**Flight test 2.1**

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

This is the first flight test that has two drones connected to one computer. In prior flight tests, we have used the software to connect to one drone with the CV implemented. This software tests the prior software, with the expectation that the software will work the same as before, just with two drones connected to it.

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 test event was scheduled on 3/25/2021, after it was canceled on 3/24/2021 for weather.

**Requirements Tested:**

## 

|  |  |  |
| --- | --- | --- |
| **Subsystem** | Requirement | Tested? |
| **Operator Interface** |  | No |
| 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 |

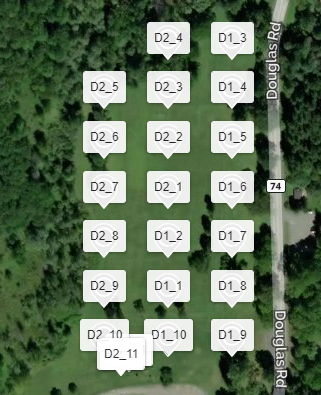
These tests are a prerequisite to any flight test without the string. Without these tests, the drone could cause serious damage to itself or others. To verify that all of these tests work, the drone will be tethered to a string to verify that the drone will not fly off and cause damage. In theory, by 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 11. 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.



D1 was the primary drone and D2 was the secondary drone. The number after the underscore was the point number; for example, D2\_3 is drone 2, 3rd point.

We plan on flying at 150 feet for one drone and 155 feet for the second drone.

**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 drone threading and another one with CV detection.

One defect we have found is with the Java threading and ZeroMQ messages being sent to the drone. The drone will be sent the appropriate message to launch, but the drone would never launch. After some analysis, we found the bug being a race condition with the messaging. A patch was issued (thread.sleep) and the software works as intended.

The second defect we found was with the CV detection algorithm. This is not the first time that the CV detection algorithm has failed. We believed in the last test that the drone height had a play with the lack of detection of people. That theory was partially correct; the first stage of the algorithm worked, to an extent. We only got false positives in the first stage, but in the second stage it did not detect anything.

**Lessons Learned:**

Overall, we have solved our threading problem on location and was able to fly the drone successfully on the second flight. However, our CV algorithm was our pain point again. After the flight test, we ran our algorithm on the images we collected that day. We have found that the slower algorithm, Okutama-Action Pedestrian, works the best on its own. For a future flight test, we plan on running it on its own to see its effectiveness. On the data set we have collected today, it is the best one. Our theory on why the two stages worked better on the other data sets is because the previous data sets had snow on the ground, leading to high contrast in the images.