/17/2021.

**Requirements Tested:**

## 

|  |  |  |
| --- | --- | --- |
| **Subsystem** | Requirement | Tested? |
| **Operator Interface** |  | Yes |
| 1 | The system SHALL display the following: |  |
| 1a | GPS point inputs |  |
| 1b | Video livestream for all drones |  |
| 1c | A map of the area |  |
| 2 | The system SHALL display an image when a person is detected |  |
| 3 | The system SHALL be automatically displayed upon software turn on |  |
| 4 | The computer SHALL be able to identify a drones location |  |
| **Drone Actions** |  | Yes |
| 5 | The drone SHALL follow all GPS points given |  |
| 6 | The drone SHALL send live video feed |  |
| 7 | The drone SHALL return to homebase after a critical battery event |  |
| 8 | The drone SHALL send GPS coordinates to the computer |  |
| 9 | After identifying the target, the drone SHALL send an image of the target |  |
| 10 | The system SHALL accept 1 or more drones |  |
| **Mission Planning** |  | Yes |
| 11 | The system SHALL create a mission for each flight |  |
| 12 | The system SHALL send the mission to each drone |  |
| 13 | The system SHALL designate a leader drone, if there is more than one drone connected |  |
| 14 | The computer SHALL break up the amount of land into the pieces, with the amount of drones being the number of spaces |  |
| 15 | The system SHALL use Wi-Fi or the Parrot SkyController to connect to the drone |  |
| 16 | The computer SHALL remember where the previous drone locations are |  |
| **CV** |  | Yes |
| 17 | The CV SHALL be able to detect a human |  |
| 18 | The CV SHALL be able to process at least 2 frames per second on an Intel I7 processor |  |
|  |  |  |
| **SOS** |  | No |
| 19 | The drone SHALL send an SOS alert after going to the ground, if still connected |  |
| 20 | After a drone send an SOS alert, the remaining drones SHALL receive new GPS locations |  |
| 21 | After the computer receives a SOS alert, the computer SHALL resize the area |  |

## 

**Pre flight checklist:**

* Verify that the battery is charged
* Verify the SkyController is charged
* Verify that the clouds are above 400 feet
  + If cloud coverage is under 400, verify that the software can only go 25 feet under the cloud coverage
* Verify that the wind is under 30 mph
* Verify the laptop is charged
* Verify that the battery is property in the drone
* Verify that the drone’s wings are correctly fitted
* Verify that the location of which we are flying is unrestricted

**Pre-mission tests:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Up | SDK | 1. Connect the drone by Wi-Fi or SkyController  2. Input a message in the UI saying that the drone should go up 25 feet  3. Verify that the drone goes up 25 feet from the starting position | Pass |
| 2 | Left | SDK | 1. Connect the drone by Wi-Fi or SkyController  2. Input a message in the UI saying that the drone should go left 25 feet  3. Verify that the drone goes left 25 feet from the starting position | Pass |
| 3 | Right | SDK | 1. Connect the drone by Wi-Fi or SkyController  2. Input a message in the UI saying that the drone should go right 25 feet  3. Verify that the drone goes right 25 feet from the starting position | Pass |
| 4 | Down | SDK | 1. Connect the drone by Wi-Fi or SkyController  2. Manually fly the drone up 30 feet  3. Input a message in the UI saying that the drone should go down 25 feet  4. Verify that the drone goes down 25 feet from the starting position | Pass |
| 5 | Camera | SDK | 1. Turn on the system  2. Connect the drone by Wi-Fi or SkyController  3. Allow the system to accept camera feed  4. Verify that the camera feed is displayed on the computer | Pass |
| 6 | Moving Camera | SDK | 1. Turn on the system  2. Connect the drone by Wi-Fi or SkyController  3. Allow for the system to accept camera feed  4. Verify that the camera feed is displayed on the computer  5. Verify that the user can move | Pass |
| 7 | Take off | SDK | 1. Verify that the drone can take off | Pass |
| 8 | Land | SDK | 1. Verify that the drone is in the air  2. Input a message to allow the drone to land  3. Verify that the drone can land | Pass |
| 9 | Battery | SDK | 1. Verify that the battery level is less than 25%  2. Verify that the drone is coming back home and then landing | Pass |
| 10 | Emergency Landing (Wi-FI) | SDK | 1. Connect the drone by Wi-Fi  2. Input message that allows the drone to fly 20 feet  3. Disconnect the drone from Wi-Fi  4. Verify that the drone can connect to the phone's Parrot App by Wi-Fi | Pass |
| 11 | Emergency Landing (Sky Controller) | SDK | 1. Connect the drone by the SkyController  2. Input message that allows the drone to fly 20 feet  3. Disconnect the drone from the SkyController  4. Verify that the drone can connect to the phone's Parrot App by the SkyController | Pass |

These tests are a prerequisite to any flight test without the string. Without these tests, the drone could cause serious damage to itself or others. To verify that all of these tests work, the drone will be tethered to a string to verify that the drone will not fly off and cause damage. In theory, by allowing the drone to fly tethered on this, flying untethered will be more successful by knowing that all basic features work.

**Safety:**

Before flying the drone, we completed all pre-mission tests successfully and the flight was recorded in the flight log. In addition, the flight test was observed by one sUAS pilot and two visual observers in different locations to verify the drones were going in the correct direction. All observations were at point 14. We were able to see the drone the entire time, as complaint with Part 107.

**Location and Set-Up:**

We will be flying at Mendon Ponds Park for this flight test.

We plan on flying at a height of 300 feet to avoid all obstacles.

**Results:**

|  |  |  |
| --- | --- | --- |
| **Operator Interface** |  | Pass/Fail/NA |
| 1 | The system SHALL display the following: | N/A |
| 1a | GPS point inputs |  |
| 1b | Video livestream for all drones |  |
| 1c | A map of the area |  |
| 2 | The system SHALL display an image when a person is detected |  |
| 3 | The system SHALL be automatically displayed upon software turn on |  |
| 4 | The computer SHALL be able to identify a drones location |  |
| **Drone Actions** |  |  |
| 5 | The drone SHALL follow all GPS points given | Pass |
| 6 | The drone SHALL send live video feed | Pass |
| 7 | The drone SHALL return to homebase after a critical battery event | N/A |
| 8 | The drone SHALL send GPS coordinates to the computer | Pass |
| 9 | After identifying the target, the drone SHALL send an image of the target | N/A |
| 10 | The system SHALL accept 1 or more drones | Pass |
|  |  |  |
|  |  |  |
|  |  |  |
| **Mission Planning** |  |  |
| 11 | The system SHALL create a mission for each flight | Pass |
| 12 | The system SHALL send the mission to each drone | Fail |
| 13 | The system SHALL designate a leader drone, if there is more than one drone connected | Pass |
| 14 | The computer SHALL break up the amount of land into the pieces, with the amount of drones being the number of spaces | Pass |
| 15 | The system SHALL use Wi-Fi or the Parrot SkyController to connect to the drone | Pass |
| 16 | The computer SHALL remember where the previous drone locations are | Pass |
| **CV** |  |  |
| 17 | The CV SHALL be able to detect a human | Fail |
| 18 | The CV SHALL be able to process at least 2 frames per second on an Intel I7 processor | Pass |
|  |  |  |
| **SOS** |  | N/A |
| 19 | The drone SHALL send an SOS alert after going to the ground, if still connected |  |
| 20 | After a drone send an SOS alert, the remaining drones SHALL receive new GPS locations |  |
| 21 | After the computer receives a SOS alert, the computer SHALL resize the area |  |

During the test, we were able to successfully pass 10 out 12 tests (83%). These failed tests have led to the discovery of some failed requirements: One with ZeroMQ messaging and the CV detection algorithm.

One defect we have found is a messaging issue with ZeroMQ. The drone will be sent the appropriate message to launch, and the drone will not launch. When the software is stopped, the drone will then launch 3 feet in the air. We would then have to manually take control of the drone or use the software to bring it down. We were able to use the software once for the 300 feet test (after some failed attempts), but when we tested the 200 feet test, the ZeroMQ did not allow the drone to take off. We believe it is a defect in the ZeroMQ messaging system and plan on fixing before our next flight test.

The second defect we found was with the CV detection algorithm. While flying at 300 feet, the software collected images that were meant to be analyzed into the algorithm later. During the test, we had a team member out in the field, in addition to two other people playing with their dogs. When we analyzed the images, we got 0 positives and 0 false positives. We believe that this is an issue of flying 300 feet, rather than the algorithm itself, but we plan on testing at lower altitudes to verify it is not the algorithm.

In addition to the flight test, we tested the field of view of the drone for granularity. The drone was up 69-72 feet and saw the 79.2 foot string.

**Lessons Learned:**

Overall, we found some pain points in our software with the ZeroMQ messaging and the maximum height the drone can go without the CV working. We plan to test with multiple drones and single drones, operator interface, and a fixed ZeroMQ messaging system.

We have also learned that the height of the drone is not 100% accurate. The Parrot software only allows the drone to report in increments of 3 feet.