3D Apple Detection from Large Point Clouds Using Deep Learning

Berkay Arpaci

Abstract

In recent years, with the increasing number of studies about autonomous vehicles, LiDAR sensor technology and 3D deep learning object detection models have kept improving steadily. As a result of these improvements, these models have started to be implemented in other fields, such as agriculture. One important potential implementation of 3D object detection is to determine various quantities related to fruits such as their number, position and size which can be utilized to optimize the orchard management. In this work, performance of a selected 3D deep learning object detection model for detection of apples in an orchard is examined to achieve in-field automatic fruit counting and localization. Additionally, the results of different types of data are compared to determine the most suitable data type for these tasks. Among many 3D object detection models, PointRCNN is selected as it supports point clouds having homogeneous characteristics in all 3 axes unlike other models using bird-eye-view or similar methods treating the height differently. Point clouds obtained from two different methods and a combination of the data obtained from them are used to see the effect of positional accuracy, cloud density and the effect of additional parameters such as color and reflectance. A LiDAR dataset was obtained by a mobile LiDAR sensor combined with a positioning sensor and merged to form a single point cloud for 11 trees. Photogrammetry dataset was obtained from a set of color images using Structure from Motion (SfM) algorithm. The experimental results in this thesis show that photogrammetry data with color information yielded scores much higher than the scores obtained from the LiDAR data. The average AP score for photogrammetry and LiDAR data is 67.62 % and 26.31 % respectively.

Index Terms

3D, point cloud, photogrammetry, LiDAR, object detection, deep learning, computer vision.

I. Introduction

PEOPLE'S search for devices to make their lives easier, which has been going on for ages, has started to be met with the development of technology. In some areas, technology is intertwined with human life, while in other areas it has not reached the desired levels due to the inadequacy of existing technology or cost disadvantages. One of these areas is agriculture, where a large portion of work is done by manpower. The optimization of technologies used in agriculture can have a significant impact on society, directly or indirectly. The most important of these are the price and quality of agricultural products. The development of artificial intelligence technology has made the usage of these technologies possible, but not without challenges.

For example, to pave the way for data mining in agricultural applications, it is necessary to collect a significant amount of information about agricultural products at short intervals. This is a major problem, especially for greenhouses occupying a large area. Using manpower is not ideal in terms of both cost and time savings. Likewise, the investment and maintenance costs of sensors placed at frequent intervals make their practical use impossible. Since the development of crops takes place over a period of time, the most logical option is the periodic scanning of a large area with the help of a mobile sensor. However, for this to be feasible, sensor data must first be converted into useful data which can be used in optimization with satisfactory efficiency. Currently, this task can be best performed by deep learning networks. This is because deep learning has achieved remarkable success in many areas such as natural language processing (NLP), instance segmentation, and image generation.

In this project, the aim is to determine the amount of apples using point clouds and to assist harvesting them automatically. 3D data are obtained in two main formats: The first one is photogrammetry data. The most significant property of the photogrammetry data is containing color information. Another approach is using a LiDAR sensor with a positioning system. Here, unlike photogrammetry data, reflectance information is available instead of color information.

LiDAR technology, with the development of autonomous vehicles, has reduced the cost of sensors, increased the performance of models and is being widely used today. Although there has been an increase in performance, the models are generally optimized for autonomous driving scenarios and there are not many such studies in the agricultural field therefore the results obtained from these studies have been unsatisfactory. However, studies in this area are expected to increase over time.

The contents of this paper are organized as follows: Section II briefly describes the previous work done in 3D object detection field and other related fields. Section III consists of three subsections where the data used, the data preprocessing method and the methodology of the project are described in detail. The experiment process is included in Section IV. In section V, the results of each experiment and the final results are presented. In Section VI, the results obtained from different datasets are evaluated and compared, and in the final section, conclusions regarding the project are drawn and future work is discussed.

Author: Berkay Arpaci, berkayarpaci@icloud.com
Advisor 1: Javier Ruiz-Hidalgo, Technical University of Catalonia (UPC)
Advisor 2 : Jordi Gené-Mola, IRTA - Institute of Agrifood Research and Technology
Thesis dissertation submitted: SEPTEMBER 2023

II. State of the art

A. Implementations

With the development of autonomous vehicle technology and the integration of LiDAR sensors into autonomous vehicles, studies on 3D object detection, segmentation, and pose estimation have increased. Although these tasks can be accomplished by using images obtained from cameras and image processing methods, LiDAR sensors have become almost indispensable for many autonomous vehicle manufacturers since images do not directly carry the three-dimensional coordinate information provided by LiDAR. Although less preferred by real-time detection tasks due to complexity of the processing algorithm, photogrammetry data is a better alternative to LiDAR data for generation of dense point clouds due to easy matching using additional color information. Some of its applications include 3D scanning and producing 3D models of tourist attractions such as famous historical buildings. Photogrammetry data is also cheaper to obtain as images can be captured by any camera sensor from any position unlike dense LiDAR data which requires expensive sensors and a special tripod or positioning system to determine the sensor pose.

Early attempts for 3D object detection were made with detectors using hand-crafted features. Among these, the detector proposed by P. Silva et al [5] stands out. However, as with many tasks, the performance and generalizability of such methods were below that of deep learning methods. One of the pioneering methods using deep learning was PointNet. [22] In this method, the points in the point set are individually represented in high-dimensional space by passing them through many parallel layers and subjecting them to coordinate changes. In this way, the input size of the network can take the desired value. Such methods are called point-based. Later, voxel-based and graph-based methods were also proposed. To be used in situations where accuracy is more important than speed, two-stage networks have emerged, as in image processing.

B. Point Cloud Processing Methods

1) Point-based Models: Such models [22], [33], [23] can achieve high scores because they do not suffer from information loss due to voxelization, but they may not be fast enough for every application due to the large amount of data they process. PointGroup [14] architecture performs clustering by utilizing the empty volumes between objects. It has a bottom-up architecture and two branches that predict offsets from the object center with semantics. During the prediction of instances, clustering is done by using old and new coordinates together. In SoftGroup [31], which is a continuation of this paper, soft prediction is used to overcome the problem of false positives in group-based models, where points can belong to more than one class. In the second stage, points that are not relevant for detection are removed. S. Liu et al. [18] aimed to prevent the loss of local and global information by using group-in-group transformers. They also used a technique called radius feature extraction (RFA) to utilize point density information.

2) Voxel-based Models: It is used in many models because of its speed [35], [4], [20]. The speed increase is achieved by using a method called voxelization, which locally reduces the space into voxels representing the prism-shaped volume around it, reducing the amount of input without significant loss of information. Therefore, it is more efficient than point-based methods. The resulting voxels are subjected to full 3D or sparse convolution [11] depending on the point density and speed requirements. In the transformer-based model called VoxSet [13], two cross-attention modules in latent space are used instead of self-attention. As an alternative to three-dimensional Voxelization, the PointPillars [16] method outputs feature vectors with two-dimensional coordinates. These vectors are called pillars. It is often used as a backbone due to its high performance and the ability to apply 2D convolution directly on the output.

3) Graph Models: It is based on the application of graph neural networks (GNNs) by transforming the point cloud into a graph. DeepGCN [17], a semantic segmentation model based on applying graph convolutions to GNNs similar to standard CNNs, has been a pioneer for other models in this field. It was able to solve the vanishing gradient problem with the help of dilated convolutions and residual connections. Another popular model, PointGNN [28], combines boxes to reduce the number of redundant vertex detections and provides translation invariance with an auto-registration system.

C. Number of Stages

1) Two Stage: As in the two-dimensional versions, these models consist of two parts: region proposal network (RPN) and RCNN (region-CNN). In the first part, a large number of bounding box proposals are generated by anchor-box or center-based heads, while in the second part, the number of bounding boxes is reduced by improving the predictions for each proposal.

The CenterPoint [33] architecture uses a heat map-based method with peak detection to determine the center of the object and the average velocity calculated by the change of the center between frames to solve the problems of anchor-based approaches where the angled bounding box does not sufficiently match the perpendicular anchors. This part is used in other architectures such as PV-RCNN++ [26]as an alternative to anchor-based RPN. For the second stage, RoI pooling, features corresponding to the center of the object and the midpoints of the bounding box edges are taken from the feature map from a bird-eye-view.

PV-RCNN++ is a slightly better and much faster, improved version of the PV-RCNN [25] architecture which combines the best features of point-based and voxel-based methods. In addition to voxel-based RPN, key points are obtained from the raw point cloud by sampling. An important bottleneck of PV-RCNN has been overcome by switching from further point sampling

(FPS) to sectorized proposal-centric sampling (SPC). In SPC, before performing FPS, the point cloud is divided at equal angles from the center of the scene and only points within a certain distance from the center of the region proposals are used. At the step called set abstraction (SA), feature vectors are created using the Vector Pool Aggregation (VPA) module using the regions of different layers of the backbone, 2D feature maps, and raw point cloud within a certain volume around the key points. Bilinear interpolation is used for pooling in 2D feature maps. Several different sizes are used for pooling volume. In PV-RCNN, this part was done using vector set abstraction (VSA), a simpler feature extractor. In the second stage, called RoI grid pooling, a grid is placed on the region proposal and the final feature vector is created with VPA around the grid points. Other popular two-stage models include Pyramid RCNN [19] and Voxel RCNN [4].

2) Single Stage: These models are lighter and more efficient than their two-length versions. They are used in applications requiring speed. The 3DSSD model [32] is a general-purpose, lightweight model that does not use anchors. As in two-stage models, fusion sampling is applied to use background points in detection. C. He et al. [12] used a parallel network that translates features back to points, allowing direct use of coordinate information. Other popular one-stage models include Voxelnet [35] and Pointpillars [16].

D. Fruit Detection

Due to the difficulty of fruit detection, harvesting robots have been produced but are not yet widely used. J. Gene-Mola [7] proposed to project Mask RCNN detections to point clouds generated by structure from motion (SfM) for apple segmentation. This helped with the detection of occlusions. P. Kursters et al. [15] used PointNet [22] on grape vines and showed that color-coordinate combination provides a significant advantage over color or coordinate alone. Y. Chen et al. [3] proposed a semantic segmentation architecture for large-scale crops based on a lightweight network called RandLA-Net. LFPNet [34] is another lightweight classification and segmentation network that can process large-scale point clouds. To improve fruit detections and reduce instability in detections, they used local and global features together and applied spatial pyramid pooling (SPP).

III. Method & Dataset

The datasets are obtained from 11 apple trees in an orchard. Datasets are provided by ”University of Lleida” and ”IRTA-Institute of Agrifood Research and Technology”. We worked with two different main datasets, these are photogrammetry and Lidar. Photogrammetry dataset consists of an RGB point cloud, which contains color information as well as position information. This data consists of 9.769.010 points, 11 trees and 958 apples. A program called CloudCompare [1] was used to analyze this data. Photogrammetry data is shown in Fig. 1.

Fig. 1: Apple Trees Photogrammetry Point Cloud

Photogrammetry data was collected by capturing images while moving along both sides of the orchard. Images are captured with some overlap, from a certain height, turning the camera around its axis from top to bottom. A 3D point cloud was generated using the Structure from Motion (SfM) algorithm. This algorithm can convert pixels in the image into 3D points in the point cloud by using photographs of the same objects taken from different angles. The algorithm extracts keypoints from images using a keypoint extractor such as SURF and matches them using a matching algorithm such as RANSAC. Then the

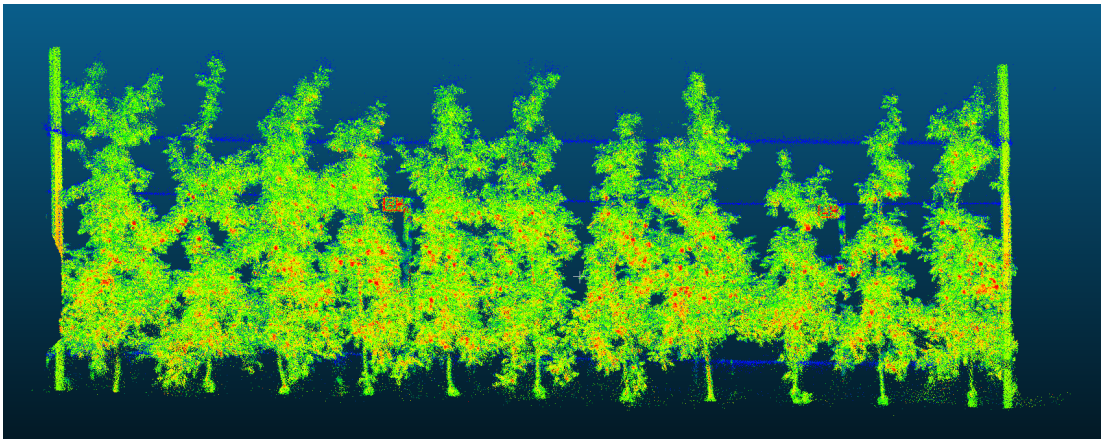


Fig. 2: Apple Trees LiDAR Point CLoud

fundamental matrix is calculated between the cameras and the points can be triangulated to obtain their coordinates in 3D space. This data set was obtained in a similar way described in the article Fuji-SfM dataset [10].

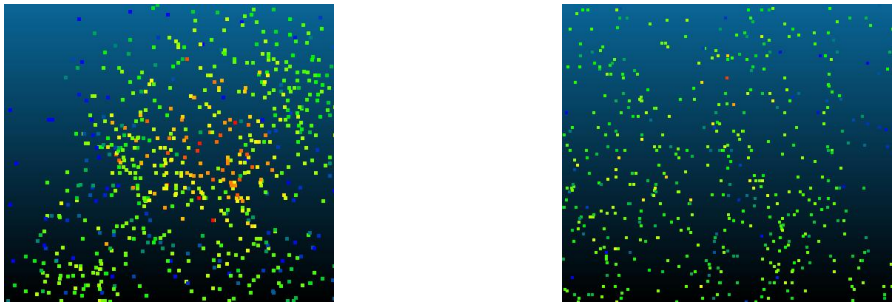
LiDAR is an acronym for ”light detection and ranging”[21] or ”laser imaging, detection, and ranging”[30]. It is a method for determining the distances of points within a 2D or 3D angular range by targeting an object or a surface with a laser and measuring the time for the reflected light to return to the receiver. The LiDAR point cloud contains the position and reflectance (x, y, z, reflectance) information [2]. This data consists of 2.067.362 points, 11 trees and 958 apples. CloudCompare program was used to analyze it. LiDAR point cloud is shown in Fig. 2 below with reflectance value go from blue to green to yellow to red as they go higher.

LiDAR data was collected as the scanning system traveled at 2.5 km/sa in the direction of the tree line with a LiDAR sensor placed perpendicular to the direction of travel at a certain height. The scanning plane of the lidar was chosen perpendicular to the plantation direction. Due to the usage of 3D LiDAR, measurements from multiple angles are available. Since LiDAR sensor and the positioning sensor recorded data synchronously, all the data collected from the tree row could be combined into a single file. This data set, is obtained in a similar way as described in the article Lfuji-air dataset [8].

Photogrammetry has a richer information content than LiDAR because it contains color and precise location information. It also contains more points than LiDAR. Therefore, the shape and size of apples are closer to real apples, making detection task easier. Due to the way LiDAR data is collected, the apples whose height are far from the measurement height lack high reflectance points which are essential in detecting them. At the same time, due to the narrow scanning angle in the plantation direction, more problems may be encountered in areas with occlusion in this dataset compared to photogrammetry dataset. Although it is difficult to detect unripe apples with color information in photogrammetry, they are still detectable due to their shape. They are mostly occluded and smaller than red apples as well. Therefore, they are hardly detectable in LiDAR point clouds as their reflectance is similar to background elements, branches and leaves. The appearances of red and brown apples in photogrammetry shown in Fig. 3 and LiDAR point clouds are given in Fig. 4. Brown apples are harder to detect due to limited number of points especially in LiDAR data. Both their reflectance and red channel values are lower. Other color channels are higher as well making color a less distinguishing feature for these apples. Furthermore, LiDAR sensor uses laser rays to collect data. However, the reflectance data depends on the angle of incidence of each ray as it is calculated from only the portion of the ray which is reflected at a right angle. The remainder of the incident rays are scattered or reflected in other directions. Therefore, a convex reflectance distribution can be seen on apple surfaces where high reflectance points are located at the center and diminishes towards the edges of the apples enough to become indistinguishable from the leaves and branches.

(a) Photogrammetry Red (b) Photogrammetry Brown

Fig. 3: Brown apples have lower number of points with less distinguishing colors.



(a) LiDAR Red (b) LiDAR Brown

Fig. 4: Brown apples are almost indistinguishable from the background as they have less number of points and has lower reflectance values than red apples.

In this project, a total of 4 datasets were used, 2 main datasets that were used in this experiment and 2 more datasets produced by removing or joining parts of their information:

- Photogrammetry: This dataset contains x_p, y_p, z_p, r ,g ,b values for each point in the point cloud as mentioned above.
- Photogrammetry coordinates: This dataset contains only x_p, y_p, and z_p channels of the photogrammetry dataset.
- Photogrammetry coordinates with reflectance: This dataset was created by adding reflectance data from LiDAR point cloud to the x_p, y_p, and z_p channels of photogrammetry point cloud by using nearest neighbors for assignment. The point cloud for a tree produced by this dataset is shown in Fig. 5.
- LiDAR: This dataset contains x_l, y_l, z_l, and reflectance values for each point in the point cloud as described above.

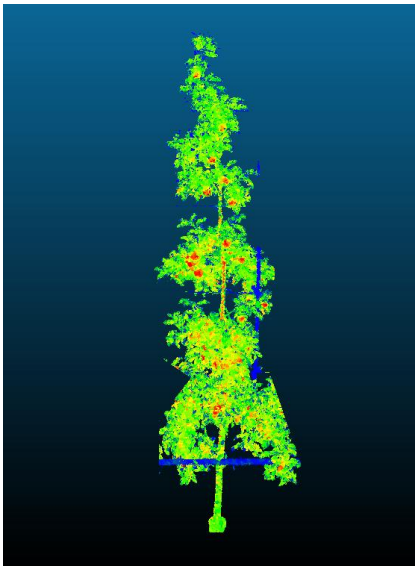
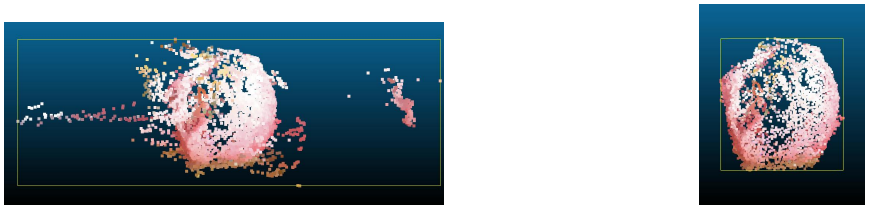


Fig. 5: Photogrammetry with Reflectance Tree Point Cloud

A. Labeling

The ground truth bounding boxes produced for the photogrammetry dataset were obtained in two stages. The first stage was not performed in this thesis. Although both datasets were collected within a small time period, some of the apples appear in different locations in the photogrammetry and LiDAR datasets due to the changes within this time period, construction of LiDAR data or errors in the allignment of the two datasets. Therefore, labels of the the LiDAR dataset was processed for one more stage to match their positions with the LiDAR point cloud more accurately.

In the first stage, the segmentation algorithm given in [9] was used and bounding box center coordinates and dimensions were calculated from the segmented points by taking the average and differences of maximum and minimum points in each of the coordinate axes. This algorithm relies on 2D instance segmentation with SfM to utilize 2D segmentation methods on 3D data. First, instance segmentation was applied on the images used to generate the photogrammetry dataset utilizing the difference between the colors of apples, which are predominantly red and leaves which are mostly green. Then, apple instances



(a) Bounding Box Prior to Correction (b) Bounding Box After Correction

Fig. 6: The removal of misplaced and wrongly segmented points can change the bounding box size and center significantly

were matched in different images and the segmented images were projected to the point cloud. Finally, apples with too few points were deleted to prevent erroneous detections.

At the end of the first stage, segmented instances contain some apples which are very difficult to detect even though they do not meet the deletion criteria such as apple groups detected as a single instance, apples containing points which are incorrectly segmented as apples, and points with incorrect positions. Since these points will be used in the creation of the bounding boxes to be used in the detection, even a single point away from the apple can greatly affect the size of the bounding box and the location of its center. The result of incorrect reconstruction and segmentation can be seen in Fig. 6a. For this reason, in the second stage, the bounding boxes were corrected by manually deleting the incorrectly segmented points. Finally, in the last stage the bounding boxes were manually shifted to match the locations having high reflectance values and point density in LiDAR point cloud. This process is prone to errors as apples are difficult to distinguish even by eye in the LiDAR dataset. Especially for brown, small and occluded apples, there are too few points with different reflectance values to distinguish them from branches and leaves.

B. Preprocessing

First of all, the photogrammetry data was shifted to a position closer to the origin to make it easier for CloudCompare to read and to reduce file sizes. The LiDAR point cloud was then segmented into trees using CloudCompare's segmentation tool, taking care not to split the apples. Since the trees were intertwined in some parts, the points of some trees were saved in the file of neighboring trees. Meanwhile, the apples within the boundaries of the trees were separated into folders. Photogrammetry data was also split into trees following the same boundaries.

The trees were shifted to the origin for the data augmentation to work properly, to facilitate training, and to avoid errors in case of switching to a voxel-based model. The center coordinates and dimensions of the orchard bounding box were used to keep the distance of the soil surface and the trunk to the sensors constant. However, since the width of the trees varied, the trees were centered in the direction of planting.

After labeling was completed, following the instructions created by OpenPCDet for custom datasets, ground truth, and point cloud files were created from data files with Python scripts. Necessary changes were made in the dataset configuration template. Finally, a Python script was prepared to assign the reflectance data of the nearest LiDAR point to the points in the photogrammetry point cloud.

C. Model

Several two-stage models were considered for 3D apple detection. These were PointRCNN [27], PV-RCNN [25], PV-RCNN++ [26], and CenterPoint [33]. PointRCNN was chosen among these since it is a general purpose 3D detection network unlike others specialized for the driving datasets by utilizing bird-eye-view based strategies as the objects in driving datasets are standing on the ground. The architecture of the model is given in Fig.7.

PointRCNN is a state-of-the-art point-based two-stage 3D detection network. First stage consists of two parts: pointwise feature generation via PointNet++ [23] and simultaneous foreground point segmentation and bounding box generation. Bounding boxes are generated for each foreground point. The second part is performed by two task specific heads. In the second stage, feature pooling is performed for each bounding box and transformed to the canonical coordinate system for better performance in the refinement stage. During pooling raw point cloud coordinates, point features and foreground mask is used as input features. The bounding box can be extended by a margin called context width to allow background point features to be included in the pooling process as well. Canonical transformation is learnable and aligns all the objects such that every face of the objects has the same normal vector. For example, the length of the car lies along x-axis, the width lies along y-axis and the height lies along z-axis. Finally, the pooled features are concatenated with point features obtained at the end of the backbone and fed into the refinement network.

Some of the details of implementation can be seen from the configuration file in the given repository. Some of the points are sampled from the raw point cloud before the backbone. At the backbone, the points are sampled again to obtain a number of sample sets with different sample sizes. For each set, two radii are used for set abstraction. The points within the radii around

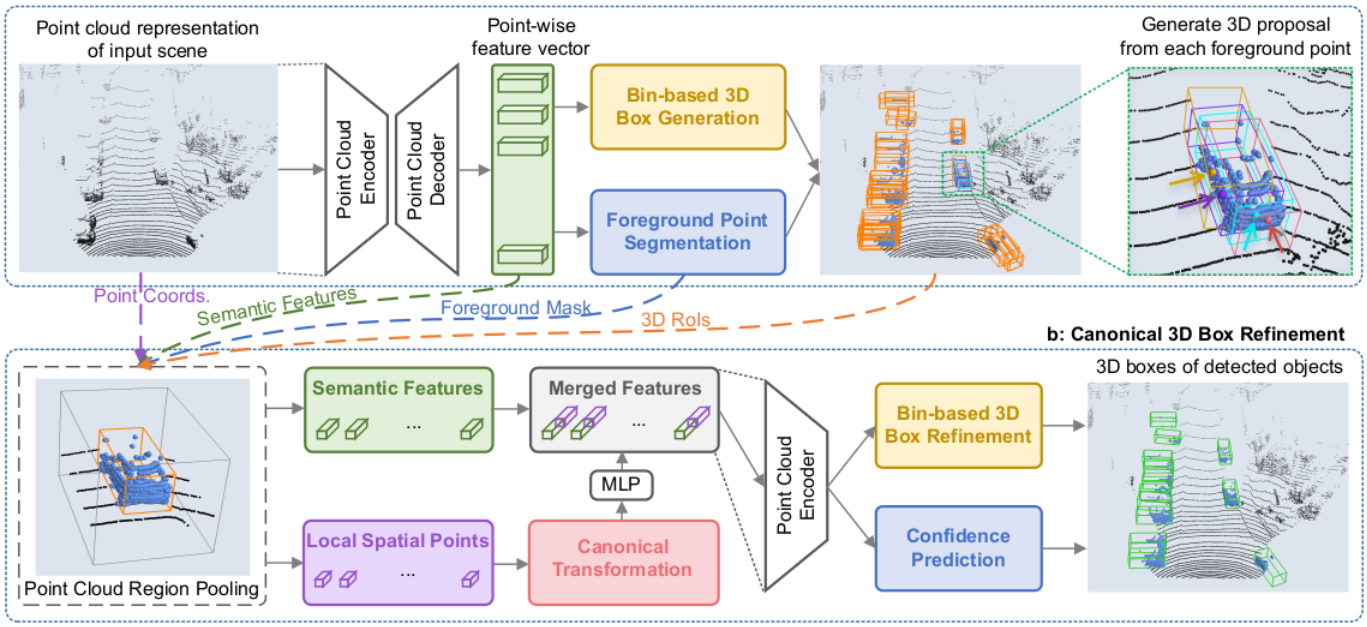


Fig. 7: PointRCNN Architecture

each sample are sampled for the last time and all the sample sets are given to the backbone as inputs. These are concatenated to obtain one feature vector per initial sample. The same process is performed at the RoI pooling step. Box coding is the process of coding differences between the predicted and ground truth bounding boxes. Mean size can be provided to the model to code the box dimensions relative to it. In order to utilize box coding and calculate the loss function, predicted bounding boxes are assigned to ground truth bounding boxes. Ground truth extra width parameter enlarges the ground truth bounding boxes during this step.

D. Evaluation metrics

The evaluation metric parameter in the training configuration was selected as KITTI [6] in the sample configuration file prepared by the author. R40 AP score for 0.5 IoU is chosen for apples due to its small size. Average Precision is ideally equal to the area under the precision-recall curve and it is used to evaluate the performance of the model for the whole range or above a IoU threshold. Since 0.5 is used in this project as the IoU threshold, the precision and recall values calculated for lower thresholds are not accounted for since low threshold means either large bounding box position error or a large size error. In reality, calculating the precision and recall values for every IoU value above the threshold is computationally expensive and a good approximation can be obtained by selecting a number of points on the curve and calculate an average by using only these points. R40 score used for this project means the number of points used for AP calculation is equal to 40. Number of detections and the number of apples are given in the experiments section as well since they allow a broader assessment of the model performance.

IV. Experiments

PointRCNN [27] model was originally built to be used for driving datasets such as KITTI [6] and Waymo Open [29] and the hyperparameters given in the configuration file in the repository of the paper are tuned according to driving datasets. However, the datasets contains a single class with very different scenes and objects with size 8x8x8 cm on average which is much smaller than vehicles, cyclists and pedestrians in a driving dataset. There are many steps in the network whose effect on each other is difficult to determine without testing. Therefore, parameters could not be calculated or chosen.

In this section, the precision of the model is tried to be maximized by altering the parameters in training and dataset configuration files. In order to find the possible values some parameters can take, the toolbox code was reviewed. In order to compare the effects of different parameter values, parameters are altered one by one and the results are compared with the reference training. Length based parameters were reduced to one tenth of their original value as their values seem independent of axis and the size of the apple is roughly one tenth of a pedestrian.

Experiment section is organized as follows. First, the implementation and important details of the experiments are given. Then, important hyperparameter tuning experiments are mentioned in the main subsection. In this subsection, after dataset characterization experiments, number of samples, regularization parameters and learning-rate experiments are examined using photogrammetry, LiDAR datasets and the optimum parameters for the final training are decided.

A. Implementation

PointRCNN model has been presented with a GitHub repository by the authors. In order to avoid any implementation errors and follow the standard pipeline followed by the authors, this repository is used for the experiments. This toolbox is called OpenPCDet [24] and it includes hyperparameters optimized by the authors of the article as well. OpenPCDet is part of a codebase called OpenMMLab, which includes many computer vision toolboxes specialized for various tasks. It supports many datasets and models and is used for benchmarking due to its state-of-the-art functionality.

OpenPCDet was written by Shaoshuai Shi in 2020 and has a modular structure to support a wide range of architectures. It relies heavily on PyTorch libraries. For the backbone, the SpConv (PyTorch Spatially Sparse Convolution Library) package is used for sparse convolution, which is a version of 3D convolution tailored for sparse point clouds. Apart from these, Open3D and Mayavi packages can be used for interactive visualization of the point cloud, ground truths, and detections. Custom dataset support is provided through the functions prepared for the preprocessing of the KITTI dataset. Since it is open source, it has become capable of supporting many datasets without any major bugs with the support of other authors.

Several changes had to be made to perform the experiments and assess the results efficiently and without errors. The visualization script is modified to efficiently compare ground truth and predicted bounding boxes. The 0.5 IoU R40 mAP value written to the tensorboard log to see it graphically and export it. The data processing script was modified to create info files of validation split information in addition to training and testing and finally, a new script was written to get the validation loss.

Training is performed according to a configuration file. Most of the parameters are located in this file. Parameters such as number of epochs, batch size, recording period, and number of epochs to test were given during runtime. There is also a dataset configuration file with data-related parameters. This file contains voxelization settings, dataset range, input feature list, and data augmentation settings. When the training configuration is imported, the dataset configuration is imported into a field within it. After preprocessing, before starting the training, the files where the samples to be used in train, validation, and test are sorted are added to a specific folder, and info files are created according to the dataset configuration.

B. Hyperparameter Tuning

The experiments on LiDAR dataset started before the second and third stages of the labeling process as the automatic labeling was expected to be sufficient for preliminary experiments. However, low scores are obtained the number of detections were too low compared to ground truth bounding boxes. After all the labeling stages are completed, the results improved significantly. After visualization, it became apparent that too few bounding boxes are predicted at the regions where apples are clustered and close enough to be in contact sometimes. As the apples have a roughly spherical shape, detection of bounding boxes with high IoU requires a high NMS threshold to prevent filtering of accurate detections. The score threshold is removed as well since there were very few false positives. With these changes, better results could be obtained.

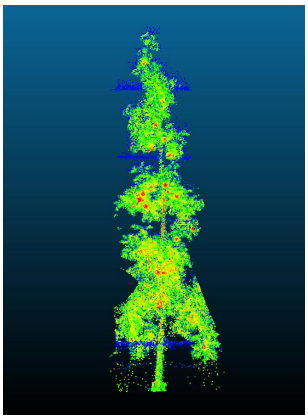
1) Dataset Characterization: Upon visual inspection, it can be seen that the color, shape and reflectance of the apples on some trees are noticably different from each other. Therefore, several experiments are performed to understand the effect of selection of test sample and obtain a reference score to compare with the scores obtained in other hyperparameter tuning experiments. As the size of the dataset is only 11, all samples can be tested individually. For this purpose, trees are assigned to test split one by one. The results can be compared with the visualization of the point clouds to see what kind of conditions make the model perform better. The average and the range can also be used to compare the model performance before and after the hyperparameter tuning.

Test Tree	AP R40 @0.5 IoU	Number of Detections	Number of GT Boxes
1	30.03	74	84
2	38.17	66	70
3	22.03	71	58
4	29.53	77	86
5	21.58	77	87
6	12.57	75	101
7	17.36	89	81
8	23.15	73	82
9	11.67	93	126
10	19.22	83	91
11	7.56	83	92
Global Mean	21.17	-	-

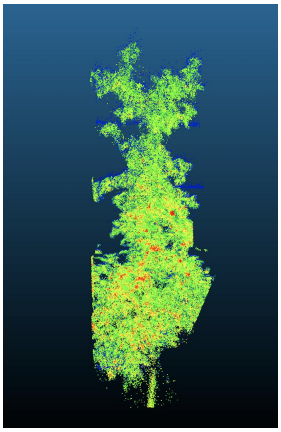
TABLE I: Lidar Dataset Sample Comparison

According to Table I, the second tree has achieved a much better score than the rest of the trees as expected from its clearly defined, mostly unoccluded apples. Thus, it is selected as the test tree for the rest of the experiments. Eleventh tree is the worst performing tree. The point cloud of the second and eleventh trees are given in Fig. 8.

Individual trees are tested for photogrammetry and photogrammetry with reflectance datasets as well to see the effect of different modalities, point density and positional accuracy on detection from different samples. These are performed using the regularization parameters chosen which will be explained later in this section. This experiment can be used to see the effect of hyperparameters unrelated to regularization by comparing the results with the final training results.



(a) LiDAR 2nd Tree



(b) LiDAR 11th Tree

Fig. 8: The apples on the 2^{nd} tree are largely unoccluded and has more high reflectance points as a result. There are also no unripe apples on the 2^{nd} tree whereas 11^{th} tree has many apples which are unripe and occluded making them very difficult to detect even visually.

Test Tree	AP R40 @0.5 IoU	Number of Detections	Number of GT Boxes
1	69.53	75	84
2	79.19	67	70
3	78.50	69	58
4	69.19	72	86
5	53.82	72	87
6	47.19	72	101
7	57.40	78	81
8	70.44	77	82
9	33.30	80	126
10	54.26	73	91
11	33.68	81	92
Global Mean	58.95	-	-

TABLE II: Photogrammetry Dataset Sample Comparison

The results given in Table II are similar in ranking to the ones obtained from LiDAR. Therefore, color and reflectance information are most likely affected by occlusion similarly. This is because the apples occluded by leaves, branches and other apples are likely to be occluded from similar angles as they are densely filling the space around the apples. Another reason is the proximity of measurement ranges and directions of LiDAR and camera sensors. Some apples such as brown ones can be more difficult to detect by both reflectance and color as well.

The scores obtained from photogrammetry data showed superiority of color data especially red and green colors in distinguishing red apple surfaces from green leaves. This clear distinction is not as clear with the LiDAR data. High reflectance does not always indicate an apple and low reflectance does not guarantee absence of one. Therefore, it has to be associated with position data clearly for the apple to be detected easier.

Test Tree	AP R40 @0.5 IoU	Number of Detections	Number of GT Boxes
1	37.07	73	84
2	44.16	69	70
3	39.32	74	58
4	28.81	84	86
5	22.20	76	87
6	17.02	82	101
7	23.90	78	81
8	39.18	79	82
9	7.20	82	126
10	25.33	82	91
11	9.12	89	92
Global Mean	26.66	-	-

TABLE III: Photogrammetry with Reflectance Dataset Sample Comparison

According to the Table III, photogrammetry with reflectance has very similar scores to LiDAR data in value as well. The density and positional accuracy of photogrammetry data could not improve the score as the reflectance values are taken from the LiDAR data with lower density and positional accuracy. Therefore, the association between the points were imperfect since the closest LiDAR neighbor of a photogrammetry point is not necessarily at the correct position.

2) Number of Samples: Number of points at the set abstraction step is a parameter controlling number of samples sampled from the initial samples where a different sample set is formed for each number in the array to be concatenated after the backbone as explained in the model section. It was increased to four times its original value during the preliminary experiments as the

memory requirement was satisfied by any node in the server. It is doubled again for this experiment to utilize the geometrical information and point density further. It is important for the model to extract as much information as possible from the scenes and sampling is where most of the information is lost. Additionally, the number of points at the sampler is doubled to avoid limiting the effect of the parameter and the number of samples for two different set abstraction radii are doubled. A score of 46.07 is reached for LiDAR dataset and 81.43 is obtained from photogrammetry dataset which is a significant enough to keep for the final training although a higher score was expected from a major upgrade like this.

3) Regularization: Regularization parameters are tuned to increase the network performance while preventing possible model overfit. The most important regularization strategy for the datasets used is data augmentation as the lack of data is the most problematic part of the experiments. There are four types of built-in data augmentation methods in OpenPCDet: Ground truth sampling where points inside a given number of ground truth bounding boxes are randomly placed inside the scene, flipping about x or x and y together, random rotation within a given range and random scaling within a given range. The number of apples are selected as 250 to see the effect of ground truth sampling clearly. Therefore, 250 additional point cloud portions within random ground truth bounding boxes will be placed inside the scene at random locations. The rotation range is selected as – pi to pi to use the whole range. Scaling ratio range is selected as 0.9 and 1.1 to allow % 10 size variation.

	None	Ground Truth Sampling	Scaling	Rotation	Flip-x	Flip-xy
AP R40 @0.5 IoU	38.17	41.46	34.24	28.93	35.72	34.18
Number of Detections	66	77	65	61	65	85
Number of GT Boxes	70	70	70	70	70	70

TABLE IV: LiDAR Data Augmentation

According to Table IV, ground truth sampling and flipping about x-axis are beneficial. Data augmentation results for the photogrammetry dataset are as follows.

	None	Scaling	Rotation	Flip-x	Flip-xy
AP R40 @0.5 IoU	79.19	78.80	71.36	76.21	77.75
Number of Detections	67	70	71	67	72
Number of GT Boxes	70	70	70	70	70

TABLE V: Photogrammetry Data Augmentation Results

According to Table V, Flipping, rotation and scaling did not increase the score. Therefore, they are not used for the rest of the experiments. Other regularization options for PointRCNN are drop out and weight decay.

Weight Decay Coefficient	0.01	0.02	0.03	0.04	0.05
AP R40 @0.5 IoU	82.25	83.84	83.81	82.12	81.35
Number of Detections	63	72	68	65	66
Number of GT Boxes	70	70	70	70	70

TABLE VI: Weight Decay Results

Drop-out Ratio	0.4	0.5	0.6	0.7	0.8
AP R40 @0.5 IoU	81.74	83.68	84.14	82.62	81.02
Number of Detections	71	72	72	70	73
Number of GT Boxes	70	70	70	70	70

TABLE VII: Drop-out Ratio

Number of Objects for Ground Truth Sampling	150	250	400	600	800	1000
AP R40 @0.5 IoU	80.89	81.66	81.35	83.42	80.28	81.04
Number of Detections	71	67	69	73	66	67
Number of GT Boxes	70	70	70	70	70	70

TABLE VIII: Photogrammetry Number of Objects in Ground Truth Sampling

According to Tables VI, VII, VIII, number of samples for ground truth sampling, drop-out ratio and weight decay coefficient are selected as 600, 0.6 and 0.02 respectively. Parameters decided here are fixed for the rest of the experiments as they are not expected to interact with other parameters.

4) Learning Rate: Learning rate is one of important parameters in the optimizer. In the experiments performed the graphs indicated saturation after a certain epoch. In order to make sure the model is able to converge to a minimum close to global minimum, the learning rate is decreased to lower values.

The scores obtained from different learning rates can be seen in Table IX. Thus, the optimum learning rate is selected as 0.0005. Lower learning rates caused training to get stuck on low gradient points and higher learning rates prevented convergence to lower minima.

Learning Rate	$5 * 10^{-3}$	10^{-4}	$5 * 10^{-4}$	10^{-5}
Photogrammetry	78.47	77.38	84.19	72.33
LiDAR	37.82	38.39	45.24	30.27

TABLE IX: Learning Rate Parameters

V. Results

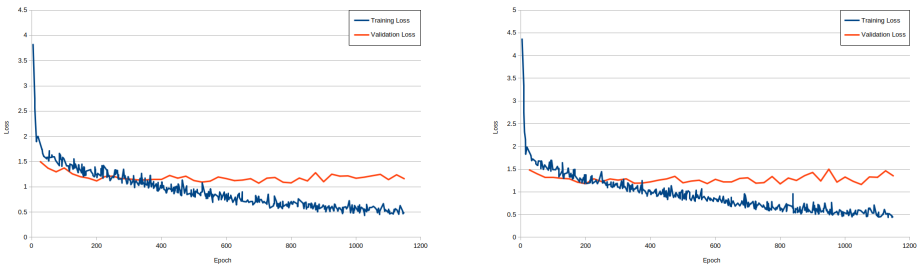
In this section, all the datasets are trained with final configuration files where one validation and one test tree selected randomly 10 times and maximum and minimum scores are shown with corresponding loss graphs and bounding box positions. Since there are only 11 samples, repetition of the test samples is avoided. Average score of all 10 experiments for each dataset is included in the tables as well.

A. LiDAR Dataset

The results of the experiment can be are given in Table X. There are several reasons for the low score. The most important of them is the dataset. The positional accuracy of the apples suffered from the positioning device of the scanning system. The shape of the apples are hard to identify and low density of points and heavy occlusions made this task very difficult for the network. Occlusions are caused by the narrow angle of measurement, density of the point cloud and proximity of leaves and branches to the apples. This result was expected as the labeling of the data was difficult for many apples as well. Reflectance is not sufficient at distinguishing apples that are not red or having too few points. High reflectance points can occur in leaves and branches as well and low reflectance values are present on apples as well. According to Fig. 9, training seems ineffective after around 200^{th} epoch for the validation tree. According to Fig. 10, the worst tree is larger than the best one and contains more apples. The apples are also very close together at many locations. Therefore, a significant number of apples could not be found.

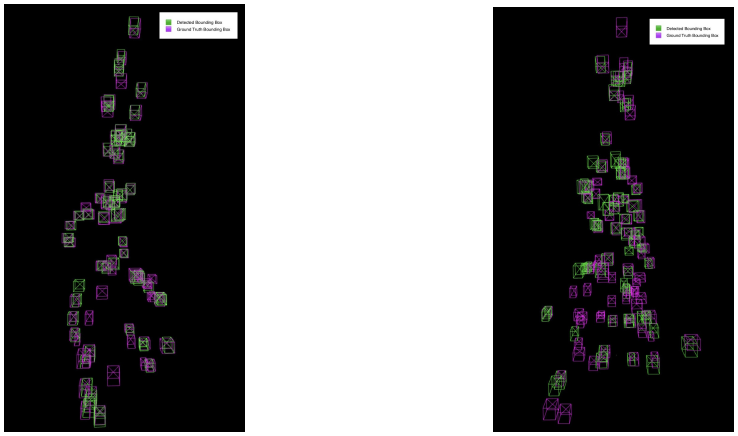
LiDAR Dataset	AP R40 @0.5 IoU
Average	26.31
Maximum Score	45.93
Minimum Score	10.23

TABLE X: LiDAR Dataset Results



(a) Best Training Score for LiDAR Dataset (b) Worst Training Score for LiDAR Dataset

Fig. 9: Loss Graphs of Best and Worst Training Scores for LiDAR Dataset



(a) Best Tree Visualization (b) Worst Tree Visualization

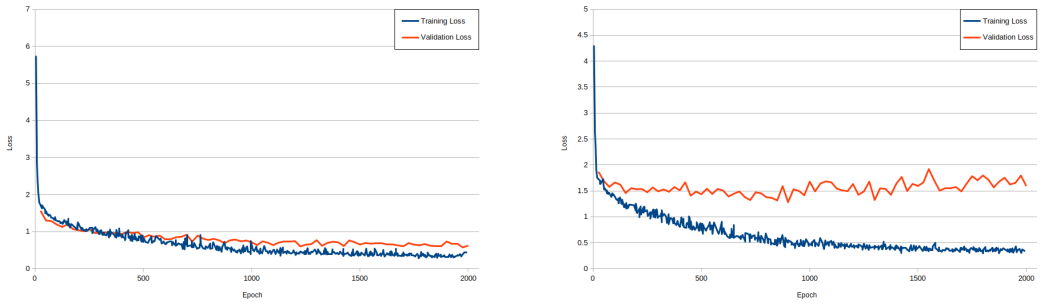
Fig. 10: Best and Worst Scores Test Tree Visualization for LiDAR Dataset

B. Photogrammetry Dataset

According to Table XI, the color information combined with mostly precise and dense point cloud enabled high scores despite occlusion. Red and green colors are so distinguishing in this dataset that apples without occlusion and brown color can be guaranteed to be detected. Thus the network can focus on harder tasks such as detecting occluded or brown apples increasing the score significantly. Similar to the LiDAR dataset, density of the point cloud and proximity of background elements to the apples limited the scores. According to Fig. 11, the validation loss for best scoring training saturates at around 1000^{th} epoch together with the training data and has a slightly higher value. However, the worst scoring training's validation loss stopped decreasing almost immediately. According to Fig. 12, the best scoring tree only missed a few apples due to its small size and easily distinguishable apples whereas the worst scoring tree has several missed detections around the center of the trunk.

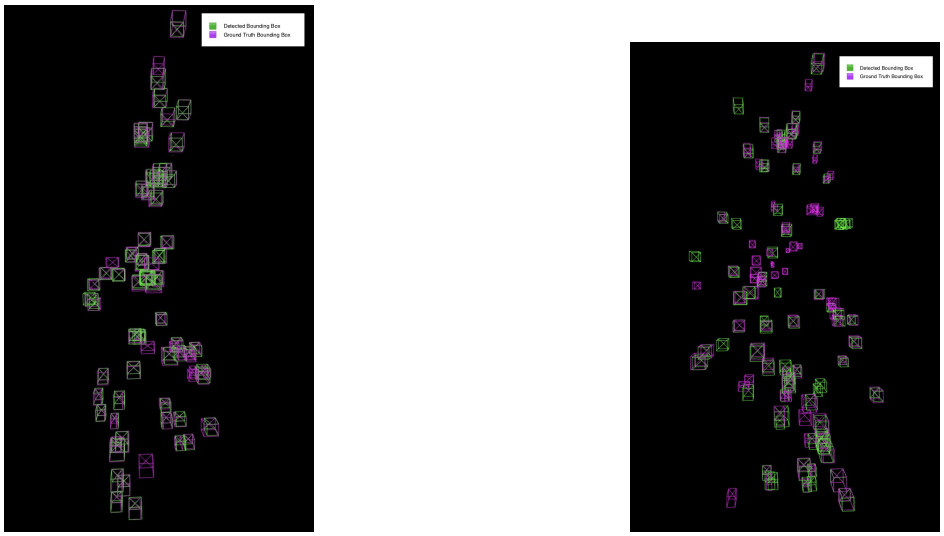
Photogrammetry	AP R40 @0.5 IoU
Average	67.62
Maximum Score	83.32
Minimum Score	34.42

TABLE XI: Photogrammetry Results



(a) Best Training Score for Photogrammetry Dataset (b) Worst Training Score for Photogrammetry Dataset

Fig. 11: Loss Graphs of Best and Worst Training Scores for Photogrammetry Dataset



(a) Best Tree Visualization (b) Worst Tree Visualization

Fig. 12: Best and Worst Scores Test Tree Visualization for Photogrammetry Dataset

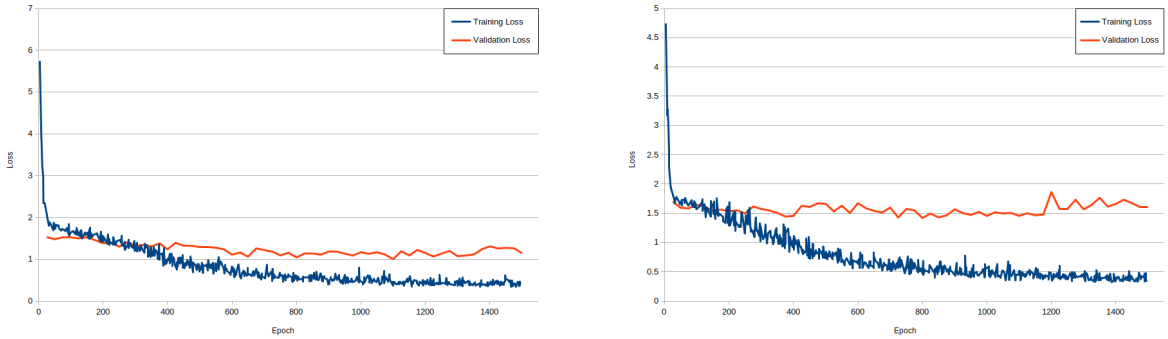
C. Photogrammetry Coordinates Dataset

According to Table XII, this dataset compensated for reflectance information with its point density and positional accuracy. SfM algorithm can accurately define 3D surfaces of apples in most cases as seen during the labeling process as well. Unfortunately, for apples which are heavily occluded, apple surfaces could not be defined as a one continuous convex surface and could not be distinguished from the leaves and branches. According to Fig. 13, the validation loss of the best score

kept decreasing until around 600^{th} epoch. However, the validation loss for the worst score did not decrease. The training loss saturated around 1000 as well. According to Fig. 14, the tree with the worst score is larger with more clustered apples. Therefore, there are more undetected apples.

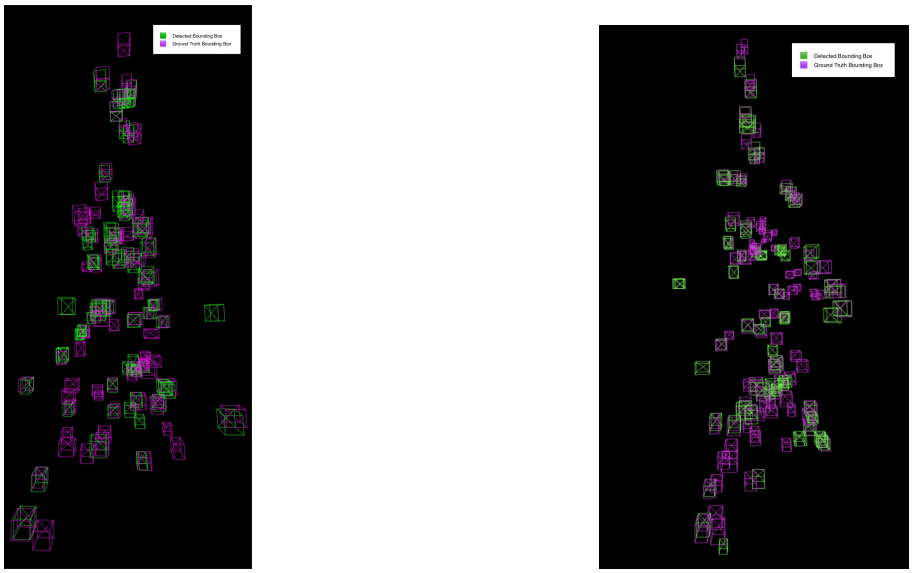
Photogrammetry Coordinates Dataset	AP R40 @0.5 IoU
Average	44.01
Minimum Score	25.68
Maximum Score	54.51

TABLE XII: Photogrammetry Coordinates Dataset Results



(a) Best Training Score for Photogrammetry Coordinates Dataset (b) Worst Training Score for Photogrammetry Coordinates Dataset

Fig. 13: Loss Graphs of Best and Worst Training Scores for Photogrammetry Coordinates Dataset



(a) Best Tree Visualization (b) Worst Tree Visualization

Fig. 14: Best and Worst Scores Test Tree Visualization for Photogrammetry Coordinates Dataset

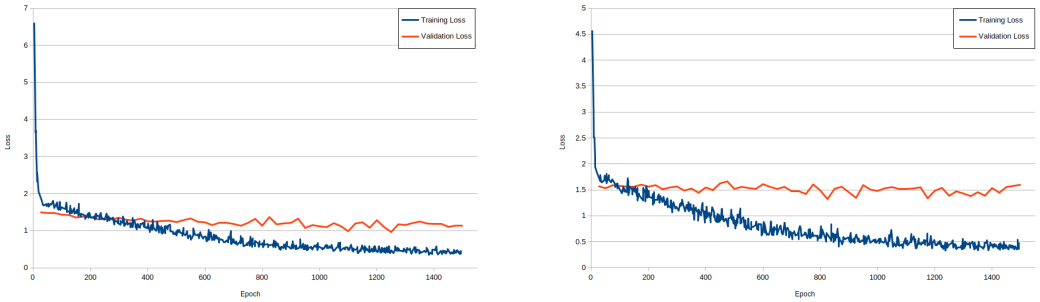
D. Photogrammetry with Reflectance Dataset

The results obtained from this dataset is given in Table XIII. Although this dataset has the reflectance information on top of the coordinate information from the photogrammetry dataset, during the assignment of the reflectance values, the measured values seems to be mixed up due to lower positional accuracy of the dataset. Therefore, some of the points which should have high reflectance have low reflectance and vice versa. The number of points in the LiDAR dataset is lower as well. As a result, reflectance of a single LiDAR point is assigned to many neighboring photogrammetry points, diluting the reflectance information. These reasons limited the contribution of reflectance information to the dataset significantly. According to Fig. 15, although the graphics look dissimilar due to a high starting point, they are very similar to each other except the validation

loss decreased to 1 for the best score. According to Fig. 16, similar to the previous comparisons, the worst scoring tree is larger and contains more apples and clusters. Therefore, many of the apples could not be detected.

Photogrammetry with Reflectance	AP R40 @0.5 IoU
Average	43.47
Minimum Score	23.06
Maximum Score	53.83

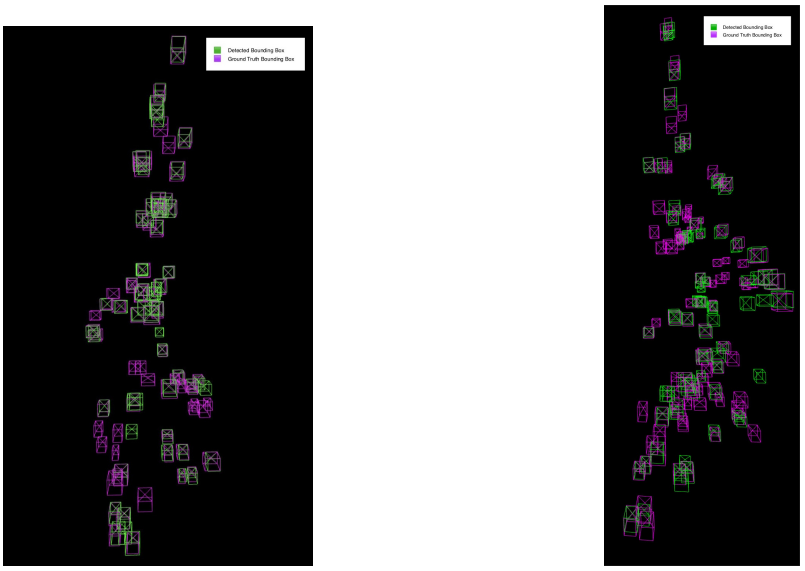
TABLE XIII: Photogrammetry with Reflectance Dataset Results



(a) Best Training Score for Photogrammetry with Reflectance Dataset

(b) Worst Training Score for Photogrammetry with Reflectance Dataset

Fig. 15: Loss Graphs of Best and Worst Training Scores for Photogrammetry with Reflectance Dataset



(a) Best Tree Visualization

(b) Worst Tree Visualization

Fig. 16: Best and Worst Scores Test Tree Visualization for Photogrammetry with Reflectance Dataset

VI. Conclusions

This section includes a short summary of the project, important results, main conclusions and future work. The aim of the project was to apply a state-of-the-art 3D object detection model to various datasets to assess its performance and feasibility. LiDAR and photogrammetry point clouds are used as well as a colorless version of the photogrammetry dataset and a version of it with reflectance information. PointRCNN is selected as the model since it achieved better scores in driving datasets and it is applicable to general 3D scenes. Hyperparameter tuning experiments are performed to make sure the model has reached its potential for the datasets. After the parameter configuration is decided, training is repeated for all the datasets and final results are presented.

Labeling had to be improved as the previous version has several errors as it relied on some assumptions such as the 2D segmentation accuracy is sufficiently high, points close in projections are close in depth and 3D reconstruction is accurate. The sources of these errors were false segmentation, 3D reconstruction process and location differences of the bounding boxes in

different datasets. Although it is difficult to label objects within dense point clouds, accurate labeling is essential for detection of small objects in the presence of heavy occlusions as even small errors can affect model performance.

Despite numerous experiments, there were only a few parameters which increased the model performance significantly. This is probably caused by the low amount of samples in the dataset as large models such as these tend to require a large amount of data for proper operation. These experiments are number of points and samples at the backbone, learning rate, drop-out ratio, weight decay and number of objects used for ground truth sampling data augmentation. Other data augmentation methods did not increase the score. This means methods used were unable to produce samples without disturbing the sample distribution of the dataset.

The photogrammetry dataset has achieved 67.62 % AP on average due to its distinguishing color information. Therefore, the task of detection of apples can be considered successful with reasonable accuracy which can be further increased with modifications of the architecture as it has non-uniform processing steps tailored for driving datasets. LiDAR dataset only reached 26.31 % AP. Data volume and quality must be improved to make it usable for agricultural purposes. Heavy occlusions, brown apples having low reflectance made this task challenging to achieve without color information. High reflectance points does not cover the entire apple surface as in photogrammetry data due to the nature of LiDAR measurement. They are only concentrated at a region much smaller than the apple size. It is suspected that reflectance channel could not be utilized efficiently because of the data volume or quality as well. Data volume was a result of high measurement system speed and quality was adversely affected by the positioning system.

Photogrammetry coordinates dataset was able to achieve similar performance with 44.01 % AP on average meaning the point density and positional accuracy are very important for this task whereas photogrammetry dataset with reflectance could only score 43.47 % AP on average. This implies naively assigning channel values to nearest neighbors may not be sufficient in this case due to the difference in datasets and the lower quality of the LiDAR dataset.

For future work, LiDAR data quality can be improved by replacing the moving measurement system with a stationary one similar to how photogrammetry data is collected, increasing the point density by scanning multiple times or moving slower. Scanning from multiple heights and angles can increase the detection performance for apples far away from the sensor and apples occluded from a certain angle. Lastly, the number of samples can be increased to utilize the model better or use few shot learning methods or machine learning architectures to diminish the effect of the low dataset size.

Large deep learning models can be used to increase the score as time is not an important constraint for this application if there is sufficient hardware. It is possible to use other architectures used for more similar tasks such as RGB-D architectures for indoor settings. The number of samples can be increased by dividing each tree into smaller pieces which makes it easier to assess the qualities of trained models by studying their differences. Manual data augmentation can be performed to control the augmentation options more freely and evaluate their performance better.

ACKNOWLEDGMENT

I would like to extend my appreciation to my thesis advisors, Javier Ruiz-Hidalgo and Jordi Gené-Mola, for their guidance and insights during our brief collaboration. I also would like to thanks to University of Lleida and IRTA for granting me access to their invaluable data, which was pivotal for this research, and to UPC for supplying the necessary equipment for my experiments and data storage.

This work was conducted within the framework of PAgPROTECT project (PID2021-126648OB-I00), funded by the Spanish Ministry of Science, Innovation and University by MCIN/AEI/10.13039/501100011033 and by “ERDF, a way of making Europe”, by the European Union.

Lastly, I want to sincerely thank everyone in my life who motivated and stood by me, enabling me to complete this project.

REFERENCES

[1] Cloudcompare (version 2.12.3) [gpl software], 2023. Available at: http://www.cloudcompare.org/.

[2] Lidar, 29 August 2023.

[3] Y. Chen, Y. Xiong, B. Zhang, J. Zhou, and Q. Zhang. 3d point cloud semantic segmentation toward large-scale unstructured agricultural scene classification. *Computers and Electronics in Agriculture*, 190:106445, 2021.

[4] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li. Voxel r-cnn: Towards high-performance voxel-based 3d object detection. 2020.

[5] J. P. Silva do Monte Lima and V. Teichrieb. An efficient global point cloud descriptor for object recognition and pose estimation. In *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 56–63, 2016.

[6] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[7] J. Gené-Mola, R. Sanz-Cortiella, J. R. Rosell-Polo, J.-R. Morros, J. Ruiz-Hidalgo, V. Vilaplana, and E. Gregorio. Fruit detection and 3d location using instance segmentation neural networks and structure-from-motion photogrammetry. *Computers and Electronics in Agriculture*, 2020.

[8] Jordi Gené-Mola, Eduard Gregorio, Fernando Auat Cheein, Javier Guevara, Jordi Llorens, Ricardo Sanz-Cortiella, Alexandre Escolà, and Joan R. Rosell-Polo. Lfuji-air dataset: Annotated 3d lidar point clouds of fuji apple trees for fruit detection scanned under different forced air flow conditions. *Data in Brief*, 29:105248, 2020.

[9] Jordi Gené-Mola, Ricardo Sanz-Cortiella, Joan R. Rosell-Polo, Josep-Ramon Morros, Javier Ruiz-Hidalgo, Verónica Vilaplana, and Eduard Gregorio. Fruit detection and 3d location using instance segmentation neural networks and structure-from-motion photogrammetry. *Computers and Electronics in Agriculture*, 169:105165, 2020.

[10] Jordi Gené-Mola, Ricardo Sanz-Cortiella, Joan R. Rosell-Polo, Josep-Ramon Morros, Javier Ruiz-Hidalgo, Verónica Vilaplana, and Eduard Gregorio. Fuji-sfm dataset: A collection of annotated images and point clouds for fuji apple detection and location using structure-from-motion photogrammetry. *Data in Brief*, 30:105591, 2020.

[11] B. Graham. Spatially-sparse convolutional neural networks. 2014.

[12] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[13] S. L. Chenhang He, Ruihuang Li, and L. Zhang. Voxel set transformer: A set-to-set approach to 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[14] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. 2020.

[15] P. Kurtser, O. Ringdahl, N. Rotstein, and H. Andreasson. Pointnet and geometric reasoning for detection of grape vines from single frame rgb-d data in outdoor conditions. 01 2020.

[16] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. 2018.

[17] G. Li, M. Müller, A. Thabet, and B. Ghanem. Deepgcns: Can gcns go as deep as cnns? 2019.

[18] S. Liu, K. Fu, M. Wang, and Z. Song. Group-in-group relation-based transformer for 3d point cloud learning. *Remote Sensing*, 14(7), 2022.

[19] J. Mao, M. Niu, H. Bai, X. Liang, H. Xu, and C. Xu. Pyramid r-cnn: Towards better performance and adaptability for 3d object detection. 2021.

[20] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[21] National Oceanic and Atmospheric Administration. What is lidar, 26 February 2021.

[22] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. 2016.

[23] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. 2017.

[24] Shaoshuai Shi. Openpcdet. https://github.com/open-mmlab/OpenPCDet/tree/master, 2023. GitHub repository.

[25] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[26] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *International Journal of Computer Vision*, 131:531–551, 2021.

[27] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. *CoRR*, 2018.

[28] W. Shi and R. Ragunathan. Point-gnn: Graph neural network for 3d object detection in a point cloud. 2020.

[29] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, and B. Caine. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[30] Travis S. Taylor. *Introduction to Laser Science and Engineering*. CRC Press, 2019.

[31] T. Vu, K. Kim, T. M. Luu, X. T. Nguyen, and C. D. Yoo. Softgroup for 3d instance segmentation on point clouds. 2022.

[32] Z. Yang, Y. Sun, S. Liu, and J. Jia. 3dssd: Point-based 3d single stage object detector. 2020.

[33] T. Yin, X. Zhou, and P. Krähenbühl. Center-based 3d object detection and tracking. 2020.

[34] Q. Yu, H. Yang, Y. Gao, X. Ma, G. Chen, and X. Wang. Lfpnet: Lightweight network on real point sets for fruit classification and segmentation. *Computers and Electronics in Agriculture*, 194, 2022.

[35] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. 2017.