Using UAV Mounted LiDAR to Estimate Plant Height and Growth

Harnaik S. Dhami

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Pratap Tokekar, Chair
Ryan K. Williams
Song Li

August 13th, 2019
Blacksburg, Virginia

Keywords: LiDAR, Plant Height Estimation, Plot Detection, K-means, Precision Agriculture

Using UAV Mounted LiDAR to Estimate Plant Height and Growth

Harnaik S. Dhami

(ABSTRACT)

In this thesis, we develop algorithms to estimate crop heights as well as to detect plots in farms. Plant height estimation is needed in precision agriculture to monitor plant health and growth cycles. We use a 3D LiDAR mounted on an Unmanned Aerial Vehicle (UAV) and use the LiDAR data for height and plot estimation. We present a general methodology for extracting plant heights from 3D LiDAR with two specific variants for the two environments: row-crops and pasture. The main algorithm is based on ground plane estimation from 3D LiDAR scans, which is then used to determine the height of plants in the scans. For row crops, the plot detection uses a K-means clustering algorithm to find the bounding boxes of these clusters, and a voting scheme to determine the best-fit width, height, and orientation of the clusters/plots. This best-fit box is then used to create a grid over the LiDAR data and the plots are extracted. For pasture, relative heights are estimated using data collected weekly. Both algorithms we evaluated using data collected from actual farms and pasture. The accuracy in plot height estimation was +/- 5.36 % and that for growth estimates was +/- 7.91 %.

This material is based upon work supported in part by NIFA grant 2018-67007-28380.

Using UAV Mounted LiDAR to Extract Plant Growth Estimations

Harnaik S. Dhami

(GENERAL AUDIENCE ABSTRACT)

Plant height estimation and measurement is a vital task when it comes to farming. Knowing these characteristics help determine whether the plants are growing healthy and when to harvest them. On similar lines, accurate estimates of the plant heights can be used to prevent overgrazing and undergrazing of pastures. However, as farm and plot size increases, getting consistent and accurate measurements becomes a more time-consuming and manually intensive task. Using robots can help solve this problem because they can be used to estimate the height. With sensors that are already available, such as the 3D LiDAR that we use, we can use aerial robots to fly over the farm and collect plant data. This data can then be processed to estimate the plant height, eliminating the need to go out and manually measure every single plant. This thesis discusses a methodology of doing exactly this, as well as detecting plots within a farm. The algorithms are evaluated using data collected from actual farms and pasture.

Dedication

To Parminder Dhami and Gagandeep Dhami, my mom and dad.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Agriculture has been a vital part of the well-being and progression of society. Farmers around the world today continue to grow crops for food, feed, and fiber. These crops can be broken down into 6 main categories; food, feed, fiber, oil, ornamental, and industrial [31]. They are extremely important in many different industries throughout the world, whether it be feeding livestock or people as well. To help optimize the growth of these crops, there is an entire industry dedicated to precision agriculture. The purpose of this is to find and use newer technologies to help optimize the growth and harvesting of crops [29]. This will get more and more important as the population grows because their will be a higher demand on crops. Because of this, it is necessary to optimize our current farms for both growth and health [5, 10, 12].

Currently, many tasks performed by farmers all around the world are extremely manually and labour intensive. Tools and machinery are constantly being developed to help augment or even replace some of these tasks. Some of the tasks performed are related to the crops and plants that they are growing. Monitoring plant health is extremely vital when growing crops so that they are ready when it comes time to harvest. One of the most important traits to monitor during a plant's growth cycle is its height. Recording the plant growth allows farmers to monitor and predict many vital features of crops such as time to reproduction and the amount of seeds it will produce [23, 24]. Also, estimating plant growth can help with high-throughput phenotyping [4]. Plant height is also important for plant breeding [4].

Another area of motivation is in the field of precision grazing. Overgrazing pastures can lead to damage of the plants so it is necessary to monitor and move livestock around the pasture. Doing so, will help prevent some of the damage that can occur and thus result in a positive benefit [6, 26, 36]. Working on improving our current methods of grazing will help improve yield rates and future growth as well.

Robotics provides a solution to help overcome some of the more manually intensive activities.

For example, with manual height measurement, a person has to go to each plant or plot and manually record the height. As the plot and farm size increases, this can take a tremendous amount of time. Because of this, fewer measurements are taken and they are extrapolated for the entire plot and farm. With robots, we could measure most, if not all, of the plants quickly. Some of the newer technologies that are being developed in precision agriculture use unmanned aerial vehicles (UAV's) to help collect various data. In this thesis, we discuss a method of collecting and estimating crop height data using a 3D LiDAR mounted on a UAV. As we developed our work, we realized a method to detect crop plots within our LiDAR scans was also needed. This method is also discussed and described in the thesis. In the next chapter, we will discuss some of the work that has already been done regarding plant and crop height measurement using LiDAR's and unmanned aerial vehicles.

Figure 1.1: DJI Matrice 600 Pro Flight Over Soybean Plants at Drone Cage

1.1 Related Work

There has already been some work done with UAV's and LiDAR's to determine crop height estimations. There methods are slightly different than the one discussed in this thesis and they will be highlighted here. A few different hardware setups have been used to estimate crop height. Some of these methods include using a 2D LiDAR mounted on a UAV [3], using 3D LiDAR's that are mounted on fixed-wing [40] and rotor-based UAV's [18, 37], and using a ground robot for navigation between rows of crops [16].

The method described by Anthony Et al. (2014) uses a 2D LiDAR mounted on the botttom of a UAV facing downwards. They used this setup to develop a method of measuring corn

heights. Since they used a 2D LiDAR, the methodology they developed is very different than ones that can be used with 3D LiDAR's due to the amount of incoming data being a lot less. For their method, the UAV would fly over the target crops while the LiDAR was scanning them from overhead. With each incoming scan, they would estimate where the ground and top of the crop was using a distribution of the scan data. The incoming height datapoints within each scan were placed given percentiles using a normal distribution. With their experiments and data, they determined the percentile of the ground was the 95th and for the crop it was 2nd. Using these for their scans, they were able to estimate the crop height by finding the difference in the data points at these percentiles [4].

Madec Et al. (2017) use a similar method to above, except with a 3D LiDAR. Here, however, the 3D LiDAR is ground based whereas the UAV has a RGB camera on it. Measurements between these two sensors are compared. A structure-from-motion algorithm was implemented to extract a 3D dense point cloud from the camera on the UAV. The LiDAR was driven along the ground over the crops and the point cloud scans were collected. Once both point clouds were available, a similar distribution analysis to Anthony Et al. (2014) was used. After, the two heights were compared with one another the authors determined that there was a strong correlation between the estimated heights [18].

Yuan Et al. (2018) used the same LiDAR that we mounted on our DJI M600 Pro. However, they had their's mounted on a ground vehicle. This vehicle was driven along the plots of the crops and, along with ultrasonic sensors, data was recorded. To determine the height of the crops using the LiDAR, they relied on finding the ground in the areas between plots. The distance of the LiDAR off the ground was manually recorded and this distance was used to estimate the heights within the scans. These ground points would then be used with the other points to find the height [37]. Their methodology was the closet to ours, but our ground estimation was the main difference.

Another methodology used was implementing a fixed wing UAV as opposed to a rotor-based. This was done by Ziliani Et al. (2018) and is vastly different to the other methods described above. The fixed wing UAV flew over the target maze crops with a Sony camera mounted facing downwards below it. It would capture image data as it passed. Throughout the target environment, there were ground control points to help calibrate the image data and used to estimate the height from the images. Also, to verify and validate the estimations, a ground-based LiDAR approach was used to collect the height data. There was a correlation of up to 0.99 between the LiDAR and structure-from-motion based approach for the RGB images. However, during the flowering of the crops there was an increased variability and the correlation was only 0.65 [40].

There has also been some work done in regards to row detection by Li Et al. (2015). A UAV flew overhead a field of corn and captured RGB images from a camera. These RGB images were then stitched together to create a singular image. Using this image, the rows between the lines of corn were detected using computer vision techniques [17].

1.2 Problem Formulation

In this section, we define the problems that we are looking to solve with our methodology. There are 2 main problems that this thesis focuses on, static height estimation and growth overtime analysis. They are discussed in detail below.

Problem 1 (Static Height Estimation)**.** *Given a set of target plots/crops, use the UAV and LiDAR to fly overhead and estimate the height of each of the targets.*

When mentioning plots in this context, it should be noted that we mean group of crops/plants within a farm. Plots of wheat can be seen in Figure 1.2. These were grown at Kentland Farm and used for some of our experiments discussed in later chapters.

The main purpose of solving this problem is to help replace the manually intensive task of height measurement. Especially for larger farms, using a UAV also allows for the collection of more data since every plant will be captured by the LiDAR. This will also let farmers and agronomists focus on more specialized tasks such as phenotyping. Phenotyping can also be done at a higher precision because of the access to more precise data. There are a few assumptions and expected inputs to this problem. They are listed below:

- Structurally aligned plot in a regular ordered grid
- Number of plots are known
- UAV flight path and height pre-determined

Problem 2 (Growth Overtime Analysis)**.** *Given a target grazing region, estimate the growth of the region overtime using the UAV and LiDAR.*

The grazing region is defined as the target field where the estimated growth is needed. It is also referred to as the pasture in some parts of this thesis.

The purpose of this algorithm was to estimate the growth of a grazing region overtime. The need to estimate this growth arises from the precision grazing motivation discussed in Chapter 1. To prevent overgrazing and undergrazing, the grazing region needs to be monitored precisely. For our implementation in solving this problem, there are a few assumptions. Those are listed below:

- Grazing region is not maintained or cut in any way
- Perimeter around the grazing region is mowed on a weekly basis
- Pre-determined flights over grazing region on a weekly basis

Figure 1.2: Wheat Plots

Chapter 2

System Description

In this chapter, the system that was used to collect and process the data is described. First, the hardware setup is discussed, followed by the software suite.

2.1 Hardware Setup

Figure 2.1: DJI Matrice 600 Pro With All Hardware Mounted

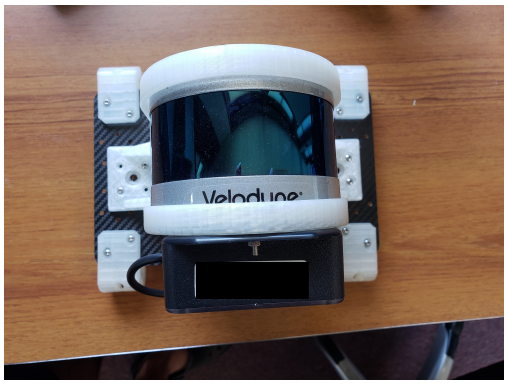
For our implementation, we used a UAV with various sensors and hardware mounted on it (Figure 2.1). The platform that was decided on was the DJI Matrice 600 Pro (Figure 2.2) [19]. There were a few reasons we decided to use this specific platform. Primarily, it has a max takeoff weight of 15.5 kg, which allows for more sensors to be used and mounted. Along with the weight, the platform has a maximum speed of 40 mph if there is no wind and a hovering time of 16 minutes with a 6 kg payload [20]. This allows for some variability and flexibility for the types of applications that the platform can be used in. It specifically works well with our application discussed in this paper.

Figure 2.2: DJI Matrice 600 Pro Hexacopter

After selecting the UAV platform, the next step was to choose an on-board computer that would interface with the system and sensors in place. We believed the NVIDIA Jetson TX2 (Figure 2.3) was the best choice. The TX2 is mounted on NVIDIA's developer kit, which has a USB 3.0 and Ethernet port that are used in our setup [14]. The computer is also equipped with a quad-core ARM processor running at 2 GHz, a 256-core NVIDIA GPU, 8 GB of LPDDR4 memory, and 32 GB flash storage [9]. The developer board also has an expansion SD card slot is used to increase the system's storage.

Figure 2.3: NVIDIA Jetson TX2 with Developer Board

A couple of sensors are also mounted on the UAV frame. The two primary sensors that were needed were a LiDAR and a stereo camera. For the LiDAR, we chose the Velodyne VLP-16 as one of our sensors [35]. The VLP-16 is a 3D LiDAR consisting of 16 channels that can refresh at a rate of 5 - 20 Hz. It also has a range of 100 meter and an accuracy of +/- 3 cm [34]. Since our UAV has a limited battery life when in air, we decided to go with a 3D LiDAR, as opposed to a 2D one, so that we could get a lot more data quickly. It should be noted that 3D LiDAR's are vastly more expensive than 2D one's. To mount the LiDAR to the DJI frame, a few more pieces were needed. Firstly, the DJI Matrice 600 Expansion Bay



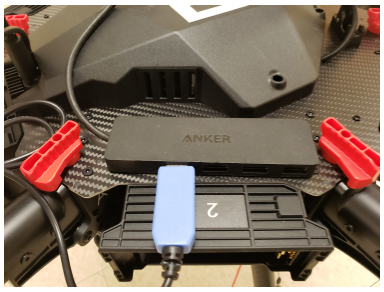
(a) Velodyne VLP-16 LiDAR in 3D Printed Frame, Mounted on DJI Matrice 600 Expansion Bay

(b) Zed Stereo Camera

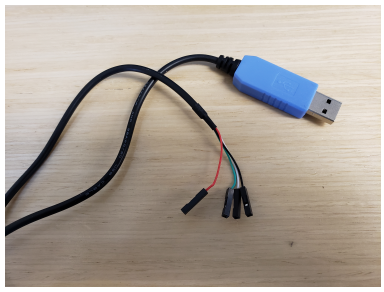
Figure 2.4: Sensors Used on DJI M600 Pro

Kit [21] was ordered. This kit contains a base plate with various mounting points that can be attached to the bottom of the M600 frame. Next, a 3D frame and brackets were printed by the Virginia Tech Laboratory for Coordination at Scale. The Velodyne LiDAR was mounted in the 3D printed frame, which was then attached to the base plate (Figure 2.4a). For the stereo camera, the ZED Stereo Camera [22] was chosen. The ZED (FIGURE 2.4b) is a cost effective camera that works seamlessly with our other hardware. It can record at various resolutions (up to 2.2k @ 15 frames per second), which gives us some more flexibility.

(a) TP-Link TL-WR802N Router



(b) Anker 4-Port USB 3.0 Hub



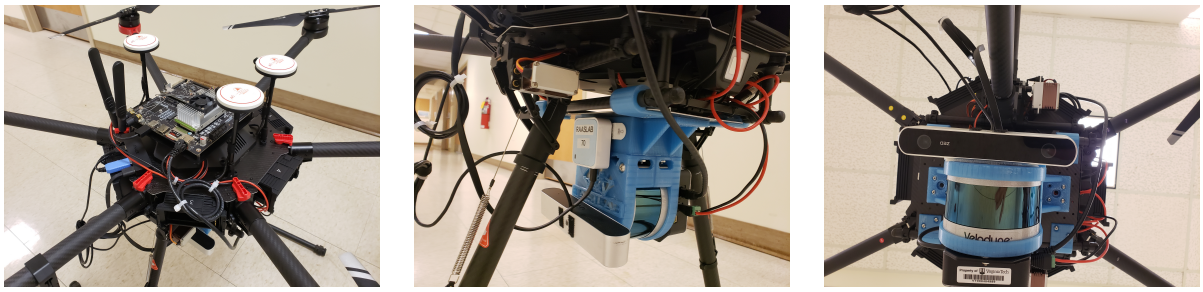
(c) Adafruit USB-to-TTL Serial Cable

Figure 2.5: Miscellaneous Hardware

Some other miscellaneous hardware was needed as well. To make interfacing with the on-board computer easier during flight, a wireless router was necessary. The TP-Link TL-WR802N (Figure 2.5a) is a small and USB powered router that fulfilled this need [1]. Due to the NVIDIA development board only having a single USB 3.0 port, an expansion hub

was needed. The Anker 4-Port USB 3.0 Hub (Figure 2.5b) was selected because it was not costly and increased the number of USB 3.0 slots to 4 [2]. Also, to interface with the flight controller of the DJI M600 Pro platform, a USB-to-TTL serial cable was required. The cable connects a USB port on the TX2 to the UART port of the flight controller on the M600 Pro. This is necessary to access the sensor information of the UAV platform, such as the GPS, local position, and imu, as well send flight commands to conduct autonomous real-time flight. DJI [13] recommended the Adafruit USB-to-TTL Serial Cable [33], so that cable was selected.

Figure 2.1 shows all the hardware mounted on the DJI M600 Pro platform. As stated previously, the Velodyne VLP-16 was mounted into a custom 3D printed frame which was then attached to the base plate. This base plate was attached to the bottom of the DJI M600 Pro using 3D printed brackets. All the other hardware was attached using double sided tape. The tape was used to allow for easy mounting and removing of the hardware. The ZED stereo camera was attached on the 3D frame of the LiDAR pointing downwards (Figure 2.6c). The TP-Link router was placed slightly above the ZED, as seen in Figure 2.6b. Due to the size of the developer kit, the TX2 was mounted on top of the M600 Pro (Figure 2.6a).



(a) NVIDIA Jetson Mounted on DJI M600 Pro

(b) Sideways View of TP-LINK Router, Zed Camera, and Velodyne LiDAR Mounted on DJI M600 Pro

(c) Underneath View of TP-LINK Router, Zed Camera, and Velodyne LiDAR Mounted on DJI M600 Pro

Figure 2.6: All Hardware Mounted on DJI M600 Pro

2.2 Software Environment

In this section, we will go over the software environment that was implemented for this system. As stated previously, our on-board computer of choice was the NVIDIA Jetson TX2. NVIDIA provides a software solution to load the TX2 with an OS and an extensive software suite. To start, we installed NVIDIA Jetpack 3.2.1 [15] on the TX2. This installs Ubuntu 16.04 [32] on the computer, as well as NVIDIA's CUDA 9.0 [25] and OpenCV [27]. These packages are necessary to run the algorithm's described in the following chapters.

The next step was to install Robot Operating System (ROS) onto the computer. This was chosen because it makes interfacing between the computer and sensors seamless. ROS Kinetic [30] was installed because it is compatible with Ubuntu 16.04. After setting up the ROS environment, a few ROS packages were needed. For the Velodyne VLP-16 LiDAR, the velodyne [11] driver was installed. This driver allows for the processing of the 3D point cloud generated by the LiDAR. To properly interface with the LiDAR, the TX2's network settings needed to be configured. A static IP was required for the Ethernet port and connected to the computer as described in the previous section. Next, ZED SDK 2.7 [39] was installed. This package is required for the ZED stereo camera to work properly with the TX2. Once that was setup, the zed-ros-wrapper [38] driver was installed to allow interfacing of the camera through ROS, similar to how the LiDAR connects. Lastly, the DJI Onboard SDK [7] was needed to connect to the DJI Matrice 600 Pro and it's flight controller. Also, the DJI SDK [8] ROS driver was installed to provide a ROS-based solution to interfacing with the UAV.

The RQT graph of the ROS environment during data collection is shown in Figure 2.7. During intial testing, it was found that recording the ZED topics would quickly fill up the buffer when saving the data to ROS bags. To get around this, the ZED SDK was used to record data into .svo files. These .svo files could be used to create the ZED topics offline. More details about the processing of the data are discussed in the algorithm chapters following this one. Due to using the ZED SDK, there are no ZED nodes shown in the RQT graph (Figure 2.7). However, the graph does show the topics generated by the DJI SDK and the velodyne driver. For the velodyne, the /velodyne_points topic provides the 3D point cloud generated by the VLP-16 LiDAR. Some of the important DJI topics include /dji_sdk/gps_position, which provides the GPS information of the UAV, /dji_sdk/local_position, which provides the local position of the UAV relative to the starting location of the UAV, and /dji_sdk/attitude, which provides the UAV principile axes information (roll, pitch, and yaw). All of these topics were necessary to develop the algorithms that are discussed in the following chapters.

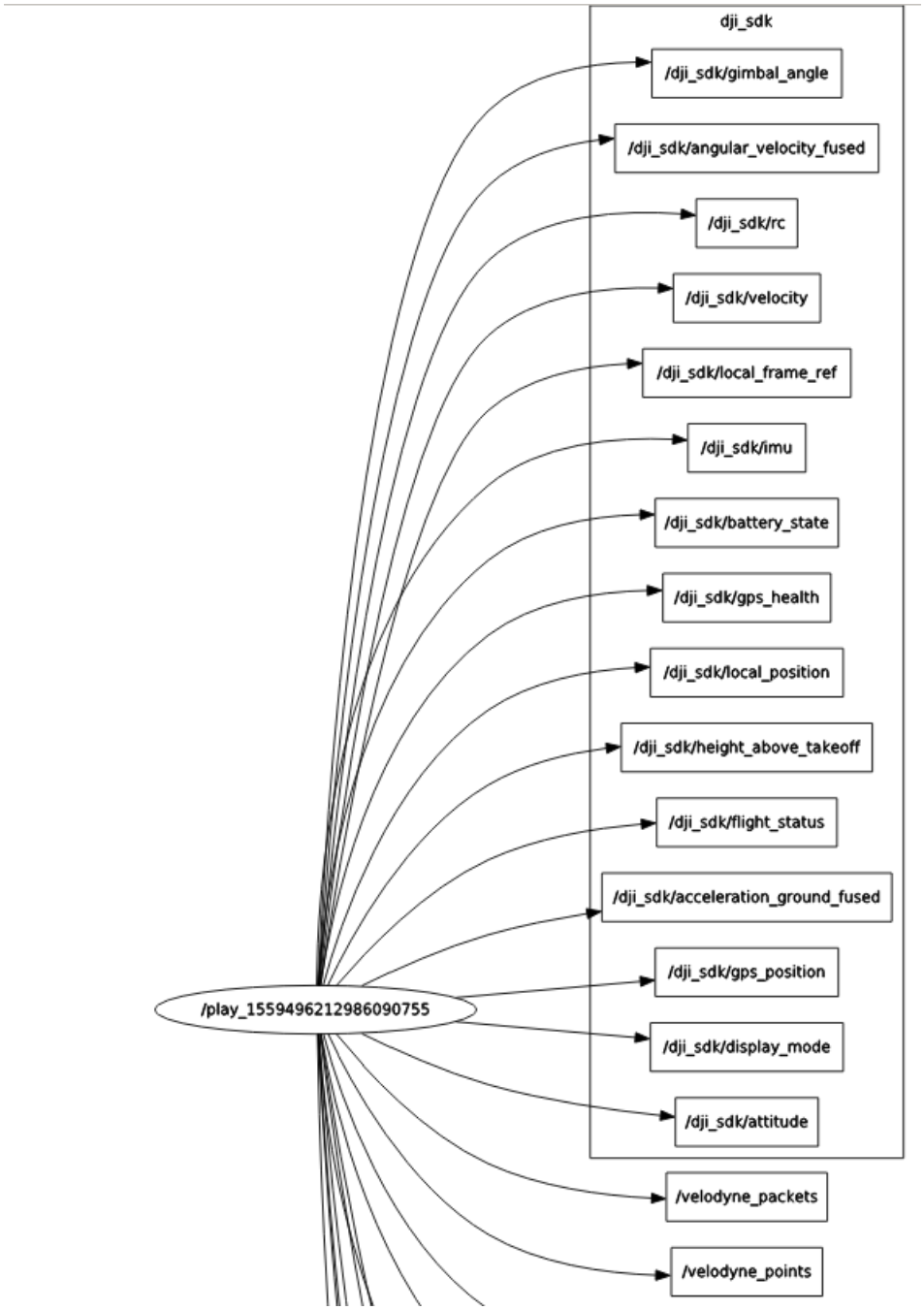


Figure 2.7: RQT Graph for the UAV

Chapter 3

Algorithm Descriptions

In this chapter, the algorithms used to solve the problems highlighted in the previous chapter are described. Figure 3.1 shows the system block diagram for the algorithm pipeline. Each of the blocks are discussed step-by-step below.

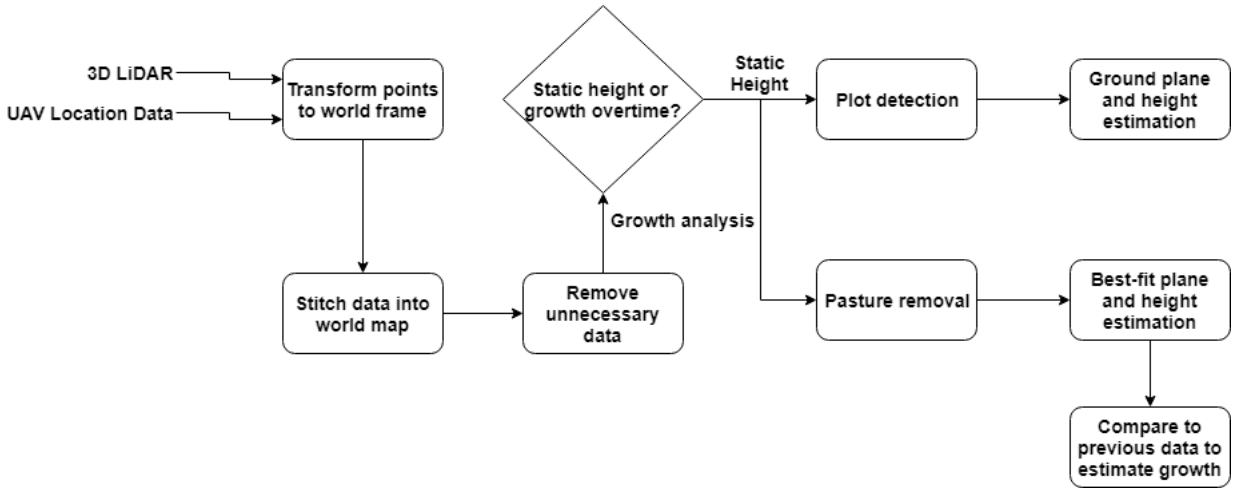


Figure 3.1: System Block Diagram

3.1 Point Cloud Processing

3.1.1 Coordinate Frames

Before any of the plant data can be extracted and worked with, the raw point cloud data needs to be processed. With the Velodyne VLP-16 LiDAR, 3D point-cloud data comes in at a rate of 5-10 Hz. Without any processing, all of the data is relative to the LiDAR being

the origin of the "world." This is perfectly fine if we only care about a single scan from the LiDAR. However, if we are looking at successive scans, each scan needs to be framed relative to one another. This is where the point cloud processing work comes into place. Because the LiDAR, is mounted facing downwards on the bottom of the DJI M600 Pro, we can use its IMU and GPS sensors to help build these reference frames.

This was done using the built-in tools provided by the ROS packages described in the previous chapter. The DJI ros package did not broadcast the hexacopter's coordinate frame relative to the start point, so this was the first step that needed to be accomplished. The DJI ros package did, however, broadcast the position of the hex relative to the start point as well as it's attitude in the form of a quaternion. Using these two data points, the coordinate frame of the hex could be built and broadcast. This coordinate frame gave a reference of the hex to where it started and helped track the motion of it through the "world" coordinate frame. Next, we had to properly transform the coordinate frame of the Velodyne LiDAR to the hex because the sensor's coordinate frame was different than the UAV's. This was done by setting the roll, pitch, and yaw of the LiDAR relative to the DJI. The roll was set to 0, pitch to 90 degrees, and yaw to 90 degrees. This would differ depending on how the LiDAR is mounted on the UAV. After all this, there were 3 total coordinate frames. The top-most was the world frame. This is the coordinate frame relative to the world. The starting point of the hex was the origin in this reference frame. A child of this coordinate frame was the one for the DJI M600 Pro. This coordinate frame represented the DJI M600 as the origin. Lastly, there was the velodyne coordinate frame. The origin of this was also the DJI and LiDAR, however it transformed the coordinates of the LiDAR to the DJI. The parent/child relations of these coordinate frames can be seen in Figure 3.2. Once the incoming LiDAR data scans could be framed properly, further algorithms were developed. These are described in detail in the next sections.

3.1.2 Map Building

One of the requirements for further algorithms was to build a map by combining all the LiDAR point scans into a single scan. Once the point cloud processing described in the previous chapter was achieved, this was a relatively easy task. Since the coordinate frame translations were working and broadcasting, each incoming scan could be translated to a world coordinate frame to due the relationship shown in Figure 3.2. This translation was done with each incoming scan so that the data would be set in reference to the world origin as opposed to the LiDAR being the origin. Each successive scan was then concatenated to the previous scan. Over an entire dataset, this would result in a single file containing data from every scan translated relative to the movement of the DJI in the world. The built map, with some further processing discussed later, is shown in Figure 3.3.

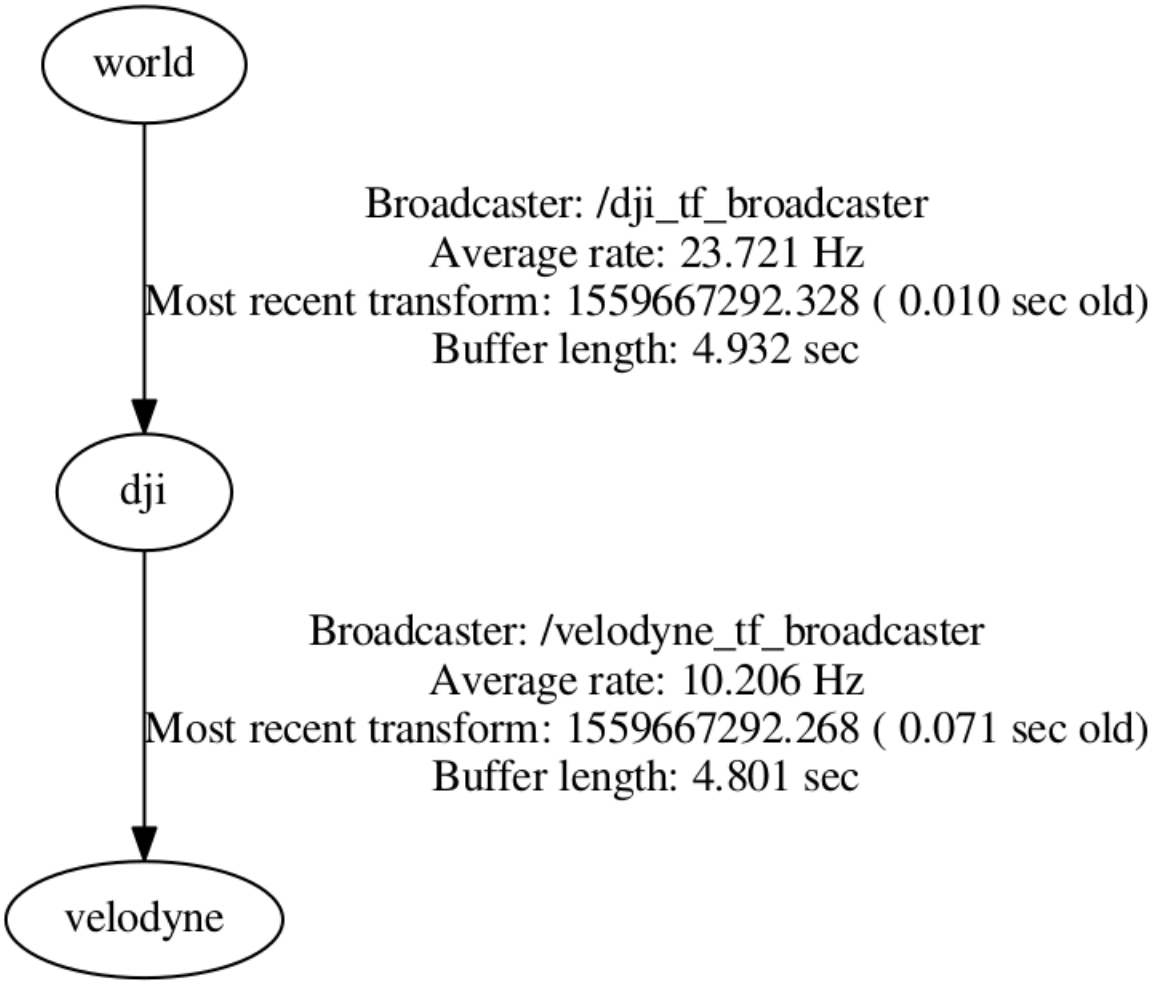


Figure 3.2: Coordinate Frames

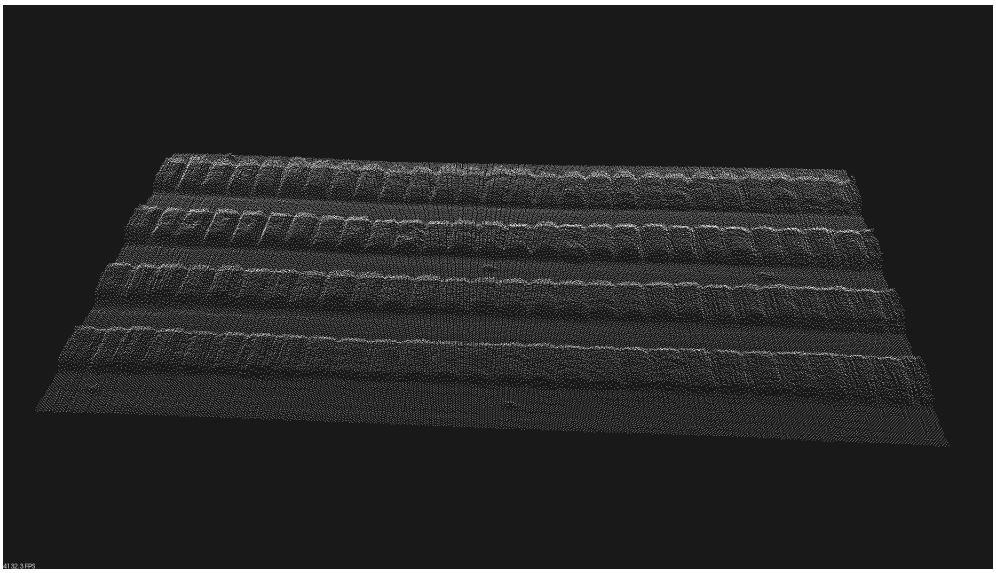


Figure 3.3: Cropped Map of Kentland Farm Wheat Plots

3.1.3 Other Processing Tools

A few other, minor tools were developed to help with the point cloud processing. Those are described a bit in this section very briefly. Due to the concatenation of the LiDAR scans, the dataset would quickly increase in size due to the immense amount of incoming data. Because of that, it was necessary to down-sample the data to help not only reduce the size, but help with some of the noise. This was done using a voxel filter. The voxel filter allowed down-sampling through averaging points in a set size grid. Down-sampling also helped increase the speed at which the data could be processed for other algorithms. Another point cloud processing tool that was developed was a crop box filter. This was used to crop the point cloud data and extract what was needed for processing. Lastly, a centering tool was also developed. This would take in a point cloud file and change the origin of the data to the center point of it. This was done so that the dataset could be visualized better after the cropping. All of these tools were developed using C++ and the Point Cloud Library(PCL). Again, Figure 3.3 shows a dataset with all of this processing applied. Each LiDAR scan has been concatenated into one point cloud file. Then, that file has been voxel filtered, cropped, and recentered.

3.2 Branching Point

In this part of the algorithm, as seen in the block diagram, a branching point is reached. Prior to this, the steps performed were exactly the same whether we were doing our static height estimation or our growth overtime analysis. From this point on, the algorithms differ. The

first algorithm discussed will be the static height estimation starting with the plot detection portion. Afterwards, the growth overtime analysis algorithm will be described.

3.3 Static Height: Plot Detection

Another algorithm that was developed was a plot detection algorithm to determine plant plots within a farm. This was needed to use the ground plane and height estimation algorithm on a plot based basis given a point cloud file of the entire farm. An example dataset for this can be seen in Figure 3.3.

3.3.1 K-means Clustering

To begin, we would start with a concatenated and down-sampled dataset within a single point cloud file. However, since we were trying to find the plots within this dataset, all points below a certain z axis value (height) were removed. This gave us a dataset similar to what is shown in Figure 3.4. The red points shown in this figure are the center points of each cluster found in the dataset. The k-value set for the k-means clustering algorithm is the number of clusters we expect, or for our case, the number of plots we are looking for.

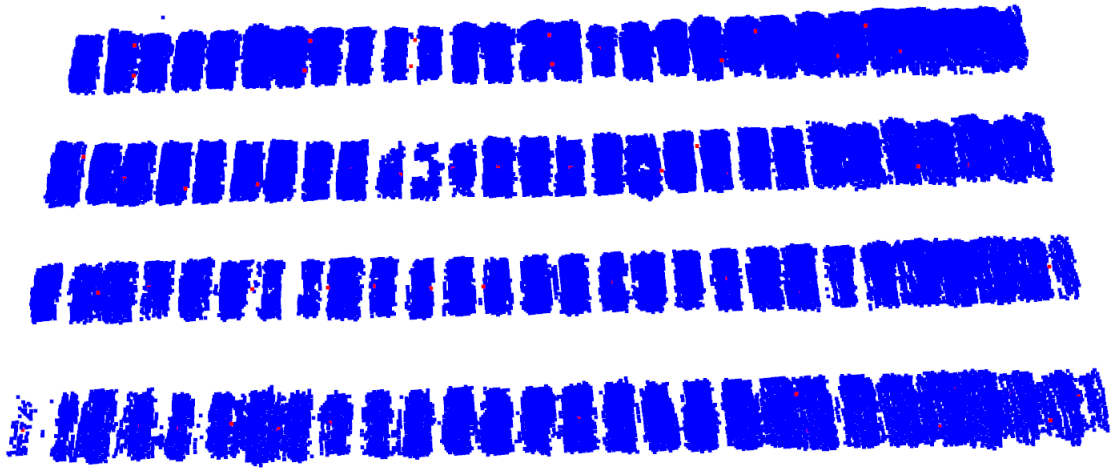


Figure 3.4: K-means Clustering

In Figure 3.5, the minimum oriented bounding box of each cluster is shown. A minimum orientated bounding box is the smallest area box that fits all the points. The red points in the figure are all the points within one of the clusters. As can be seen in this figure, not all plots have been clustered correctly, but a fair amount of them have. Because of this, a

voting scheme needed to be implemented to help determine the plot size using the bounding boxes.

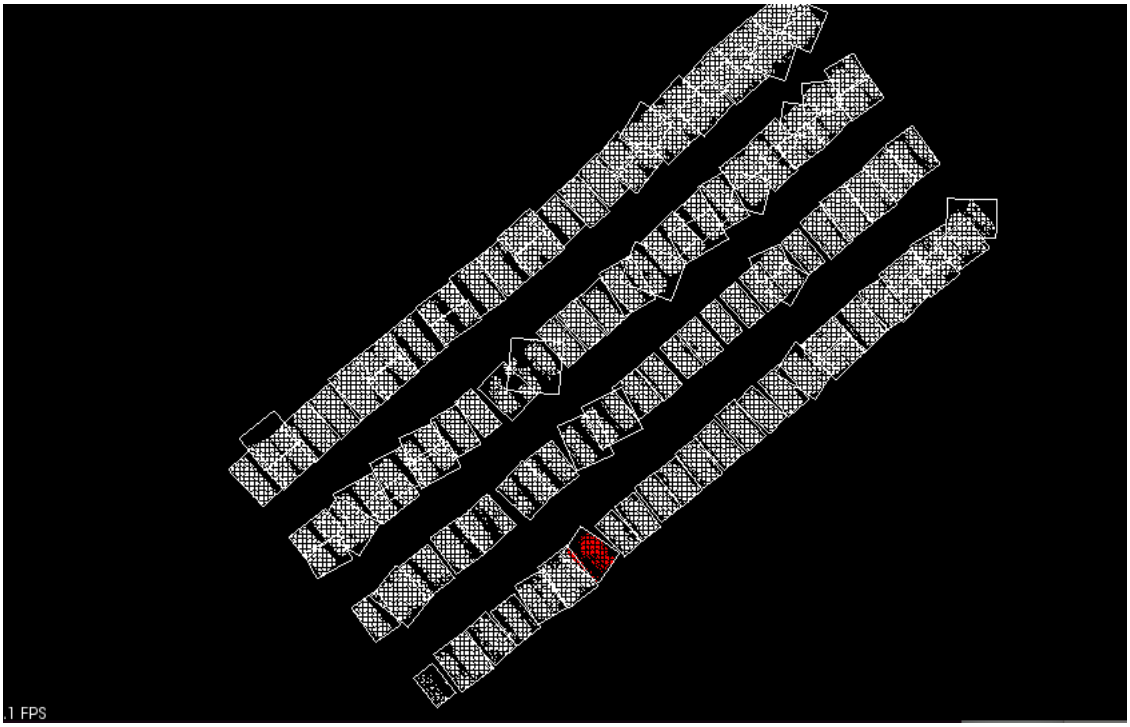


Figure 3.5: Bounding Box of Clusters

3.3.2 Voting Scheme for Correct Clusters

For the voting scheme, we used the bounding box dimensions and orientations of each of the clusters determined in the previous step. These were all broken into separate ranges and the number of clusters that fit into each of the ranges were counted. For example, the orientation data was broken into ranges of 0.3 radians. If the orientation for a bounding box was 0.0 - 0.29 radians, the vote for that range would increase by 1. This was done for each cluster and then after, the range with the most number of votes was used as the orientation that would be set for the plots. A similar method was done for the height and width of the clusters to determine the height and width of the plots.

3.3.3 Static Height: Grid Selection

Now that the orientation, width, and height of the plots were determined, these parameters were used to fill in a grid for the dataset. The vertical and horizontal distances between the plots was manually found to help create the grid. It should be noted that even though

the grid is currently manually overlaid on top of the target dataset, work is being done to automate it. After this grid selection was done, the results looked similar to what is shown in Figure 3.6. These bounding boxes could then be used on the original dataset. The heights of the boxes were extended so that they would cover the ground between plots. All of the points in these boxes were then extracted one box at a time and then fed into the ground plane and height estimation algorithm discussed previously.

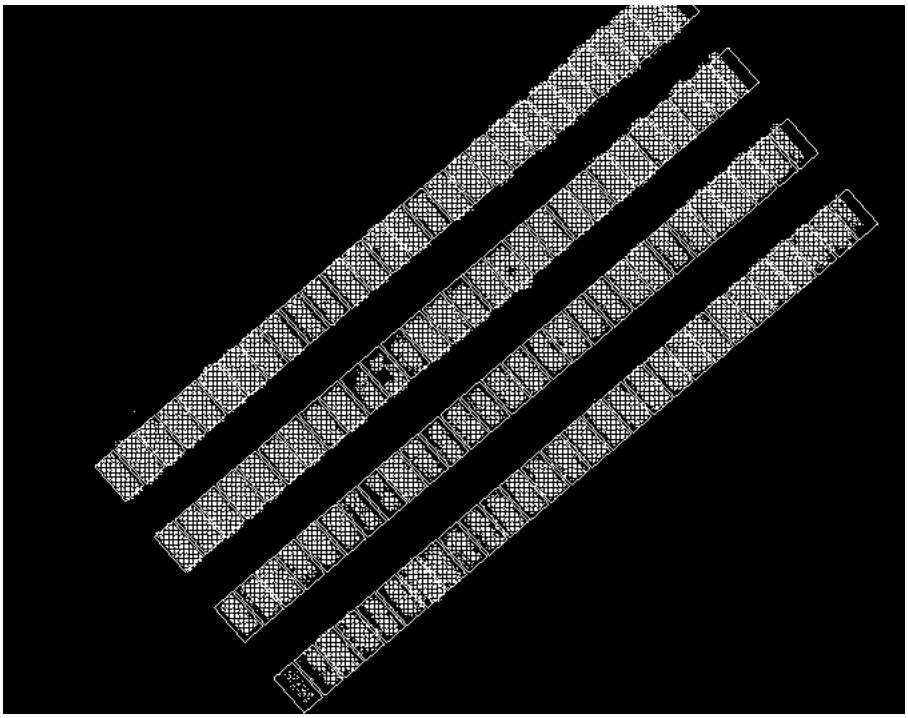


Figure 3.6: Correct Bounding Boxes

3.4 Static Height: Ground Plane and Height Estimation

One of the more extensive algorithms developed during this work was the ground plane and height estimation algorithm. The purpose of this algorithm was to take a point cloud file, find the ground plane in it, and then determine height data for objects that were not part of the ground. The process of this algorithm is described in this section.

3.4.1 Detect Points in Ground Plane

The very first step to accomplish this algorithm was to detect the ground plane in a LiDAR scan. Again, the Point Cloud Library in C++ was used to help with this work. The tutorial described in the PCL documentation was followed for a sample plane model segmentation [28]. This segmentation implementation uses a RANSAC algorithm to find a plane within the given dataset. After certain parameters were optimized, the developed algorithm would give a similar output to what is shown in Figure 3.7. The DJI was flown over a table that was placed on a flat field. The red points indicate what the algorithm output as points in the plane; whereas, the blue points were outliers that were ignored by the algorithm. As seen from the figure, this algorithm could successfully be used to determine the ground plane in LiDAR scans.

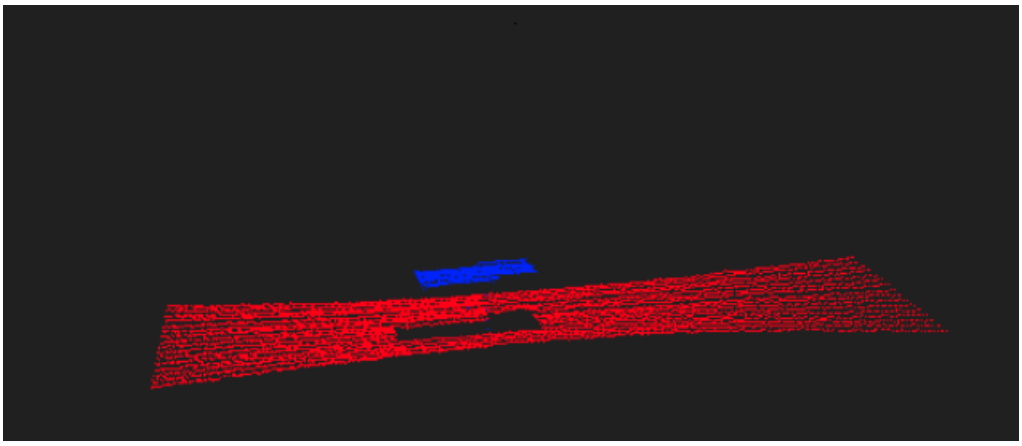


Figure 3.7: Ground Plane Extraction

3.4.2 Best-Fit Plane of Inlier Points

Next, the inlier points that were found from the planar model segmentation discussed in the previous step were used to find the equation that described a best-fit plane. This was successfully done using the linear least squares approximation on the inlier dataset. Each of the "red" points were taken and used for this approximation to output the centroid of the data, as well as the normal vector of the best-fit plane. Now, with a equation describing the best-fit plane, height of the outlier points could be determined.

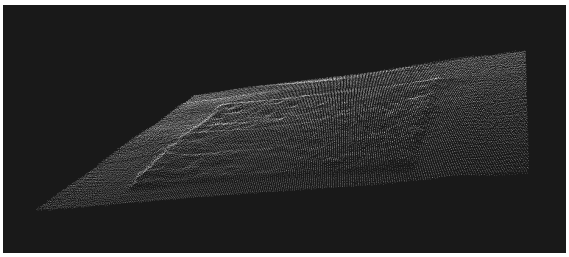
3.4.3 Height Estimation

With a centroid and normal vector for the best-fit plane, the height of each of the outlier points could be determined. This could be done by finding the distance of each outlier point

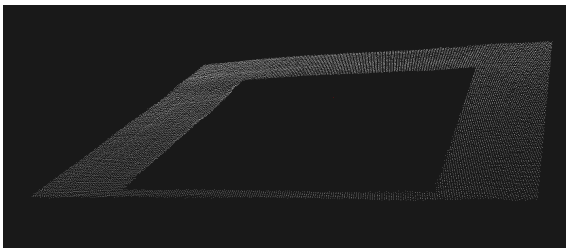
to the best-fit plane. Since the plane described the ground, the distance from the outlier point to the plane would be the height of that point over the ground. Doing this for all of the outlier points would indicate each of their heights relative to the ground plane. For example, in Figure 3.7, the distance of the blue points to the best-fit plane of the red points can be found to determine their height off of the ground.

3.5 Growth Overtime Analysis: Pasture Removal

Given a dataset that was processed as previously discussed, the first step that needed to be done for this algorithm was to remove the pasture/grazing region. The data was processed using the common steps shown in Figure 3.1 to output a processed point cloud of our target grazing region. This is shown in Figure 3.8a. The crop-box tool was used to remove the pasture within this dataset to get only the perimeter of the region; the output of this is shown in Figure 3.8b.



(a) Processed Grazing Region Dataset



(b) Grazing Region Removed

3.6 Growth Overtime Analysis: Best-fit Plane and Height Estimation

Using the perimeter points found and shown in Figure 3.8b, a best-fit ground plane was determined. Similar to the static height estimation algorithm, a linear least squares approximation was used to determine this best-fit plane. All of the perimeter points were used in this approximation. After the centroid and normal vector was output, the distance between all of the pasture points and the best-fit plane was computed. These were used to estimate the height of the pasture points.

3.7 Growth Overtime Analysis: Weekly Comparison

The last step was to compare data weekly to get a growth estimate. All of the previous steps for the growth overtime analysis algorithm were performed on each captured dataset. These datasets would be captured at different times; in our case, this was at weekly intervals. After all the data had been processed and the height of the pasture points was computed, the data was compared with each weekly set to get a growth estimate.

Chapter 4

Experiments and Results

Experiments were conducted in 3 main locations. The Drone Cage at Virginia Tech provided a quick location to test out the hardware as well as new implementations of some of the algorithms. Most of the initial experiments were done here while the system was still being developed. The more important data collection was conducted at Virginia Tech's Kentland Farm. Kentland Farm provided real life examples of plotted crops that we could use to collect data. The experiments that were done at each of these locations are described in detail in this chapter, along with their results.

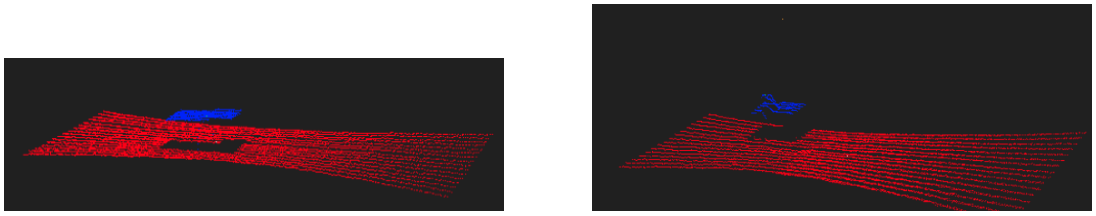
4.1 Drone Cage

4.1.1 Inital Table Experiment

One of the first algorithms that was developed was the ground plane and height estimation algorithm. To properly test whether this algorithm worked, we flew at the Drone Cage on March 4th, 2019. A simple test was done using two tables placed within the drone cage. One of the tables did not have anything on it, whereas the other one had a backpack and other material to give it a different height. The DJI flew directly over both of these tables with the Velodyne LiDAR facing downwards and scanning point cloud data. The scans of the two separate tables were fed into the ground plane and height estimation algorithm to see how accurate it was. The results of this measurement are shown in Table 4.1. The ground plane and height estimation algorithm being run can be seen in Figure 4.1 with both tables. The ground is correctly being found; this is indicated by the red points being the plane model that is found.

	Measured Height	Estimated Height	Percent Difference
Table w/ backpack	44 inches	43.1 inches	-2.07 %
Empty table	29 inches	29.6 inches	+2.05 %

Table 4.1: Drone Cage Table Flight Results



(a) Empty Table (b) Table with Backpack

Figure 4.1: Ground Plane and Height Estimation During Drone Cage Table Flight

4.1.2 Point Density Experiment

During testing and experiments, it was determined that if the UAV flew over the test subject too high, the data points in the LiDAR scan were too sparse. Because of this, a optimal flight height needed to be found so that there were enough data points for the algorithms, but high enough so that the down wash was minimal. A point density experiment was conducted to determine this optimal height. The experiment consisted of the DJI flying over an artificial plant. It would increase its height over the subject while remaining directly on top of it. With the LiDAR collecting the data points, the ground plane detection algorithm was used. For each incoming scan, the ground plane points were determined and removed. Therefore, the remaining points directly underneath the UAV were only a part of the plant. The height of the DJI M600 Pro as well as the number of points on the plant were saved as the 2 datapoints. Figure 4.2 graphs these data points. It can be inferred from this figure that the optimal flying height of the DJI is 3 - 3.5 meters above the target to get the most amount of points. The reason that there aren't that many points when the altitude is really low is because there is a 1 meter minimum range for the LiDAR. Also, as the LiDAR is directly over and close to the plant, the scans do not reach further down to the lower portions of the plant.

4.1.3 Soybean Experiment

After determining that the ground plane and height estimation worked properly and the optimal flying height was found, the next step was to use the algorithm on real plants. Soybean plants were grown by Dr. Song Li and his graduate student, Qian Zhu in a greenhouse located on Virginia Tech's campus. These plants were then moved to the Drone Cage and

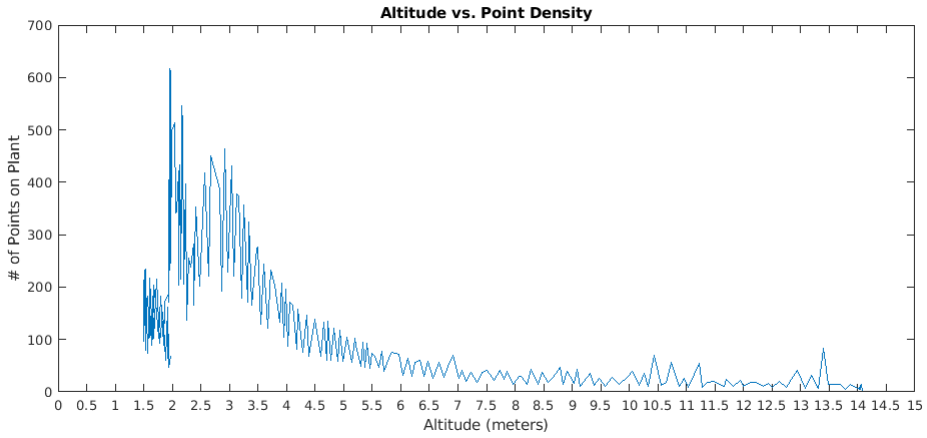


Figure 4.2: Point Density of LiDAR Scan

lined up into 2 rows. A top and side view of these plants can be seen in Figure 4.3. After the data was collected, each of the plants was manually found within the LiDAR data scans. Once each of the 10 plants was found, the LiDAR scan was fed into the ground plane and height estimation algorithm. This was used to output the estimated height of the soybean plant. The output of the algorithm can be seen in Figure 4.4. The results of the experiment are shown in Table 4.2. As can be seen, their is a constant underestimation of the algorithm based height estimation compared to the manual measurements. There are a few reasons this is the case. The two main reasons are due to both horizontal and vertical wind. The day the flights were conducted was relatively windy. Because of this, their was a crosswind that would horizontally strike the plants causing them to flutter. With any wind hitting the plants, their height changes due to plant and stem movement. Secondly, there was a down wash from the UAV's rotors spinning and flying overhead of the plants. This wind would push the plants downwards also causing their height to reduce. On average, there was a 10.23 % underestimation of the soybean plants with our ground plane and height estimation algorithm.

(a) Top View of Soybean Plants

(b) Side View of Soybean Plants

Figure 4.3: Soybean Flight

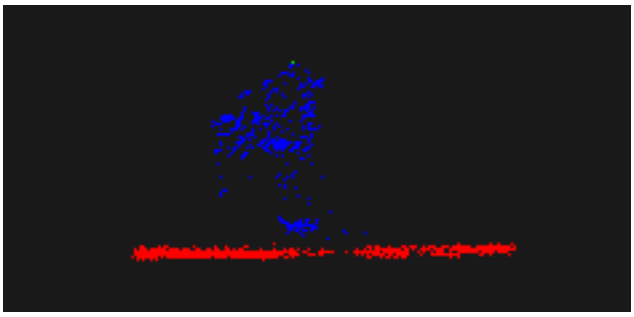


Figure 4.4: Ground Plane and Height Estimation Algorithm on Soybean Plant

Plant Number	1	2	3	4	5	6	7	8	9	10
Manual Height Measurement (m)	0.94	0.97	0.79	1.12	1.09	1.07	0.97	0.93	1.02	1.22
LiDAR Height Estimation (m)	0.85	0.91	0.69	0.99	0.96	0.96	0.88	0.85	0.93	1.04
Percent Difference (%)	-10.15	-6.17	-13.23	-12.19	-12.59	-10.89	-9.29	-8.48	-8.49	-15.87

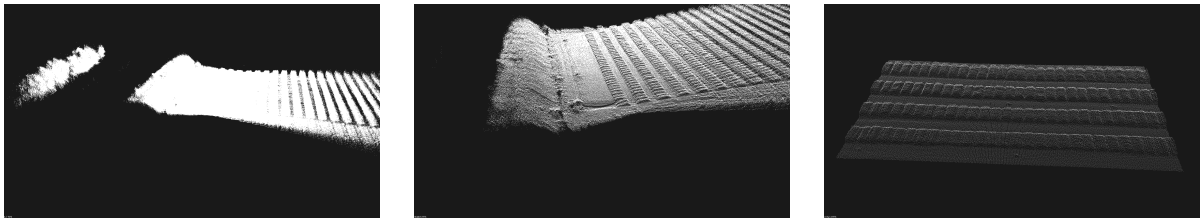
Table 4.2: Drone Cage Soybean Flight Results

4.2 Kentland

4.2.1 Wheat Experiment

At Virginia Tech's Kentland Farm, a section of land was used to grow plots of wheat. We took advantage of our access to this area to conduct a more real-world test of some of our algorithms. Using a plot-based environment revealed a new challenge and extra step before we could use our ground plane and height estimation algorithm. Since there were multiple plots in our concatenated point cloud file, we could not feed the entire cloud to our algorithm. We were interested in finding the height for each of the plots of wheat. To do this, we needed to be able to detect the plots within our scan. The approach taken was discussed in the algorithm chapter previously. Figure 4.5 shows the point cloud data that we worked with. Figure 4.5a is the raw scans all concatenated into a single file and then Figure 4.5b shows the down sampled data when fed into a voxel filter. Lastly, Figure 4.5c is the cropped point cloud data so that it only includes the wheat plots that we are looking at. There are 112 plots that are shown in Figure 4.5c. The goal of our plot detection algorithm was to correctly find a bounding box for each of these plots. Figure 3.5 highlights the bounding boxes when using just a K-means clustering algorithm to determine clusters within the point cloud data. As stated before, many of the plots/clusters are found, but there also many that do not fit at all. The dimensions of the bounding boxes were fed into the voting scheme step of the plot detection algorithm. The output of the voting scheme found 52 plots that fit within a certain orientation as well as width range. Whereas, 54 fit within a height range. All of the values within the range were averaged out to determine the best-fit orientation, width, and height of the bounding boxes for the plots. The grid was then manually setup to overlay on top of the point cloud data to extract the points within each plot. This grid fitting was

shown previously in Figure 3.6.



(a) Concatenated Point Cloud of Wheat Farm

(b) Voxel Filter Applied to Point Cloud

(c) Cropped Point Cloud to Include Only Target Plots

Figure 4.5: Wheat Point Cloud

The height estimation on the farm field gave promising results. Figure 4.6 shows the percent difference between the hand-measurements and the height estimation. For the height estimation, a few different methods were tried. The first method was to use the voxel filtered data set shown in Figure 4.5. The maximum height within each plot was used as the height estimate. This is shown in the first plot of Figure 4.6. However, since it was a down-sampled dataset, the output was all underestimations and the average percent difference between the manual measurements and height estimations was +/- 13.35%. It can also be seen in the figure that the percent differences were centered around the -15 to -10 % bin. The next step was to fit the raw data shown in Figure 4.5a. That data was cropped and recentered, but not down-sampled using the voxel filter. Afterwards, the plot detection bounding boxes were used to extract the points within each plot. These were then concatenated with the already voxel filtered dataset in Figure 4.5. The maximum height within each plot was again used. The results of this methodology are shown in the second plot in Figure 4.6. These gave more accurate results, but not within ideal specification. Because the raw data was used, the new dataset was a lot noisier and the average percent difference was +/- 13.81%. The percent differences were centered around the +10 to +15 bin for this method, but there was also a single datapoint within the +100 to +105 % bin because of the noise. So, to get more accurate results, the 99th percentile height point was extracted from each plot. This gave the most accurate height estimations with the percent difference shown in the bottom plot of Figure 4.6. The average percent difference across all plots was +/- 5.36%. The percent differences for this method centered around the -5 to 0 % bin of the figure. These results were a better fit and even an improvement from the Soybean Drone Cage experiment.

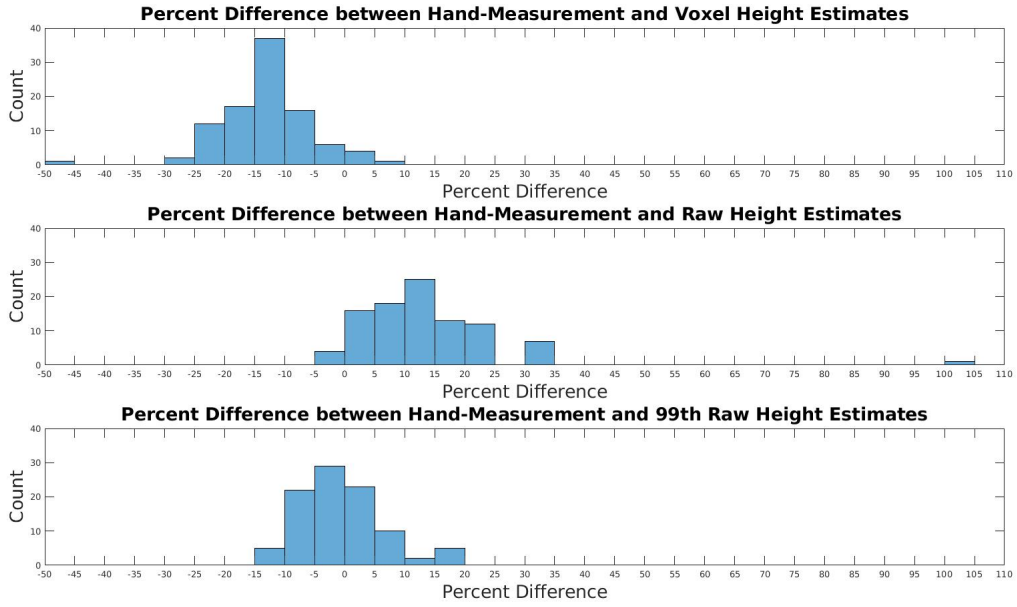


Figure 4.6: Hand-Measurements vs. Height Estimations

4.3 Turfgrass Research Center

4.3.1 Pasture Experiment

A 30 feet x 30 feet pasture was grown at Virginia Tech's Turfgrass Research Center. This region was allowed to grow untouched over a period of time. The area or perimeter around it was constantly maintained and mowed. We used this region to test our growth overtime analysis algorithm. Our UAV flew over this target region on a weekly basis and captured data using the LiDAR. Over the course of 5 weeks, 4 flights were conducted. The growth overtime analysis algorithm was applied to each of these flights. Manual measurements were taken by hand at 9 areas around the plot. These manual measurements were taken each time a flight was done. The average of these hand measurements can be seen in the first row of Table 4.3. In subsequent rows, various percentiles of the height estimations are shown. These are also plotted in Figure 4.7. The line all the way at the top of the figure is the manual measurements. Each line below represents the percentile heights in decreasing order (99.5th, 99th, ... , 50th).

As can be seen, the height estimations are all underestimates. However, since our target information for this algorithm was growth; these underestimations are not as important. To estimate the growth, the weekly changes were observed. To get these changes the difference in height between each consecutive week was computed. The growth is shown in Table 4.4. Again, the first row shows the average growth in the manual measurements of the plot. It

	May 15th	May 29th	June 5th	June 12th
Hand Measurement (cm)	**48.00**	**68.17**	**74.11**	**69.04**
50th Percentile (cm)	3.97	10.68	12.18	9.60
75th Percentile (cm)	6.31	16.01	18.41	14.60
90th Percentile (cm)	8.73	22.08	26.11	19.93
95th Percentile (cm)	11.03	27.48	31.53	24.48
97.5th Percentile (cm)	13.43	32.22	36.64	30.42
99th Percentile (cm)	15.62	36.43	41.65	36.17
99.5th Percentile (cm)	16.70	39.23	44.29	39.42

Table 4.3: Turfgrass Height Measurements and Estimations

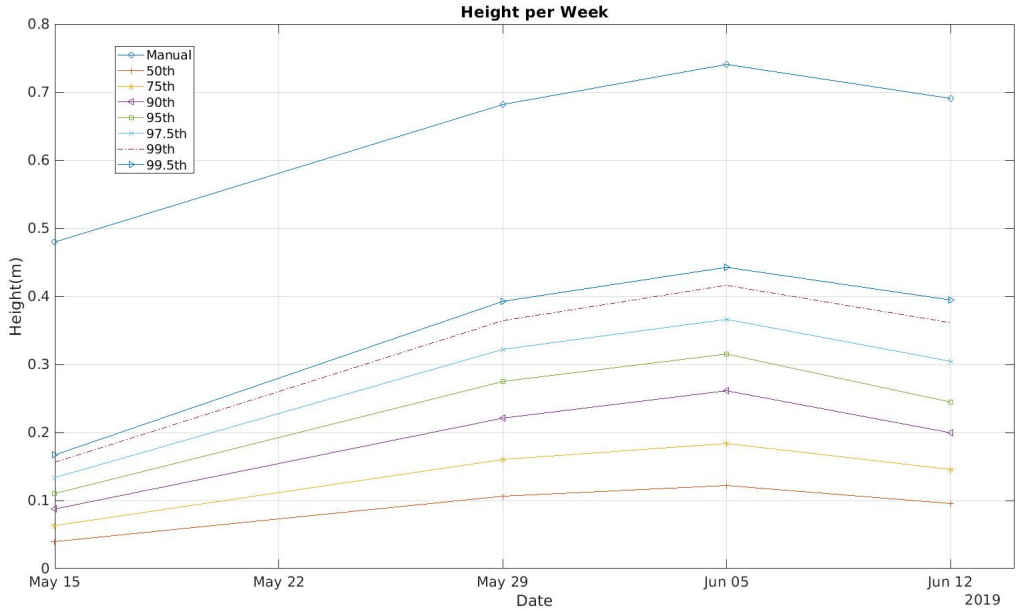


Figure 4.7: Hand-Measurements vs. Height Estimations of Turfgrass Grazing Region

	Growth from Week 1 - 2	Growth from Week 2 - 3	Growth from Week 3 - 4
Hand Measurement (cm)	**20.17**	**5.94**	**-5.07**
50th Percentile (cm)	6.72	1.49	-2.58
75th Percentile (cm)	9.70	2.40	-3.81
90th Percentile (cm)	13.34	4.03	-6.18
95th Percentile (cm)	16.45	4.05	-7.05
97.5th Percentile (cm)	18.79	4.40	-6.21
99th Percentile (cm)	20.80	5.22	-5.48
99.5th Percentile (cm)	22.54	5.05	-4.87

Table 4.4: Turfgrass Growth Measurements and Estimations

	Growth from Week 1 - 2	Growth from Week 2 - 3	Growth from Week 3 - 4
50th Percentile (%)	100.04	119.76	-65.11
75th Percentile (%)	70.12	84.80	-28.37
90th Percentile (%)	40.72	38.35	-19.72
95th Percentile (%)	20.32	37.95	-32.62
97.5th Percentile (%)	7.04	29.79	-20.13
99th Percentile (%)	**3.10**	**12.99**	**-7.65**
99.5th Percentile (%)	11.10	16.18	-3.97

Table 4.5: Percent Difference of Turfgrass Growth Measurements and Estimations

should be noted that between weeks 3 and 4, a decline in the growth was observed. This is because that was towards the end of the growing cycle and some of the plants had begin dying and drying out. Each subsequent row shows the various percentile growth estimates starting at the 50th percentile and ending with the 99.5th percentile.

Next, the percent difference between the manual measurements and the percentiles was computed. This is shown in Table 4.5. The best results were when the 99th percentile of the growth estimations were used. Ideally, the percent difference would be 0.0 %, indicating that there was no difference between the manual measurements and the growth estimations. For the 99th percentile estimations, the average percent difference with the manual measurements was +/- 7.91 %. This shows that the growth overtime analysis algorithm output a relatively accurate growth height estimation compared to the actual growth observed.

Chapter 5

Future Work

While we were happy with the initial results provided by these algorithms, there is definitely a need to improve the work and further develop them. There are a few key areas with our developed algorithms that future work would help make better. This is true for both the plot detection algorithm as well as the ground plane and height estimation method. Regarding the plot detection algorithm, the manual grid selection needs to be worked on. The rest of the algorithm relies on autonomous and code based solutions. The grid selection is the only portion that requires user input and manual work. If focus is put here to replace it with a code based method, the entire algorithm would be a lot more streamlined. For our work, the main focus was to find the best fitting plot dimensions and orientations. The voting scheme and clustering algorithms accomplished this really well, but again, the grid selection slowed down the work process.

The ground plane and height estimation could also use some refining for future work. Currently, it is very particular to the type of environments that are being used. Both the Drone Cage and Kentland Farm are relatively flat and consistent. With terrains that are more variable on a local basis, the algorithm would not be as effective. The plot detection algorithm was used to help combat some of this strictness by using the plots as localities to feed into the algorithm. Not all environments are going to be as organized as a farm, and even some farms won't have rows like the ones used for the wheat plants.

Regarding the growth overtime analysis, ideally it would be tested on a larger dataset. Currently, we only compared it to 4 weeks of flights which provided 3 growth measurements and estimations. To further validate the algorithm, different pastures over a longer period of time are needed to collect even more data.

Another area of improvement would be the implementation of multiple UAV's or UGV's to help combat the limited battery life that is present on a single robot. The larger the farm is, the harder and more unreasonable it is for a single robot to cover the entire farm on one set of batteries. This is especially true for air-based robots. To work on this, the data collection

can be sped up if more robots are used and coordinated with one another.

Working on these improvements will help make the algorithms more applicable to a wider range of applications. This is necessary before the developed methodologies can be implemented on a wider basis. As an initial experiment, the methodologies look promising for helping solve our initial problem statement of the manually intensive height measurements farmers have to do to monitor plant growth for precision agriculture.

Bibliography

[1] 300mbps wireless n nano router. https://www.tp-link.com/us/home-networking/wifi-router/tl-wr802n/. (Accessed on 06/01/2019).

[2] 4-port ultra-slim usb 3.0 hub. https://www.anker.com/products/variant/4port-ultraslim-usb-30-hub/A7516011. (Accessed on 06/01/2019).

[3] David Anthony, Sebastian Elbaum, Aaron Lorenz, and Carrick Detweiler. On crop height estimation with uavs. In *Intelligent Robots and Systems, 2014 IEEE/RSJ International Conference on*, pages 971–976. IEEE, 2012.

[4] David Anthony, Sebastian Elbaum, Aaron Lorenz, and Carrick Detweiler. On crop height estimation with uavs. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[5] Riccardo Bommarco, David Kleijn, and Simon G. Potts. Ecological intensification: harnessing ecosystem services for food security. *Trends in Ecology and Evolution*, 28(4):230–238, 2013.

[6] D.d. Briske, J.d. Derner, J.r. Brown, S.d. Fuhlendorf, W.r. Teague, K.m. Havstad, R.l. Gillen, A.j. Ash, and W.d. Willms. Rotational grazing on rangelands: Reconciliation of perception and experimental evidence. *Rangeland Ecology and Management*, 61(1):3–17, 2008.

[7] Dji onboard sdk documentation. https://developer.dji.com/onboard-api-reference/index.html. (Accessed on 06/02/2019).

[8] dji_sdk. http://wiki.ros.org/dji_sdk, December 2017. (Accessed on 06/02/2019).

[9] Dustin Franklin. Nvidia jetson tx2 delivers twice the intelligence to the edge. https://developer.nvidia.com/embedded/buy/jetson-tx2-devkit, March 2017. (Accessed on 06/01/2019).

[10] A. J. Franzluebbers, L. K. Paine, J. R. Winsten, M. Krome, M. A. Sanderson, K. Ogles, and D. Thompson. Well-managed grazing systems: A forgotten hero of conservation. *Journal of Soil and Water Conservation*, 67(4), 2012.

[11] Getting started with the velodyne vlp16. http://wiki.ros.org/velodyne, January 2019. (Accessed on 06/02/2019).

[12] H. C. J. Godfray. Food and biodiversity. *Science*, 333(6047):1231–1232, 2011.

[13] Hardware setup guide. https://developer.dji.com/onboard-sdk/documentation/development-workflow/hardware-setup.html, February 2019. (Accessed on 06/01/2019).

[14] Harness ai at the edge with the jetson tx2 developer kit. https://developer.nvidia.com/embedded/buy/jetson-tx2-devkit. (Accessed on 06/01/2019).

[15] Jetpack 3.2.1 release notes. https://developer.nvidia.com/embedded/jetpack-3_2_1. (Accessed on 06/02/2019).

[16] Erkan Kayacan, Sierra N. Young, Joshua M. Peschel, and Girish Chowdhary. High-precision control of tracked field robots in the presence of unknown traction coefficients. *Journal of Field Robotics*, 35(7):1050–1062, 2018.

[17] Zhengqi Li, Nikolaos Stefas, Haluk Bayram, and Volkan Isler. Mapping corn fields with uavs.

[18] Simon Madec, Fred Baret, Benot De Solan, Samuel Thomas, Dan Dutartre, Stphane Jezequel, Matthieu Hemmerl, Gallian Colombeau, and Alexis Comar. High-throughput phenotyping of plant height: Comparing unmanned aerial vehicles and ground lidar estimates. *Frontiers in Plant Science*, 8, 2017.

[19] Matrice 600 pro. https://www.dji.com/matrice600-pro. (Accessed on 06/01/2019).

[20] Matrice 600 pro specs. https://www.dji.com/matrice600-pro/info. (Accessed on 06/01/2019).

[21] Matrice 600 upper expansion bay kit. https://store.dji.com/product/matrice-600-upper-expansion-bay-kit. (Accessed on 06/01/2019).

[22] Meet zed. https://www.stereolabs.com/zed/. (Accessed on 06/01/2019).

[23] Angela T. Moles and Michelle R. Leishman. The seedling as part of a plants life history strategy. *Seedling Ecology and Evolution*, pages 217–238, 2008.

[24] Angela T. Moles, David I. Warton, Laura Warman, Nathan G. Swenson, Shawn W. Laffan, Amy E. Zanne, Andy Pitman, Frank A. Hemmings, and Michelle R. Leishman. Global patterns in plant height. *Journal of Ecology*, 97(5):923–932, 2009.

[25] Nvidia cuda toolkit 9.0.176. http://developer.download.nvidia.com/compute/cuda/9.0/Prod/docs/sidebar/CUDA_Toolkit_Release_Notes.pdf, March 2018. (Accessed on 06/02/2019).

[26] Lawrence G. Oates, Daniel J. Undersander, Claudio Gratton, Michael M. Bell, and Randall D. Jackson. Management-intensive rotational grazing enhances forage production and quality of subhumid cool-season pastures. *Crop Science*, 51(2):892, 2011.

[27] Opencv. `https://opencv.org/`, 2019. (Accessed on 06/02/2019).

[28] Plane model segmentation. `http://pointclouds.org/documentation/tutorials/planar_segmentation.php`. (Accessed on 08/06/2019).

[29] Nicole Rogers. What is precision agriculture?

[30] Ros kinetic kame. `http://wiki.ros.org/kinetic`, January 2018. (Accessed on 06/02/2019).

[31] National Geographic Society. crop, Oct 2012.

[32] Ubuntu 16.04.6 lts (xenial xerus). `http://releases.ubuntu.com/16.04/`, 2018. (Accessed on 06/02/2019).

[33] Usb to ttl serial cable - debug / console cable for raspberry pi. `https://www.adafruit.com/product/954`. (Accessed on 06/01/2019).

[34] Velodyne. *Velodyne LiDAR PUCK*, 2015. Rev. A.

[35] Puck. `https://velodynelidar.com/vlp-16.html`. (Accessed on 06/01/2019).

[36] Julie E. Woodis and Randall D. Jackson. Subhumid pasture plant communities entrained by management. *Agriculture, Ecosystems and Environment*, 129(1-3):83–90, 2009.

[37] Wenan Yuan, Jiating Li, Madhav Bhatta, Yeyin Shi, P. Baenziger, and Yufeng Ge. Wheat height estimation using lidar in comparison to ultrasonic sensor and uas. *Sensors*, 18(11):3731, 2018.

[38] zed-ros-wrapper. `http://wiki.ros.org/zed-ros-wrapper`, March 2019. (Accessed on 06/02/2019).

[39] Zed sdk 2.7. `https://www.stereolabs.com/developers/release/2.7/`, November 2018. (Accessed on 06/02/2019).

[40] Matteo Ziliani, Stephen Parkes, Ibrahim Hoteit, and Matthew Mccabe. Intra-season crop height variability at commercial farm scales using a fixed-wing uav. *Remote Sensing*, 10(12):2007, 2018.