Estimation of plant points using Pointcloud and Image data

Sai Sourabh Tiruvaipati
Embedded Systems Engineering
Albert-Ludwigs-Universität Freiburg

Abstract

Autonomous robots will be deployed on a large scale in agriculture in the near-future. To become commercial offering in the marker, they need to be able to navigate the farms independently. This requires a robust on-board algorithm which can recognise ans distinguish plants in the field. This report presents the experiment results on Image and Pointcloud data acquired from the sensor setup atop a robot in an agricultural field. The data from the two sensors is fused and various features were extracted and analysed. Unsupervised learning algorithms have been deployed to form plant and non-plant clusters. Performance of algorithms have been compared between datasets containing plants of different sizes. A segmentation using MLESAC was done to estimate a ground plane of the field. The algorithms are able to segment the fused data into the respective classes. Further ground work has been done to identify individual plants

Index Terms

PCL, ROS, agricultural robots, TGI, curvature, clustering, SAC segmentation , Feature Extraction

CONTENTS

List of Figures

List of Tables

I. Acknowledgements

Working in the pandemic is a new, tough reality we all have to face and get used to. This project is one of the many works which was also affected. I would like to thank Wera and Freya for their valuable guidance and patience shown to me during the duration of this project. In the course of this project, I have learnt many aspects in the field of autonomous devices, which I will always refer to, if and when needed.

I would also like to thank Ms Martina Nopper for her help with the organization of this project.

II. Introduction

Agriculture is one of the oldest and most important industries that human civilization has established. Fundamentally, agriculture relies on the efficient utilization of natural resources. To live up to the demand by an increasing population while being efficient with respect to the environment, a sustainable approach to agriculture is required.

Agricultural mechanization is the use of machines to perform tedious operations. It has contributed to improving agricultural productivity through more efficient use of the workforce, improved work timeliness, and more efficient input management. With the continuous advances in agricultural mechanization and automation technology over the last few decades, agriculture has entered the era of robotic agriculture. Agricultural robots generally have a set of intelligent behaviors similar to human operators, such as perceptual, reasoning, and operational functions for performing certain operations and tasks in an agricultural environment, with or without human supervision.Such robotic technology has the potential to further reduce labor use, increase the accuracy and efficiency of production inputs, and contribute to improving agricultural productivity and long-term sustainability of the industry.

A. Motivation

In the future, mobile robots will be able to traverse unaided on the farm, while participating in various types of agricultural activities. Since most crops are grown in rows, an important step in achieving this long-term aim is to develop a system to recognize plant rows and individual plants. This allows the robot to follow sequential plants accurately.

For many agricultural activities such as spraying, planting and harvesting to be conducted successfully, it is essential to accurately negotiate the crop. One solution for recognizing plants is to rely on repetitive, globally accurate localization of the sensor platform. The plants being stationary, can be ”recognized” by its location in the farm.

These recognized plants can also act as landmarks to localise the robot globally. To enable long-term autonomy for smaller and more economical robots, it is desirable to develop GPS-independent systems. General purpose non-GPS-based techniques such as simultaneous localization and mapping (SLAM) are a good example.

This work aims at contributing to the aforementioned goal by using pointcloud data from Lidar sensors and image data covering three bands inside the visual spectrum (RGB) from a color camera. A combination of data from both the sensors is used to identify features and thus segment the estimated plant points from the Pointcloud.

The experiments have been carried out primarily in ROS using C++, and Python 2.7 along with libraries such as PCL and sklearn.

The remainder of this work is structured as follows. The background and related work is presented in Section section III. In Section 3 the setup and methodology is presented. The experimental results are proposed in Section 4. Finally, the conclusions are presented in Section 5.

Fig. 1: Data acquiration using the Bonirob in an Agricultural Field [5]

III. Fundamentals

A. Related Work

Prior works in the application area of using similar data to work for agricultural robotics, especially crop row detection were found in the work by Winterhalter et al. [16].

Many approaches tend to use image color information provided by the scanner for LiDAR data segmentation. The image-based approaches have key benefits namely, 2D image processing is often more effective than processing in 3D. Verdoja et al.[15] portrays significant imaging techniques such as edge detection applied in the 2D color images before projecting back. Nex and Rinaudo [10] put forward segmentation and feature extraction approach using LiDAR data and muti-image matching.

LiDAR segmentation is also achieved through model-based approaches such as, Random sample consensus and the Hough transform (HT) are two well known algorithms which use basic geometry for grouping purposes, such as planes, spheres, and other predefined shapes. Points with the same numeric representation are grouped into a single segment. RANSAC sets the minimum value arbitrarily and repeatedly to identify the best parameter that fits the candidate's mathematical model. Then the parameters are tested to determine the best model match. [3]

To understand the dataset and the setup of BoniRob as in Figure 1, the dataset from an experiment with similar setup using BoniRob was consulted in the work by Chebrolu et al. [5].

B. Technologies

Introducing the technologies familiarized with, in the context of this project.

1) Robot Operating System: Robot Operating System or ROS represents a robotic-specific middleware which is developed as part of the STAIR[1] project as well as the Personal Robots Program[2] at Stanford University in cooperation with the robotic manufacturer Willow Garage Quigley et al[11].

This framework as shown in Figure 2 provides a set of tools to support many different common robotic problems like path planning, collision detection, image processing etc. By using packages where each of them provides one particular functionality ROS allows developers to implement robotic code which is reusable.

[1]Stanford Artificial Intelligence Robot: http://stair.stanford.edu/

[2]http://personalrobotics.stanford.edu/

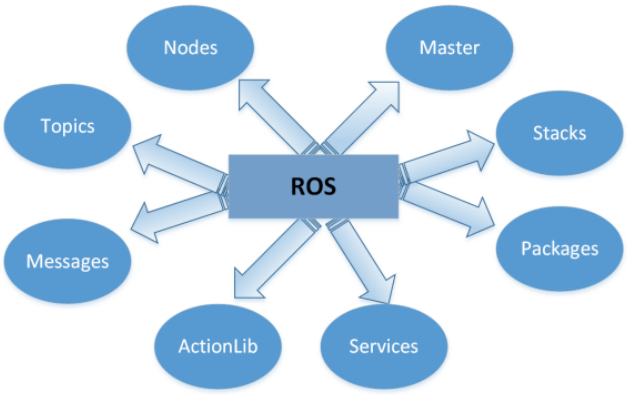


Fig. 2: Core concepts of the ROS framework [8]

In ROS, processes are called nodes which are organized in a Peer-to-Peer network without a central point of communication. Any node in the system can access this network as well as is able to interact with other nodes. The master node provides name registration and lookup for all custom nodes. Without the master node, it is not possible to communicate with nodes, services, messages, and others. The nodes communicate via topics or services following the publisher/subscriber protocol.

ROS provides libraries in several programming languages for the implementation of nodes and inter-process communication.

- rospy is a pure Python client library for ROS and is designed to provide the advantages of an object-oriented scripting language to ROS.
- roscpp (C++) presents the most used client library which is mainly used to implement high-performance functionality in ROS
- ROS also provides Lisp support along with experimental support for Java, Lua, EusLisp, R as well.

ROS allows using simulation time. That means, one can handle the acceleration of the simulation. there is a ROS topic called ‘clock’ responsible for handling simulation time. To make this topic active one has to set a parameter called ‘use_sim_time’ to true. Now, time will not progress until there is a message on ‘clock’ topic. The ‘clock’ server is an any node in a ROS network that publishes message on a ‘clock’ topic. There should only be one clock server in the ROS network and the clock server should be initialized first before any other nodes are running, if otherwise, issues regarding time related computations could be encountered.

RViz is a 3d visualization tool for ROS. All the nodes, parameters, topics in ROS network that have the visual properties can be visualized using this tool. A typical RViz window is shown in Figure 3

2) Point Cloud Library: For different processing steps, the Point Cloud Library (PCL) [13], which is fully integrated in ROS, is used. It supports a lot of different tasks like filtering, model fitting, segmentation, surface reconstruction, registration and feature estimation. Additionally it provides different tools for an increase in performance like OpenMP and Threading Building Blocks. Due to the integration in ROS, most algorithm can be executed as own nodelets.

3) Scikit-learn: Scikit-learn provides a range of supervised and unsupervised learning algorithms via an interface in Python. Scikit-learn was initially developed by David Cournapeau as a Google summer of code project in 2007. It’s built upon some of the existing technology like NumPy, pandas, and Matplotlib. The functionality that scikit-learn provides include Regression, Classification, Clustering, Model selection and Preprocessing.

C. Dataset

The dataset to be working on is acquired from previous work. It was recorded on a BoniRob platform.

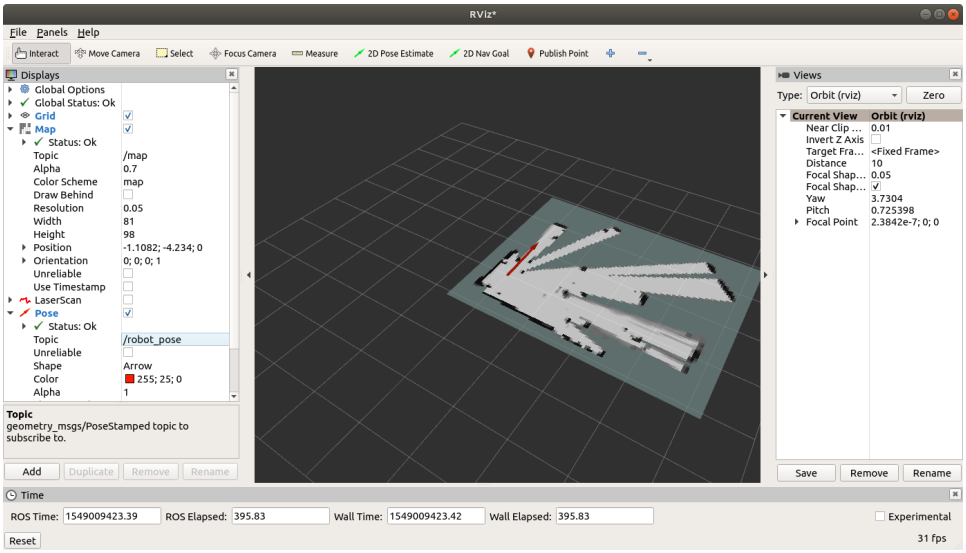


Fig. 3: RViz window

1) Sensor setup: The sensor setup that included two LiDAR Velodyne Sensors mounted on the front and the rear and a PointGrey Blackfly color camera. The BoniRob platform is a multi-purpose robot by Bosch DeepField Robotics. BoniRob is developed for applications in precision agriculture. It provides mounts for installing different sensors for task-specific applications. [5] .

The main purpose of the Velodyne sensors is to acquire laser data in order to create a 3D map of environment which can be used for applications such as navigation tasks. This sensor provides distance and reflectance measurements obtained by a rotating column of 16 laser diodes. The BoniRob is mounted with two of these sensors, one in the front right top corner of the chassis and the other in the rear left top corner. They have a slight tilt towards the ground to better detect objects close to the robot. [5]

2) LiDAR: LiDAR is a powerful remote sensing technology for the acquisition of terrain surface. Recently, Light Detection and Ranging has become a common distance measurement technology in a lot of different domains. To measure the distance, the sensor emits a signal and measures the time t until the signal is returned. Based on this information, it is possible to determine the path distance d of the signal with the Equation 1, where c is the speed of the light.

$$d = \frac{c \cdot \Delta t}{2} \tag{1}$$

LiDAR sensor not only generates 3D pointclouds but also detects the laser impulse reflection data, known as intensity.

3) ROSbag: A rosbag is a tool for data recording provided by ROS. The introduction and working of ROS is done in subsubsection III-B1 . The dataset which is in the format of a rosbag (*.bag) file contains the pointcloud data from the LiDAR sensor, the raw image data from the Camera, the odometry of the robot and the robot model. Along with the data, a TF tree, which is the collection of transformations between coordinate frames corresponding to different sensors and the robot base at each time stamp is also available.

The data in a rosbag, hereon forth also referred to as, bag file, is organized by topics, where a topic is determined by sensor, type of sensor, position, datatype and if necessary the transport method, for example, compressed or raw. Each instance of the data, known as a message has a header which holds the important information regarding the data, such as the time recorded, known as timestamp, the sequence of the the message and the identity or the name given to the coordinate frame.

4) Point Clouds: The measurement data received from the input LiDAR sensor is a general structure called the point cloud. Point Clouds are a collection of data points for a particular coordinate system. In addition to spatial coordinates, additional point cloud data parameters cloud be rgb-Color and intensity of a receiving laser beam for particular points. However, this information is sensor-specific. This causes the following restrictions for the point cloud:

- The sensor range is finite. As a result, the point cloud only represents the nearby area.
- The density of the data points gets reduced with ongoing distance due to the circular spread of the sensor beams.

IV. Implementation

The first tasks part of this work consisted of familiarizing with the evnvironment of ROS using C++ and catkin, which is a build system of ROS. A build system generates 'targets' from raw source code that can be used by an end user. catkin combines CMake macros and Python scripts to provide added features on top of CMake's workflow.

A. Preprocessing

The data received was two Rosbag files containing results sensor data acquisition as in subsubsection III-C1 from the Bonirob setup with a duration of $\tilde{8}$0secs each. One bag contained data recorded when the robot traversed fields with large plants and the other bag file contained data when the robot traversed part of the field with small plants.

1) Image data stream: The first step was to rectify the image data stream recorded from the Blackfly sensor. Rectification of images is the transformation of multiple images onto a common coordinate system For this purpose, the *image_proc*, a package part of ROS was used. *image_proc* removes camera distortion from the raw image stream. The application of this package is to De-mosaic and undistort the raw camera image stream. *rectify* nodelet in this package takes an unrectified image stream and its associated calibration parameters, and produces rectified images.

Nodelets work similar to nodes, but benefit from the cost of copying between nodelets of the only nodelet handler. In other words, different algorithms run in different nodelets (within the same nodelet handler), without any copy or serialization cost between these nodelets. Given the raw image as input, it was rectified and colorised.

The message type representing images in ROS is the *sensor_msgs/Image* type. This message type contains an uncompressed image. The *sensor_msgs* package defines messages for commonly used sensors such as cameras and Laser sensors. Notable message types defined in this package include Image, Pointcloud, Imu, JointState, etc.

2) PointCloud data: The raw Pointcloud data, hereon forth also referred to as Pcl, from the Velodyne sensor used contains 16 scan lines from each of the laser diodes. Each point contains information about its X,Y,Z coordinates and also the intensity. These lines are too sparsely spaced to have a full 'picture' of the scanned area. In order to interpolate for the points in the areas in between each scan lines, the pointcloud needed to be 'integrated'. The inherent idea is to use the data not from a single pointcloud, but a series of them together. This could be interpreted as a individual Pcl corresponding not just to its time stamp but having a decay time.

To acheive this, a Pcl integrator C++ class was written wherein, the pointclouds were collected into a buffer of specific maximum length, such that this buffer could be understood as a vector of pointclouds. The correct way to do this requires each pointcloud to be first in the same frame and transformed such that they correspond to the same time. Concatenating the point clouds in the buffer at each step would result in an integrated pointcloud at each timestamp. To accomodate a new Pcl once the buffer is full, the first entered Pcl is ejected from the buffer and the new Pcl is then pushed at the last position of the buffer, just like that of a queue.

(a) small plants dataset

(b) large plants dataset

Fig. 4: Rectified and colorised images from datasets

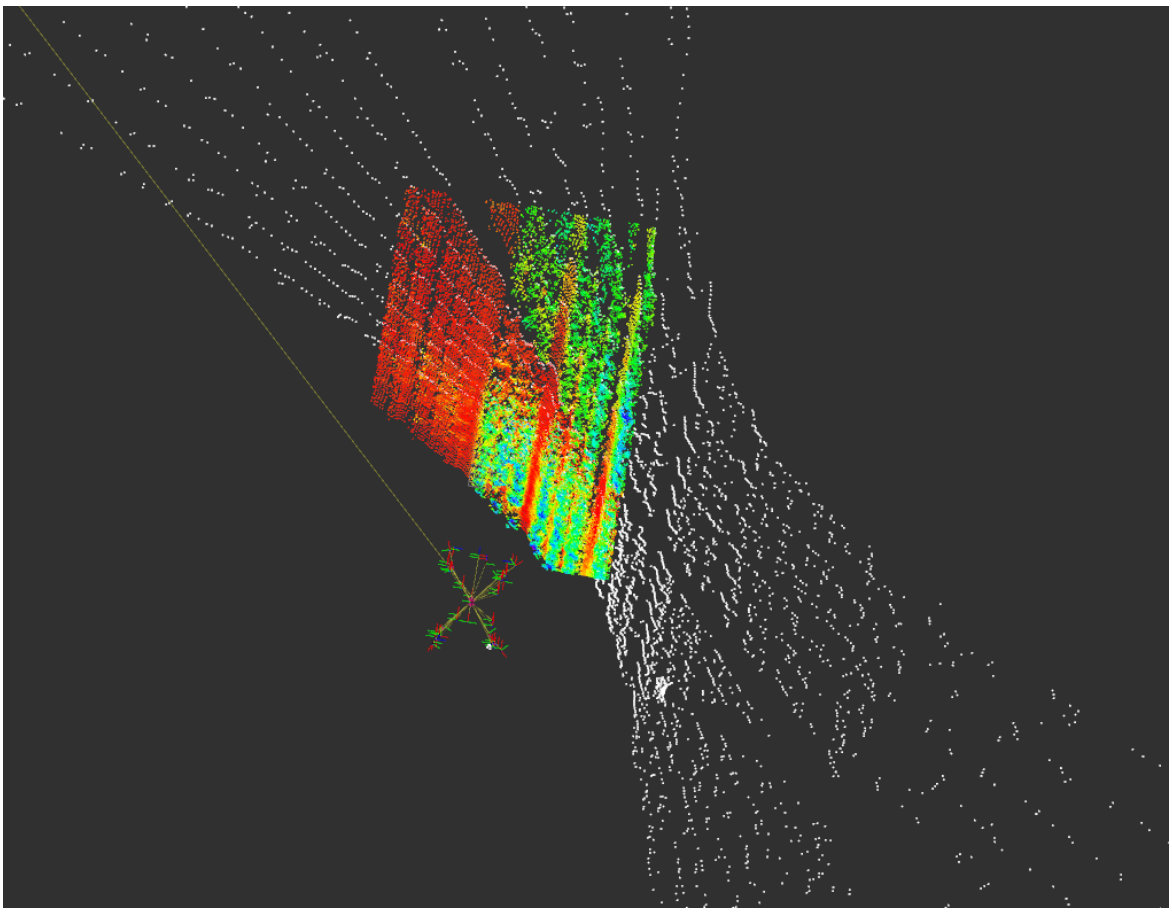


Fig. 5: Integrated and cropped Pointcloud (color) with respect to the raw pointcloud data(white) with cropping window dimensions as 8mx6m

A windowing process was also implemented as a C++ class, which takes in a pointcloud as input, crops the pointcloud data with respect to set limits in XY plane in a set coordinate frame. The windowing for

this work was done with limits in the *base_footprint* frame of the robot. The integrated Pcl is processed through a window of given sizes. This makes sure further processing is carried out only on a defined area in front of the robot, while also reducing the number of points to process per Pointcloud. It also trims out the edges which tend to have less density of points as stated in subsubsection III-C4

The Pointcloud data is represented using the Pointcloud2 message type in *sensor_msgs* package in ROS. It is a standard message type for representing Poinclouds in ROS. This message type is a container for a collection of N-dimensional points, which may even contain additional information such as normals, intensity, color, etc. Point cloud data can be organized in 2D (like an image) or in 1D(unordered). Individual point data is stored in binary blobs, with the intrinsic 'fields' array acting as a lookup, describing the layout of the contents. This message type is quite flexible in holding different kinds data, and is also easier to manipulate whenever necessary.

The intensity property from the LiDAR sensor, as mentioned in subsubsection III-C2 measures the laser reflection data, thus, it depends on the material the laser is being reflected upon, thus different intensities for different surfaces/materials. [7]. It is a feature that can help distinguish and classify objects, given significant difference in reflectivity. Figure 6 shows an integrated windowed pointcloud, the colour of the points on the pointcloud represents intensity.

In this work, the transformation of points between coordinate frames is implemented using Tf2 library in ROS. Tf2 is the second generation of the transformation library, which allows the user to monitor multiple coordinate frames over time. Tf2 maintains the relationship between coordinate frames in a time-protected tree structure known as TF tree, and allows the user to switch points, vectors, etc. between two coordinate frames at any required time.

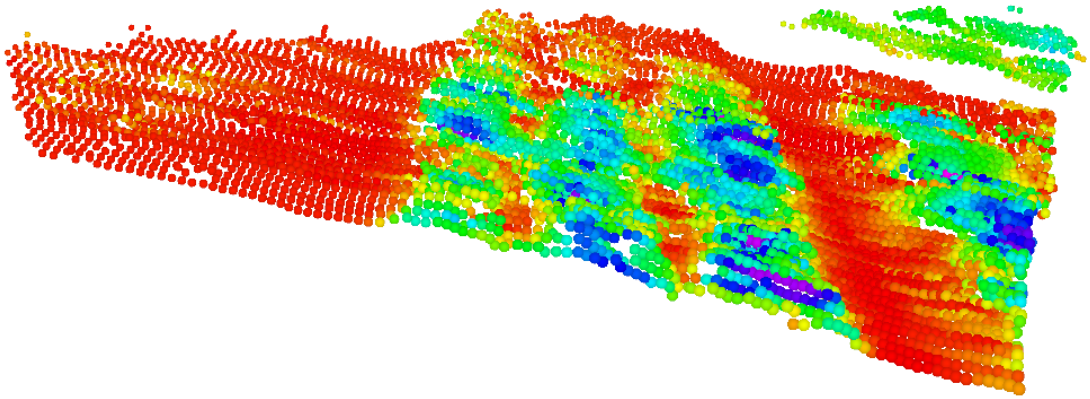


Fig. 6: Pointcloud with the color representing magnitude of intensity of points

B. Colored Point Cloud

The integrated Pointcloud upon being tested manually for its consistency and correct representation in time was primed for the next steps. The integration process increases the resolution of the pointcloud such that it can be projected onto by a image such that each point in the pointcloud could be associated with a respective RGB information from the image data. The increased resolution does come at a cost, which being that the points in the Pcl are not the exact representation of the object scanned by the sensor, since it fills the sparsity between scan lines of the raw pcl by approximating the data with the last available scan line.

So as to achieve a colorised pointcloud, the *image_geometry* package in ROS is utilized. In this package, there are C ++ and Python libraries for interpreting images geometrically . It interfaces the calibration parameters in CameraInfo messages in *sensor_msgs* package with OpenCV functions such as image rectification.

The CameraInfo message defines meta information for a camera. It should be in the same name-space as the camera in ROS. The Camera-Info depends on calibration parameters, which are fixed during camera calibration. Their values will be the same in all messages until the camera is re-calibrated.

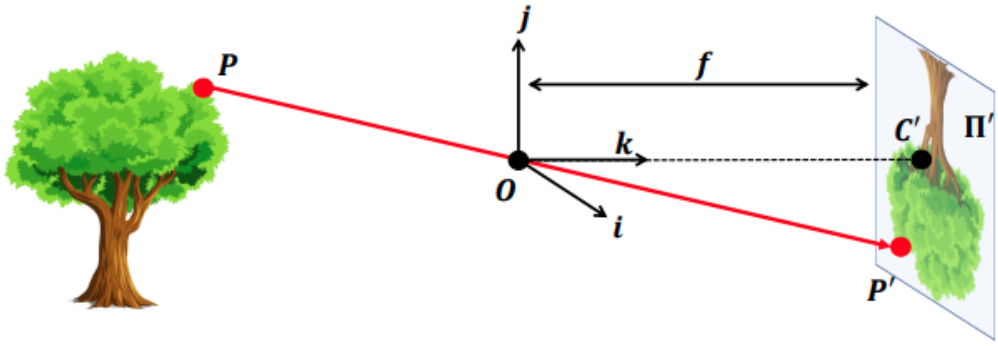


Fig. 7: A construction of the pinhole camera model [2]

The parameters in CameraInfo message are for full-resolution images. To process a region of interest directly using these parameters results in a highly complicated rectification map and requires adjustment of the projection matrix. Thus motivating the use of *image_geometry* package. To model the camera in *image_geometry* we use the Pinhole Camera Model defined in this package.

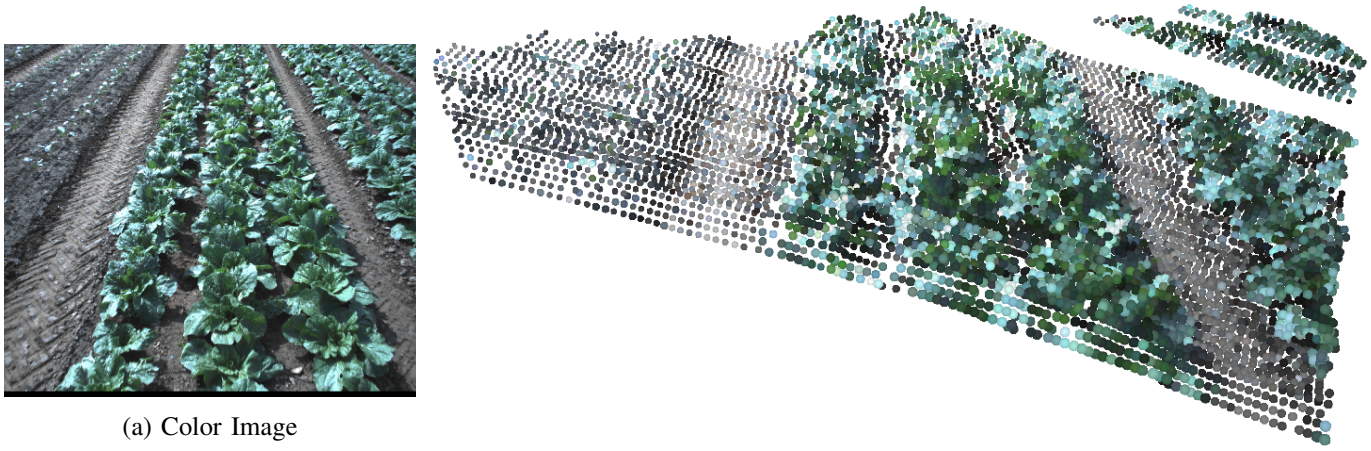
The pinhole model as depicted in Figure 7 is the basic camera model used in computer vision. Pinhole cameras allow to take photographs of objects, which usually requires long exposure times due to the small aperture. The principles behind pinhole cameras have been known, at least since the fourth century BCE. [14]

Once the model is built, the image message from ROS is converted into an OpenCV message by using CvBridge package, with OpenCV being an open source library of programming functions mainly aimed at real-time computer vision. The final step in this process is to project each 3d-point in an individual pointcloud to rectified pixel coordinates. If the projected pixel lies inside the image, then the corresponding color data of this pixel is concatenated to the point data in Pcl, thus representing not only X,Y,Z information but also RGB details of each point. For the matching to be accurate and consistent, it is essential for the calibration between the Lidar and Camera sensors to as precise as possible.

This process requires correlation not only with the image pixels, but also with the acquisition time to ensure that the image and the laser scan correspond to the same actual data. This is achieved by using the time synchronizer filter in *message_filters* package, which synchronizes incoming channels by the timestamps and outputs them in the form of a single callback that takes the same number of channels. A message filter takes in a message that may or may not be output at a later point in time. The *message_filters* package collects some common ‘filters’ in ROS.

The time synchronizer can be instructed by a ‘policy’ that determines how to synchronize the messages. There are two implemented policies for the same in this package, namely, exact time policy and approximate time policy. The exact time policy requires messages to have exactly the same timestamp in order to output, whereas, an approximate time policy uses an adaptive algorithm to match messages based on their timestamp.

Here, the approximate time policy is used since the time stamps of the images and respective pointclouds do not match in the order of milliseconds.



(a) Color Image

(b) Colorised Point-cloud

Fig. 8: Correspondence achieved through Approximate Time Policy

C. Point Features

Once the colored pointcloud is obtained, the next step is to categorize the points into either plant or non-plant data. This classification requires features of the points to be identified.

Computing the features among a set of 3D cartesian points has a big computational cost, thus the pointcloud was subsampled first through windowing process and implicitly further subsampled in the process of colorising the PCl

1) Normal and Curvature Computation: The surface normal at a particular point is a vector perpendicular to the tangent plane to the surface at that point. Normals basically define the orientation of the surface at all points and specify the local texture of the surface.

Computation of normals on Pointclouds is best handled by PCL introduced in subsubsection III-B2, the Normal Estimation class in PCL computes the least-squares plane fit for a given set of points.

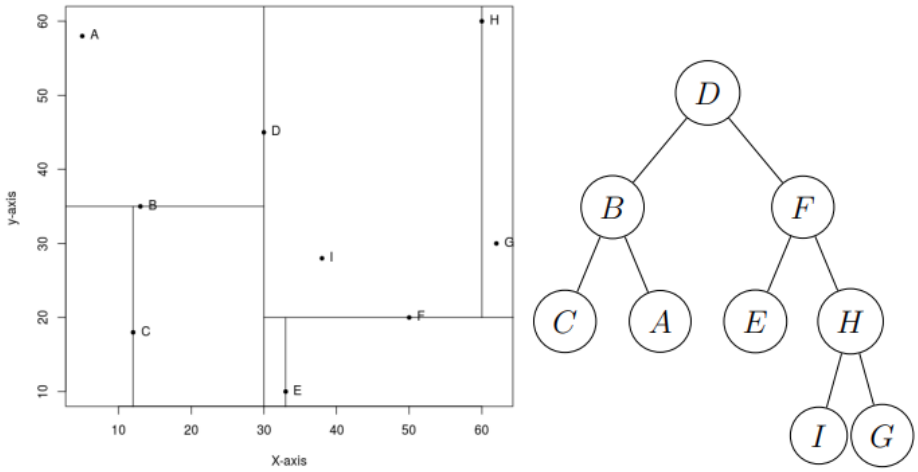


Fig. 9: K-D tree on the right, the result of subdivision of points in a two dimensional space on the left.

The least squares method is relatively easy to derive and implement. The main idea is to find the best fitting plane through a small neighborhood of the point of interest and to take a vector perpendicular to it. The best fitting plane is determined by minimizing the squares of the distances from the points in

interest to it. Given a point with coordinates $(x, y, z)^T$ lying on the plane and a set of k points in the neighborhood, least squares method outputs a normal vector $\mathbf{n} = (nx, ny, nz)^T$, fulfilling Equation 2 such that $|\mathbf{n}| = 1$

$$min \sum_{i=1}^{k} (p_i^T \mathbf{n} - d)^2 = e \tag{2}$$

The best fitting plane is given by the equation:

$$n_x x + n_y y + n_z z = d \tag{3}$$

The solution to Equation 2 is given by the smallest eigen vector of M,

$$\mathrm{M} = \frac{1}{k} \sum_{i=1}^{k} (p_i - \bar{p})((p_i - \bar{p}))^T \quad \text{where,} \qquad \bar{p} = \frac{1}{k} \sum_{i=1}^{k} p_i \tag{4}$$

This solution for the normal vector is linear on the number of neighboring points k taken into consideration.

The k- nearest neighbour search is conducted on a K-D tree fit to the points of the pointcloud. The search operation has a complexity close to O(log(n)). The Figure 9 shows an example of K-D trees.

K-D trees are a widespread technique to organize data points in a k-dimensional space. The main idea is to split the data points at one specific dimension per each level of the tree.

Curvature (σ), another important feature to determine characterstics of a surface is estimated as a relationship between the eigenvalues of covariance matrix M defined in Equation 4. The relationship is given by,

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{5}$$

where λ_x denote the eigenvalue of the covariance matrix, with λ_0 corresponding to the smallest eigen vector.

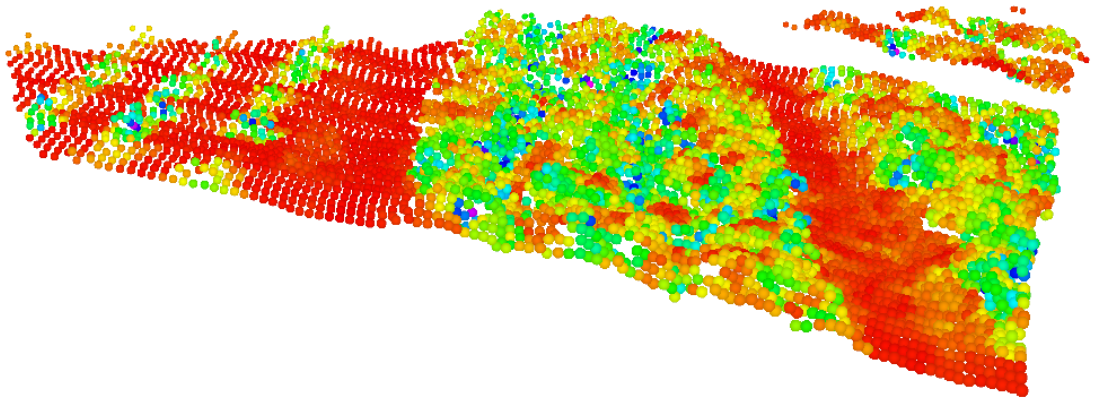


Fig. 10: Pointcloud with the color representing magnitude of curvature of points

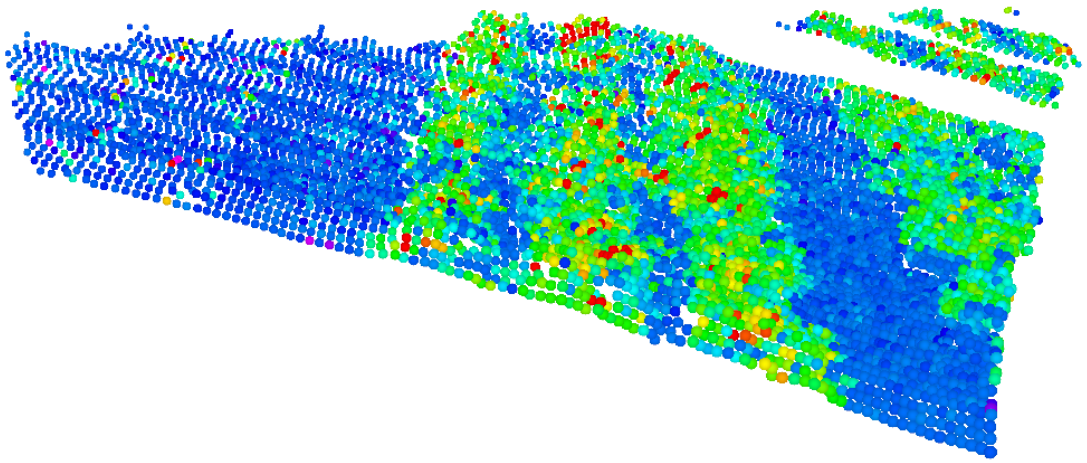


Fig. 11: Pointcloud with the color representing magnitude of TGI of points

2) RGB based features: The requirement using RGB data is to use the data in the visible spectra, represented be used to identify specifically the plants. For this specific reason, spectral indices that correspond to defining features of vegetation, such as chlorophyll are chosen.

Vegetation index is used for combining or filtering multiple spectral datasets into one value at each point. Usually the index is visualised using false-color image of a field. The vegetation index maps make it easy to find plants and at the same time, problem areas in a field. [9]

The triangular greenness index (TGI) was developed based on the area of a triangle surrounding the spectral features of chlorophyll. It is also used to estimate vegetation fraction of croplands. [6] Using the wavelengths of a typical CMOS camera and normalizing by the green signal, the TGI equation is,

$$TGI = R_{green} - 0.39R_{red} - 0.61R_{blue} \tag{6}$$

Visible Atmospheric Resistant Index (VARI) part of the method aimed to calculate the leaf area index during the growing season of plants. VARI is used to estimate the fraction of vegetation in a scene with low sensitivity to atmospheric effects. It uses the formula,

$$VARI = \frac{R_{green} - R_{red}}{R_{green} + R_{red} - R_{blue}} \tag{7}$$

where R_{xxx} is the reflectance of the object in that color. These two indices depend on only the reflectance information from the visible spectra and thus can be used directly.

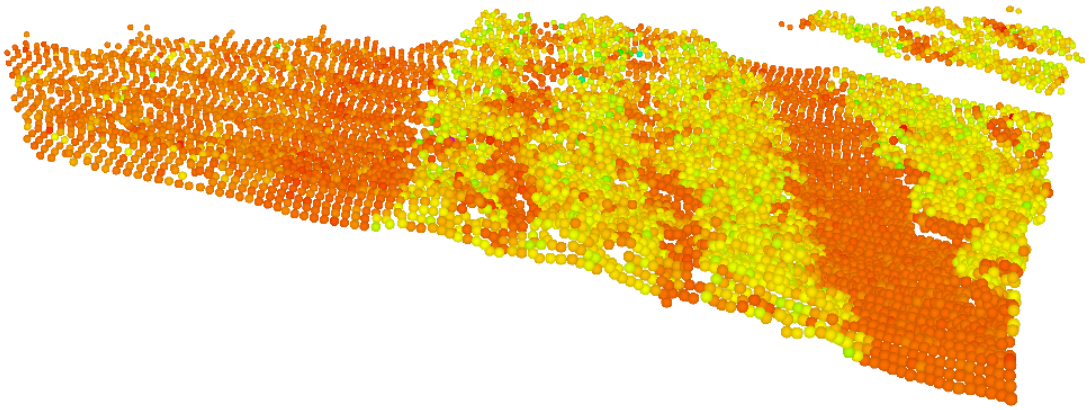


Fig. 12: Pointcloud with the color representing magnitude of VARI of points

D. Point Clustering

Classification of the points into plant and non-plant clusters requires features which were computed as described in the previous subsection IV-C. Since the data was processed from raw stage, there is no existing labeling for the points. Thus a ground truth could only be established by manual labelling. This was a motivation to undertake Unsupervised learning algorithms for achieving the clustering.

Unsupervised learning has much in common with exploratory data analysis and data mining, it's only an observation of the data. There are only input data and not an available observation or label. The data is unlabeled, hence, there is no question of evaluating the analysis but rather, restricted to understanding what can be learned from the data. These algorithms group the data from an unlabeled data set based on the underlying hidden features in the data. The Unsupervised algorithms chosen for this task are K-means algorithm and Gaussian Mixture Modelling.

The Kmeans algorithm is an iterative algorithm that tries to divide the data set into K different non-overlapping subgroups (clusters), where each data point belongs to only one group. It tries to make the data points within the cluster as similar as possible while also keeping the clusters as different as possible. The K, i.e., number of clusters is pre-defined. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid is at the minimum.

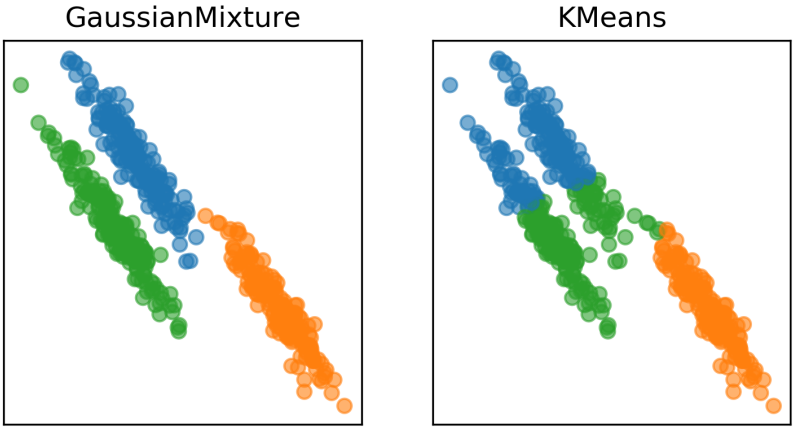


Fig. 13: Gaussian Mixture Modelling vs K-means clustering example. [1]

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions. The mixture models are a generalization of k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the gaussians. The Gaussian mixture model can be used to cluster unlabeled data in basically the same way as the k-means clustering algorithm. However, there are significant variations, Gaussian mixture models take into account the covariance of the clusters and thus can handle even oblong clusters. Gaussian mixture model also gives the edge over K-means by defining probabilities that each point belongs to its cluster.

Apart from the aforementioned methods, Supervised Learning methods can be used to classify points into different clusters. A supervised learning method learns the associations of the input with respect to the output/classes with the help of the features. Supervised learning requires training data and corresponding labels.

1) Building Ground Truth: The clustering result from the RGB-based feature combination of TGI and VARI was chosen to act as a 'ground truth' for the classification of plant points using the features Intensity and Curvature. By doing this, Supervised Learning algorithms could also be introduced for classification purposes further comparison.

In order to be more reliable, the procedure to compute the ground truth involved the combination of Gaussian Mixture Models as well as K-means clustering.

When the label of points agreed from the prediction of both the algorithms, it was passed onto the ground truth. On the other hand, when the algorithms disagreed with each other, then the probabilistic nature of gaussian mixture models was leveraged. By having a threshold on the probability, in this case, a prediction probabilty greater than '0.88', the true label was chosen to be the one as predicted by GMMs, otherwise the label passed onto the ground truth would be the output of K-means algorithm.

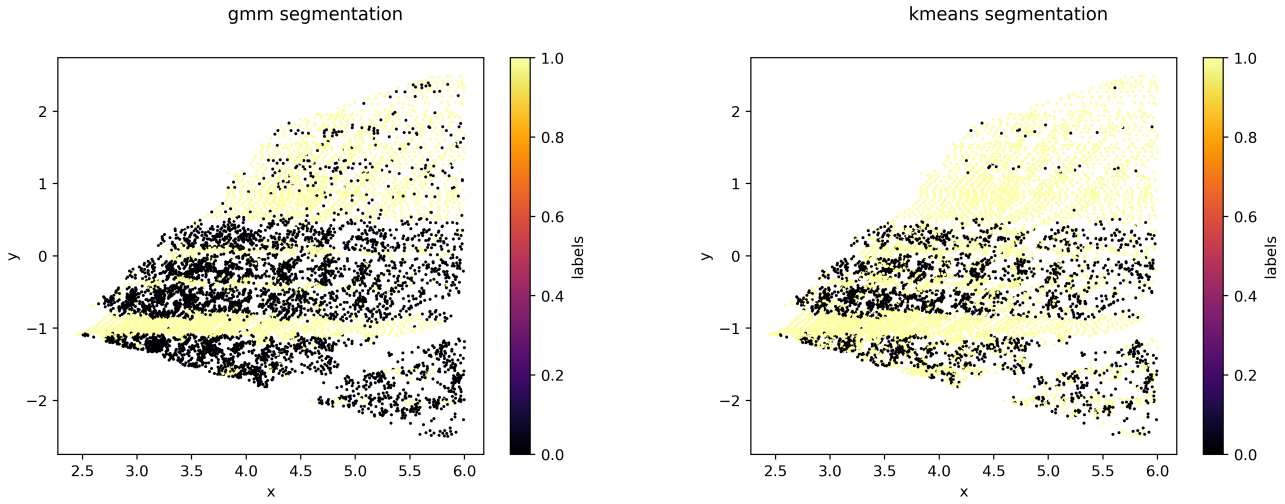
The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors

2) Plane segmentation using RANSAC: The abbreviation of 'RANdom SAmple Consensus' is RANSAC, and it is an iterative method that is used to estimate parameters of a mathematical model from a set of data containing outliers. It contains following the steps:

1) Select n random unique data points from P, where P is the Set of all data points. With n always less than number of points in P.
2) Create a mathematical model x with the selected n data points.
3) $\forall p \in$ P, Test if p fits in the model x (± some threshold d)
4) Save the quantity of matching p as quality value for current model x
5) Repeat steps 1 - 4 to find a better model x

In the classical RANSAC, the quantity of matching data points is used as quality value. On the other hand, MLESAC(Maximum Likelihood Estimation SAmple Consensus) takes the deviation of matching data points from the model into consideration. Thus, data points with higher deviation influence the quality value proportionally.

V. Results



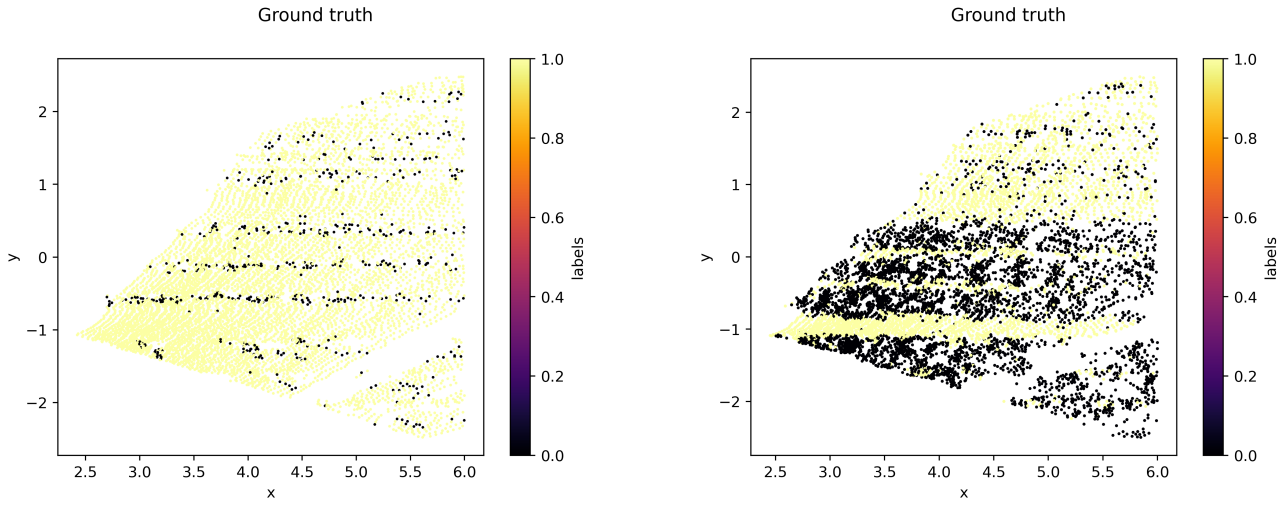
(a) Gaussian Mixture Modelling

(b) K-means clustering

Fig. 14: Clustering of Points into Plant, non-plant types using the RGB based features TGI and VARI

The outputs of the point clustering methods as detailed in subsection IV-D are presented in this section, primarily focusing on the largeplants dataset, since it contains also small plant rows in the left corner when using the same view as in the Figure 8a

A paired feature combination with, The RGB based features, i.e., Triangular Greenness Index and Variable Atmospheric Resistant Index as explained in subsubsection IV-C2 forming one combination for analysis and the Lidar sensor pointcloud based-features such as Intensity and Curvature of points were the other combination of features, while, acting as the test data.



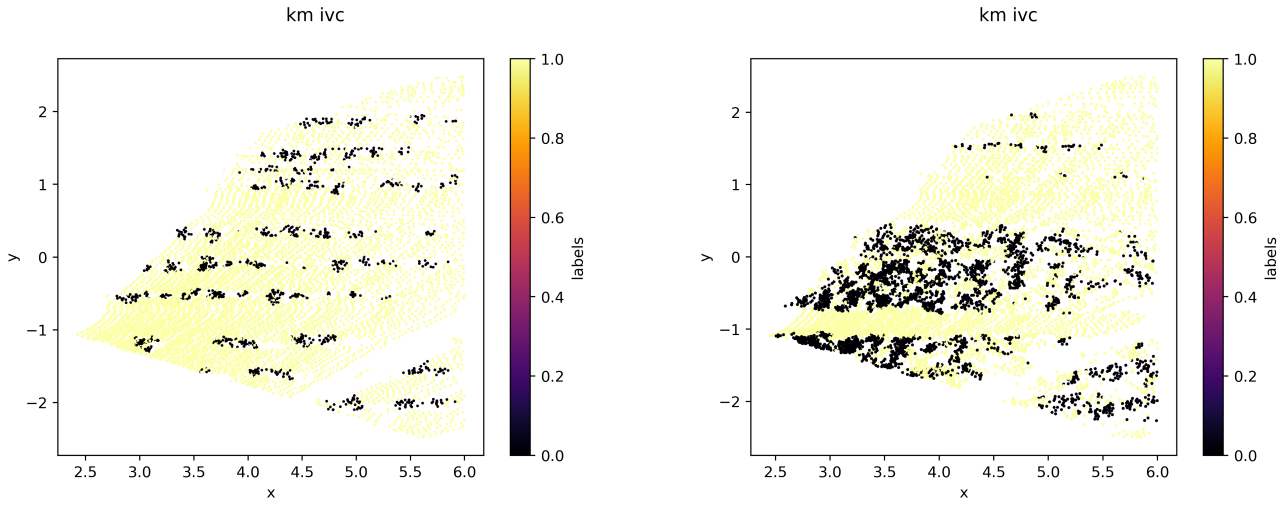
(a) Ground Truth small-plants

(b) Ground Truth large-plants

Fig. 15: Ground truth constructed using the features TGI and VARI on GMM, K-means algorithms

Figure 14 presents the results of Gaussian Mixture Modelling (GMM) and K-means clustering side-by-side. Both the algorithms require an initial specification of number of clusters.It can be observed that the clustering algorithms identify the crop rows and can differentiate between plant and non-plant points.

For the GMM algorithm, The weights, the means and the precisions are initialised by a initial kmeans run. The covariance matrix is of the shape (n components, n features, n features) with ‘n components’ i.e., number of points in this case, and ‘n features’ being number of features. Whereas, the K-means algorithm is implemented using a mini-batch method, with a batch size of ‘100’ for faster convergence [4].



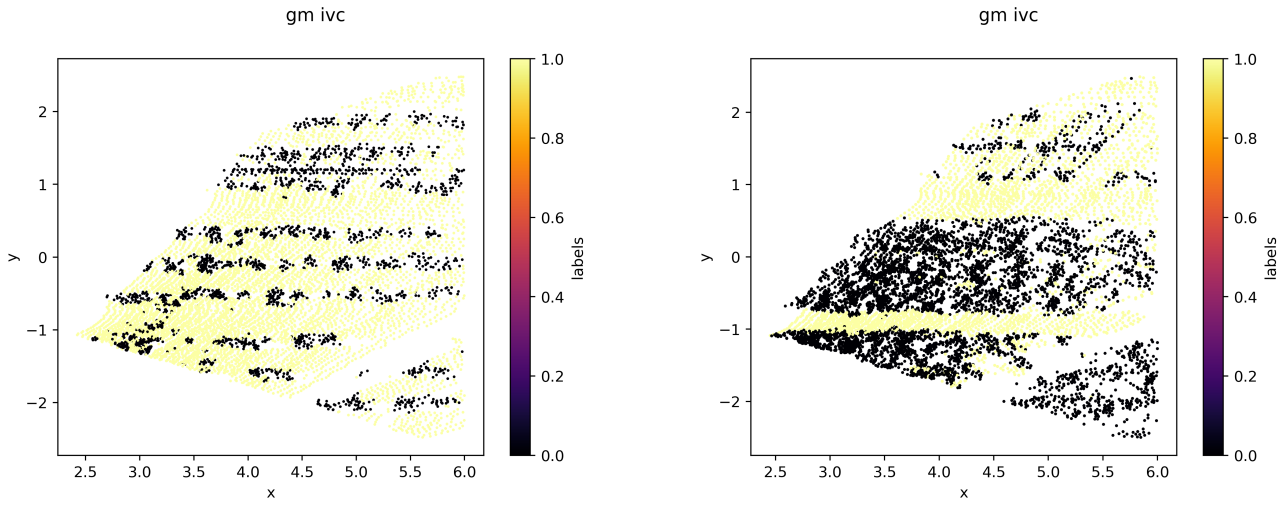
(a) small-plants

(b) large-plants

Fig. 16: Clustering of Pointcloud into Plant, non-plant points using the features Intensity and Curvature using K-means

The ground truth to be assumed in further experiments was constructed as explained in subsubsection IV-D1, in order to have higher reliability and reduced bias. The results of which can be observed in Figure 15 for both large-plants and small-plants datasets.

The next set of results in Figure 16 and Figure 17 portray the performance of unsupervised clustering algorithms using features of Pointcloud data, namely, Intensity and surface curvature at each of the points.



(a) small-plants (b) large-plants

Fig. 17: Clustering of Pointcloud into Plant, non-plant points using the features Intensity and Curvature using Gaussian Mixture Modelling

Supervised learning algorithms, as described in require to be trained with data with corresponding labels, for this purpose the true labels obtained from ground truth are utilized. A support vector classfier (SVC) trained with the Radial Basis Function (RBF) kernel is deployed with a regularization parameter of '1.0'. The performance metrics of this model on the datac can be seen in Figure 18

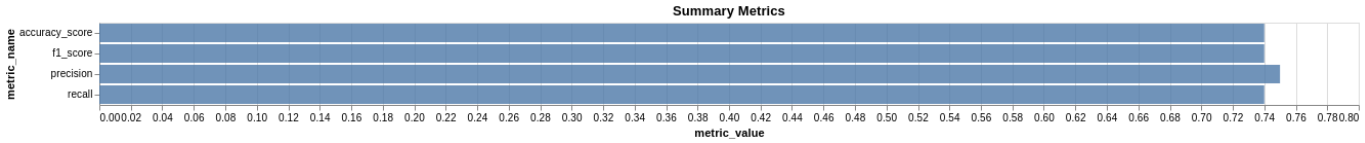


Fig. 18: Metrics of the Support Vector Classifier on the Pointcloud test-data with features Intensity and Curvature

The Matthews correlation coefficient(MCC) is used as an index to measure the quality of binary and multi-class classification in machine learning. It takes into account, the true and false positives and negatives. It can be used even if the size of the classes differs greatly.(+1 represents a perfect prediction, 0 an average random prediction and -1 and inverse prediction)

Table I summarizes the performace of models to cluster the Pointcloud data into with features being Intensity and Curvature.

TABLE I: Performance Metrics of the models with features as Intensity and Curvature

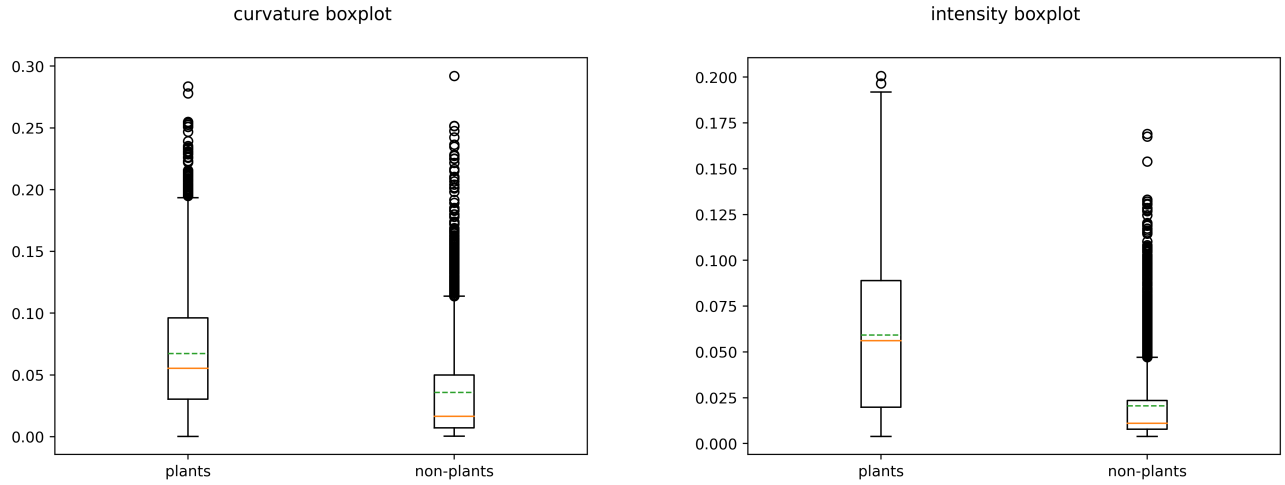
Model	Large-plants		Small-plants	
-	Accuracy	MCC	Accuracy	MCC
K-means	0.659	0.374	0.827	0.374
GMM	0.752	0.477	0.718	0.131
SVC	0.735	0.472	0.921	0.0

A. Analysis of segmented Pointcloud

The Pointcloud segmentation assumed as ground truth is analysed in this subsection. This analysis can used for further processing of either plant data or non-plant data separately. The aim is to provide insights

into how the data is distributed, with respect to the features.

1) Boxplot analysis: Figure 19 describes the information on the variability or dispersion of the data in Intensity and Curvature channels and shows the significant differences in values between plant and non-plant points respectively.



(a) Curvature

(b) Intensity

Fig. 19: Boxplots of pointcloud features with respect to Plant, non-plant data. Here the orange line describes median, while green line denotes mean

2) T-test analysis: The T-test is a measure of how significant the differences between groups are; In other words, it describes if those differences (measured in means) could have happened by chance. The T-score is the ratio between the differences between the two groups and the differences within the group. The larger the t score, the greater the difference between the groups. The smaller the t score, the more similarities between groups. A p-value is the probability that the results from your sample data occurred by chance. A lower p-value indicates the results did not occur by chance.

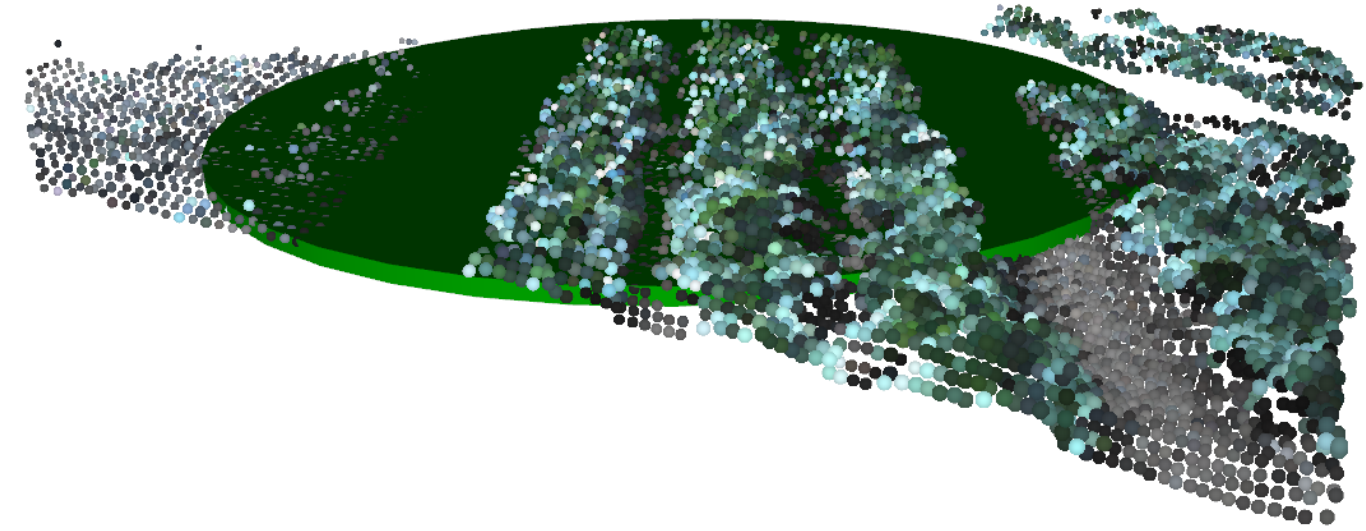
Independent T-tests have been conducted for both Intensity and Curvature features, with respect to plant and non-plant points to understand the differences between these groups. The results of these tests are shown in Table II

TABLE II: Performance Metrics of the models with features as Intensity and Curvature

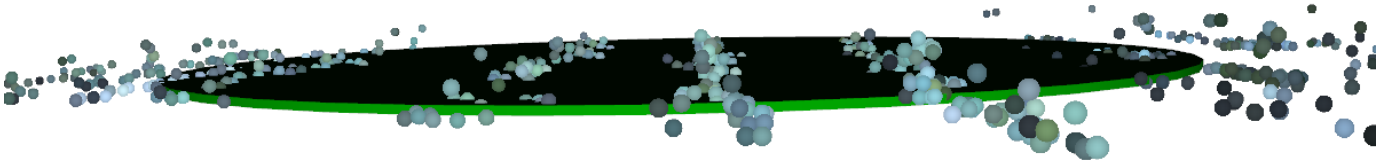
Feature	Large-plants		Small-plants	
-	T-score	p-value	T-score	p-value
Intensity	51.608	0.0	1.105	0.269
Curvature	29.925	2.064e-185	9.236	4.495e-19

3) Ground Plane segmentation: A separation is neede between points that are on the ground plane and the rest of the points. To simplify this task, the ground is assumed as an even plane. by ignoring small elevations by adding a threshold parameter. A MLESAC algorithm as described in subsubsection IV-D2 is deployed on the non-plant points as clustered previously, in order to obtain an estimate of the ground plane of the farm. The results of which are displayed in Figure 20 and Figure 21 for the large-plants and small-plants pointclouds respectively.

The Sample Consensus segmentor used was the SAC Segmentation From Normals method in PCL, which uses the normal information at each point to further refine the ground plane segmentation. This method also returns the coefficients of the segmented plane which can be used to determine the elevation of points from the ground plane. This can a feature, especially when segmenting for individual plants from



(a) Full pointcloud



(b) Plant points

Fig. 20: Ground plane(Green marker) with thickness that of specified threshold for the large-plants dataset

the plants cluster by calculating the elevation of each point wrt. estimated ground plane. The elevation was calculated for the largeplants, with promising results as shown in Figure 22

VI. Conclusion and Outlook

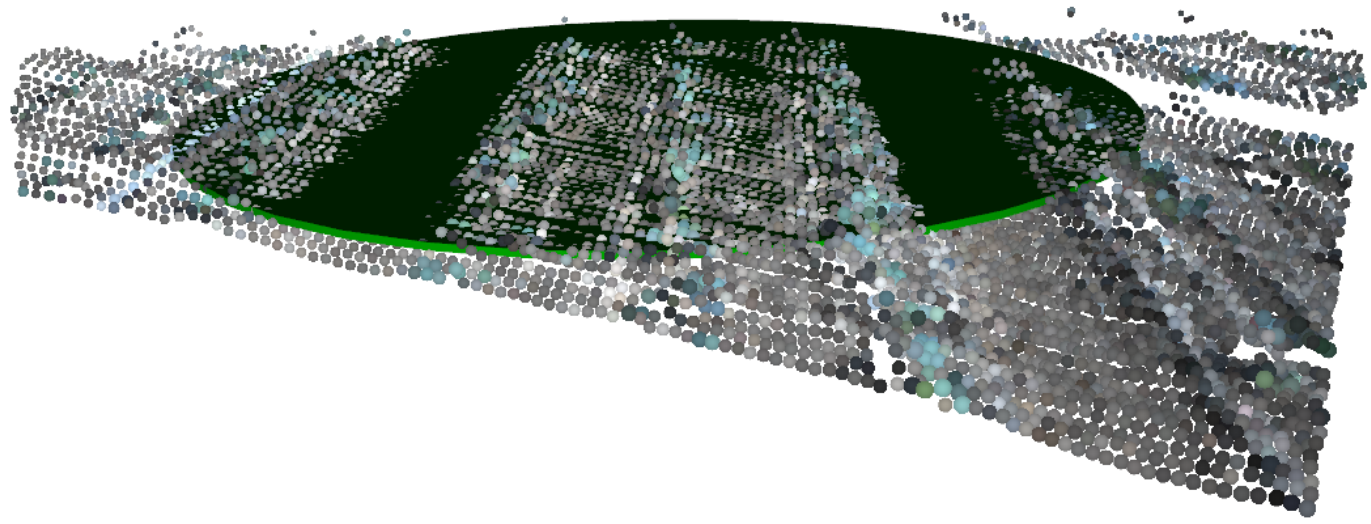
The project develops an approach using ROS and PCL to segment pointclouds between plant and non-plant data given Image and Pointcloud data. One can observe that the algorithms can identify the plant rows fairly accurately, further processing is required to identify individual plants.

The significant sparseness of the raw scans due to the nature of the lidar data means that accurate surface normal calculations are less probable, and hence, all the features with respect to this are only estimations.

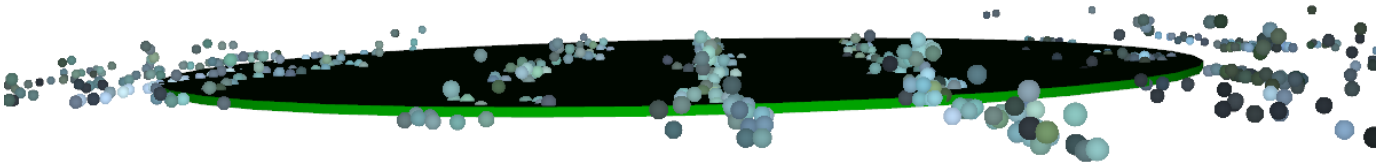
The approach designed in this implementation could also be extended to identify individual plant clusters with some tuning. This depends also on the nature of the data, where the plant foliage must be sparse as required.

Algorithms like DBSCAN also have been tested for clustering purposes with under-performing results. This could be enhanced by using a parameter search to tune the algorithm as necessary. For further research purposes, FPFH, fast point feature histogram descriptors have been computed for key points in the plant cluster. Point Feature Histograms (PFH) descriptor [12], is a feature that encodes a point's k-neighbourhood geometrical properties based on normal and curvature around the point.

The next step in the pipeline would be the implementation of estimating individual plant centers using the extensions and feature enhancement as mentioned above, which in turn would deem individual plants on the field to identified as landmarks for increasing accuracy of operations such as localisation or mapping. Since an agricultural field mostly has no distinct features at this level, this could be a boost to the progress in autonomous agri-robots.



(a) Full pointcloud



(b) Plant points

Fig. 21: Ground plane(Green marker) with thickness that of specified threshold for the small-plants dataset

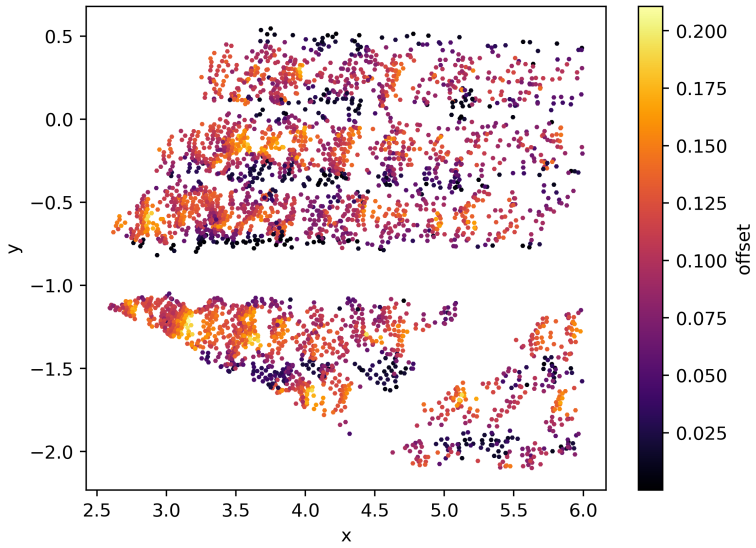


Fig. 22: Elevation of plant points with respect to Ground plane

REFERENCES

[1] *COMS W4995 Applied Machine Learning*.
[2] *CS231A: Computer Vision, From 3D Reconstruction to Recognition*.
[3] Yahya Alshawabkeh. Linear feature extraction from point cloud using color information. *Heritage Science*, 8, 2020.
[4] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier

Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[5] Nived Chebrolu, Philipp Lottes, Alexander Schaefer, Wera Winterhalter, Wolfram Burgard, and Cyrill Stachniss. Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields. *The International Journal of Robotics Research*, 36(10):1045–1052, 2017.

[6] E. Raymond Hunt, Paul C. Doraiswamy, James E. McMurtrey, Craig S.T. Daughtry, Eileen M. Perry, and Bakhyt Akhmedov. A visible band index for remote sensing leaf chlorophyll content at the canopy scale. *International Journal of Applied Earth Observation and Geoinformation*, 21:103 – 112, 2013.

[7] Hui Li, Liping Di, Xianfeng Huang, and D. Li. Laser intensity used in classification of lidar point cloud data. volume 2, pages 1140–1143, 01 2008.

[8] Aaron Martinez and Enrique Fernández. *Learning ROS for Robotics Programming*. Packt Publishing, September 2013.

[9] Tom Mckinnon. Comparing rgb-based vegetation indices with ndvi for drone based agricultural sensing. 2017.

[10] Francesco Nex and Fulvio Rinaudo. Photogrammetric and lidar integration for the cultural heritage metric surveys. volume 38, 06 2010.

[11] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. volume 3, 01 2009.

[12] Radu Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. pages 3212 – 3217, 06 2009.

[13] Radu Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). 05 2011.

[14] Peter Sturm. *Pinhole Camera Model*, pages 610–613. Springer US, Boston, MA, 2014.

[15] F. Verdoja, D. Thomas, and A. Sugimoto. Fast 3d point cloud segmentation using supervoxels with geometry and color for 3d scene understanding. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1285–1290, 2017.

[16] W. Winterhalter, F. V. Fleckenstein, C. Dornhege, and W. Burgard. Crop row detection on tiny plants with the pattern hough transform. *IEEE Robotics and Automation Letters*, 3(4):3394–3401, 2018.