THE MECHANICAL AND ALGORITHMIC DESIGN OF IN-FIELD ROBOTIC LEAF SAMPLING DEVICE

BY

JUNZHE WU

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Agricultural and Biological Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Master's Committee:

Associate Professor Girish Chowdhary, Chair and Advisor
Assistant Professor Cody Michael Allen
Assistant Professor Matthew Jon Stasiewicz

ABSTRACT

Leaf samples analysis is a significant tool to acquire the actual nutrition information of crops. After that, farmers can adjust fertilization programs to prevent nutritional problems and improve the yield of crops. Traditional way for leaf sampling is manual, and researchers need to go to the field and use paper hole punchers with a catch-tube to collect leaf samples. The temperature in summer is hot, and some crop like corn is difficult for researchers to walk through, therefore the manual way of leaf sampling is not a good option.

In this thesis, an automatic method of leaf sampling is presented to solve the difficulty of leaf sampling. The contributions of this thesis are the following: (1) Build the end effector of leaf sampling device to punch and store leaf samples separately, (2) Train a neural network to detect the leaves with high horizontal level, (3) Combine point cloud data from depth camera and vison data from camera via the sensor fusion to get the leaf rolling angle and grasp point. The method in this thesis can produce a consistent leaf rolling angle estimate quantitatively and qualitatively on multiple corn leaves, especially on leave with multiple different angles.

To My Mom and Dad:

I haven't been home for two years since the COVID-19 pandemic, and I really missed all of you each time I had a video chat with you. You always encourage me in the video chat which gives me strength to face the difficulties in study and life. I hope I can come back home soon and eat the food you cooked again.

To my girlfriend:

Your presence brings me a lot of happiness and makes me not feel alone anymore in this specific time period. With your mental support, I was able to overcome the difficulties I met in the process of writing this thesis.

ACKNOWLEDGMENTS

The author wants to acknowledge the following people:

- **Dr. Girish Chowdhary** for giving me the opportunity to become a member of DASLab, and providing the support and guidance which help me confirming the pipeline of algorithmic design and clear my mind in the process of writing this thesis.
- **Dr. Matthew Jon Stasiewicz** for proposing this interesting project which gives me the opportunity to solve a complicated real problem.
- **Sri Theja Vuppula** for printing the 3D model of designed parts of end effector, and buying the equipment needed for the end effector and sensor fusion such as solenoid, stepper motor, 3D Lidar (VLP-16) and the chessboard for the calibration.
- **Mateus Valverde** for collecting the data of corn leaves for me in the field which is particularly hot in summer, and helping me to do the joint calibration of Lidar and camera.
- **Rajan Aggarwal & Genny Korn** for revising the initial design of enf effector and assembling the 3D-printed parts of end effector.

TABLE OF CONTENTS

CHAPTER 1: Introduction

1.1 Motivation

From the United Nations, the world population is increasing by around 1.13 percent per year and will grow from 7.4 billion in 2016 to 8.1 billion in 2025. By 2050, the world population is projected to reach 9.7 billion (UN, 2015). Urbanization will continue to develop at a rapid pace with the numerous income increases, and urban populations will make up about 70% of the world's population, compared to 49% today. To feed these increased populations, especially the richer population in the urban area, food production must increase by 70% (Tripathi et al., 2019). In order to solve this problem, the efficiency of food production should be improved in technical ways. Fertilization could produce higher yields to enhance crop reserve and buffer national food provision (Renard and Tilman, 2019). In practice, fertilizer programs should be changed in the different stages of plants according to nutritional condition of crops and soil to improve yield. Leaf sample analysis, frequently used after soil testing, is critical for the indication of nutrient status which tells us how well the plants get certain elements from soil-applied fertilizer and what elements it needs (Obreza et al., 1992).

One way for traditional leaf sampling is manual, researchers need to go to the field and use paper hole punchers with a catch-tube to collect leaf samples. The operation is simple, but researchers need to take many tubes to isolate the sample. The temperature in summer is hot, and some crops like corn are difficult to walk through. Therefore, the manual way of leaf sampling is not a good option. Another way is using sampling machines to punch and isolate the leaf samples automatically. An example of these two ways can be seen in figure 1.1. From the picture, we can find this machine is bulky and heavy so that researchers still need to go to the field to bring the leaves back. Hence an automatic and flexible leaf sampling device is needed to solve the problem.

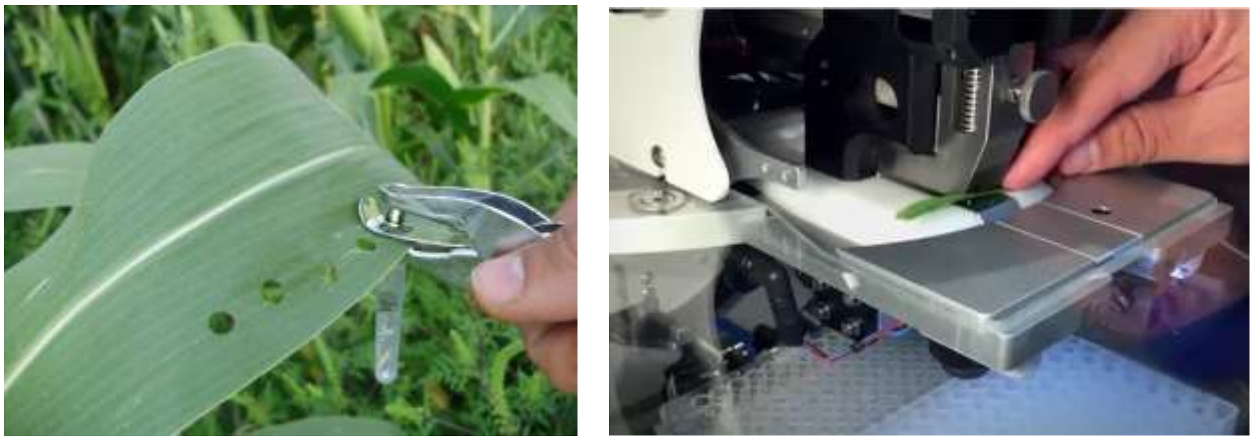

(a) paper hole punchers with a catch-tube (b) sampling machine

Figure 1.1: Two traditional ways for leaf sampling

1.2 Related Work

1.2.1 Automatic Harvesting of fruits

The mechanical design process for leaf sampling device and automatic harvesting device is same where the manipulator and end effector need to be designed according to the object we want to grasp. As shown in the figure 1.2, a manipulator with 5 DOF was designed to grasp the apple in the high position and avoid the obstacle in the process of approaching object apple. With the utilization of pneumatic devices, the spoon-shaped end effector can quickly change the open and close status. In the experiment, this apple harvesting system uses 15.4 s on average for the picking process and achieves a 77% success rate (De-An et al., 2011).

For the harvesting of green pepper, an end effector was developed to grasp the stem of the fruit and cut it which is shown in the figure 1.3. In this way, this process doesn’t cause any harm to the fruit since all the operations are done at the stem position. The manipulator of the green pepper harvesting system is also in 5 DOF since it is enough to approach the object in the automatic harvesting of fruits and vegetables.

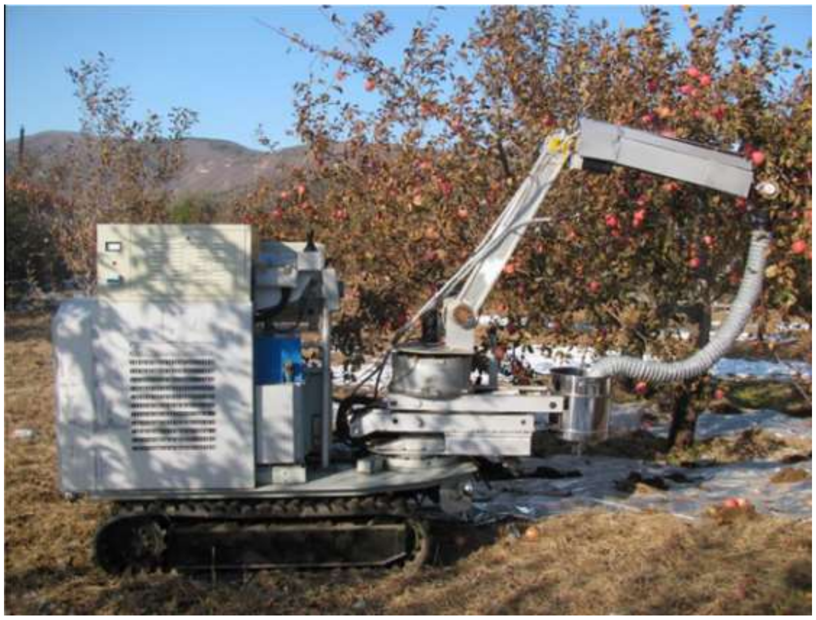
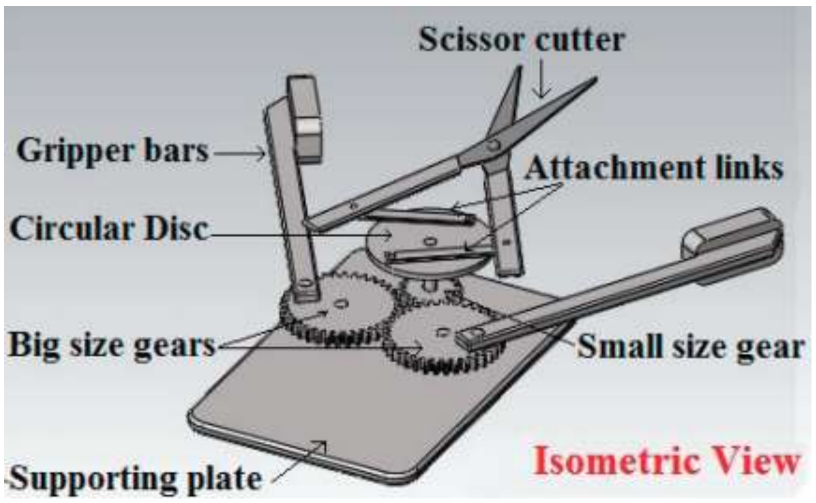
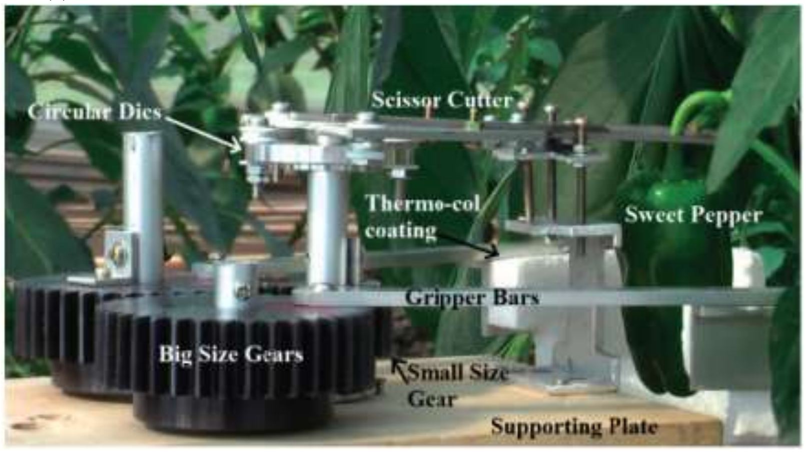


Figure 1.2: Apple harvesting system(De-An et al., 2011)



(a) The 3D model in Solidworks



(b) The end effector in the real system

Figure 1.3: The end effector for green pepper harvesting (Bachche and Oka, 2013)

The automatic harvesting of rice, wheat, soybean, corn, rapeseed is already realized by the combine-harvester from John Deere. With the adoption of the CTS technology, cutting flow threshing cylinder and plate teeth longitudinal axial flow separation cylinder structure was developed to enhance the separation performance of the machine and reduce the rate of grain breakage. Compared to the fruit of these crops which are hard or possess a hard shell, other fruits and vegetables like apple and green pepper are softer and easily damaged which means the large agricultural machinery cannot be applied for the harvesting of these fruits.

Low accuracy, low speed, and high cost are the main reasons of why automatic harvesting robot are not as popular as the large agricultural machinery. However, with the rapid development of machine vision, machine learning, and sensor fusion, automatic harvesting machinery are capable of detecting the fruit with high success rate. Ji et al. (2012) presented a real-time vision detection system where a segmentation method based on color feature and shape feature is applied to process filtered apple image. After that, a SVM classifier was utilized to classify apple in the segmented objects with 89% success rate. However, the recognition rate of fruits could be further improved by the application of neural networks. In the study of Jia et al. (2015), they used K-means algorithm to segment apple from the background and utilize a radial basis function (RBF) neural networks which is optimized using genetic algorithm and least mean square algorithm for recognition. For blocked and overlapping apple samples, the detection rates are 95.38% and 96.17%, respectively. While for all kinds of apple samples, the detection rate can reach 96.95%. Compared to apple, the recognition of green pepper is harder since it has the same color with the background of leaves. In the study of Song et al. (2014), a bag-of-words (BoW) model is trained to extract features from images and produce frequency distribution as an input to the SVM classifier. This method of image analysis can achieve a recognition rate of 96.5%. After the detection of object, a flexible and robust manipulator is needed to approach the object rapidly after the detection of object to overcome the low speed of automatic harvesting robot. Considering the high cost of fruit harvesting robot, a more general end effector should be designed to grasp different kinds of fruits so that it could be mass produced to reduce the cost. There are still many significant works needed to be done on the manipulators and end effector in order to construct the intelligent farm in the future.

1.2.2 Sensor Fusion

Sensor fusion is widely adopted in the autonomous driving field because the types of road scenarios in real urban environments are diverse and can change rapidly where only one kind of sensor is not capable of getting all the significant information from environment and keeping pace with the rate of road environment changing. Research shows that the autonomous driving system with more sensors for sensor fusion system benefits with better perception performance and the robustness of the planning solution (Kocić et al., 2018). For the safety of vehicle and pedestrian, sensor fusion is indispensable for the autonomous vehicle.

The key sensors in the sensor fusion of autonomous driving system are camera, radar, and lidar. The autonomous vehicle generally has the largest number of cameras in the sensor list which are mounted on the different part of vehicles with different angles to achieve 360-degree observation of the environment. The high-resolution cameras can provide informative data for the machine learning and deep learning algorithm. One shortcoming of cameras is that they consume a large amount of GPU memory in the deep-based detection algorithm, and the GPU with high memory is usually expensive.

For Radar, the most important characteristic is that it can measure the speed of other objects directly using the Doppler effect while camera and Lidar need several frames of data to calculate the speed. Another advantage is that radar is rarely affected by environmental factors and can operate in all weather conditions, such as complex light, rain and snow. Moreover, radar can measure objects in a longer distance than camera and Lidar. The drawback of Radar is that the resolution and accuracy of point cloud data from Radar are low which increases the difficulty of object classification.

LiDAR is the abbreviation of Light Detection and Ranging which emits laser beams to detect the target's position. With the utility of laser beams, it can get point cloud data with high resolution and accuracy, but the operation of LiDAR is easily influenced by weather and atmosphere. The attenuation of laser is generally small in sunny weather, but dramatically increased in heavy rain, smoke, fog and other bad weather which greatly affect the spread distance. Moreover, the LiDAR which can produce high resolution point cloud data is extremely expensive. We can find that three different sensors have their own merit and drawback, thus sensor fusion is needed to take advantage of all merits from each sensor and make up for the weakness.

The classification job using sensor fusion can be divided into two types: applying sensor fusion before the classification or applying sensor fusion after the classification. In the study of Gao et al. (2018), an object detection method which applied sensor fusion before the classification was proposed. Firstly, point cloud data after upsampling was projected to the image plane to produce a depth image. Then this depth image became the fourth channel of the RGB image, and a collection of new RGB-D images were fed into a deep convolutional neural network (CNN) for training. The process of producing this RGB-D image are shown in the figure 1.4. The classification results of this CNN can reach 100%, 100%, 98.6%, 88.6% and 97.2% at pedestrian, cyclist, car, truck and others, respectively.

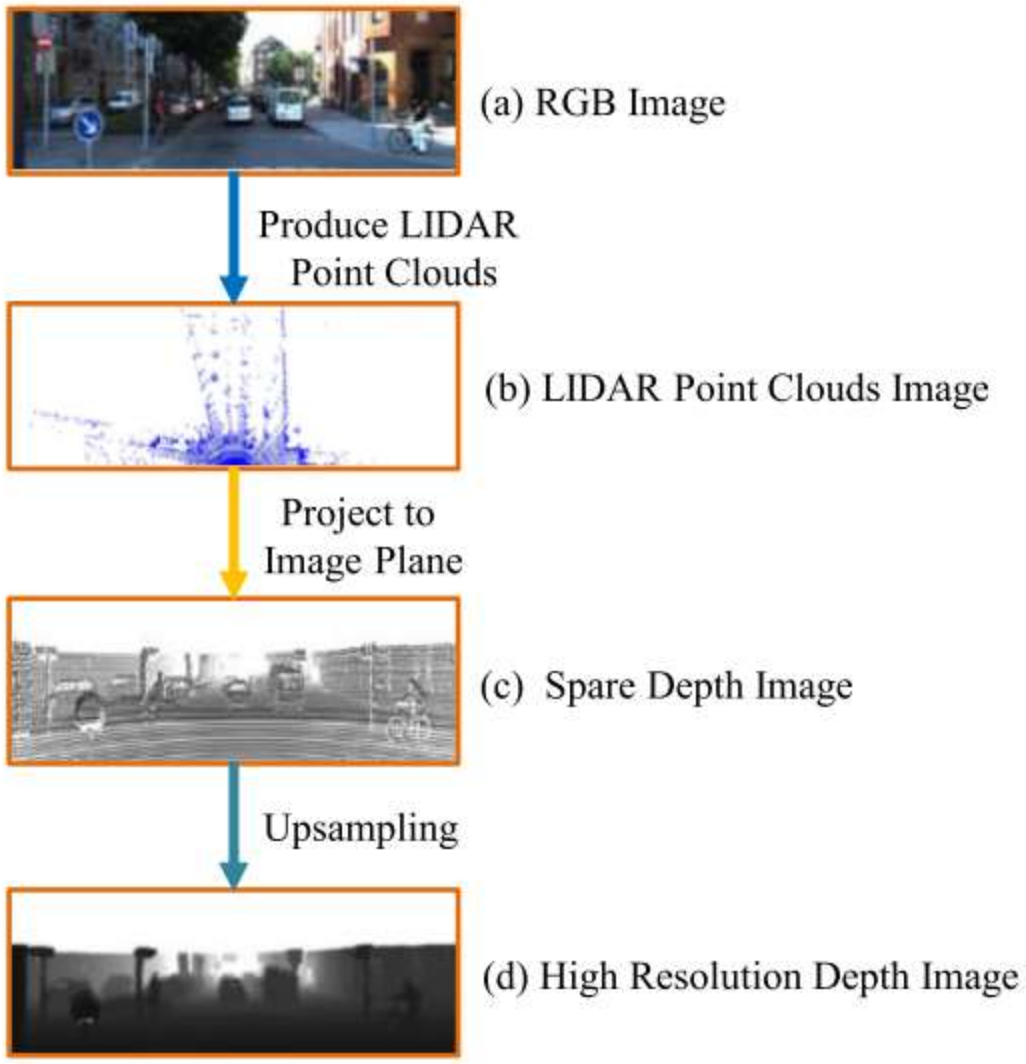

(a) RGB Image
Produce LIDAR Point Clouds
(b) LIDAR Point Clouds Image
Project to Image Plane
(c) Spare Depth Image
Upsampling
(d) High Resolution Depth Image

Figure 1.4: The process of producing this RGB-D image (Gao et al., 2018)

In another research of Xiao et al. (2015), a road detection method which applied sensor fusion after the classification was proposed. They first established the correlation between points from LiDAR and pixels in images using the intrinsic matrix and extrinsic matrix from the calibration. The fused result after projection is shown in figure 1.5. Then they trained boosted decision tree separately on image data and point cloud data for classification. After that, the two classification scores were fused to be the unary potentials of the corresponding pixel nodes to construct the conditional random field. The fused conditional random field can be easily solved with graph cut to predict the road area. In figure 1.6, green represents the predicted road areas, and blue represents ground truths. The predicted result from fused CRF works best since the green area is almost overlapped with blue area.

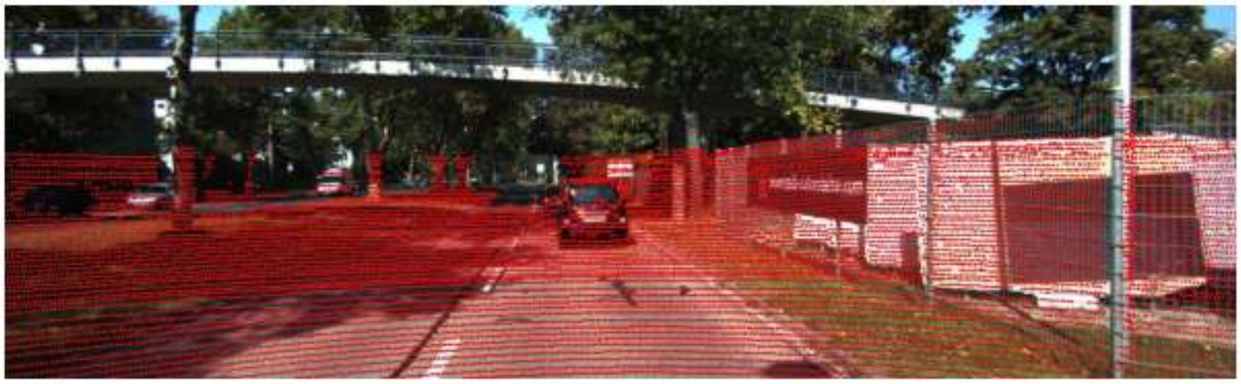
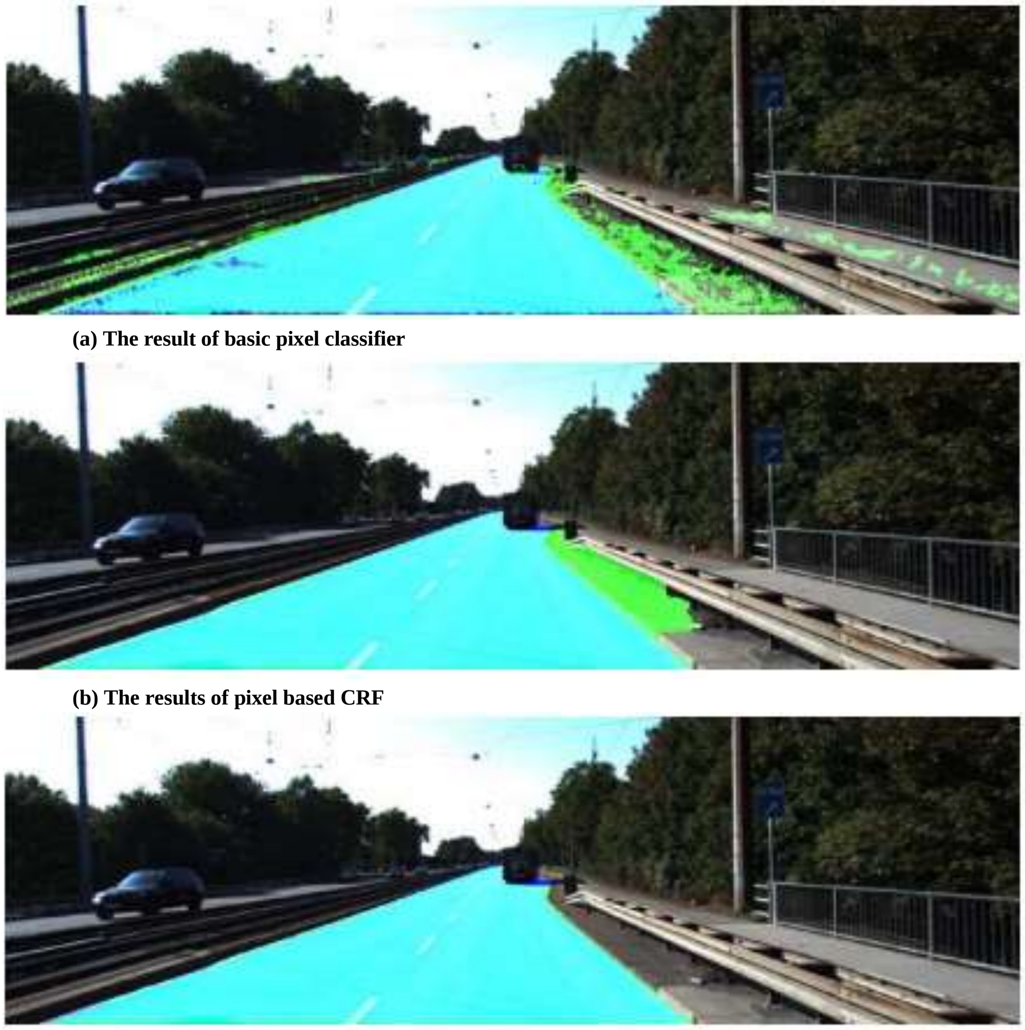


Figure 1.5: The fused result after projection (Xiao et al., 2015)



(a) The result of basic pixel classifier

(b) The results of pixel based CRF

(c) The results of the proposed fused CRF

Figure 1.6: The prediction results of road area (Xiao et al., 2015)

The same sensor fusion technology can also be applied for recognition of fruits in the automated harvesting, and the exact 3D coordinates of detected objects can be obtained from point cloud data which benefits the future steps in automated harvesting. Tao and Zhou (2017) developed an automatic apple recognition method which applied sensor fusion before the classification. Depth camera termed Kinect v2 was used in this study to produce fused colored point cloud data, and an RGB-based region growing segmentation algorithm was used to get colored point cloud data of apple as dataset. Then they utilized an improved 3D descriptor (Color-FPFH) to extract color and 3D features from dataset and fed these features to a classifier based on the support vector machine (SVM) which was optimized by a genetic algorithm for training. This classifier is capable of predicting apple in 3D bounding box (fig. 1.7) with 92.3% accuracy.

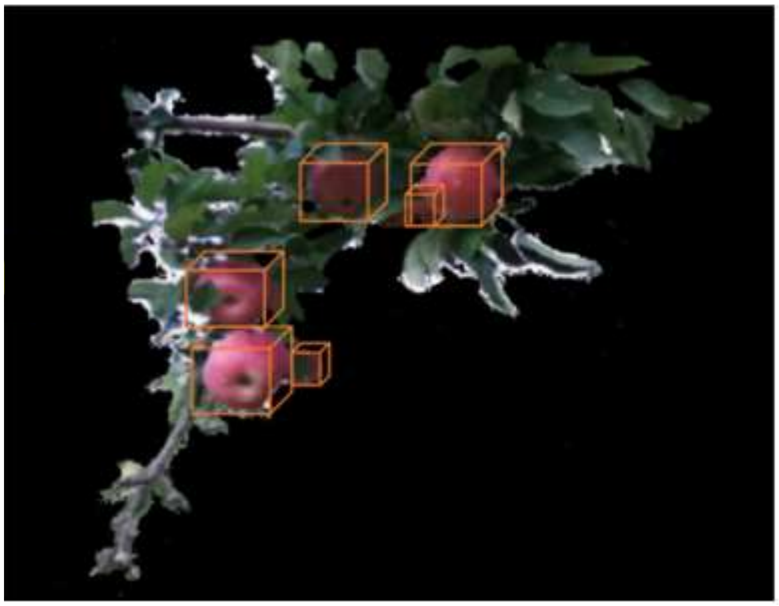


Figure 1.7: The prediction results of apple in 3D bounding box (Tao and Zhou, 2017)

Eizentals and Oka (2016) proposed another method which applied sensor fusion before the classification to recognize the green pepper and estimate the 3D pose of stem for the next cutting step. They first used a machine vision technology named image analysis block to segment the green pepper from the background of leaves. After that, they applied a coherent point drift (CPD) algorithm to project the points from LiDAR to the image plane, and filtered the point cloud inside the recognition area to calculate the 3D coordinates of detected green pepper's stem. The detection result and filtered point cloud are shown in figure 1.8, and we can find the detection result is not accurate enough where some leaves are mistakenly identified as green peppers. Hence, machine learning or deep learning method should be used for the recognition to obtain a more accurate consequence from the image.

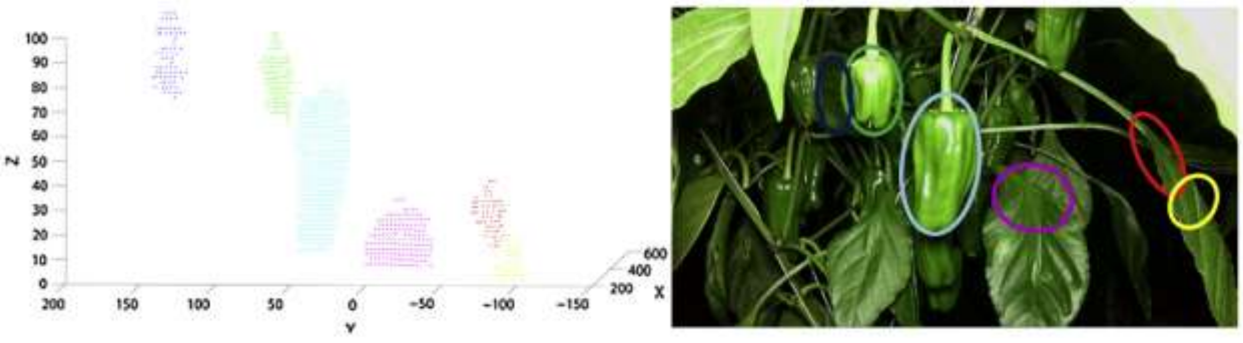


Figure 1.8: The detection result and filtered point cloud (Eizentals and Oka, 2016)

1.3 Field Robotic System

A Field Robotic System, termed TerraSentia, was developed in the Distributed Autonomous Systems (DAS) Laboratory (Zhongzhong et al., 2020). The robot in the figure 1.9 is the second version of TerraSentia. This agricultural robot is designed to autonomously navigate between the rows of corn and collect information such as corn number, stem height and stem width for crop breeders, plant protection product developers, crop scientists, and field agronomists.

This field robotic system is a four-wheel ground-based mobile robot, whose size is 30 cm tall × 50 cm long × 35 cm wide, with a 15 cm ground clearance, and weighs 6.5 kg. It is also equipped with a range of sensors. One real-time kinematic (RTK) GPS (ZED-F9P) has been mounted on the rear of the top of the robot. This module has multi-band RTK with fast convergence times and reliable performance and multi-band GNSS receiver which delivers centimeter level accuracy in seconds. The 3D LiDAR (VLP-16) or depth camera (RealSense D435i) can also be installed on this place for the sensor fusion task. Two 2D LiDARs (Hokuyo UST-10LX) are installed on the top and trailing end of the robot respectively to acquire points from the horizontal plane and moving vertical plane. The resolution angle of the 2D LiDAR is 0.25°, the range of it is 270°, and it measures data at 40Hz. Images are recorded with a USB board camera (ELP-USBFHD01M-L21) mounted on the three sides of the robot to get information from the surrounding environment. The frame rate of the camera is 30 fps when the resolution is 1920 x 1080 pixels, and the length size of the camera is 2 megapixels.

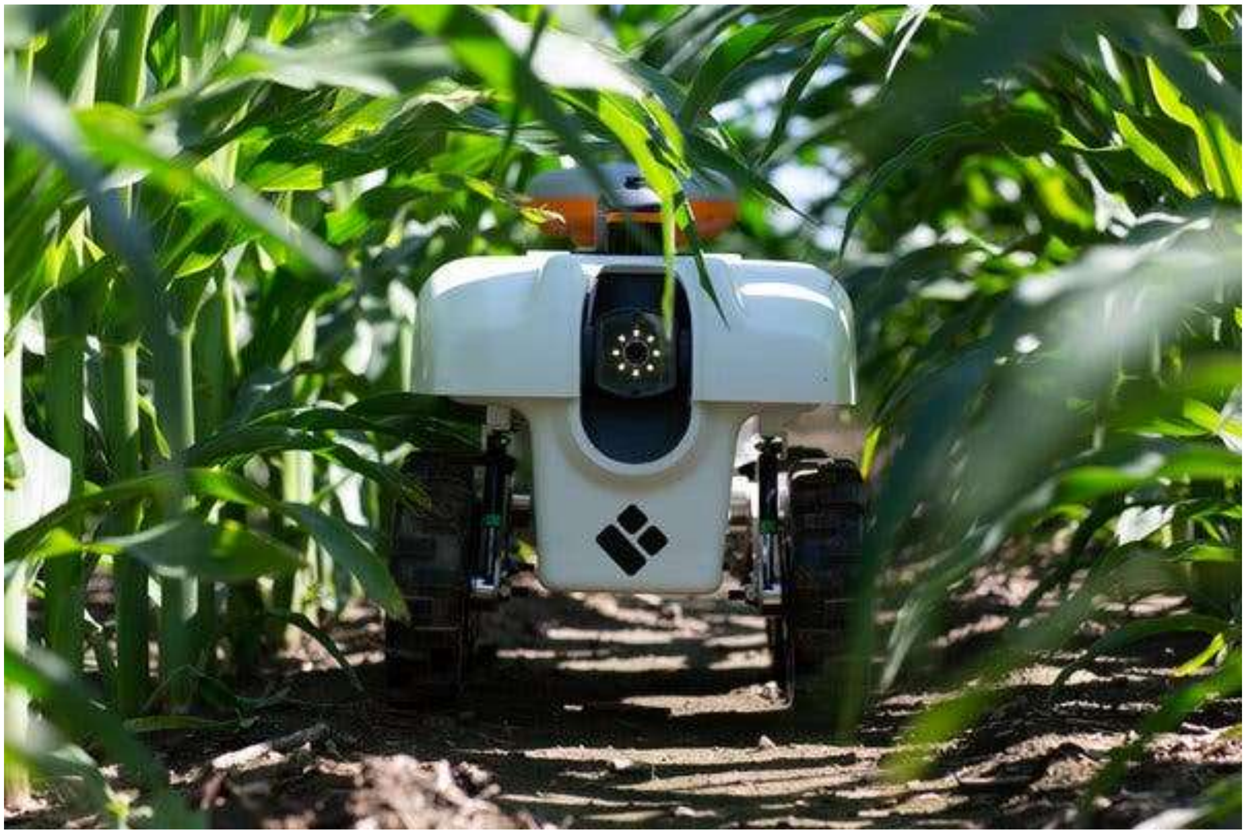


Figure 1.9: The field robotic system, termed TerraSentia

1.4 Objectives

The main objective of this thesis is to develop an automatic and robust method of leaf sampling in the field. A more flexible and smaller end effector should be designed to punch and store leaf samples separately in our leaf sampling device. The end effector with mechanical arm can be installed on the ground-based robot platform in our lab which can navigate automatically in the field. Furthermore, a deep based sensor fusion algorithm should be developed to choose object leaves, calculate the leaf rolling angle, and confirm grasp point.

1.5 Overview

This paper is organized as follows: Chapter 2 explores the mechanical design of the leaf sampling manipulator according to the robotic leaf sampling requirements. Chapter 3 details the deep based detection of object leaves including the selection of neural network architecture and the preparation of dataset. Chapter 4 describes a sensor fusion algorithm which uses the intrinsic matrix and extrinsic matrix from the calibration for leaf rolling angle estimation and grasp point selection. Chapter 5 provides an overview of data collection and experimental leaf rolling angle measurement. Lastly, chapter 6 concludes the paper and suggests future work.

CHAPTER 2: Design of the Leaf Sampling Manipulator

2.1 Robotic Leaf Sampling Requirements

This leaf sampling project was proposed by Stasiewicz Food Safety Laboratory and we had a discussion with them to confirm Robotic Leaf Sampling Requirements below. The primary design plant is lettuce, however, there is no recorded lettuce data in our lab. Instead, we choose corn as the design plant for experiment since there is a mass of recorded data of corn at different stages in our lab.

a. Take samples < 25 mm from edge of leaf.
b. Collect circular leaf sample with 20 mm diameter.
c. The end effector are capable of storing 20 samples per run.
d. Object leaf surface angle relative to the ground is around 0° for the easy control of end effector, but the designed accessible leaf surface angle should be up to 90°.
e. Each sample should be isolated and labeled with time and GPS information from the ROS system.
f. The overall size of end effector with leaf samples storage part should be less than 20 cm x 20 cm x 20 cm.
g. The end effector mass should be less than 2 kg.
h. The end effector mass should be dismountable to get the leaf samples inside the storage part easily.

2.2 End Effector Design

2.2.1 Punch Mechanism Design

For the automation of punching, a solenoid is applied to drive the punch. In order to punch the corn leaves successfully, we need to use the corn leaf thickness, PSS of corn leaves and leaf samples diameter to calculate the required punch force. Then we can use this punch force to select the appropriate type of solenoid. From table 2.1, the average corn leaf thickness with no shading treatment is 127.82 μm (Lihua et al., 2012). From table 2.2, the total average of Punch-and-Die Shear Strength (PSS) for leaf sheaths in different growth periods is 5.35 MPa. The leaf samples diameter is 20 mm according to the requirements in the section 2.1.

Table 2.1: Leaf epidermal thickness and leaf thickness of corn under different shading treatment in greenhouse (Lihua et al., 2012)

Shading Treatment	Upper Epidermal Thickness/μm	Lower Epidermal Thickness/μm	Leaf Thickness/μm
No	27.27± 2.15	20.80± 2.22	127.82± 9.12
With 30% luminousness	25.89± 2.15	27.27± 2.15	27.27± 2.15
With 10% luminousness	27.27± 2.15	27.27± 2.15	27.27± 2.15

Table 2.2: Mean for PSS of leaf sheaths originating from different internodes of sugarcane stalk from different growth periods (October, November, and December 2010) (Mou et al., 2013)

	Punch-and-Die Shear Strength (PSS)/MPa		
Internode	October	November	December
6	4.42	3.59	4.45
5	4.89	4.21	4.98
4	5.68	5.02	6.83
3	5.48	4.91	7.13
2	5.98	5.33	6.49
1	5.65	5.15	6.18
Total Average		5.35	

After these three parameters are acquired, the punch force is calculated as follows:

$$F = \pi d t (PSS) \tag{2.1}$$

Where:

F = punch force (N)

d = diameter of the leaf samples (m)

t = thickness of the corn leaves(m)

PSS = punch-and-die shear strength (MPa)

The final result of punch force is 42.95 N which is equal to 154.47 oz. Based on this punch force, a sealed linear solenoid with 191 oz. force at 10% stroke length is selected to push the punch. The voltage of this sealed linear solenoid is 12 V while the power is 59 W. The mounting orientation can be any angle, horizontal, inverted or vertical which meets the requirement of rotating end effector to access the leaf with a large leaf surface angle relative to the ground. The size of this sealed linear solenoid (the unit is inch) is shown in the figure 2.1.

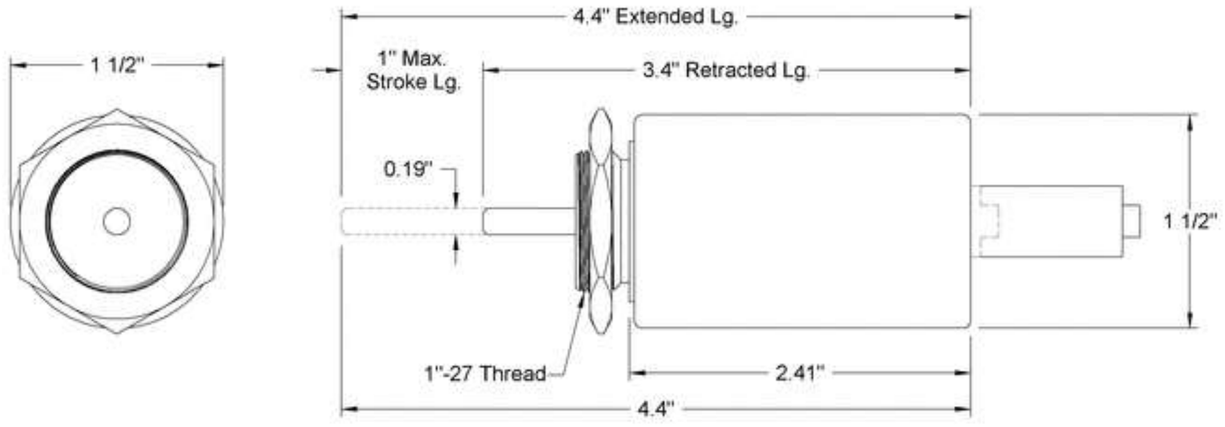
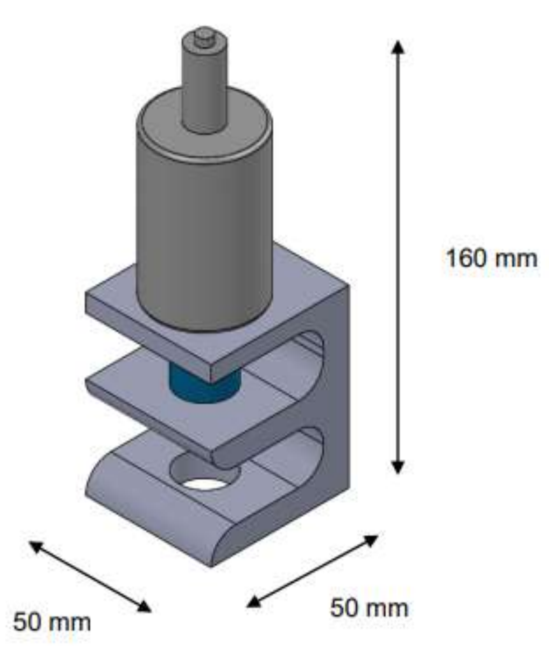
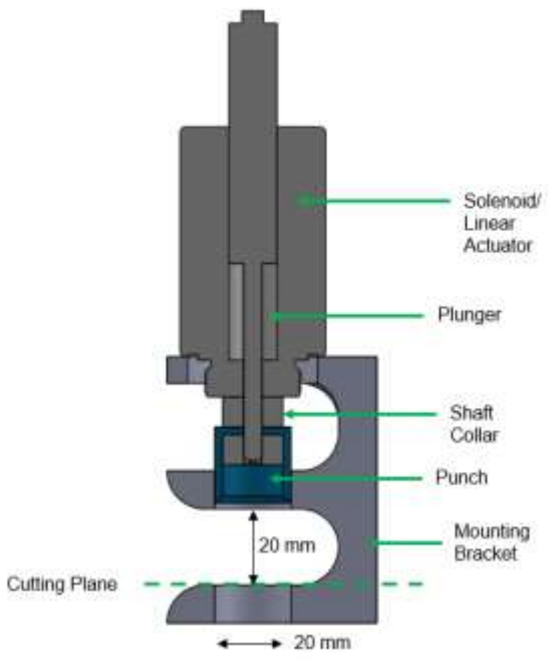


Figure 2.1: The size of sealed linear solenoid (https://americas.rsdelivers.com)

Furthermore, the punch is connected to the linear solenoid with shaft collar, then a bracket is designed to fix the combination of linear solenoid and punch (fig. 2.2). In practical work, the object leaf should be on the cutting plane for punching after the implementation of detection algorithm and the control of manipulator and end effector.





(a) **(b)**

Figure 2.2: Punch mechanism design

2.2.2 Storage Assembly Design

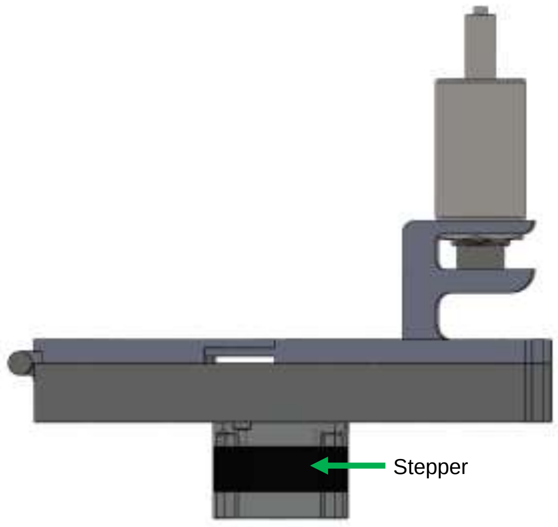
For the storage assembly (fig. 2.3), a 20x 21mm circular sample tray is designed to store leaf samples. This circular samples tray fits with a three-leaf screw shaft in the middle which is connected to a stepper in the bottom and can rotate at 18°. The round circular lid which is hinged with the bottom tray is on the top to cover the circular sample tray for the protection and isolation. The punch mechanism is combined with round circular lid and coaxially fit with one hole in the circular samples tray initially. The assembled end effector in the real world is shown in the figure 2.4, and its working process has following steps:

(1) Punch mechanism does the cutting job, and a leaf sample is forced into one hole in the sample tray

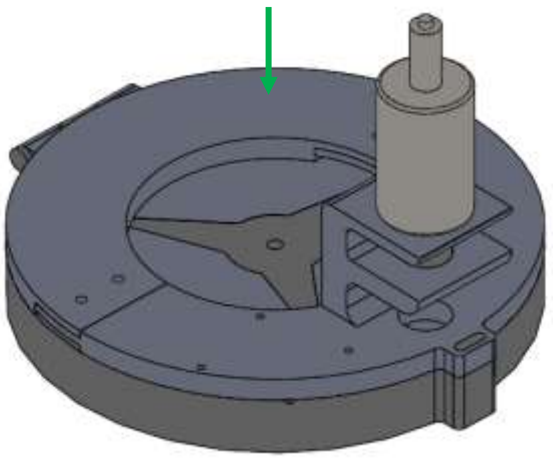
(2) The stepper rotates 18° which drives the rotation of the three-leaf screw shaft and the sample tray, and now the punch mechanism is coaxially fit with a new hole.

(3) Repeat 20 times of step (1) and step (2)

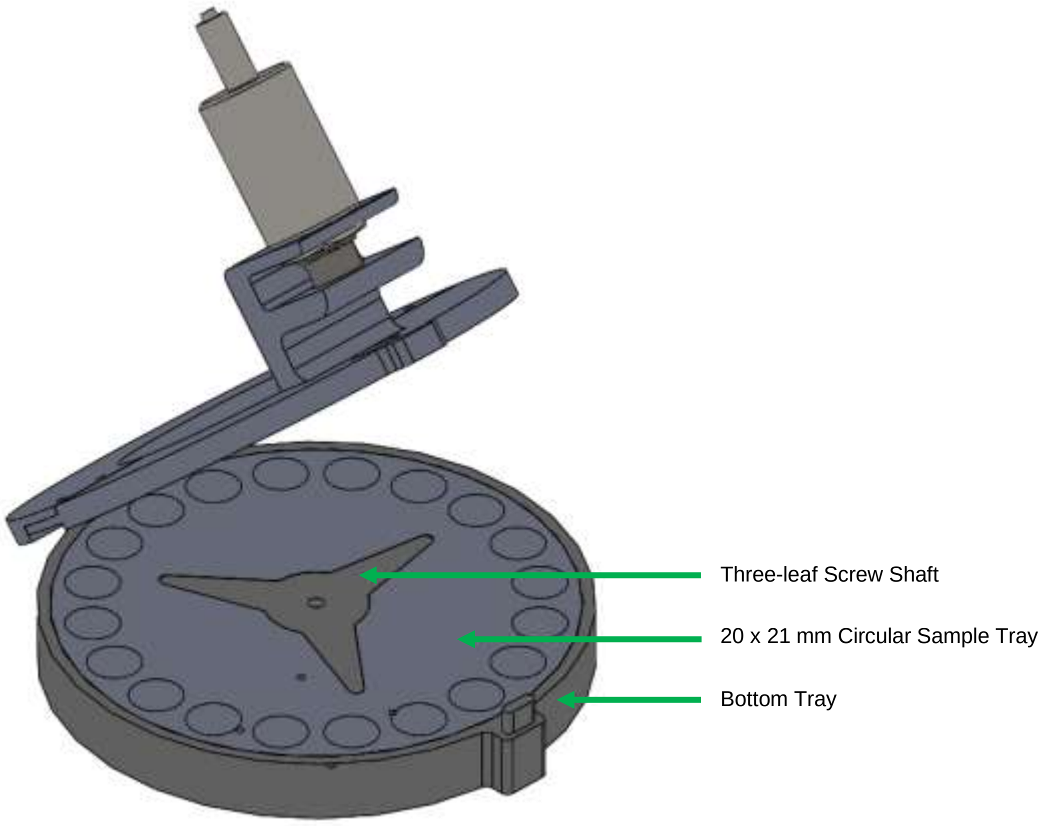
(4) Once a run is finished, researchers can lift the round circular lid, replace the circular samples tray with a new one, and start a new run again.



(a)



(b)



(c)

Figure 2.3: Storage assembly design

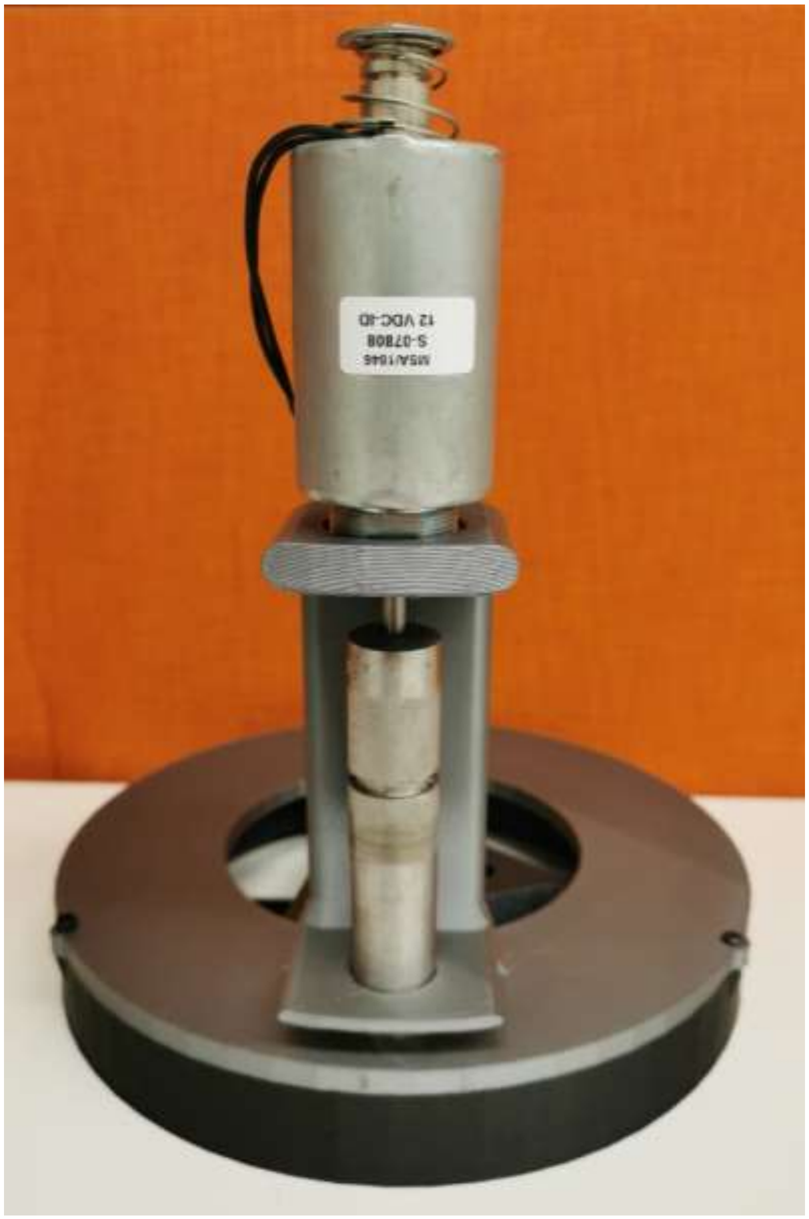


Figure 2.4: The assembled end effector in the real world

Moreover, we need to calculate the estimated torque to confirm the type of stepper. The circular sample tray and bottom tray are 3D printed using polylactic acid (PLA). The sliding friction coefficient of PLA is 0.492 (Pawlak, W. 2018), the total mass of the circular samples tray and the three-leaf screw shaft is 0.71 kg, and the radius of circular samples tray is 0.1 m, therefore the torque can be estimated using the following equation:

$$M = \frac{2}{3}\mu m g r \tag{2.2}$$

Where:

M = the torque of a disc to overcome friction (N·m)

μ = friction coefficient

m = mass of disk (kg)

g = gravity coefficient

r = diameter of disk (m)

The final result of torque is 0.22 N·m, thus a stepper motor (Nema 17 Stepper Motor) with 0.46 N·m is selected since we need to take the break-out torque into consideration. The rotation angle for one step is 1.8°, thus 10 steps are needed for one rotation since the required rotation angle is 18°. The picture of stepper motor is shown below:

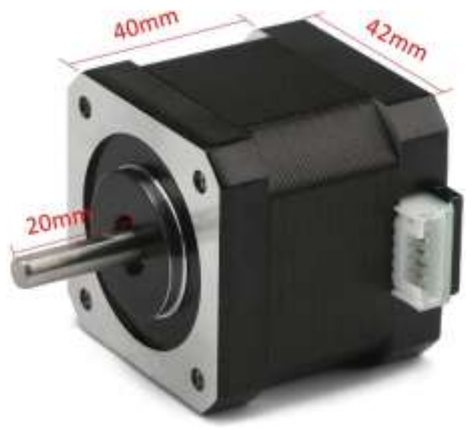


Figure 2.5: Nema 17 stepper motor

2.3 Future Work

In this chapter, only end effector is designed while the appropriate manipulator is not selected yet, therefore the future work includes the design of manipulator for corn leaves, install the manipulator and end effector to the field robotic system in our lab, and test the whole leaf sampling process in the field.

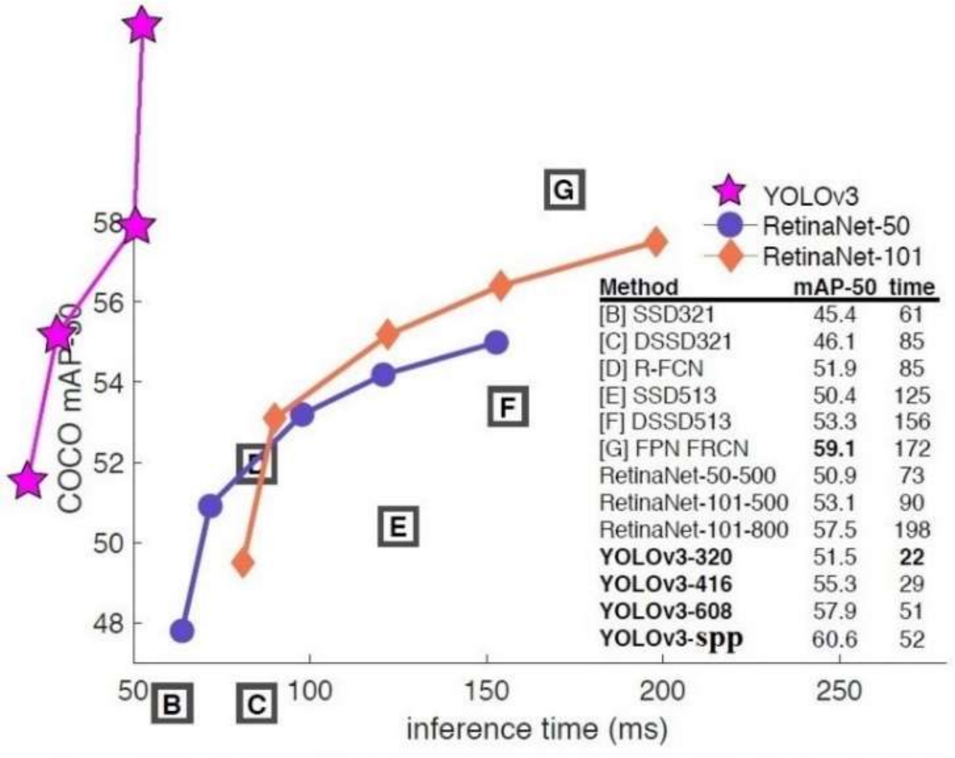
CHAPTER 3: Detection of Object Leaves

3.1 Selection of Neural Network Architecture

There are diverse neural network architectures for the image processing currently benefited from the rapid development of deep learning technology in recent years. If you want to classify the object and generate the mask for instance segmentation in the meantime, Mask RCNN is a good option. It takes image as an input, pass it through a conv-net to get ROI proposals at multiple locations and scales, reshape the ROIs, and pass them through a fully connected network to do bounding box classification and mask predicting inside the ROI (Girshick, 2015; Ren et al., 2015; He et al., 2017). However, You Only Look Once (YOLOv3) is utilized in this thesis based on the following reasons:

Firstly, YOLOv3 uses a new approach compared to the prior neural network architectures such as Mask RCNN which utilize classifiers or localizers for the recognition. However, Yolov3 use a totally different approach. It applies a single neural network to the full image which is the meaning of You Only Look Once. In this model, the input image is divided into regions and bounding boxes and probabilities are predicted for each region. The predicted probabilities then become the weight of these bounding boxes (Redmon and Farhadi, 2018).

Secondly, YOLOv3 is extremely fast and accurate. As illustrated in figure 3.1, in mAP measured at .5 IOU, YOLOv3 is on par with Focal Loss but about 4x faster (Redmon and Farhadi, 2018). IOU is the abbreviation of Intersection over Union which is equal to area of overlap divided by area of union for the predicted bounding box and the ground-truth bounding box. In figure 3.1, the line of Yolov3 is very high and far to the left, so it is much faster and more accurate than other methods. Yolov3-spp can get 60.6 IOU score at 52 second. Moreover, user can easily tradeoff between speed and accuracy simply by changing the size of the model. For example, if fast speed is more important, Yolov3-320 could be chosen. If best accuracy is more important, Yolov3-spp could be chosen.



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51
YOLOv3-spp	60.6	52

Figure 3.1: Speed/accuracy tradeoff on the mAP at .5 IOU (Redmon and Farhadi, 2018)

3.2 Dataset

In the detection process, an appropriate corn leaf should be selected at first. The corn leaves with high horizontal level are the perfect object since we do not need to rotor the end effector of the leaf sampler too much. A total of 1100 images are labeled, and 1000 images are randomly selected as the training dataset while the rest 100 images are the testing dataset. In the dataset, all corn leaves with high horizontal levels are labeled even though some part of leaves might be coved by other leaves. Figure 3.2 shows some examples of labeled picture.

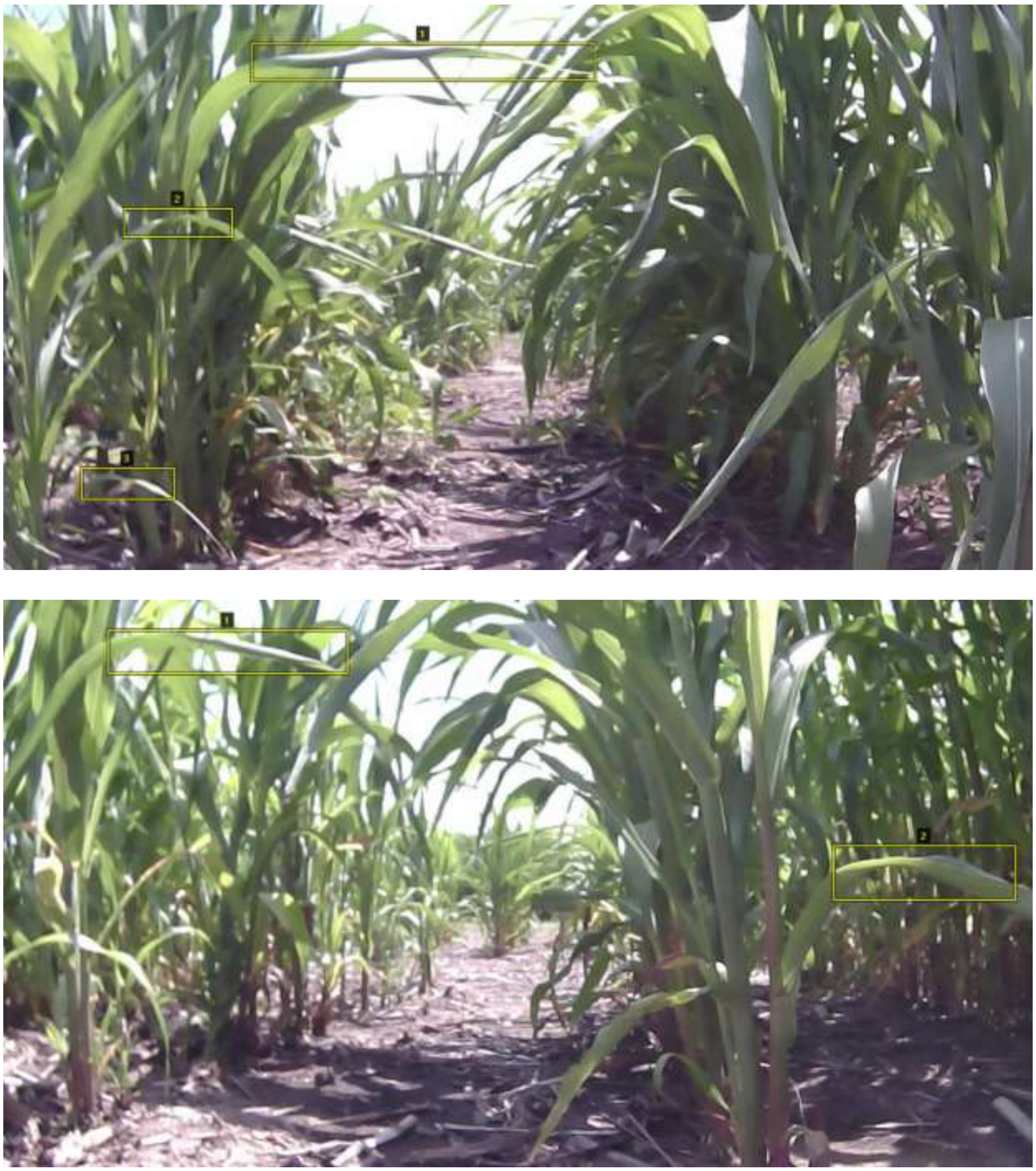


Figure 3.2: Some examples of labeled picture

3.3 Training process

Hyper-parameters are crucial for the training process and we need to adjust the hyper-parameters according to the loss on training dataset and validation dataset. There are many tips on the adjustment of hyper-parameters. Firstly, we can decrease the number of epochs, using data augmentation, and using regularization to solve the overfitting problem. As indicated in figure 3.3, the overfitting problem is a phenomenon where the loss of training set will finally go to zero with increasing number of iterations, but the loss of validation set will decrease first and then increase. In this thesis, no validation dataset is divided since the limitation of labeled image quantity, thus the accuracy on testing dataset is used to substitute the loss on validation dataset. Moreover, if the oscillation amplitude of the loss curve is large and cannot converge to a stable value, we can try to decrease the initial learning rate or increase batch size. Lastly, we can choose different optimizers to control the output curves and final result. Using Adam optimizer can accelerate the convergence of the model, but using stochastic gradient descent (SGD) with a apposite learning rate can get a more accurate model on the testing dataset. The adjusted hyper-parameters for training are shown in the table 3.1.

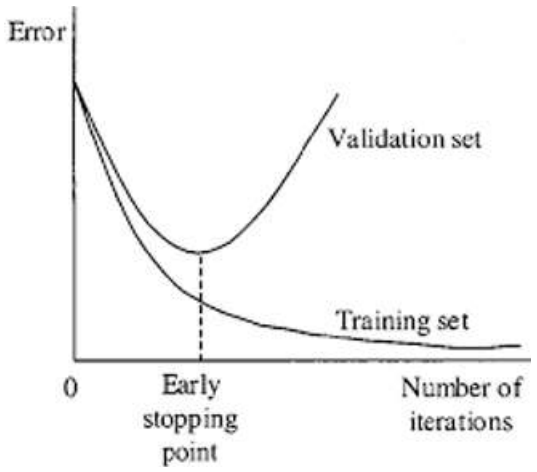


Fig. 3.3: Tendency of error for training set and validation set (https://www.sohu.com/a/276252380_473283)

Table 3.1: The adjusted hyper-parameters for training

Category	Hyper-parameters	Value
Normal	Number of Epochs	200
	Batch Size	16
	Image Size	736
Optimizer	Name	SGD
	Initial Learning Rate	0.001
	Momentum	0.9
	Regularization Coefficient	0.0005
Learning Rate Scheduler	Step Size	10
	Scale Coefficient	0.1
Data augmentation	Scale of Image Brightness	0.18
	Scale of Image Translate	0.12
	Image Scaling	0.16
	Probability of Horizontal Flip	0.5

Data augmentation technology is applied in the training to enlarge the dataset since we only have 1000 images in the training dataset. In neural networks, images with minor changes, such as flips, translations, rotations, and scaling, are treated as different images. For instance (fig. 3.4), the corn leaves image after horizontal flip become a different image for YOLOv3. However, not all data augmentation methods are effective for the data set, we need to determine the category based on our object to make sure not to add irrelevant data. In this research, we want to detect corn leaves with high horizontal level, thus the images should not be rotated since the leaf angle will change. Similarly, the hue and saturation of images should not be changed since the leaf color will not be green.

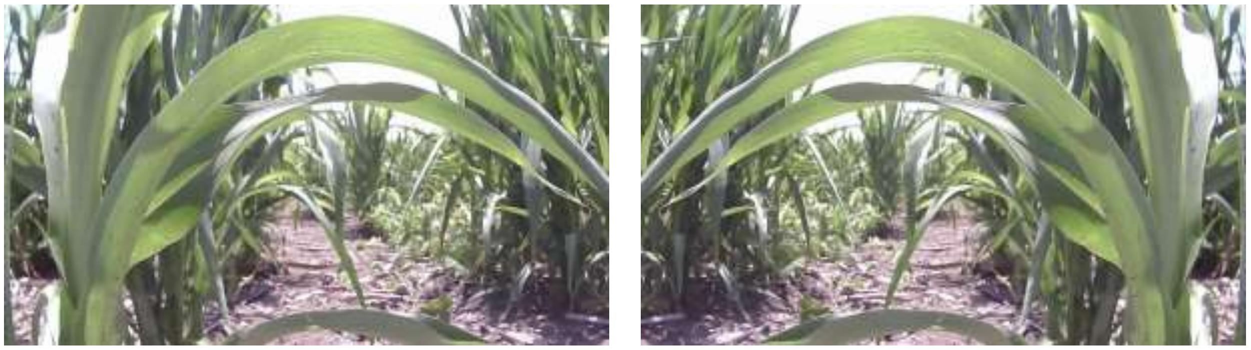
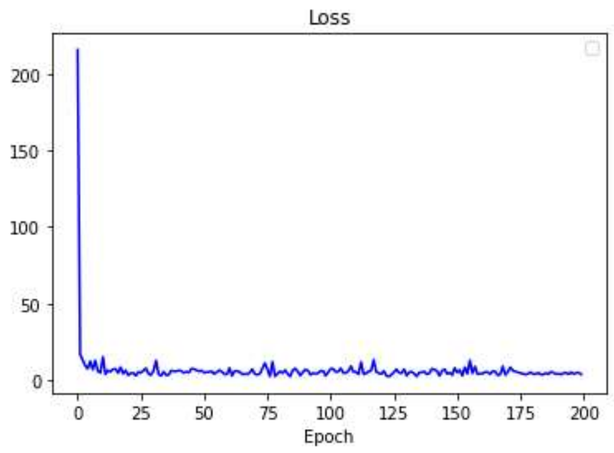
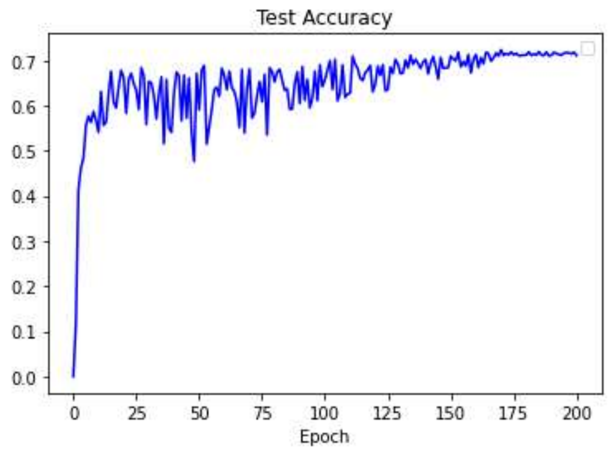


Figure 3.4: An example of horizontal flip

As illustrated in figure 3.5, the train loss and test accuracy are converged at last, and the average precision measured at .5 IOU threshold of 200th iteration is 71.10% which is a relatively low value in the image recognition field. Possible reasons are as follows: (1) The size of training dataset is not big enough even though augmentation technology is applied. (2) Transfer learning is not utilized since the trained weight from a related task is difficult to find. However, this accuracy is enough for the object detection in this thesis since there is no need to recognize all the corn leaves with high horizontal level in the environment if we merely want to choose some appropriate corn leaf for sampling.



(a)

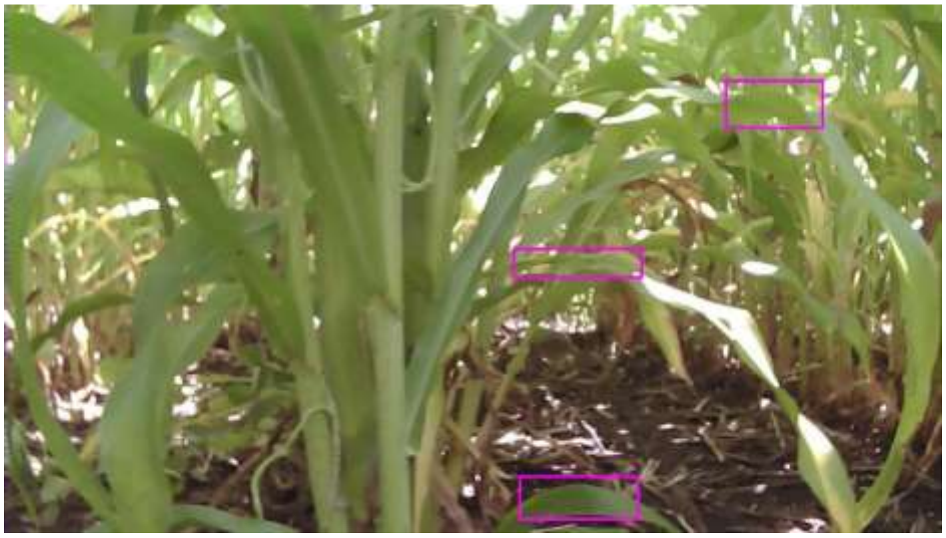


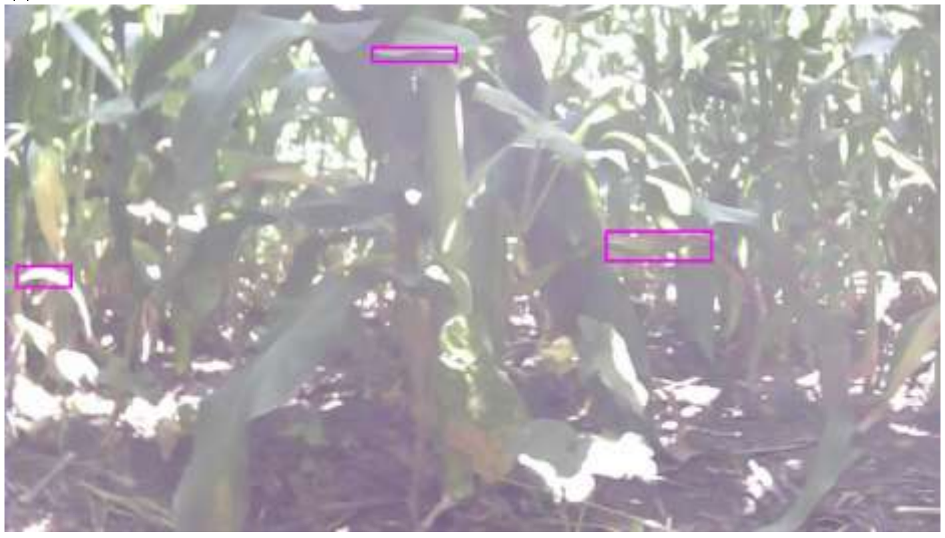
(b)

Figure 3.5: Results of train loss and test accuracy

3.4 Object Detection

The figure 3.6 below are the test results of using weight in 200th iterations. The results show that Yolov3 is very powerful even there are only 1000 images in the training dataset. It also works well when the environment is in high brightness condition or part of leaf is covered by other leaves.


(a) Normal condition


(b) High brightness condition

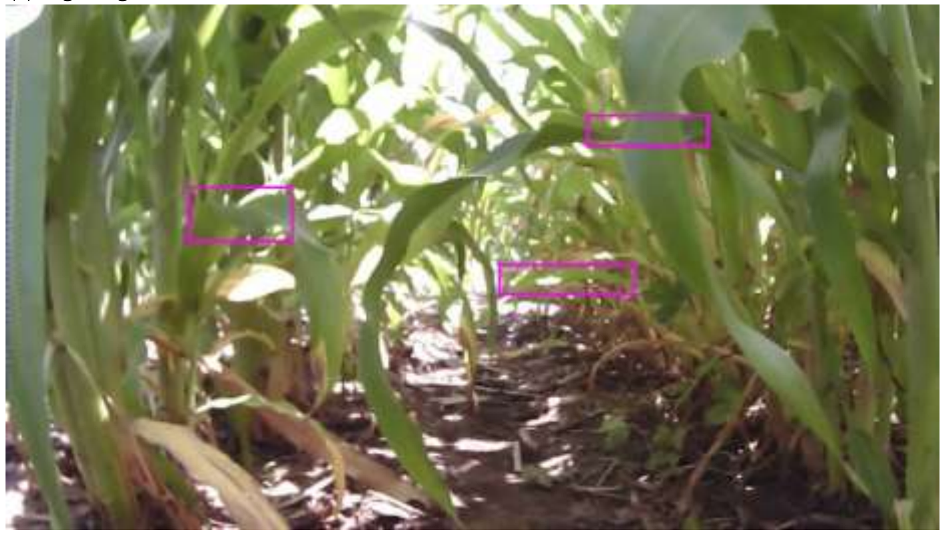

(c) Condition where part of leaf is coved by other leaves

Figure 3.6: Test result from the trained YOLOv3 model

3.5 Future Work

Transfer learning is a machine learning technique where a model trained on one task is re-purposed and improved on a second related task through the learned knowledge inside the pretrained weight (Torrey and Shavlik, 2010). Transfer learning is a very useful technique when the size of training dataset is small. You can download the pretrained weight, freeze all the layers in the model except the last fully connected layer, and then start training the last layer using the pretrained weight. However, using the model pretrained on the COCO dataset (Lin et al., 2014) for the transfer learning, the train loss is high and the test accuracy is low which means no knowledge is distilled from the pretrained model for the detection of the corn leaves with high horizontal level. Therefore, future work includes finding the model pretrained on a large and related dataset and using this model to apply the transfer learning technology. If an appropriate pretrained model for our dataset is not available, more labeled data would be needed to further improve the model accuracy of detecting required object. Moreover, neural network architectures are updated in YOLOv4 (Bochkovskiy et al., 2020) and YOLOv5 (Jocher, 2020), thus our dataset can be fed into the new models to get better performance for horizontal leaves detection.

CHAPTER 4: Detection of Leaf Rolling Angle and Grasp Point

4.1 Framework

In this research, image data and 3D point cloud data are used for the sensor fusion. As illustrated in the framework (fig. 4.1), image data are first fed into YOLO network for object detection, then the corresponding 3D point cloud data are projected into the image plane. After that, only the points inside the bounding boxes are remained and the DBSCAN algorithm is used to cluster these residual points. Lastly, each cluster of points can be used to estimate leaf rolling angle and grasp point for the attitude control of end effector.

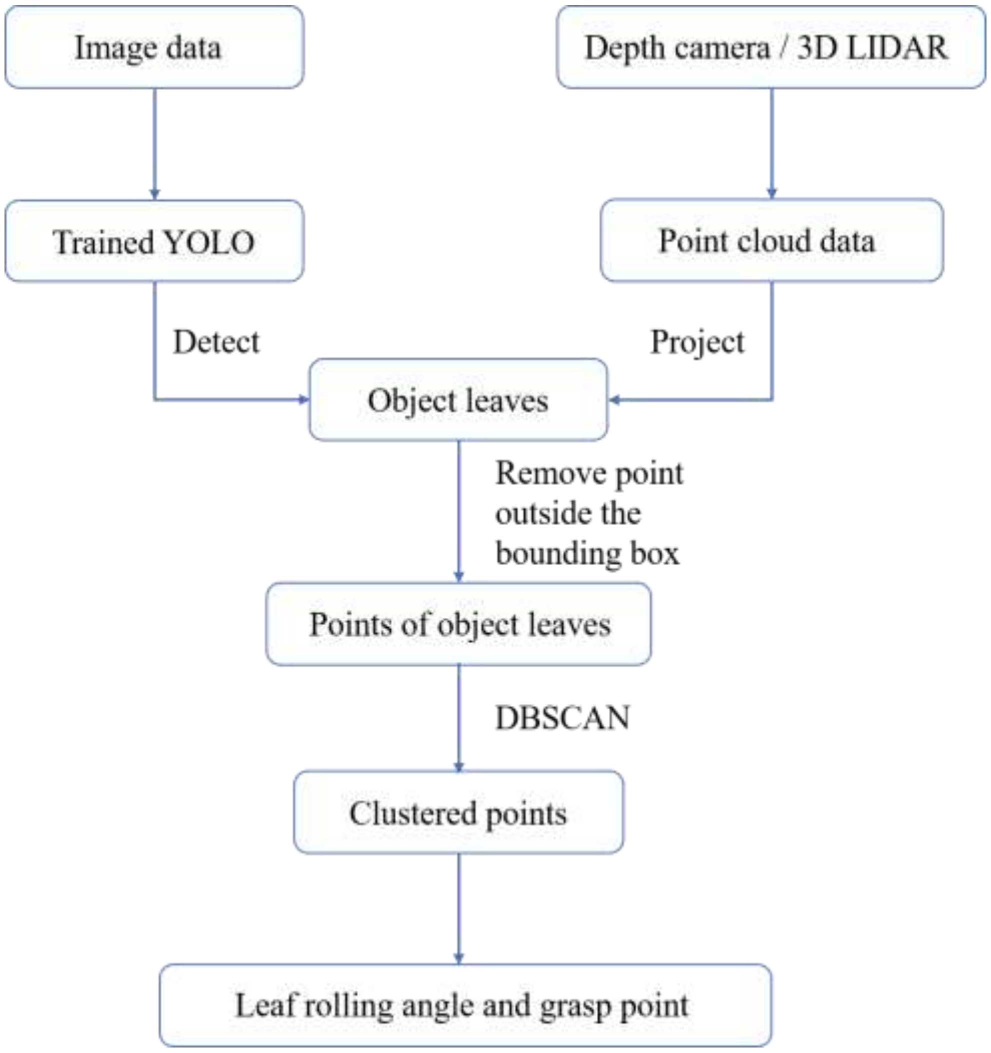


Figure 4.1: Framework of sensor fusion method in this research

4.2 Calibration

The following equations are required in order to project a 3D point X in world coordinates to a point Y in the image plane:

$$kY = PT_{\mathrm{velo}}^{cam}X \tag{4.1}$$

$$P = \begin{pmatrix} \alpha & \beta & c_u & 0 \\ 0 & \gamma & c_v & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{4.2}$$

$$T_{\mathrm{velo}}^{cam} = \begin{pmatrix} \mathrm{R}_{\mathrm{velo}}^{cam} & \mathrm{t}_{\mathrm{velo}}^{cam} \\ 0 & 1 \end{pmatrix} \tag{4.3}$$

Where:

$X = (x, y, z, 1)^T$, a 3D point in world coordinates (m)

$Y = (u, v, 1)^T$, a 2D point in image plane (pixel)

$k =$ normalized coefficient

P = the intrinsic projection matrix of the camera

T_{velo}^{cam} = the extrinsic matrix between Lidar coordinates and camera coordinates

(c_u, c_v) = the principal point where the optic axis intersects the image plane (pixel)

$\mathrm{R}_{\mathrm{velo}}^{cam}$ = the rotation matrix between Lidar coordinates and camera coordinates

$\mathrm{t}_{\mathrm{velo}}^{cam}$ = the translation matrix between Lidar coordinates and camera coordinates

Through the calibration of camera, P which is intrinsic matrix can be calculated. Similarly, the extrinsic matrix T_{velo}^{cam} can be computed via the joint calibration of camera and Lidar.

4.2.1 Calibration of Camera

In the calibration of camera, distortion coefficients are also considered since real lenses usually have some distortion, mostly radial distortion and slight tangential distortion. As shown in figure 4.1 and figure 4.2, the light is more curved away from the center of the lens than near the center in radial distortion, and tangential distortion is caused when the camera plane and lens are unparallel.

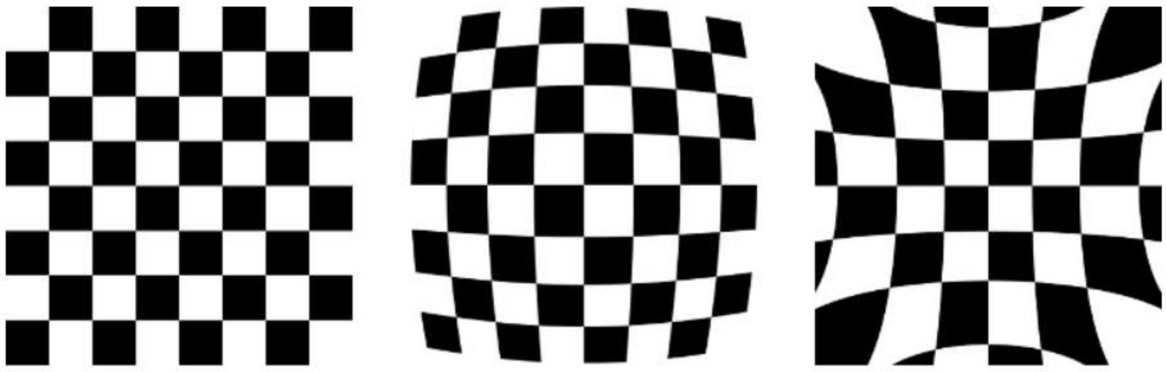

(a) No distortion (b) Positive radial distortion (c) Negative radial distortion

Figure 4.1: An example of radial distortion

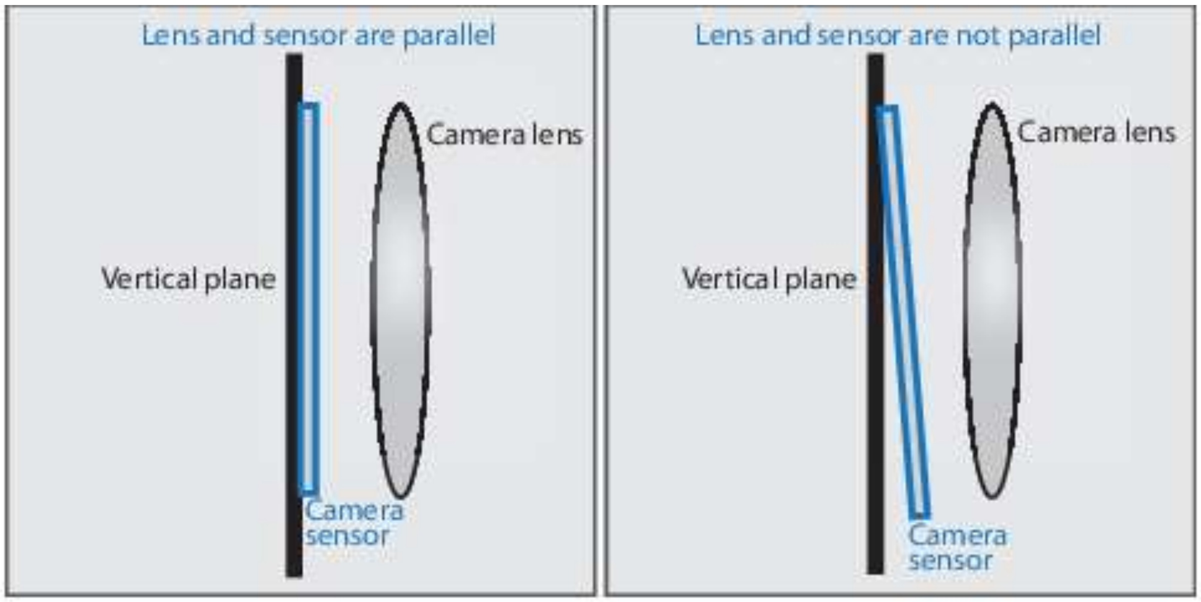


Figure 4.2: An example of tangential distortion (taken from http://www.mathworks.com/help/vision/ug/camera-calibration.html)

Taking radial distortion and tangential distortion into consideration, the equation (4.1), (4.2) and (4.3) become following equations:

$$(x_1, y_1, z_1)^T = \mathrm{R}_{\mathrm{velo}}^{cam}(x, y, z)^T + \mathrm{t}_{\mathrm{velo}}^{cam} \tag{4.4}$$

$$x_2 = x_1/z_1, y_2 = y_1/z_1 \tag{4.5}$$

$$x_3 = x_2 \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + 2p_1 x_2 y_2 + p_2(r^2 + 2{x_2}^2) \qquad (4.6)$$

$$y_3 = y_2 \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + 2p_2 x_2 y_2 + p_1(r^2 + 2{y_2}^2) \qquad (4.7)$$

$$(u, v, 1)^T = \begin{pmatrix} \alpha & \beta & c_u \\ 0 & \gamma & c_v \\ 0 & 0 & 1 \end{pmatrix} (x_3, y_3, 1)^T \qquad (4.8)$$

Where:

$r^2 = {x_2}^2 + {y_2}^2$

$k_1, k_2, k_3, k_4, k_5, k_6 =$ radial distortion coefficients

$p_1, p_2 =$ tangential distortion coefficients

In the camera calibration method from Zhang (2000), k_4, k_5 *and* k_6 are not modeled since they are usually slight. In this method, either the camera or the chessboard plane can be moved freely to get *n* images of chessboard plane under different orientations (fig. 4.3) where *m* feature points on the chessboard are detected, then the intrinsic projection matrix and remained distortion coefficients can be solved by minimizing the following equation using gradient descent method:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} |Y_{i,j} - \hat{m}(P, R_\mathrm{i}, t_\mathrm{i}, X_{i,j})|^2 \qquad (4.9)$$

Where:

$\hat{m}(P, R_\mathrm{i}, t_\mathrm{i}, X_{i,j})$ = the projection of point $X_{i,j}$ in image *i* according to the equation (4.4) to equation (4.8) where the z axis of chessboard plane is set to zero.

$R_\mathrm{i}, t_\mathrm{i}$ = the rotation matrix and translation matrix between chessboard plane in image *i* and camera coordinates.

The calibration results are shown below:

$$P_1 = \begin{pmatrix} 654.98866183279665 & 0 & 662.02409727903864 \\ 0 & 639.94394769101689 & 394.28107906715690 \\ 0 & 0 & 1 \end{pmatrix}$$

$$D_1 = (-0.50423877743644596, -0.0015373510649772425, -0.048608818411876836,$$
$$-0.0025829818360963387, 0.35058193751798555)$$

$$P_2 = \begin{pmatrix} 612.547119140625 & 0 & 319.91510009765625 \\ 0 & 611.4494018554688 & 236.07823181152344 \\ 0 & 0 & 1 \end{pmatrix}$$

$$D_2 = (-0.434515956086, 0.156558117566, -0.00678458464834, 0.00792117520961, 0)$$

Where:

$P_1, D_1 =$ the intrinsic projection matrix and distortion coefficients of the USB board camera (ELP-USBFHD01M-L21). $D_1 = (k_1, k_2, p_1, p_2, k_3)$.

$P_2, D_2 =$ the intrinsic projection matrix and distortion coefficients of the RGB camera in RealSense D435i. $D_2 = (k_1, k_2, p_1, p_2, k_3)$.

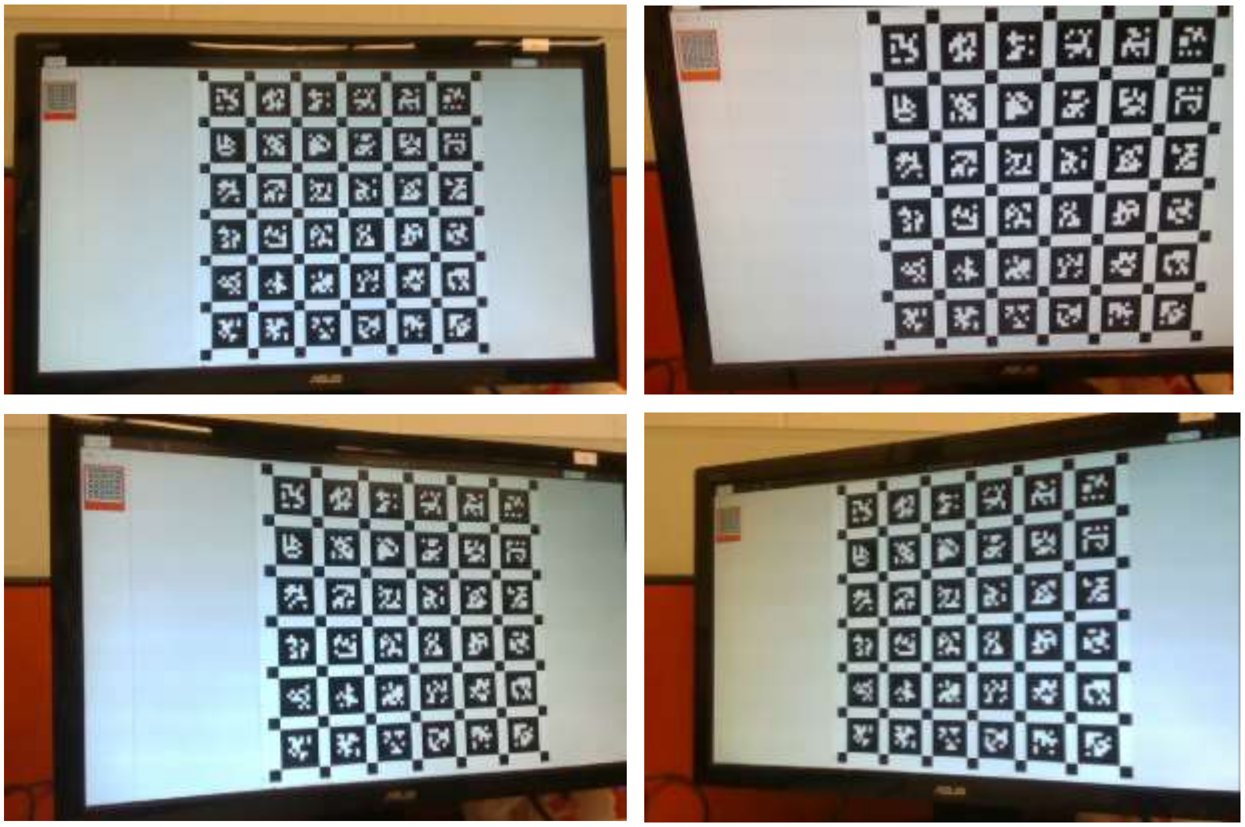


Figure 4.3: Images of chessboard plane under different orientations

4.2.2 Joint Calibration of Camera and Lidar

The camera-lidar calibration toolkit in Autoware (Kato et al., 2014) is used for the joint calibration of Camera and Lidar. As indicated in figure 4.4, the chessboard plane needs to be captured in both image and 3D point cloud data in the calibration process. The capture in image is automatically generated by the program, then the capture in 3D point cloud needs to be drew out by hand with the green circle, and the red points are captured points of the chessboard plane. The green circle contains at least two lines which form a plane where the calibration chessboard plane is located. The attitude of LIDAR can be inferred from the Angle of this plane, and the position of LIDAR can be calculated according to the position of these captured points, then the extrinsic matrix between Lidar coordinates and camera coordinates can be computed in the case where the intrinsic projection matrix of the camera is already known. The result of extrinsic matrix between Lidar coordinates and camera coordinates is shown below:

$$T_{\text{velo}}^{cam} = \begin{pmatrix} -0.55984359921 & -0.11251772247 & 0.82092320381 & 5.17331840968 \\ -8.28536642582 & 8.81032941671 & -5.52960072206 & -3.82111782884 \\ -1.01082305669 & -9.89736112145 & -1.42549121322 & -5.99329646312 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

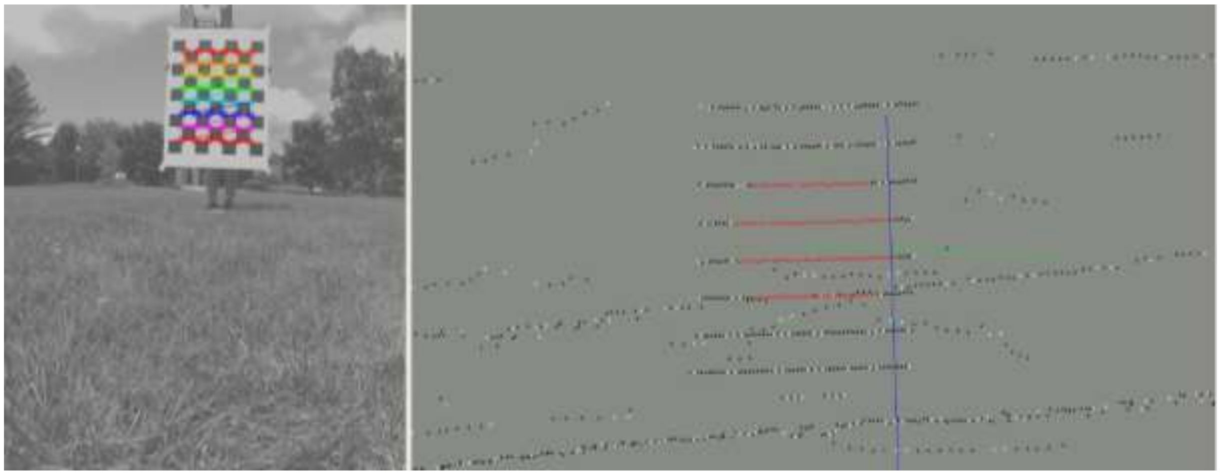


Figure 4.4: One frame of captured chessboard plane in image and 3D point cloud

For RealSense D435i, the camera part and the depth camera part are integrated, thus the extrinsic matrix between depth camera coordinates and camera coordinates is fixed and provided by the RealSense. The result is shown below:

$$T_{\text{depth}}^{cam}=\begin{pmatrix} 0.999996185303 & 0.002035569865 & 0.001876220456 & 0.0147790554911 \\ -0.00203593517 & 0.999997913837 & 0.000192823587 & 0.0001829413085 \\ -0.00187582406 & 0.000196642708 & 0.999998211861 & 0.0002062431449 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4.3 Projection of Point Cloud data

As indicated in the projection result of VLP-16 in the environment for calibration (fig. 4.5), the point cloud of chessboard is projected to the image plane accurately since the chessboard's edges in image and projected point cloud are coincident.

However, for the projection results of VLP-16 in corn field (fig. 4.6), the VLP-16 doesn't work very well since the resolution is too low to identify corn leaves. That's why RealSense D435i is used instead of VLP-16 in this research.

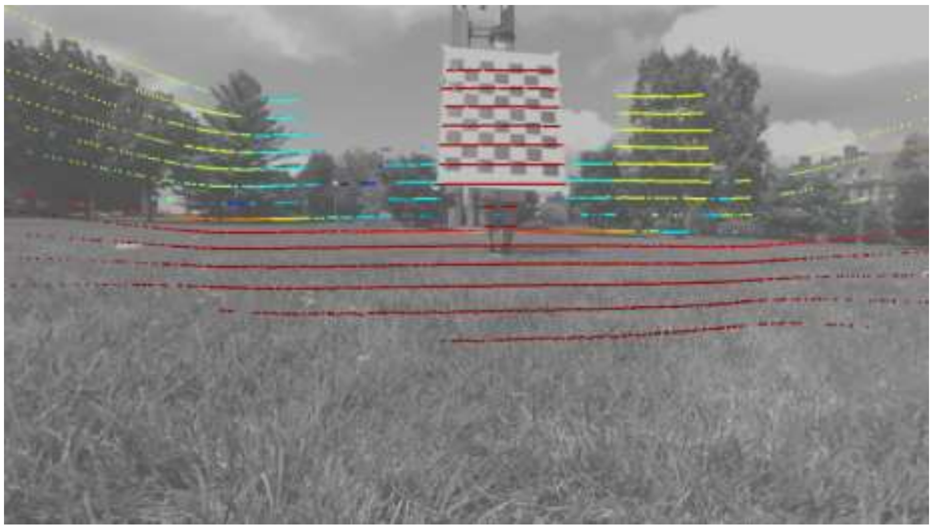


Figure 4.5: Projection of point cloud from VLP-16 in the environment for calibration

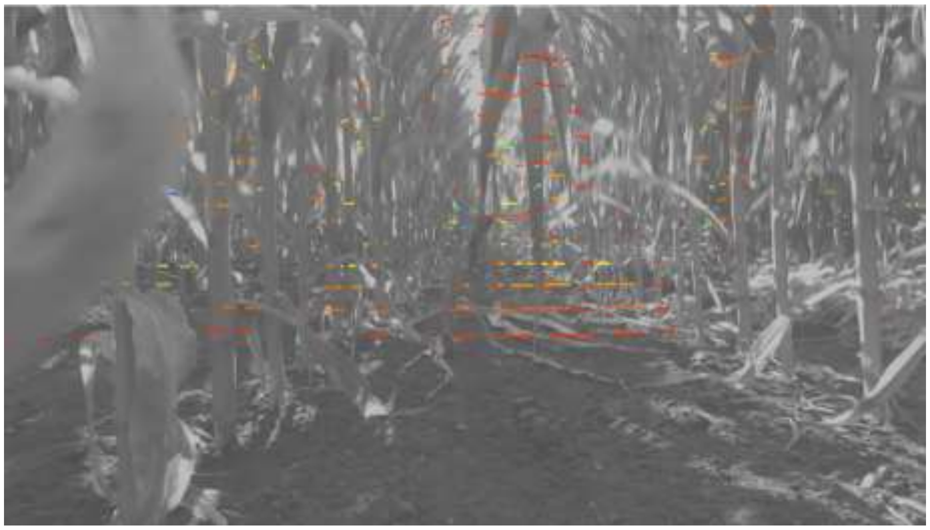
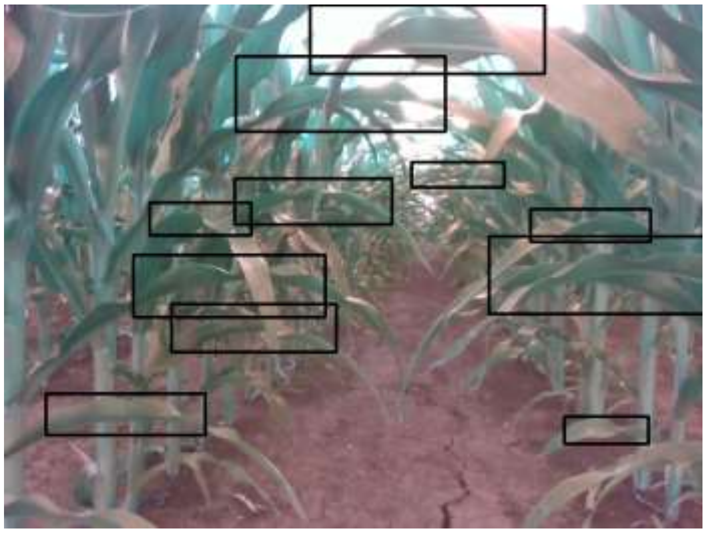
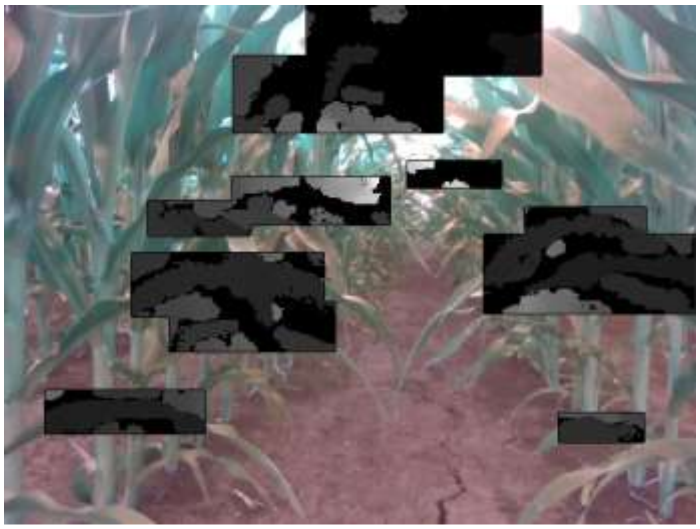


Figure 4.6: Projection of point cloud from VLP-16 in the corn field

For the Projection of point cloud from RealSense D435i in the corn field (fig. 4.7), the projected points have the same shape as the leaves in the bounding boxes which demonstrates the accuracy of extrinsic matrix. In figure 4.7(b), the black part is the missed points in measurement from depth camera, and for other part inside the bounding boxes, the color is whiter when the distance is larger. The point cloud data from depth camera is accurate when the distance is small, but the number of missed points is increasing when the distance become larger, thus the points whose distance is larger than the threshold is deprecated in the next DBSCAN algorithm. The residual point cloud inside the bounding boxes is shown in the figure 4.8.



(a)


(b)
Figure 4.7: Projection of point cloud from RealSense D435i in the corn field

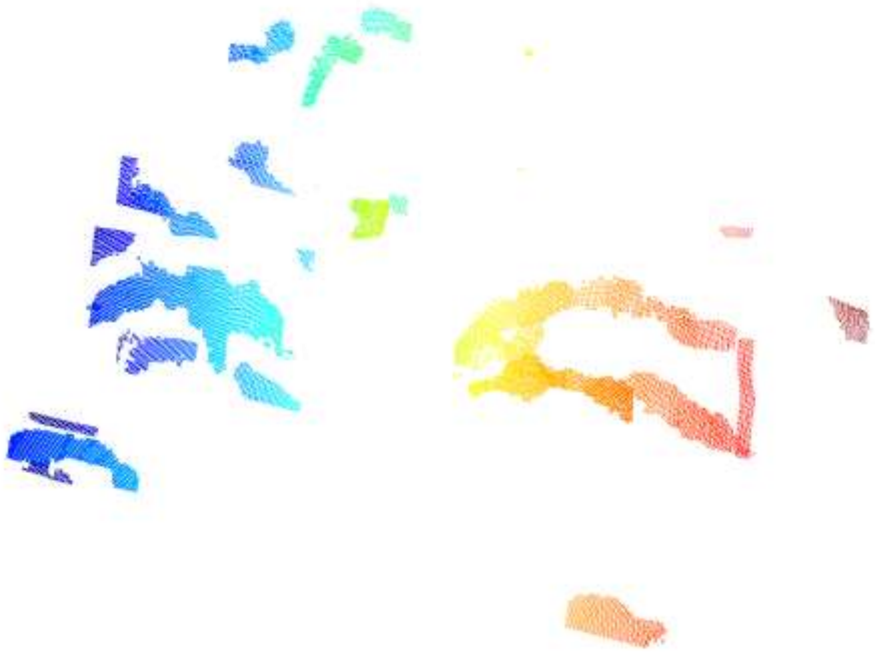


Figure 4.8: The residual point cloud inside the bounding boxes

4.4 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a pioneering density-based algorithm (Ester et al., 1996). It can cluster 2D and 3D points in any shape and size since it is density-based. Moreover, it can also classify noise and outliers in datasets. However, the initial DBSCAN needs to be improved in several aspects: (a) user needs to determine density parameter that is used to find neighboring points and minimum number of points to form a cluster.; (b) it is difficult for users to set density parameter and minimum number of points to form a cluster when datasets include points with varying densities; (c) and the computational cost is high (Khan et al., 2014).

Many enhanced DBSCAN algorithm were proposed to overcome these shortcomings. For instance, in DCBRD (Fahim et al., 2006), clustering can be implemented efficiently without any input parameters from user. For datasets including points with varying densities, Liu et al. (2007) presented VDBSCAN which utilized K-distance plotting to calculate the density parameter automatically while the computational complexity was same as that of DBSCAN. GRIDBSCAN (Uncu et al., 2006) is also capable of dealing with various densities, but the time cost is expensive. In contrast, the time complexity of FDBSCAN (Liu, 2006) is linear which is much less than that of DBSCAN: $O\ (n * \log n)$. In this algorithm, kernel function is introduced to reduce the time complexity and improve the accuracy. Moreover, there are other modified DBSCAN algorithm which aim to improve the accuracy. EI-Sonbaty et al. (2004) provided an enhancement version of DBSCAN to get a better performance from large size of datasets using CLARANS (Ng and Han, 1994) for the pre-processing of datasets. Mahran et al. (2008) presented a grid-based clustering algorithm to get higher accuracy with the utilization of high degree of parallelism.

In this research, FDBSCAN (Liu, 2006) is used to cluster the residual point cloud inside the bounding boxes because of the fast speed and high accuracy and the pseudo-code is shown in Algorithm 1. In the clustering result (fig. 4.9), one color represents one cluster of data, and the algorithm works well on most part of the point cloud except the point cloud of overlapped leaves. After that, one cluster of data is selected based on a factor which is proportional to quantity of points and inversely proportional to distance. In this circumstance, the selected data is the purple cluster inside the red box in figure 4.9 whose front view and top view are shown in figure 4.10.

Algorithm 1: FDBSCAN

```
Input      : D → a dataset including n objects
             eps → the radius of confirming neighborhood of an object
             MinPts → density threshold of neighborhood
Initialize : C = NOISE
Output     : (C_1, …, C_N) → a list of clusters based on density
sort(D)
foreach unvisited point P in dataset D do
    mark P as visited
    NeighborPts = all points within P's eps-neighborhood (including P)
    if sizeof(NeighborPts) < MinPts then
        mark P as NOISE
    else
        C_old = getfirstcoreId(NeighborPts)
        if C_old is not classified then
            C = next cluster
            add NeighborPts to cluster C
        else
            foreach point P' in NeighborPts do
                if P' is not visited then
                    mark P' as visited
                    NeighborPts' = all points within P''s eps-neighborhood (including P')
                    if sizeof(NeighborPts') >= MinPts then
                        NeighborPts = NeighborPts joined with NeighborPts'
                    end
                end
                if P' is not yet member of any cluster then
                    add P' to cluster C
                end
            end
        end
    end
end
```

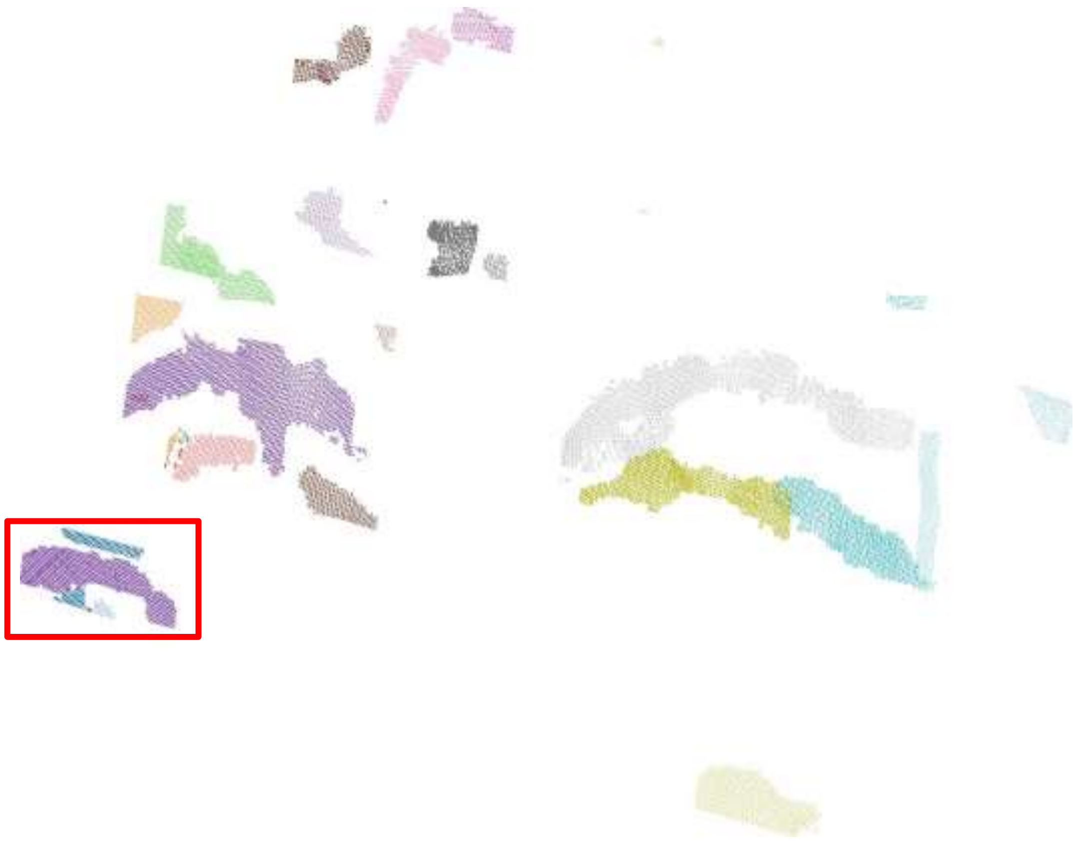
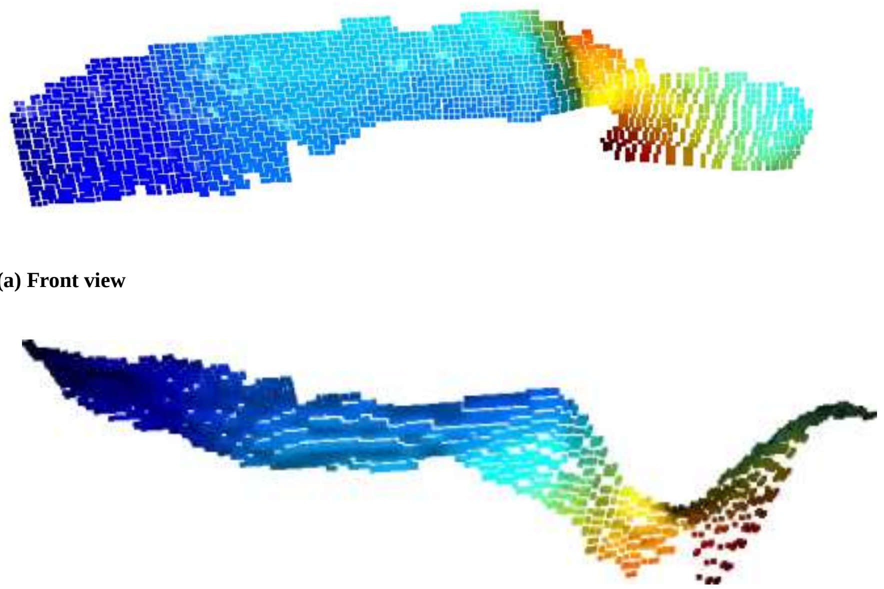


Figure 4.9: The clustering result obtained using DBSCAN


(a) Front view

(b) Top view

Figure 4.10: Front view and top view of selected cluster in figure 4.9

4.5 Leaf Rolling Angle Detection and Grasp point Detection

After object leaf is confirmed, leaf rolling angle needs to be calculated for the attitude control of end effector designed in the chapter 2. As illustrated in figure 4.11, selected leaf is slightly rolling and there are three different rolling angles in this leaf which are obvious in the point cloud of this leaf, thus point cloud data is used to compute the average rolling angles for these three parts.

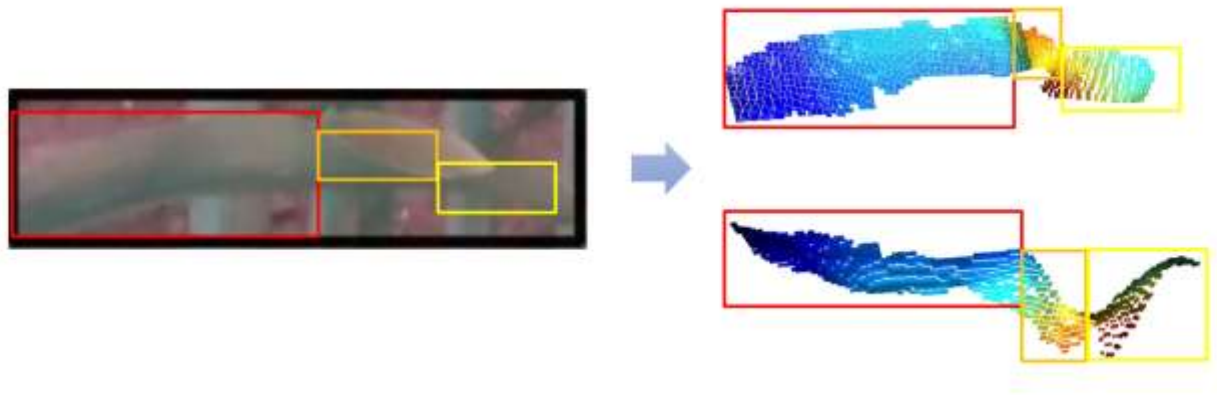


Figure 4.11: The correlation between selected leaf's image and point cloud

Computing the leaf rolling angle is equal to computing the normal vector of point cloud surface which is usually implemented in two ways:

(1) Using surface reconstruction technique to obtain the surface corresponding to the sampling points from dataset, then normal vector is calculated from the surface model.

(2) Normal vector of Surface can be estimated directly from normal vectors of each point in point cloud dataset.

In this research, the second method is chosen since it is much easier to implement. The normal vectors of a point can be approximated by calculating the normal vectors of the plane fitted according to the points in the neighborhood, thus the original problem is transformed into the least square plane fitting estimation problem presented as follow equation:

$$\min_{A,B,C,D} \sum_{i=1}^{n}(Ax_i + By_i + Cz_i + D)^2 \quad \text{s.t.} \quad A^2 + B^2 + C^2 = 1 \tag{4.10}$$

Where:

$x_i, y_i, z_i =$ coordinates of points in the neighborhood

$A, B, C, D =$ the coefficients of 3D plane

By taking the derivation, setting it equal to 0, and eliminating D in the equation set, the following linear system of equations can be obtained:

$$M\begin{bmatrix}A\\B\\C\end{bmatrix} = \begin{bmatrix}\overline{x^2} - \bar{x}^2 & \overline{xy} - \bar{x}\bar{y} & \overline{xz} - \bar{x}\bar{z} \\ \overline{xy} - \bar{x}\bar{y} & \overline{y^2} - \bar{y}^2 & \overline{yz} - \bar{y}\bar{z} \\ \overline{xz} - \bar{x}\bar{z} & \overline{yz} - \bar{y}\bar{z} & \overline{z^2} - \bar{z}^2\end{bmatrix}\begin{bmatrix}A\\B\\C\end{bmatrix} = 0 \quad \text{s.t.} \quad A^2 + B^2 + C^2 = 1 \tag{4.11}$$

Where:

$M =$ covariance matrix

$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

$\overline{xy} = \frac{1}{n}\sum_{i=1}^{n} x_i y_i$, and the rest algebraic expressions are in a similar fashion.

In general, the covariance matrix is nonsingular, so there is no exact solution to the above equation, but PCA (Dunteman, 1989) can be used to obtain the estimated solution which is the normalized eigenvector corresponding to the minimum eigenvalue of the covariance matrix M. One plane can have two opposite directions as normal vectors, and either one could be correct without knowing the global structure of the geometry, thus origin is set as point of sight to orient the normal vectors. The normal vectors after unity of direction are shown in the figure 4.12.

Then the angles between each normal vector and angle vector [0,1,0] are computed to get the leaf rolling angle distribution (fig. 1.13). We can assume this distribution is the mixture of several Gaussian distribution and use EM algorithm (Xuan et al., 2001) to get the average and variance of these Gaussian distribution where k-means algorithm (Wagstaff et al., 2001) is used for the initialization. However, the right number k of clusters is not obvious, thus G-means algorithm is applied to automatically choose k. In this algorithm, k is initialized to 0, and k will keep increasing until the clusters assigned to each k-means center follow the Gaussian distribution. Anderson-Darling statistic test is utilized to detect whether the cluster around the center are sampled from Gaussian distribution. The pseudo-code of k-means algorithm, G-means algorithm and EM algorithm are illustrated below.

Algorithm 2: *k*-means

```
Input      : $X \rightarrow$ a dataset including $n$ objects
             $k \rightarrow$ number of clusters
Initialize : $Iter = 0$
             $MaxIter$ = Maximum allowable iterations
             $(\mu_j)_{j=1}^{k} = randn(dim_x, k) \rightarrow$ initialize $k$ centers randomly
Output     : $(l_j)_{j=1}^{k} \rightarrow$ a list of clusters
             $(\mu_j)_{j=1}^{k} \rightarrow$ a list of centers
while $Iter < MaxIter$ do
    foreach $x_i$ in $X$ do
        Compute $d_{ic} = ||x_i - \mu_c||^2$ for $c = 1, \dots, k$
        $z_i = argmin_c(d_{ic})$
    end
    Update clusters: $l_j = \{ x_i \mid z_i = j \}$
    foreach $l_j$ do
        Compute $\mu_j = \frac{1}{|l_j|} \sum_{x \in l_j} x$
    end
    $Iter$ += 1
end
```

Algorithm 3: G-means

```
Input      : $X \rightarrow$ a dataset including n objects
             $\alpha \rightarrow$ the confidence level
```

```
Initialize  :  $k = 1$
               $C = \{\mu_1\} = \{\bar{X}\}$
Output      :  $C = (\mu_j)_{j=1}^{k}$ → a list of centers
while True do
    $(\mu_j)_{j=1}^{k}, (l_j)_{j=1}^{k}$ = kmeans(X, k, C)
    foreach $l_j$ do
        if $ADstat\left((l_j)_{j=1}^{k}\right) > \alpha$ do
            delete $\mu_j$ from C
            randomly choose two centers from $l_j$
            add these two centers to C
            $k$ += 1
    end
    if length of C does not change do
        break
    end
end
```

Algorithm 4: EM

```
Input       :  $X$ → a dataset including $n$ objects
               $k$ → number of Gaussian components
               eps → improvement lower bound
Initialize  :  $(\pi_j)_{j=1}^{k} = 1/k$ → the probability over j^th Gaussian component
               $(\Sigma_j)_{j=1}^{k} = I$ → the covariance matrix of j^th Gaussian component
               $(\mu_j)_{j=1}^{k}$ given by k-means → the mean of j^th Gaussian component
               $R = O$ → the responsibility matrix
Output      :  $(\pi_j)_{j=1}^{k}, (\Sigma_j)_{j=1}^{k}, (\mu_j)_{j=1}^{k}$ → updated probability, covariance matrix and mean
change = 2eps
while change > eps do
    for i in range(n) do
        for j in range(k) do
            $p_{\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j}(x_i) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_j|}} \exp\left(-\frac{1}{2}(x_i - \mu_j)^\top \Sigma^{-1}(x_i - \mu_j)\right)$
            $R_{ij} = p_\theta(y_i = j \mid x_i) = \frac{p_\theta(y_i = j, x_i)}{p_\theta(x_i)} = \frac{\pi_j p_{\mu_j,\Sigma_j}(x_i)}{\sum_{l=1}^{k} \pi_l p_{\mu_l,\Sigma_l}(x_i)}$
        end
    end
    for j in range(k) do
        $\pi_j := \frac{\sum_{i=1}^{n} R_{ij}}{\sum_{i=1}^{n}\sum_{l=1}^{k} R_{ij}} = \frac{\sum_{i=1}^{n} R_{ij}}{n}$
        $\mu_j := \frac{\sum_{i=1}^{n} R_{ij} x_i}{\sum_{i=1}^{n}\sum_{l=1}^{k} R_{ij}} = \frac{\sum_{i=1}^{n} R_{ij} x_i}{n\pi_j}$
        $\Sigma_j := \frac{\sum_{i=1}^{n} R_{ij}(x_i - \mu_j)(x_i - \mu_j)^\top}{n\pi_j}$
    end
    change = max(norm($\mu_{1new} - \mu_{1old}$), …, norm($\mu_{knew} - \mu_{kold}$))
end
```

Using the EM algorithms above, the obtained means are [68.16245498, 92.82219438, 49.73753005], the obtained variances are [30.41397885, 66.67874526, 45.74359508], and the obtained weights are [0.41315852, 0.0973933, 0.48944817]. The regular density function of these three Gaussian components is shown in the figure 4.14. According to the calculation result, the average leaf rolling angles of the selected leaf are [68.16245498, 92.82219438, 49.73753005], and the obtained means, variances, and weights are fed into the Gaussian Mixture Model for the classification. As shown in the figure 4.15, the left and right clusters in the classification result from GMM model have some misclassified points since there might be similar angles in different part of leaves. This misclassification could be solved by using the DBSCAN algorithm again and then choosing the biggest cluster. Finally, the coordinates of grasp points are computed by averaging the coordinates of points in three chosen clusters, and the computed result is [(-0.19778947, 0.11839437, 0.51080566), (-0.1522579, 0.11742342, 0.52530924), (-0.13044321, 0.12841471, 0.52953533)]. The coordinates of best grasp point is (-0.19778947, 0.11839437, 0.51080566) since the manipulation of end effector is easier for larger clusters in one leaf.

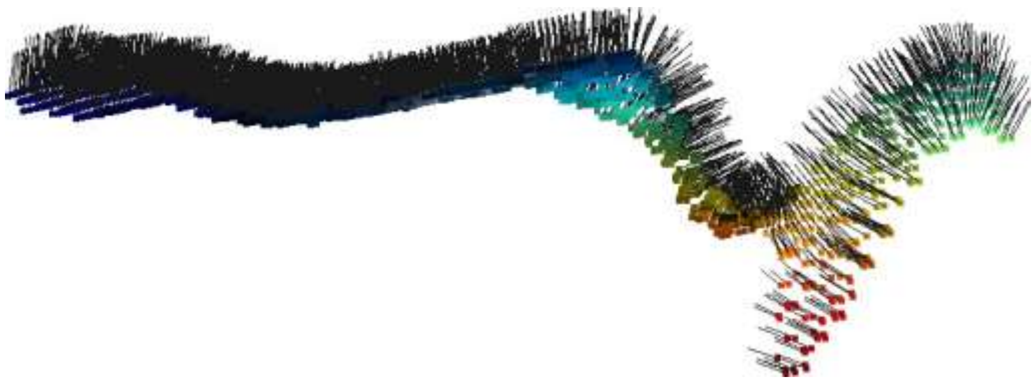


Figure 4.12: The normal vectors of selected point cloud

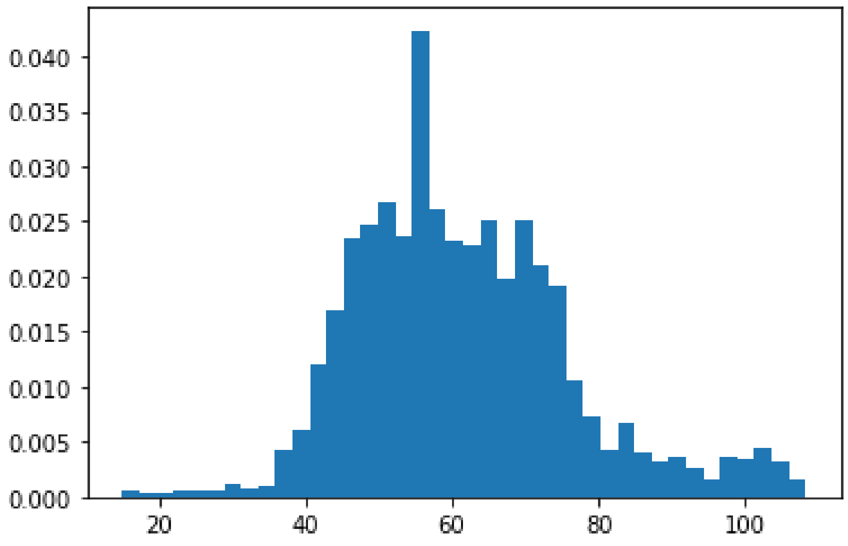


Figure 4.13: The leaf rolling angle distribution

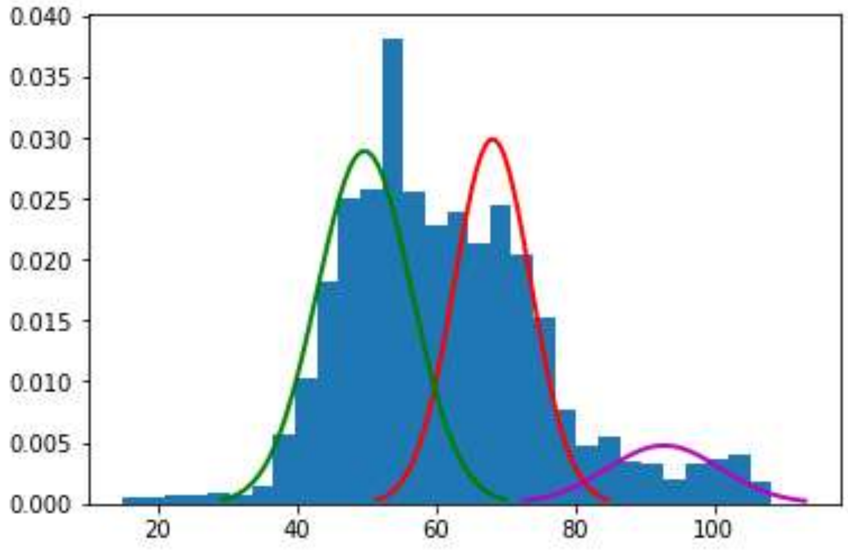


Figure 4.14: The regular density function of three Gaussian components

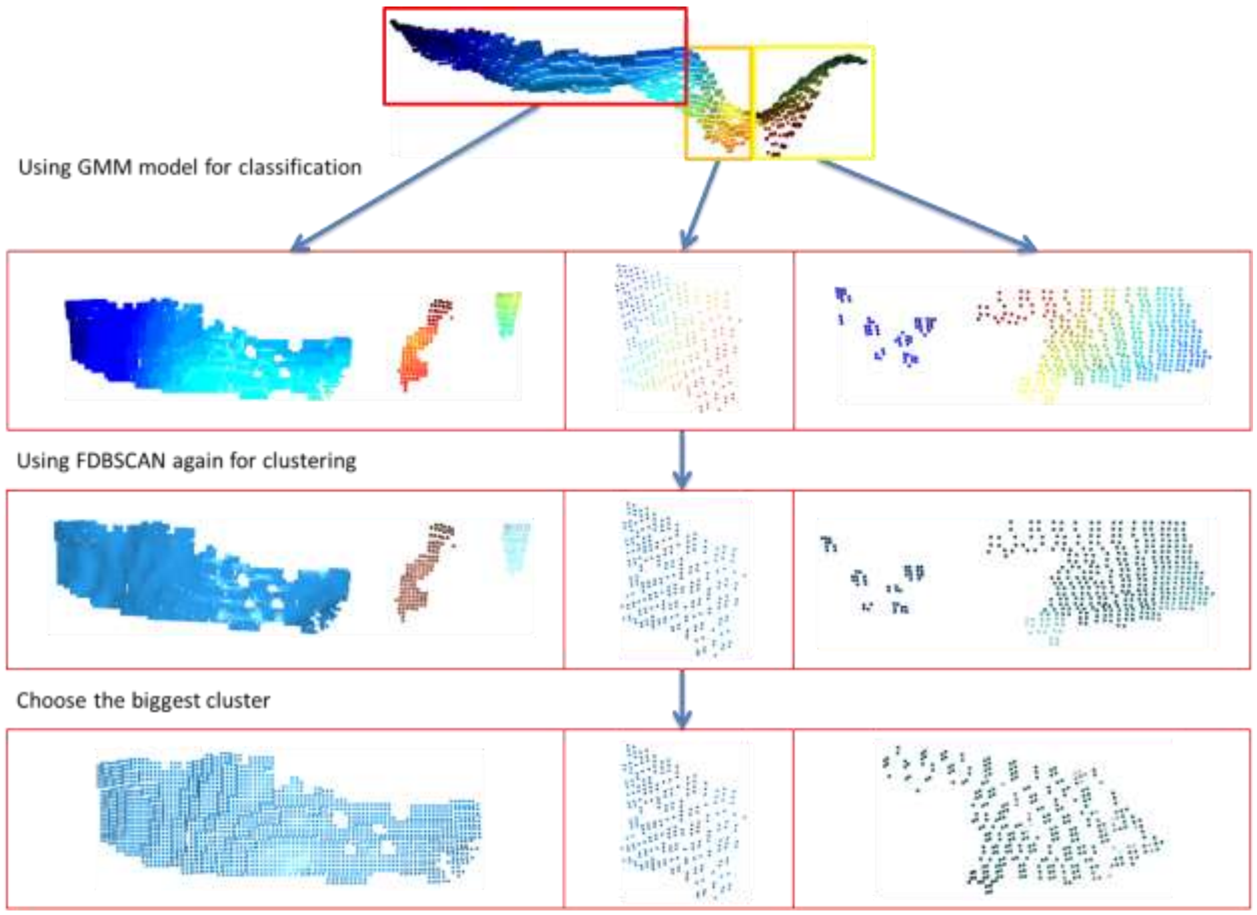

Using GMM model for classification
Using FDBSCAN again for clustering
Choose the biggest cluster

Figure 4.15: The process of computing grasp point

4.6 Future Work

The point cloud of overlapped leaves cannot be separated by DBSCAN algorithm which can cause problems in the calculation of leaf rolling angle and grasp point. In order to avoid these problems, the labeled overlapped leaves in the existing dataset should be deleted, and another YOLOv3 model need to be trained to no longer detect overlapped leaves.

CHAPTER 5: Data Collection and Field Test

5.1 Data Collection

Most data were collected in the corn field outside the energy farm (St. Race, Urbana, Illinois) at the growing and mature stages of corn. After the corn withered, the plastic corn models in our lab are used for the data collection. The following experimental leaf rolling angle measurement is also conducted using the corn models.

5.2 Experimental Leaf Rolling Angle Measurement

In this experiment, data were collected using the depth camera (RealSense D435i) installed on TerraSentia, then the data were fed into the pipeline to get the detection result and computed leaf rolling angle. After that, the detected leaves in the real world were found according to the detection result in the image, then a protractor was used to measure the leaf rolling angle (fig. 5.1). As indicated in the experimental result, totally 46 angles from 24 leaves were measured, and the root mean square error (RMSE) is 6.53 which is acceptable considering the error in the measurement process. The scatter diagram of measured angle and computed angle is shown in the figure 5.2.

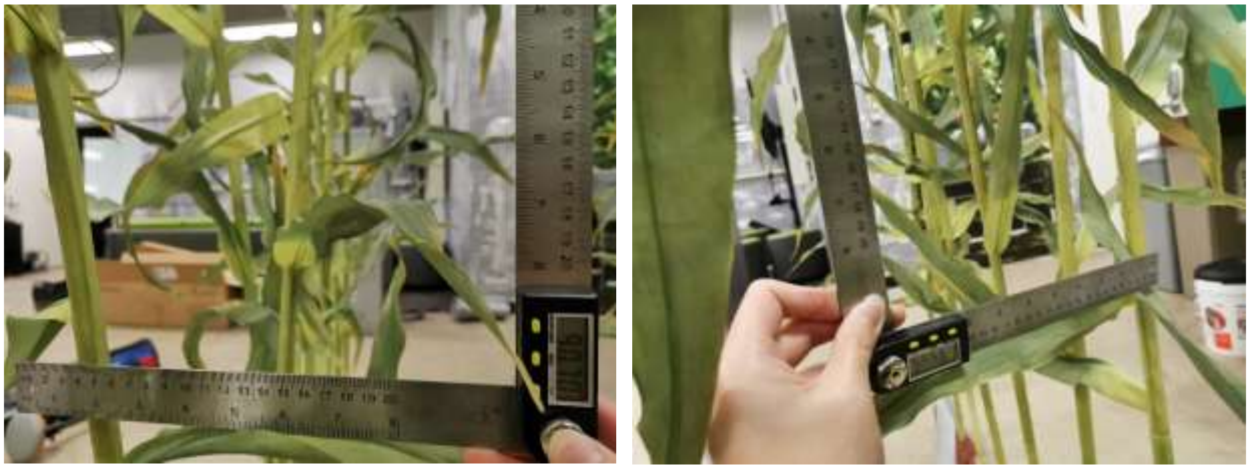


Figure 5.1: Leaf rolling angle measurement

Table 5.1: The experimental result of leaf rolling angle

Leaf number	Measured Angle/°	Computed Angle/°
1	2.60	6.73
1	18.15	18.10
1	36.00	37.74
2	11.30	13.04
2	46.15	44.25
3	18.25	13.10
3	41.90	41.41
4	103.50	102.13
4	162.15	155.85
5	9.05	12.09
6	46.15	40.37
6	88.25	83.12
7	30.15	31.49
7	17.60	14.63
8	11.20	15.20
8	31.70	31.18
9	13.50	10.74
10	4.95	5.31
10	9.65	8.79
11	14.45	14.94
11	9.50	9.63
12	4.90	9.49
13	50.15	52.23
13	12.50	12.86
14	6.40	6.94
14	161.90	163.44
15	94.75	93.61
15	177.25	166.68
16	51.10	48.37
16	25.45	15.31
17	153.10	139.08
17	81.80	88.73
18	19.95	22.42
18	86.75	70.61
19	54.05	38.52
19	25.50	18.50
20	50.15	44.36
20	18.20	19.68
21	20.85	17.49
21	122.65	117.93
22	58.85	51.06
22	36.95	20.36
22	126.25	111.06

23	**1.85**	**8.51**
23	**20.60**	**25.72**
24	**19.75**	**17.72**
RMSE	**6.53**	

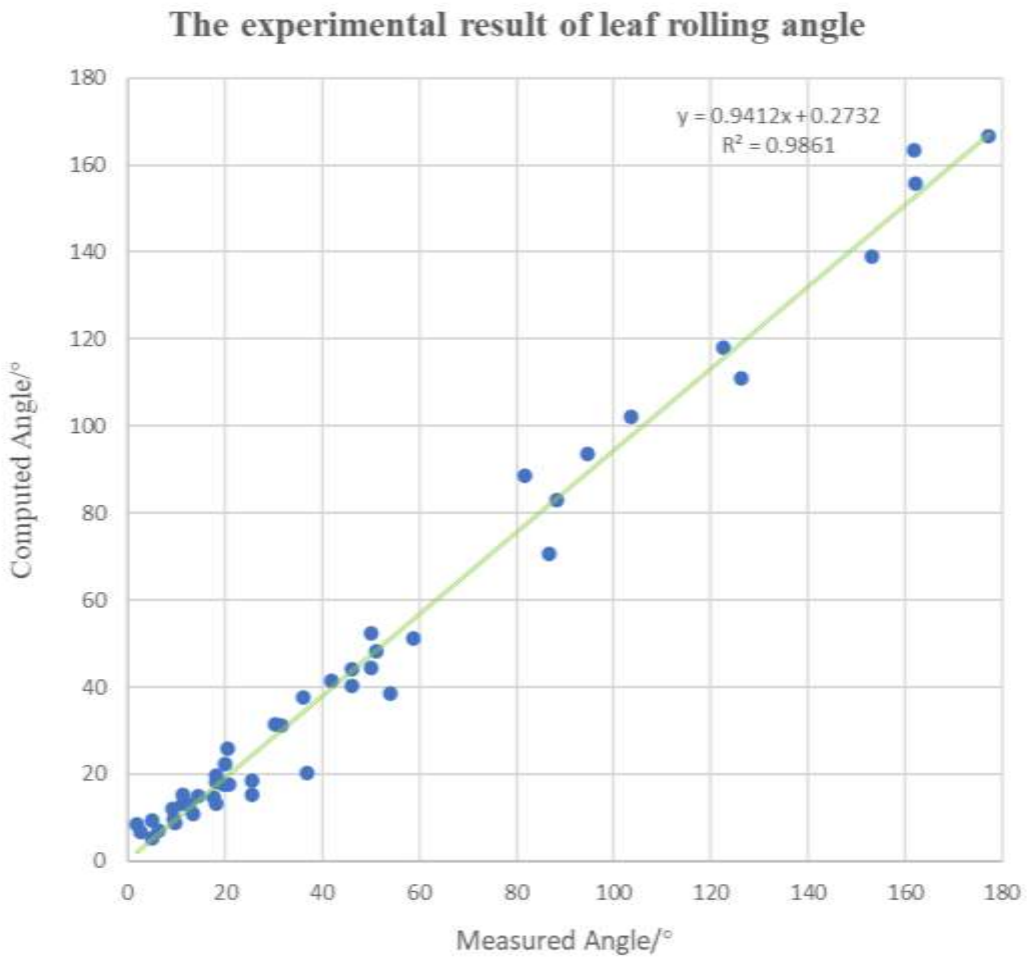


Figure 5.2: The scatter diagram of measured angle and computed angle

CHAPTER 6: Conclusion

This thesis expanded on the process of developing an automatic method of leaf sampling in the field. In terms of hardware, the design of a novel end effector aimed at punching and storing leaf samples separately was presented in detail. In terms of software, an YOLOv3 model was well trained for the detection of leaves with high horizontal level. Moreover, a innovative pipeline using sensor fusion was developed to compute the leaf surface orientation and optimal punching position. In this pipeline, different sensors were calibrated to unified coordinate system, then the point cloud data were projected to the image plane to match detected leaves. With these isolated leaf point cloud inside the bounding boxes, FDBSCAN was utilized for clustering, and the normal vectors of each point in one cluster were calculated to get the leaf rolling angle distribution, then a Gaussian mixture model was applied to compute the multiple different rolling angles in one leaf. Finally, 46 rolling angles from 24 leaves were measured, and the RMSE is 6.5535 which is acceptable since there are also errors in the measurement process.

Reference

[1] Abel, J. (2018). In-Field Robotic Leaf Grasping and Automated Crop Spectroscopy.

[2] Ahlin, K., Joffe, B., Hu, A. P., McMurray, G., & Sadegh, N. (2016). Autonomous leaf picking using deep learning and visual-servoing. *IFAC-PapersOnLine*, *49*(16), 177-183. https://doi.org/10.1016/j.ifacol.2016.10.033

[3] Bachche, S., & Oka, K. (2013). Design, modeling and performance testing of end-effector for sweet pepper harvesting robot hand. *J. Rob. Mech.*, *25*(4), 705-717. https://doi.org/10.20965/jrm.2013.p0705

[4] Birant, D., & Kut, A. (2007). ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & know. eng.*, *60*(1), 208-221. https://doi.org/10.1016/j.datak.2006.01.013

[5] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv: 2004.10934.* https://arxiv.org/abs/2004.10934

[6] Borah, B., & Bhattacharyya, D. K. (2004, January). An improved sampling-based DBSCAN for large spatial databases. In *Proc. ICISIP, 2004.* (pp. 92-96). IEEE.

[7] Borah, B., & Bhattacharyya, D. K. (2007, February). A clustering technique using density difference. In *2007 ICSPCC* (pp. 585-588). IEEE.

[8] De-An, Z., Jidong, L., Wei, J., Ying, Z., & Yu, C. (2011). Design and control of an apple harvesting robot. *Biosyst. eng.*, *110*(2), 112-122. https://doi.org/10.1016/j.biosystemseng.2011.07.005

[9] Dunteman, G. H. (1989). *Principal components analysis* (No. 69). Sage. https://doi.org/10.4135/9781412985475

[10] Eizentals, P., & Oka, K. (2016). 3D pose estimation of green pepper fruit for automated harvesting. *Comput. Electron. Agric.*, *128*, 127-140. https://doi.org/10.1016/j.compag.2016.08.024

[11] El-Sonbaty, Y., Ismail, M. A., & Farouk, M. (2004, November). An efficient density based clustering algorithm for large databases. In *16th ICTAI* (pp. 673-677). IEEE.

[12] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd* (Vol. 96, No. 34, pp. 226-231)

[13] Fahim, A. M., Salem, A. M., Torkey, F. A., & Ramadan, M. A. (2006). Density clustering based on radius of data (DCBRD). *Wor. Acade. Sci., Eng. Tech.*.

[14] Gao, H., Cheng, B., Wang, J., Li, K., Zhao, J., & Li, D. (2018). Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment. *IEEE TII*, 14(9), 4224-4231

[15] Gao, H., Cheng, B., Wang, J., Li, K., Zhao, J., & Li, D. (2018). Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment. *IEEE TII*, *14*(9), 4224-4231.

[16] Girshick, R. (2015). Fast r-cnn. *Proc. IEEE ICCV* (pp. 1440-1448).

[17] Gordon, D., Kembhavi, A., Rastegari, M., Redmon, J., Fox, D., & Farhadi, A. (2018). Iqa: Visual question answering in interactive environments. *Proc. IEEE CVPR* (pp. 4089-4098).

[18] Hamerly, G., & Elkan, C. (2004). Learning the k in k-means. *Adv. neur. info. Proc. Syst.*, *16*, 281-288

[19] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. *Proc. IEEE ICCV* (pp. 2961-2969).

[20] Ji, W., Zhao, D., Cheng, F., Xu, B., Zhang, Y., & Wang, J. (2012). Automatic recognition vision system guided for apple harvesting robot. *Comput. & Electr. Eng.*, *38*(5), 1186-1195. https://doi.org/10.1016/j.compeleceng.2011.11.005

[21] Jia, W., Zhao, D., Liu, X., Tang, S., Ruan, C., & Ji, W. (2015). Apple recognition based on K-means and GA-RBF-LMS neural network applicated in harvesting robot. *Transac. Chin. Soc. Agric. Eng.*, 31(18), 175-183.

[22] Jocher, G. (2020). Yolov5. Code repository https://github. com/ultralytics/yolov5

[23] Kato, S., Tokunaga, S., Maruyama, Y., Maeda, S., Hirabayashi, M., Kitsukawa, Y., ... & Azumi, T. (2018, April). Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th ICCPS* (pp. 287-296). IEEE.

[24] Khan, K., Rehman, S. U., Aziz, K., Fong, S., & Sarasvady, S. (2014, February). DBSCAN: Past, present and future. In *The fifth ICADIWT* (pp. 232-238). IEEE.

[25] Kocić, J., Jovičić, N., & Drndarević, V. (2018, November). Sensors and sensor fusion in autonomous vehicles. *2018 26th TELFOR* (pp. 420-425). IEEE

[26] Krishna, K., & Murty, M. N. (1999). Genetic K-means algorithm. *IEEE SMC, Part B (Cybernetics)*, *29*(3), 433-439.

[27] Lihua, Z., Zhonghui, H., Chengyun, L., & Hanyong, Z. (2012). The Effect of Different Shading Degrees on Maize Leave Structure. *Chin. Agric. Sci. Bulletin*, 28(6), 43-46.

[28] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *ECCV* (pp. 740-755). Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1_48

[29] Liu, B. (2006, August). A fast density-based clustering algorithm for large databases. In *2006 ICMLC* (pp. 996-1000). IEEE.

[30] Liu, P., Zhou, D., & Wu, N. (2007, June). VDBSCAN: varied density based spatial clustering of applications with noise. In *2007 ICSSSM* (pp. 1-4). IEEE.

[31] Mahran, S., & Mahar, K. (2008, July). Using grid for accelerating density-based clustering. In *2008 8th IEEE ICICT* (pp. 35-40). IEEE.

[32] Mou, X., Liu, Q., Ou, Y., Wang, M., & Song, J. (2013). Mechanical properties of the leaf sheath of sugarcane. *Transactions of the ASABE*, *56*(3), 801-812. https://elibrary.asabe.org/abstract.asp?aid=42742

[33] Ng, R., & Han, J. (1994, September). Efficient and effective clustering method for spatial data mining. In *Proc. 1994 Int. Conf. Very Large Data Bases, pages144-155, Santiago, Chile*. https://dl.acm.org/doi/10.5555/645920.672827

[34] Obreza, T. A., Alva, A. K., Hanlon, E. A., & Rouse, R. E. (1992). Citrus grove leaf tissue and soil testing: sampling, analysis, and interpretation. *Fla. Coop. Ext. Serv., IF AS, Univ. of Florida*. https://ufdc.ufl.edu/IR00004632/00001

[35] Pawlak, W. (2018). Wear and coefficient of friction of PLA-graphite composite in 3D printing technology. *Eng. Mech.*. https://www.engmech.cz/improc/2018/649.pdf

[36] Pham, D. T., Dimov, S. S., & Nguyen, C. D. (2005). Selection of K in K-means clustering. *Proc. IME, Part C: J. Mech. Eng. Sci.*, *219*(1), 103-119. https://doi.org/10.1243/095440605X8298

[37] Ram, A., Sharma, A., Jalal, A. S., Agrawal, A., & Singh, R. (2009, March). An enhanced density based spatial clustering of applications with noise. In *2009 IEEE IACC* (pp. 1475-1478). IEEE.

[38] Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, October). Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV* (pp. 525-542). Springer, Cham. https://doi.org/10.1007/978-3-319-46493-0_32

[39] Redmon, J., & Angelova, A. (2015, May). Real-time grasp detection using convolutional neural networks. In *2015 IEEE ICRA* (pp. 1316-1322). IEEE.

[40] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proc. IEEE CVPR* (pp. 7263-7271). https://doi.org/10.1109/CVPR.2017.690

[41] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767.* https://arxiv.org/abs/1804.02767

[42] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proc. IEEE CVPR* (pp. 779-788).

[43] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE TPAMI*, *39*(6), 1137-1149.

[44] Renard, D., Tilman, D. National food production stabilized by crop diversity. (2019). *Nature*, 571, 257–260. https://doi.org/10.1038/s41586-019-1316-y

[45] Shental, N., Bar-Hillel, A., Hertz, T., & Weinshall, D. (2004). Computing Gaussian mixture models with EM using equivalence constraints. *Adv. Neur. Info. Process. Syst.*, *16*(8), 465-472.

[46] Shinzato, P. Y., Wolf, D. F., & Stiller, C. (2014, June). Road terrain detection: Avoiding common obstacle detection assumptions using sensor fusion. *2014 IEEE IV* (pp. 687-692). IEEE

[47] Song, Y., Glasbey, C. A., Horgan, G. W., Polder, G., Dieleman, J. A., & Van der Heijden, G. W. A. M. (2014). Automatic fruit recognition and counting from multiple images. *Biosyst. Eng.*, 118, 203-215. https://doi.org/10.1016/j.biosystemseng.2013.12.008

[48] Tao, Y., & Zhou, J. (2017). Automatic apple recognition based on the fusion of color and 3D feature for robotic fruit picking. *Comput. electron. agric.*, 142, 388-396. https://doi.org/10.1016/j.compag.2017.09.019

[49] Torrey, L., & Shavlik, J. (2010). Transfer learning. *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques* (pp. 242-264). IGI global.

[50] Tripathi, A.D., Mishra, R., Maurya, K.K., Singh, R.B., Wilson, D.W. (2019). Estimates for world population and global food availability for global health. *The Role of Functional Food Security in Global Health,* (pp. 3-24). https://doi.org/10.1016/B978-0-12-813148-0.00001-3.

[51] UN Department of Economics and social Affairs. (2015). World population projected to reach 9.7 billion by 2050. http://www.un.org/en/development/desa/news/population/2015-report.html

[52] Uncu, O., Gruver, W. A., Kotak, D. B., Sabaz, D., Alibhai, Z., & Ng, C. (2006, October). GRIDBSCAN: GRId density-based spatial clustering of applications with noise. In *2006 IEEE SMC* (Vol. 4, pp. 2976-2981). IEEE.

[53] Viswanath, P., & Pinkesh, R. (2006, August). l-dbscan: A fast hybrid density based clustering method. *18th ICPR* (Vol. 1, pp. 912-915). IEEE.

[54] Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001, June). Constrained k-means clustering with background knowledge. *ICML* (Vol. 1, pp. 577-584).

[55] Xiao, L., Dai, B., Liu, D., Hu, T., & Wu, T. (2015). CRF based road detection with multi-sensor fusion. *2015 IEEE IV* (pp. 192-198). IEEE.

[56] Xiaoyun, C., Yufang, M., Yan, Z., & Ping, W. (2008, October). GMDBSCAN: multi-density DBSCAN cluster based on grid. In *2008 IEEE ICEBE* (pp. 780-783). IEEE.

[57] Xuan, G., Zhang, W., & Chai, P. (2001, October). EM algorithms of Gaussian mixture model and hidden Markov model. In *Proc. 2001 ICIP* (Vol. 1, pp. 145-148). IEEE.

[58] Yu, X., Zhou, D., & Zhou, Y. (2005, June). A new clustering algorithm based on distance and density. In *Proc. ICSSSM, 2005.* (Vol. 2, pp. 1016-1021). IEEE.

[59] Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE TPAMI*, *22*(11), 1330-1334.

[60] Zhang, Z., Kayacan, E., Thompson, B., & Chowdhary, G. (2020). High precision control and deep learning-based corn stand counting algorithms for agricultural robot. *Auto. Rob.*, *44*(7), 1289-1302. https://doi.org/10.1007/s10514-020-09915-y

[61] Zhou, Q. Y., Park, J., & Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847*. https://arxiv.org/abs/1801.09847