Lab 1 - 10.10.2017

UAS Based Farm Monitoring

Shean Lin / slin63 / 664451629

ABE 498 Fall 2017

Q1: Differentiate between a fixed wing and rotorcraft UAS indicating their weaknesses and strengths. What are some commercially available examples? What applications can you think of these different UAS. (10 points)

Fixed wing UAS have the advantage of being aerodynamically easier to predict and control, not requiring constant power to stay airborne, leading to greater endurances than rotorcraft UAVs. They are also capable of maintaining higher average speeds and can cover much larger distances over a given time span than a fixed rotorcraft.

Fixed wing UAS are highly limited by their takeoff/landing procedures. Larger UAS require some type of suitable surface to land or takeoff from and these requirements only increase as the size of the UAS increases.

Rotorcraft UAS do not have this issue. Their ability to land and takeoff vertically makes deployment in a wide variety of terrain environments feasible and quick. However, their maximum average airspeed is limited by their mechanical complexity and aerodynamic properties. Their endurance is also limited by these same factors.

Fixed wing craft are highly suitable for long distance and relatively obstruction free applications such as aerial surveys. Rotorcraft shine in tight environments. They can be highly useful for close to the ground inspections and situations where the UAS would do best standing still.

An example of a rotorcraft UAS:

Above: The Threod KX-4 LE Multirotor
(http://www.threod.com/products/kx-4-le/description)

This rotorcraft is advertised for its ability to lift up to 6.1 kg. Its endurance while fully load is estimated at around 25 minutes. Its average top air speed sits at 12 m/s.

An example of a rotorcraft UAS:

Above: The Threod EOS mini-UAS
(http://www.threod.com/products/kx-4-le/description)

This fixed-wing UAS is advertised for its 2+ hour flight time, good aerodynamics, and high compatibility with aerial mapping hardware.
Its advertised airspeed is 50-100 km/h with wind penetration 14m/s at 25m/s speed. Its maximum payload is only 1.5 kg.

Q2: Draw a trade-off diagram indicating the choice of the UAS based on mission endurance, mission range, payload capacity, and mission altitude (10 points)

Requirement	Fixed Wing	Rotorcraft
Endurance	Great	Bad
Range	Great	Bad
Payload Capacity	Bad	Great
Altitude	Great	Bad

Q3: What is FAA Part 107? Why is knowing this important? What is a remote pilot in command? (10 points)

FAA Part 107 is an FAA document that outlines a set of limitations, requirements, and responsibilities for UAS operation and ownership. Not knowing FAA Part 107 makes it very easy for an operator to accidentally conduct illegal UAS activity, such as flying in class B, C, D, or E airspace without ATC permission.

The term *remote pilot in command* refers to the operator of the UAS.

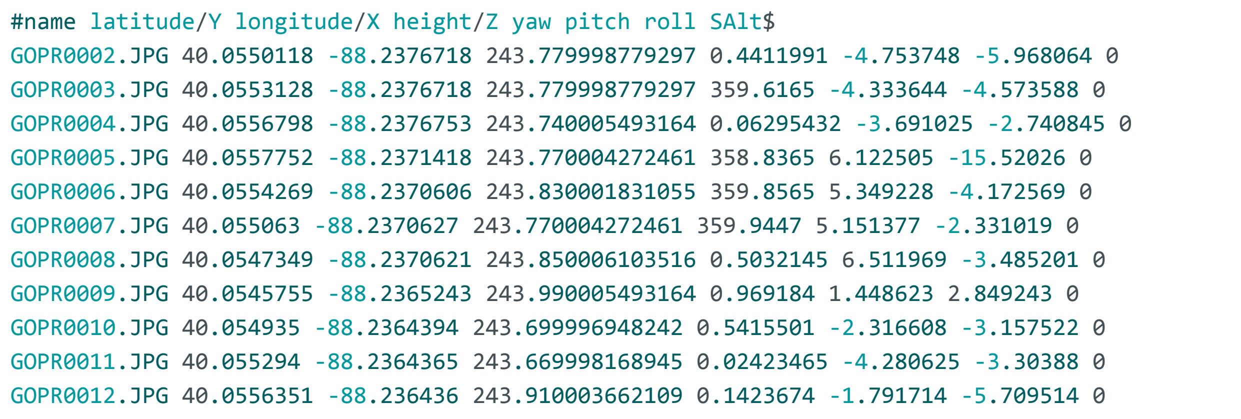
Q4: What are waypoints? What apps are available for programming waypoint missions on the DJI drone? How does one program waypoint missions? (20 points)

Waypoints are single GPS point coordinates. Waypoints may be linked together to form a path that can be traversed by a UAS.

DJI drones can be programmed to follow waypoints by using the *Waypoint Mission* setting on the proprietary *DJI Go* app. Users simply select points on a map in the order they would like the UAS to go. Once point selection is complete, the UAS is released and moves to complete the mission.

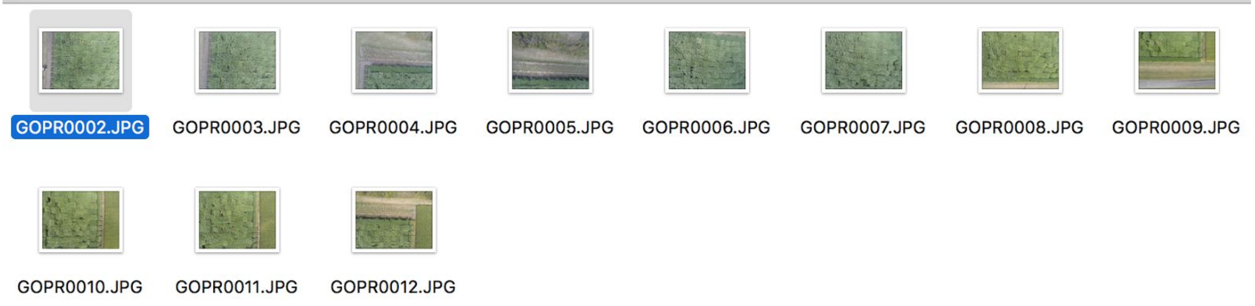
Q5: Participate in the execution of mission and obtain data (20 points)

The mission was executed and the data was uploaded to Box. The two main useful data points are the *.tsv* file with file names and their corresponding GPS coordinates (figure 1) and the actual undistorted photos (figure 2).



```
#name latitude/Y longitude/X height/Z yaw pitch roll SAlt$
GOPR0002.JPG 40.0550118 -88.2376718 243.779998779297 0.4411991 -4.753748 -5.968064 0
GOPR0003.JPG 40.0553128 -88.2376718 243.779998779297 359.6165 -4.333644 -4.573588 0
GOPR0004.JPG 40.0556798 -88.2376753 243.740005493164 0.06295432 -3.691025 -2.740845 0
GOPR0005.JPG 40.0557752 -88.2371418 243.770004272461 358.8365 6.122505 -15.52026 0
GOPR0006.JPG 40.0554269 -88.2370606 243.830001831055 359.8565 5.349228 -4.172569 0
GOPR0007.JPG 40.055063 -88.2370627 243.770004272461 359.9447 5.151377 -2.331019 0
GOPR0008.JPG 40.0547349 -88.2370621 243.850006103516 0.5032145 6.511969 -3.485201 0
GOPR0009.JPG 40.0545755 -88.2365243 243.990005493164 0.969184 1.448623 2.849243 0
GOPR0010.JPG 40.054935 -88.2364394 243.699996948242 0.5415501 -2.316608 -3.157522 0
GOPR0011.JPG 40.055294 -88.2364365 243.669998168945 0.02423465 -4.280625 -3.30388 0
GOPR0012.JPG 40.0556351 -88.236436 243.910003662109 0.1423674 -1.791714 -5.709514 0
```

(Figure 1): Images and their corresponding GPS coordinates.



(Figure 2): Actual image files.

Q6: Write a report with processed data presenting any field anomalies you found using the data. The report should present a tiled or a stitched image of the field and indicate anomalies on the field that you found. What are some of the challenges in dealing with the data that you faced? (30 points)

The images were tiled using an algorithm implemented in Python 3.5.

Images are converted into “nodes” containing GPS information as well as links to nodes geographically above, below, and to the right of the current node (figure 3).

The algorithm picks a root node at an extreme location and recursively finds all adjacent nodes from that first root (traversing up, down, then right). The final result is a collection of doubly-linked nodes whose linkages represent the actual geographical shapes of their corresponding image files.

A more detailed pseudocode explanation of the algorithm is presented in figure 4 and the complete codebase is available in the appendix as well as on Github: *(https://github.com/slin63/TileProject).*

```
# Member node of linked list of image tiles
class TileNode(object):
    def __init__(self, file_name, lat, longi, height, below_node,
above_node, right_node):
        self._file_name = file_name
        self._coord = LatLong(lat=lat, longi=longi)
        self._height = height

        self._below = below_node
        self._above = above_node
        self._right = right_node
```

(Figure 3): Data object representing the doubly-linked nodes containing image and GPS information

```
# Convert list of GPS coordinates (figure 1) into list of TileNode objects
tile_nodes = parse_coordinates_into_tile_nodes("locations.txt")

# Select the "West-most" (minimum longitude) node as our root node
root_node = tile_nodes.get_west_most_node()

# Define our search space by making a copy of all existing nodes
search_nodes = tile_nodes.copy()

# Begin our search for adjacent nodes with the current node
current_node = root_node

# Initialize list for linked nodes
paired_nodes = []

# Search for adjacent nodes till every node is paired
while (search_nodes is not empty):
    # Remove the current node from the search set
    search_nodes.remove(current_node)

    # "find_nodes_*()" creates linked lists of nodes above and below the "current_node"
    # Newly paired nodes are also removed from search set
    current_node.find_nodes_above(IN SET: search_nodes)
    current_node.find_nodes_below(IN SET: search_nodes)

    # "current_node" is now linked to all nodes above and below it
    # Add it to the list of linked nodes
    paired_nodes.append(current_node)

    # Look up the node to the east of the current node.
    # We will be using this node as the current node for the next iteration
    current_node = find_east_node(current_node)

    # If we managed to find an adjacent node, we continue to the next iteration
    if current_node is not None:
        continue

# openCV image matrix we'll be writing to
imageData = ImageData()

for linked_node in paired_nodes:
    # Convert each linked node to png and add to the image
    imageData.insert(linked_node.to_png())

# Present the final stitched image
imageData.show()
```

(Figure 4): Pseudocode detailing the tiling algorithm.

The main challenges in implementing the algorithm are outlined as follows:

1. How do we determine a meaningful value for the difference between two tiles' latitudes and longitudes; i.e., what magnitude of difference means that a tile is adjacent to another tile?
2. What node should be selected as our root node?

In dealing with challenge (1), several assumptions had to be made.

1. Photos will be taken at semi-predictable intervals
2. Photos will always have the same resolution
3. Photos will always be taken with the camera in the same orientation
    a. That is, the top of the photo will always be the north, the bottom the south, etc.
4. No redundant photos will be taken; i.e., in the approximately same GPS coordinates

These assumptions allowed for the determination of a meaningful value for longitudinal and latitudinal differences between adjacent tiles. The algorithm that was created to determine this value begins by finding the maximum difference between two tiles. This maximum difference is then increased and decreased by some user defined percentage to define the floor and ceiling of the meaningful difference between two tiles. This is to deal with discrepancies involving the consistency of the UAS' navigation as well as the GPS readings of the UAS during flight. Pseudocode describing the algorithm is available in figure 6.

```
def find_meaningful_difference(DIFF_TOLERANCE):
        difference_max = 0

        # Iterate through the nodes and find the maximum difference
        for current_index, current_node in enumerate(self._nodes):
            next_node = self._nodes[current_index + 1]
            difference = |current_node.longitude() - next_node.longitude()|

            if difference > difference_max:
                difference_max = difference

        # Define the ceiling and floor for the meaningful difference value
        diff_ceiling = difference_max + (difference_max * DIFF_TOLERANCE)
        diff_floor = difference_max - (difference_max * DIFF_TOLERANCE)

        return Range(diff_floor, diff_ceiling)
```

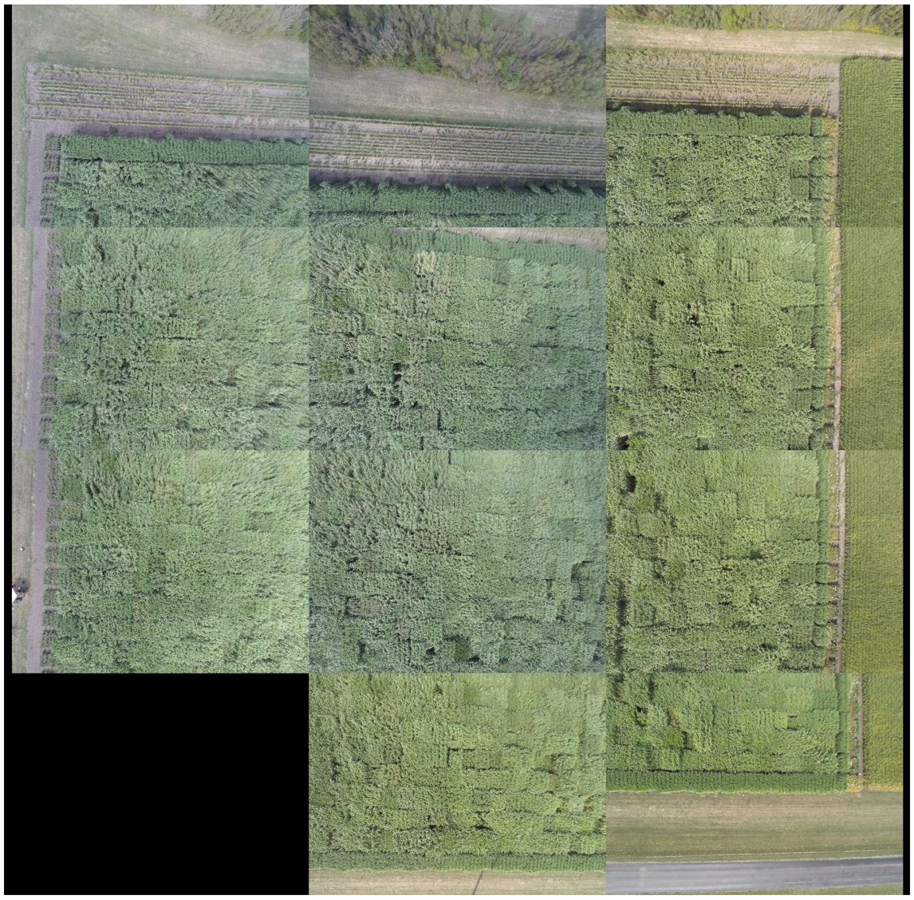
(Figure 6): Pseudocode detailing the meaningful difference algorithm.

Challenge (2) was interesting due to the conflicting mathematical ideas and real world data. The initial assumption was that the node with the maximum latitude and minimum longitude would represent the North-West most node. In this instance, that would mean that the node with the maximum latitude would also have the minimum longitude.

From there, that North-West most node would be used as the root node. The node-traversal would then move down, then right, until all nodes were linked. However, due to GPS drift and modest imprecisions in automated UAS navigation this was not the actual case. The nodes with maximum latitude and minimum longitude were two different nodes.

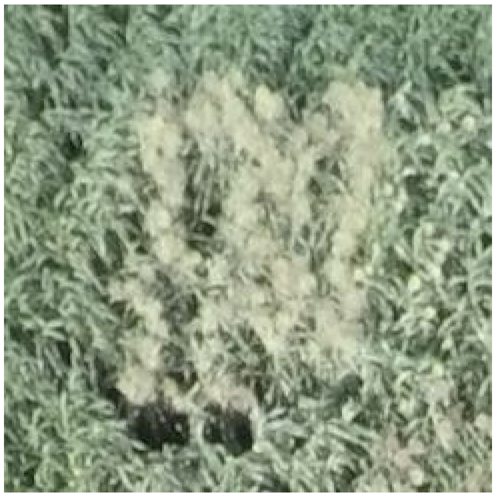
The final approach was to simply select the West most node as the root node and use doubly linked nodes to traverse down, then up, then right, until all nodes were linked.

After all these difficulties were accounted for and the implementation was completed, a tiled image of the surveyed field was produced (shown below in figure 5).

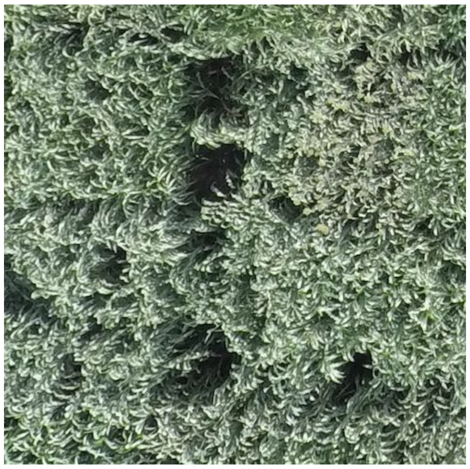


(Figure 7): The final tiled image from the algorithm described in figure 4.

One of the phenomena visible in the final tiled image is the inconsistency in coloration of the field. This is likely a result of changing cloud cover rather than discoloration in some parts of the field, as evidenced by the equally discolored surrounding terrain.


(Figure 8): Sorghum plot displaying early blooming

Visible in figure 8 (and figure 7 at the middle, second to top-most tile) is a patch of sorghum which seems to have bloomed early. Also visible from this angle are various areas of the field that have experienced severe lodging. An example of this is shown in figure 9.


(Figure 9): Sorghum plot displaying signs of lodging

Bonus (20 points): What is stitching and orthorectification? Why and when are they needed? Stich and orthorectify images using software that you can obtain from the web.

Stitching is the process of combining multiple overlapping images to create one seamless, complete image. Stitching is particularly useful in situations like aerial surveying where capturing an entire area of interest in one photo would not be possible.
Orthorectification is the removal of artificial image distortions created by either photographer tilt or terrain irregularities. The result is an image where features are of constant scale and rest in their “true” positions.