Crop Height and Plot Estimation from Unmanned Aerial Vehicles using 3D LiDAR

Harnaik Dhami, Kevin Yu, Tianshu Xu, Qian Zhu, Kshitiz Dhakal, James Friel, Song Li, and Pratap Tokekar

Abstract—In this paper, we present techniques to measure crop heights using a 3D LiDAR mounted on an Unmanned Aerial Vehicle (UAV). Knowing the height of plants is crucial to monitor their overall health and growth cycles, especially for high-throughput plant phenotyping. We present a methodology for extracting plant heights from 3D LiDAR point clouds, specifically focusing on row-crop environments. The key steps in our algorithm are clustering of LiDAR points to semi-automatically detect plots, local ground plane estimation, and height estimation. The plot detection uses a $k$-means clustering algorithm followed by a voting scheme to find the bounding boxes of individual plots. We conducted a series of experiments in controlled and natural settings. Our algorithm was able to estimate the plant heights in a field with 112 plots within ±5.36%. This is the first such dataset for 3D LiDAR from an airborne robot over a wheat field. The developed code can be found on the GitHub repository located at https://github.com/hsd1121/PointCloudProcessing.

I. INTRODUCTION

Agriculture has been a vital part of the well-being and progression of society, for food, feed, fiber, oil, ornamental, and industrial uses [27]. Precision agriculture is a field of study dedicated to help optimize the growth and harvest of crops using new technologies [26], [28]. This area will become especially important as the population grows leading to a higher demand from farms. Because of this, it is necessary to optimize our current farms for both growth and health [4], [9], [11].

Currently, tasks performed by farmers all around the world are labor-intensive. People constantly develop tools and machinery to help automate or even replace some of these manual tasks. Monitoring plant health is vital when growing crops to determine the optimal time for crop management such as application of fertilizers, pesticides and to determine the optimal time for harvest. One of the most important traits to monitor during a plant’s growth cycle is crop height. Recording the plant growth allows farmers to monitor and predict vital features of crops such as flowering time and yield [19], [20].

An immediate area where estimating plant growth is critical is high-throughput plant phenotyping and breeding [2], [6]. High-throughput phenotyping refers to collecting large scale phenotypic information about plants, including their heights, which are used for selective breeding and associative mapping through, for example, genome-wide association studies. A bottleneck of high-throughput phenotyping is the data collection process. Manual height measurements are labor-intensive and quickly become infeasible as the plot and farm size increases. Manual height measurements are also sometimes biased. Using wheat as one example, it is only feasible to measure a few selected plants in each plot where hundreds of wheat plants are growing. In this paper, our goal is to alleviate this bottleneck by using an Unmanned Aerial Vehicle (UAV) equipped with a 3D LiDAR for plant height estimation.

We present a technique for determining crop heights using a 3D LiDAR mounted on a UAV. In particular, we focus on farms organized into smaller plots as shown in Figure 1. The plots themselves may be organized in rows. Our technique starts with raw LiDAR scans obtained by a UAV flown above the farm and produces as output bounding boxes around individual plots as well as height estimates for the plants within the plot. We do not assume prior knowledge about the terrain, nor do we require the terrain to be perfectly horizontal.

While there has been recent work on plant height estimation with LiDARs, as we describe in Section II, this is the first such work using a 3D LiDAR mounted on an Unmanned Aerial Vehicle that also estimates the individual plots. We describe our algorithm for plot and height estimation along with the underlying assumptions in Section IV. We conducted several experiments with potted plants and actual wheat plots. We describe the hardware and software setup in Section III. Lastly, we present the experimental results and observations in Section V. Our technique was able to estimate the plant heights for the wheat crops within ±5.36% of the ground truth measured manually.

II. RELATED WORK

There has already been some work done with UAVs and LiDARs for crop height estimations. However, these methods are different from the one presented in this paper. We highlight the differences in this section. Some of these methods include using a 2D LiDAR mounted on a UAV [1], using RGB cameras mounted on fixed-wing [33] and rotor-based UAVs [17], [32], and using a ground robot for navigation between rows of crops [15].

The method described by Anthony et al. uses a 2D LiDAR mounted on the bottom of a UAV facing downwards [2]. They used this setup to develop a method of measuring corn heights.
Fig. 1. This overhead picture was taken during a flight over wheat crops by our UAV. The farm is organized into plots, indicated by white boxes. The plots are organized as a grid. Our technique finds the plots and estimates the height of the crops within a plot.

Since they used a 2D LiDAR, the methodology they developed is different from ones that use 3D LiDARs, such as ours, due to the smaller amount of incoming data. With each incoming scan, they estimate where the ground and top of the crop were using a distribution of the LiDAR point cloud. They fit the incoming height data using a normal distribution. We use a similar percentile analysis for our method. With their methodology, they reported an accuracy of height estimations within 5 cm.

Madec et al. used a similar method, except with a ground-based, 3D LiDAR on a row-based wheat plot. [17]. The researchers implemented a structure-from-motion algorithm to extract a 3D dense point cloud from a camera on the UAV. They compared the measurements between UAV and ground-based LiDAR. They drove the LiDAR along the ground near the border of the crop fields to collect the point cloud scans. Once both point clouds were available, they used a similar distribution analysis to Anthony et al. [2]. After, they compared the two heights with one another the authors determined that there was a strong correlation between the estimated heights. The accuracy of the LiDAR estimations compared to the manual measurements was an RMSE of 3.5 cm.

Yuan et al. used the same LiDAR as us but mounted on a ground vehicle [32]. They also had a UAV with a camera to perform a structure-from-motion analysis as a comparison. The ground vehicle drove along the plots of the row-based wheat crops and, along with ultrasonic sensors, they recorded the data from the sensors. 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 they used this distance to estimate the heights within the scans. They used these ground points with the other points to find the height. Their methodology is similar to ours, but we have developed a unique ground estimation method. As opposed to manually determining the distance between the LiDAR and the ground, our method automates this process through the algorithm. Their final accuracy for the ground-based LiDAR system was an RMSE of 5 cm; whereas the accuracy of the UAV approach was an RMSE of 9 cm. They deemed the ground-based ultrasonic approach was too inaccurate with an RMSE of 34 cm.

Another methodology used was implementing a fixed-wing UAV as opposed to a rotor-based. Ziliani et al. implemented this and it is different from the other methods described above [33]. Their goal was to determine intra-field variability within a farm of maize crops. The fixed-wing UAV flew over the target maize crops with a Sony camera mounted facing downwards. This is unlike our work since they are not using a LiDAR on the UAV. It would capture image data as it passed. Throughout the target environment, they used 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, they used a ground-based LiDAR approach 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. During the flowering of the crops, there was an increased variability and the correlation was only 0.65.

There has also been some work done in regards to row detection by Li et al. [16]. 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, they detect the rows between the lines of corn using computer vision techniques. Our method accomplishes this using a 3D LiDAR as opposed to an RGB camera.

Unlike these methods, we use a 3D LiDAR mounted on a UAV to estimate height from overhead. As opposed to pre-determining the height between the LiDAR and the ground, we will estimate this using our algorithm. Our techniques can estimate the heights within ±5.36% (approximately, 4.69 cm error) on average. We also detect the plots of wheat using computer vision methods we developed for 3D LiDAR, which has not been developed so far.

III. SYSTEM DESCRIPTION

In this section, we describe the system used to collect and process the data. First, we discuss the hardware setup, followed by the software suite.

The UAV platform that we use is the DJI Matrice 600 Pro. The on-board computer is the NVIDIA Jetson TX2 [12]. The 3D LiDAR of choice is the Velodyne VLP 16. There were reasons we decided to use this specific platform. Primarily, the DJI Matrice 600 pro has a maximum takeoff weight of 15.5 kg. 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 [18]. 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 meters and an accuracy of ±3 cm [30]. Figure III shows all the hardware mounted on the DJI M600 Pro platform.

We used NVIDIA Jetpack 3.2.1 [14] on the TX2. This installs Ubuntu 16.04 [29] on the computer, as well as NVIDIA’s CUDA 9.0 [22] and OpenCV [23]. We also used Robot Operating System (ROS) Kinetic and the ROS drivers for our sensors. They are the Velodyne driver [10], DJI Onboard SDK [7], and the DJI SDK [8]. We also used the Point Cloud Library (PCL) [24] and C++ [5] to process the LiDAR.
scans. We used Python [25] as well for the clustering and grid selection described in Section IV.

IV. ALGORITHM DESCRIPTIONS

In this section, we describe the algorithms used to solve the problems highlighted in the previous section. Figure 3 shows the system block diagram for the algorithm pipeline. We discuss each of the blocks step-by-step below. There are assumptions and expected inputs to this problem. We list them below:

- We assume that the plots are aligned in a regular ordered grid, at least locally.
- We assume that we know the number of plots. In principle, we only need to know the number of plots for a small subset of the environment.

A. Point Cloud Processing

1) Coordinate Frame Transformation: Before we can extract and work with any of the plant data, we need to process the raw point cloud data. 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 fine if we only care about a single scan from the LiDAR. However, if we are looking at successive scans, we need to frame each scan relative to one another. This is where the point cloud processing work comes into place. Because we mounted the LiDAR facing downwards on the bottom of the DJI M600 Pro, we can use its IMU and GPS sensors to help build these reference frames.

We use a static transformation between the UAV body frame to the LiDAR frame. Thus, the pose of the UAV published by the DJI ROS driver can be used to convert the LiDAR data in the world frame. We show the frame transformation in Figure 4.

Fig. 3. The system block diagram describing our developed methodology.

Fig. 4. The coordinate frame relationships within the ROS environment.

2) Map Building: We translate each incoming scan to a world coordinate frame due to the relationship shown in Figure 3. The algorithm does this translation with each incoming scan so that it sets the data about the world origin as opposed to the LiDAR being the origin. We concatenated each successive scan to the previous scan. Over an entire dataset, this results in a single file containing data from every scan translated relative to the movement of the DJI in the world. We show the built map, with some further processing, discussed later, in Figure V-B.1.

We tried other simultaneous localization and mapping (SLAM) packages, such as Berkeley Localization and Map-
ping (BLAM) [21], but found them to be too erroneous. This is possibly due to the lack of global features in our outdoor environments. Instead, the open-loop map building works well enough for our purposes. This is likely because the UAV has multiple GPSs used for correction and that the GPS has a clear line of sight when operating outdoors over a farm.

3) Other Processing Tools: We developed several other tools to help with the point cloud processing. We describe them in this section briefly. Due to the concatenation of the LiDAR scans, the dataset quickly increases in size due to the immense amount of incoming data. Because of that, it was necessary to down-sample the data to not only reduce the data size but also reduce the noise in the data. We do this using a voxel filter [31]. The voxel filter allows down-sampling through averaging points in a set size grid. Down-sampling also helps increase the speed at which we process the data for other algorithms. Another point cloud processing tool that we developed is a crop box filter. We use this to crop the point cloud data and extract what we need for processing. A centering tool is also needed. This takes in a point cloud file and changes the origin of the data to the center point of it. We do this so that we can visualize the dataset better after the cropping. We developed all of these tools using C++ and the Point Cloud Library. Again, Figure V-B.1 shows a dataset with all of this processing applied. We concatenate each LiDAR scan into a point cloud file. Then, that file is voxel filtered, cropped, and re-centered.

B. Plot Detection

After the pre-processing described in the previous subsection, the next step in the algorithm is to detect individual plots within a farm. Recall, that a “plot” refers to a cluster of plants as shown in Figure 1. We need this to use the ground plane and height estimation algorithm on each plot given a point cloud file of the entire farm. We show an example dataset for this in Figure V-B.1

1) K–means Clustering: We start with a concatenated and down-sampled dataset within a single point cloud file. Since we seek to find the plots within this dataset, we remove all points below a certain z-axis value (height). This gives us a dataset similar to what we show in Figure 5. 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.

In Figure 6, we show the minimum oriented bounding box of each cluster. A minimum oriented 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 shown in this figure, the algorithm has not clustered all plots correctly, but it has correctly clustered a fair amount of them. Because of this, we need to implement a voting scheme to help determine the plot size using the bounding boxes.

2) Voting Scheme for Correct Clusters: For the voting scheme, we use the bounding box dimensions and orientations of each of the clusters determined in the previous step. These are all broken into separate ranges and we count the number of clusters that fit into each of the ranges. For example, we break down the orientation data into ranges of 0.3 radians. If the orientation for a bounding box is between 0.0 - 0.29 radians, the vote for that range increases by 1. We do this for each cluster and then after, we use the range with the largest number of votes as the orientation that we set for the plots. We use a similar method for the height and width of the clusters to determine the height and width of the plots.

3) Grid Selection: Using the estimated orientation, width, and height of the plots, we fill in a grid for the dataset. The vertical and horizontal distances between the plots are manually fit to help create the grid. We highlight that even though the grid is currently manually overlaid on top of the target dataset, we are working to automate this process. After we do this grid selection, the results look similar to what we show in Figure 7. We use these bounding boxes on the original dataset. We extend the heights of the boxes so that they cover the ground between plots. All of the points in these boxes are then extracted one box at a time and then fed into the ground plane and height estimation algorithm which we discuss next.

We assume that we know the number of plots in the farm, i.e., we know the correct number of clusters for k–means. However, this is not a strict requirement. We do not need to know the total number of clusters in the entire farm. During pre-processing, we can set the crop box to a small area – small enough to manually count the number of plots. Then, we can use the k–means clustering followed by the voting scheme to determine the size of the cluster. Once we determine the
size of the cluster and the grid pattern for the smaller cropped area, we can extrapolate that grid to the rest of the (uncropped) dataset.

![Fig. 7. The estimated grid overlaid on top of the plots.](image)

**C. Ground Plane and Height Estimation**

The purpose of this algorithm is to take a point cloud file, find the ground plane in it, and then determine height data for objects that are not part of the ground. We describe the process of this algorithm in this section.

1) **Detect Points in Ground Plane:** We detect the ground plane in a LiDAR scan using the Point Cloud Library in C++ by applying a sample plane model segmentation in the PCL library [24]. This segmentation implementation uses a RANSAC algorithm to find a plane within the given dataset. After we optimize the distance threshold parameter, the algorithm gives a similar output to what we show in Figure 8. We flew the DJI over a table that we placed on a flat field. The red points indicate what the algorithm output as points in the plane; whereas, the algorithm ignored the blue points as outliers. As seen from the figure, we successfully use this algorithm to determine the ground plane in LiDAR scans.

2) **Best-Fit Plane of Inlier Points:** We find the inlier points from the planar model segmentation and use them to find the equation that describes a best-fit plane. We do this using the linear least-squares approximation on the inlier dataset. We take each of the “red” points and use them for this approximation to output the centroid of the data, as well as the normal vector of the best-fit plane. Now, with an equation describing the best-fit plane, we determine the height of the outlier points.

3) **Height Estimation:** With a centroid and normal vector for the best-fit plane, we determine the height of each of the outlier points. We do this 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 is the height of that point over the ground. Doing this for all of the outlier points indicates each of their heights relative to the ground plane. For example, in Figure 8 we find the distance of the blue points to the best-fit plane of the red points to determine their height off of the ground. As stated in the grid selection subsection, we extended the bounding box lengths. We did this so that the ground plane estimation works on a localized scale relative to each plot. This helps improve the accuracy of the algorithm by adjusting for the disparities in the ground over a large field.

**V. EXPERIMENTS AND RESULTS**

We conducted experiments in 2 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. We conducted more data collection at Virginia Tech’s Kentland Farm. Kentland Farm provided real-life examples of plotted crops that we could use to collect data. We describe the experiments at each of these locations in detail in this section, along with their results.

**A. Drone Cage**

1) **Point Density Experiment:** During testing and experiments, we determined that if the UAV flew too high over the test subject, the data points in the LiDAR scan would be too sparse. Because of this, we needed to find an optimal flight height so that there were enough data points for the algorithms, but high enough so that the downwash was minimal. We conducted a point density experiment 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, we used the ground plane detection algorithm. For each incoming scan, we determined and removed the ground plane points. Therefore, the remaining points directly underneath the UAV were only a part of the plant. We saved the height of the DJI M600 Pro as well as the number of points on the plant as the 2 data points. Figure 9 graphs these data points. We 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 are not that many points when the altitude is low is that 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.

2) **Soybean Experiment:** Next, we tested the algorithm on real plants. We grew soybean plants in a greenhouse located on Virginia Tech’s campus. These plants were then moved to the Drone Cage and lined up into 2 rows. We show a top and side
view of these plants in Figure 10. After we collected the data, we manually found each of the plants within the LiDAR data scans. Once we found each of the 10 plants, we fed the LiDAR scan into the ground plane and height estimation algorithm. We used this to calculate the estimated height of the soybean plant. We show the output of the algorithm in Figure 11.

We show the results of the experiment in Table I. There is a constant underestimation of the algorithm based height estimation compared to the manual measurements. The main reason is horizontal and vertical wind. The day we conducted the flights was windy. Because of this, there 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 downwash 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.

| Plant Number | Manual Height (m) | LiDAR Height (m) | % Difference |
|--------------|-------------------|-----------------|--------------|
| 1            | 0.94              | 0.85            | -10.15       |
| 2            | 0.97              | 0.91            | -6.17        |
| 3            | 0.79              | 0.89            | -13.23       |
| 4            | 1.12              | 0.99            | -14.19       |
| 5            | 1.09              | 0.96            | -12.39       |
| 6            | 1.07              | 0.96            | -10.89       |
| 7            | 0.97              | 0.88            | -9.29        |
| 8            | 0.93              | 0.85            | -8.48        |
| 9            | 1.02              | 0.93            | -8.49        |
| 10           | 1.22              | 1.04            | -15.87       |

TABLE I

The results of the Drone Cage soybean flight.
average percent difference between the manual measurements and height estimations was +/- 13.35%. We show in the figure that the percent differences center around the -15 to -10 % bin. The next step was to fit the raw data shown in Figure [V-B.1] We cropped and recentered that data, but did not down-sample it using the voxel filter. Afterward, we used the plot detection bounding boxes to extract the points within each plot. These were then concatenated with the already voxel filtered dataset in Figure [V-B.1] The maximum height within each plot was again used. We show the results of this methodology in the second plot in Figure [13] These gave more accurate results, but not within ideal specification. Because we used the raw data, the new dataset was a lot noisier and the average percent difference was +/- 13.81%. The percent differences center around the +10 to +15 bin for this method, but there was also a single data point within the +100 to +105 % bin because of the noise. So, to get more accurate results, we extracted the 99th percentile height point from each plot. This gave the most accurate height estimations with the percent difference shown in the bottom plot of Figure [13] 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.

![Fig. 12. Point cloud processing done on the wheat field dataset. (a) The concatenated point cloud of all the input LiDAR scans. (b) The point cloud after a voxel filtering. (c) The point cloud after cropping.](image)

![Fig. 13. The results of the Kentland wheat farm experiment. The top histogram shows the percent difference between the manual measurements and the voxel-only estimations. The middle histogram shows the percent difference using the raw data estimations. The bottom histogram shows the percent difference using the 99th percentile of the raw data estimations.](image)

VI. FUTURE WORK

While the algorithm works satisfactorily in the real world, there are rooms for improvement. This is true for both the plot detection algorithm as well as the ground plane and height estimation method. Regarding the plot detection algorithm, we need to work on the manual grid selection. The rest of the algorithm relies on autonomous and automated solutions. The grid selection is the only portion that requires user input and manual work. If we focus here to replace it with an automated 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 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, the algorithm is particular to the type of environments used. The ground planes in both the Drone Cage and Kentland Farm are flat and consistent. With terrains that are more variable on a local basis, the algorithm would not be as effective. We used the plot detection
algorithm to help combat some of this strictness by using the plots as localities to feed into the algorithm. While most farms are organized, the rows of crops may not be consistent over a larger area. Locally, they might still be in rows but the orientation of the rows can change from one region to another. This requires a separate pre-processing algorithm. Currently, there are already RGB image-based tools to identify the orientation of rows in such cases [13].

Another area of improvement would be the implementation of multiple UAVs 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, we can speed up the data collection if we use more robots 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 we implement the developed methodologies 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.

REFERENCES

[1] David Anthony, Sebastian Elbaum, Aaron Lorenz, and Carrick Tweitler. On crop height estimation with uavs. In Intelligent Robots and Systems, 2014 IEEE/RSJ International Conference on, pages 971–976. IEEE, 2012.
[2] David Anthony, Sebastian Elbaum, Aaron Lorenz, and Carrick Tweitler. On crop height estimation with uavs. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014.
[3] David Ball, Ben Upcroft, Eldert Henten, Anton den Hengel, Pratap Tokekar, and Jnaneswar Das. Jr special issue on agricultural robotics. Journal of Field Robotics, 34(6):1037–1038, 2017.
[4] Riccardo Bonmarco, David Kleijn, and Simon G. Potts. Ecological intensification: harnessing ecosystem services for food security. Trends in Ecology and Evolution, 28(4):230–238, 2013.
[5] C++.
[6] Jnaneswar Das, Gareth Cross, Chao Qu, Anurag Makineni, Pratap Tokekar, Yash Mulgaonkar, and Vijay Kumar. Devices, systems, and methods for automated monitoring enabling precision agriculture. In Proceedings of IEEE Conference on Automation Science and Engineering (CASE), pages 462–469. IEEE, August 2015.
[7] DJI on board 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] A. J. Franzluethers, 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.
[10] Getting started with the velodyne vlp16. http://wiki.ros.org/velodyne January 2019. (Accessed on 06/02/2019).
[11] H. C. J. Godfray. Food and biodiversity. Science, 333(6047):1231–1232, 2011.
[12] 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).
[13] Volkan Isler. Mapping corn fields with uavs, Aug 2015.
[14] Jetpack 3.2.1 release notes. https://developer.nvidia.com/embedded/jetpack-3_2_1 (Accessed on 06/02/2019).
[15] 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.
[16] Zhengqi Li, Nikolaos Stefan, Haluk Bayram, and Volkan Isler. Mapping corn fields with uavs.
[17] Simon Madec, Fred Baret, Benot De Solan, Samuel Thomas, Dan Dutartre, Stephane Jezquel, Matthieu Hemmer, 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.
[18] Matrice 600 pro specs. https://www.dji.com/matric600-pro/info (Accessed on 06/01/2019).
[19] 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.
[20] Angela T. Moles, David I. Warton, Laura Warman, Nathan G. Swenson, Shaw 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.
[21] Erik Nelson. erik-nelson/blam, Aug 2016.
[22] NVIDIA CUDA Toolkit documentation. https://developer.download.nvidia.com/compute/cuda/9.0/Prod/docs/sidebar/CUDA_Toolkit_Release_Notes.pdf March 2018. (Accessed on 06/02/2019).
[23] Opencv. https://opencv.org/ 2019. (Accessed on 06/02/2019).
[24] Plane model segmentation. http://pointclouds.org/documentation/tutorials/planar_segmentation.php (Accessed on 08/06/2019).
[25] Python 2.7.0 release.
[26] Nicole Rogers. What is precision agriculture?
[27] National Geographic Society. crop, Oct 2012.
[28] Pratap Tokekar, Joshua Vander Hook, David Mulla, and Volkan Isler. Sensor planning for a symbiotic UAV and UGV system for precision agriculture. IEEE Transactions on Robotics, 32(6):1498–1511, 2016.
[29] Ubuntu 16.04.6 lts (xenial xerus). http://releases.ubuntu.com/16.04/ 2018. (Accessed on 06/02/2019).
[30] Velodyne. Velodyne LiDAR PUCK, 2015. Rev. A.
[31] Downsampling a pointcloud using a voxelgrid filter.
[32] 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.
[33] Matteo Zilian, 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.