Applications of Deep Learning for Robotic Agriculture

Tanvir Pal Singh Parhar

CMU-RI-TR-20-37

Submitted in partial fulfillment of
the requirements for the degree of
Masters of Science in Robotics

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, 15213

Aug 2020

Thesis Committee:
George A. Kantor (Chair)
Jeff Schneider
Dinesh Reddy Narapureddy

Abstract

Agricultural automation is a varied and challenging field, with tasks ranging from detection to sizing and from manipulation to navigation. These are also precursors to effective plant breeding and management. Making plant measurements by manually scouting is labor-intensive and intractable at large scale. While the cost of gene sequencing has gone down several orders of magnitude in the last couple of years, measuring the physical traits of plants remains a labor-intensive task.

This calls for robotic solutions that can measure plant traits with a high throughput. Accelerated by the current developments in the area of deep learning and computer vision, in this thesis, we propose the applications of these advancements for non-contact phenotyping, tactile phenotyping, and vision-based navigation in agricultural fields. For non-contact phenotyping, we propose an architecture that can count plants within 10% of human ground truth counts and measure stalk widths with a mean absolute error of 2.76mm. We then present a deep-learning-based architecture for segmenting and grasping stalks, for tactile measurements, achieving an average grasping accuracy of 74.13% and a stalk detection F1 score of 0.90.

We then propose a deep reinforcement-learning based architecture that can learn policies to navigate in agricultural fields without human supervision. We test the same algorithm in simulation and various outdoor scenes. The robot learns to navigate in simulation in as low as 9 minutes of training time. We report the performance of the same algorithm in outdoor track, vineyard and hops plantation. The agent can learn to follow tracks in as low as 1hr 35 minutes on the real robot, learning from scratch.

Acknowledgements

I would like to begin by thanking my fantastic advisor, George Kantor, for advising me, helping me to shape ideas into concrete research. I would also like to thank Jeff Schneider and Dinesh Reddy for technical advice, feedback and discussions. Thanks to my amazing labmates, Harjatin Baweja, Anjana, Zania Pothen, Abhisesh Silwal, Tim Mueller-Sim, Francisco Yandun, Omeed Mirbod, Merritt Jenkins and Justin Abel for great discussions, technical contributions and debugging help, feedbacks and multiple data collection trips.

A big thanks goes to Stephen Nuske for giving me a research opportunity and for his work on automation in vineyards that acted as foundations to my research. A shout out to Rachel Burcin and Nora Kazour for bringing me to Robotics Institute and providing all the administrative support. Thanks to my parents, brother, and Milo, for supporting me and always encouraging me to push my limits.

I gratefully acknowledge USDA NIFA Speciality Crops Research Initiative and DOE ARPA TERRA grant for supporting this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The advent of mechanization in agriculture has no doubt made it more productive and efficient. But even with these advancements agriculture still remains to be one of the most labor intensive tasks. According to the current projections, the world population is expected to reach 9.15 billion individuals in 2050, as compared to today's 7 billion. Much of this growth taking place in regions with already scarce resources [1]. Deforestation, land degradation and erratic changes in the climate due to global warming, would further contribute to global food insecurity. Due to shrinking margins, the number of people directly employed in the agriculture is reducing, with only 1% of the population in the USA deriving their livelihood from agriculture(USDA-NASS, 2014). Thus, crop breeding for traits like resilience and higher yield is critical for global food, climate and energy security. For Instance, breeding plants for more potent bio-fuels would reduce the emissions of green-house gasses [2] and would be a beneficent in the long term by replacing the non-renewable fossil fuels with a renewable resource. Improved agricultural efficiency will be critical to global health and economic stability in the 21st century, and plant breeding will play a central role.

Genetic data can be used to breed better off-springs, by crossing plants with desirable physical traits. Geneticists benefit from finding correlations between desirable physical traits of a plant and genetic markers in gene sequences to make more informed decisions for ideal germplasms selection. As shown in Figure 1.1, there has been a dramatic reduction in the costs of gene-sequencing over the last 15 years. But measuring these physical traits (phenotypes) still remains an arduous and repetitive task. When performed at large scales, as shown in Figure 1.2 this, becomes labor-intensive and is prone to inaccuracies. Thus, phenotyping becomes a bottleneck for large scale genetic inferences in plants.

Robotics and computer vision have the potential to tackle this bottleneck by providing consistent, fast and high-throughput phenotypic data. But, outdoor plant

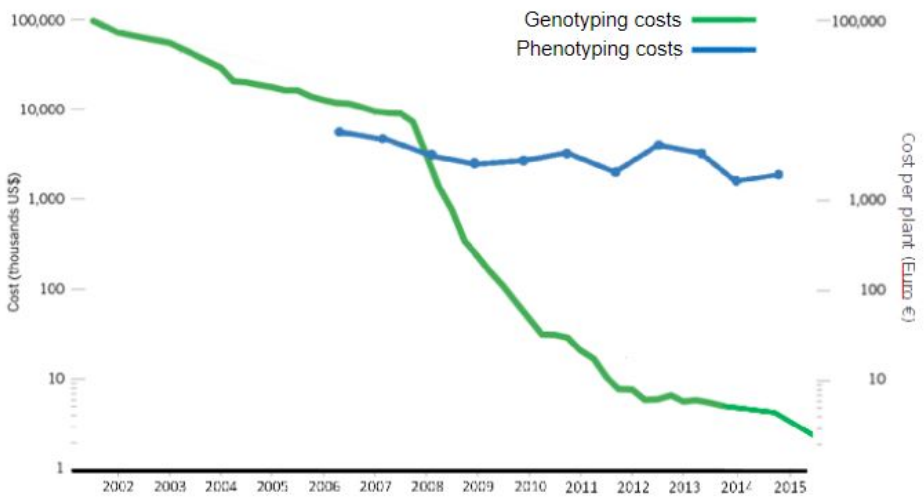

Figure 1.1: Comparison between gene sequencing costs and physical phenotyping costs from 2002 to 2015

phenotyping remains a challenging task, owing to inconsistencies in the environment and a high degree of noise like changes in lighting conditions, movements due to winds etc. Yet, recent improvements in computer vision algorithms are making it feasible to develop algorithms and robotic platforms that can operate in in these environments.

Figure 1.2: Aerial view of a vineyard depicts the large scale nature of plant phenotyping.

Images are a very rich sensory modality and are ideal for visual phenotypic estimation. But extracting relevant information from image data is a challenge. With the recent advancement in Deep-Learning based techniques for detection and segmentation and reduction in running times due to efficient CUDA implementations of these algorithms, it has become possible to use images effectively. There are 3 major tasks that need to be accomplished for robotic agricultural phenotyping, viz. *Detection, Manipulation* and *Navigation*. All of which can be done with information contained in images. They are briefly described here and then explained in detail in later sections:

Detection: This refers to identification and sizing of the relevant features in the images. This could be anything from plant height to fruit size. Once the object is detected in the image, depth estimate from the corresponding stereo pair can be used to size it. This thesis proposes a Deep-Learning based end-to-end architecture for detecting and sizing plant traits. Figure 1.3 shows different parts of plants being detected and sized.

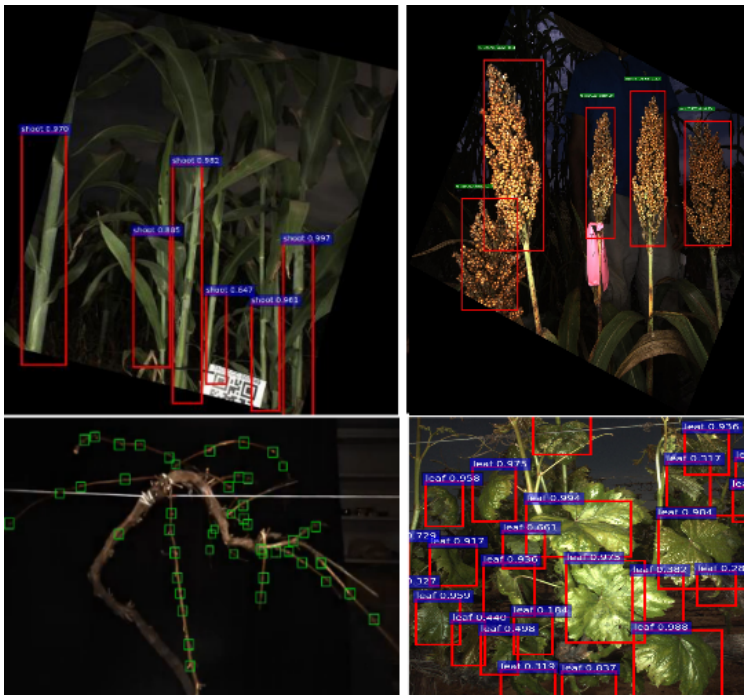

Figure 1.3: Top: Left: Sorghum stalks, Right: Sorghum Panicles, Bottom: Left: Grape Vine buds and Right: Grape Vine leaves.

Manipulation: Once the objects are detected and located in 3D, they can be acted upon physically by a manipulator. This can be used to measure tactile traits like plant strength, hyper-spectral reflectance, pruning etc. This thesis proposes a Conditional Generative Adversarial Network based end-to-end pipeline for manipulating plant stalks. Figure. 1.4 shows a Sorghum stalk being grasped.

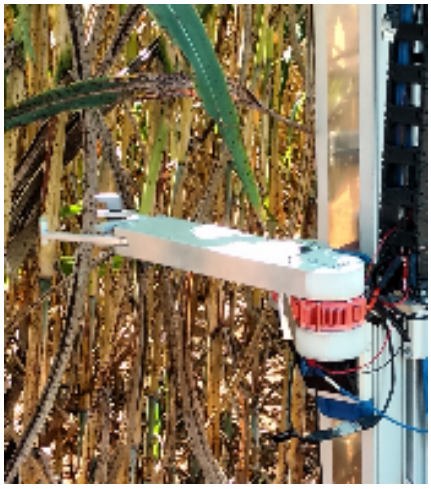

Figure 1.4: Sorghum Stalk being grabbed for strength measurement.

Navigation: The sensors and manipulators need to be transported to the plants in the field. While it is possible to use to conventional control algorithms like ILQRs, PIDs etc, they always assume full state observation of the robot. This can be attained via sensors like RTK-GPS. But, owing to high water retention in agricultural fields, GPS signals get attenuated. Images do not have this problem. Thus, this thesis explores the use of Deep Reinforcement Learning for human free image based navigation. The proposed architecture is made with keeping generalizability across different environments in mind. Figure 1.5 shows this.

This thesis thus offers the following contribution to the research community:

- An end-to-end pipeline for metric plant phenotyping.
- And end to end pipeline for grasping plant stalks.
- A Deep RL based human free pipeline for vision based navigation that is environment agnostic.

Figure 1.5: Left: The robot navigating in a Hop field. Right: The robot navigating in a vineyard.

Chapter 2

Background

The related work for this thesis can be broadly divided into three sections. They are plant phenotyping, robotic manipulation and vision based navigation.

2.1 Plant Phenotyping

Computer Vision being the mainstay of most Artificial Intelligence based phenotyping initiatives [3], a wide variety of computer vision techniques have been developed for measuring plant traits. Our group amongst several research groups is investigating the applications of computer vision to push the boundaries of crop yield estimation [4] [5] and phenotyping. Jimenez et al. [6] provide an overview of many such studies.These can be classified into classical machine vision approaches, that depend on the structure of the objects being observed and machine learning based approaches, which are more data driven.

In classical computer vision approaches, some work has been done to obtain precise yield estimates in vineyards by accurately measuring fruit density measurement using classical image processing techniques [7] [8]. Xu et al. used a monocular camera and morphological image manipulation techniques to detect buds in grape vines [9]. There have been a wide variety of attempts for estimation of leaf count and leaf area of plants. Various methods deploy techniques varying from image processing methods to digital sensor based data acquisition techniques. There has been an attempt on leaf segmentation by fitting quadratic curves to a combination of depth data and IR data [10]. RGBD sensors like Microsoft Kinect have been used for leaf segmentation and leaf count estimation using center of divergence methods, which employ parametric snakes [11][12].But the drawback of this approach is limitation of Kinect to indoors and the slow convergence of parametric snakes.

There exist techniques involving a combination of log-polar transformation and unsupervised learning using triangle encoding and K means clustering [13] [14] [15].

But these techniques presume that leaves are of a distinct shape, like circular or elliptical. Active Shape Modelling [16] technique has been used to detect multiple leaves, which might be occluded.The leaf boundaries and veins are detected using gradient change. Since this method tries to fit a model to leaves, it assumes all leaves to be roughly of similar shape. This is highly unlikely in the case of grape vines in outdoor conditions. Other techniques, such as those used by [17] and [18] make use of Frangi Filter [19] to detect shoots in grape vines. Pothen et al. [20] and Mirbod et al. [21] use gradient of texture generated by flash light, to detect and size grapes respectively. Jenkins et al. [22] use multiple stereo images taken form a moving platform for point-cloud generation and exploit geometric structure of the stalks for detecting them. The authors report a precision of 0.95. Bargoti et al. [23] provide a pipeline for trunk detection in apple orchards.The pipeline uses Hough Transforms on LIDAR data to initialize pixel wise dense segmentation into a hidden-semi Markov model (HSMM). Hough transform proves to be a coarse initializer for intertwined growth environments with no apparent gaps between adjacent plants. The disadvantage with these classical approaches is that they require a lot of parameter tuning to work well and are hand-crafted to work only for certain crops. Thus they are hard to generalize across crops.

Sodhi et al. [24] combine classical vision techniques with machine learning techniques to segment Sorghum stalks from images. In this work, plants are imaged from various view points and the 3D point clouds are reconstructed. Segmentation is performed over these point clouds using Support Vector Machines (SVMs) with Conditional Random Fields (CRFs). Accuracy of 80.2% is reported on field data.Paulus et al. use SVMs for classification if histograms of point-clouds for plant organ parametrization [25]. Reported Area Under the Curve (AUC) accuracy is 0.966.

Recent advances in deep leaning have induced a paradigm shift in several areas of Machine Learning, especially Computer Vision. With deep learning architectures producing state-of-the-art performances in almost all major computer vision tasks such as image recognition [26], object detection [27] and semantic segmentation [28] has made it possible for researchers to use these for image based plant phenotyping. A broad variety of studies ranging from yield estimation [29] [30], to plant classification [31], to plant disease detection have been conducted [32]. Chen et al. [33] use a 2 CNN based architecture, where the first CNN detects blobs where the fruits might be present and the second CNN uses these blobs to estimate the fruit count in those regions. A regressor network then regresses the total fruit count in the image. Pound et al. [34] propose using vanilla CNNs (Convolutional Neural Networks) to detect plant features such as root tip, leaf base, leaf tip, etc. Though the results achieved are impressive, the images used are taken in indoor environments thus escaping the challenges of field environments such as occlusion, varying lighting

amongst several others. This is addressed by Baweja et al., [35] where they use the same sensor as [5], [20] and [21] for drowning out ambient light. They developed a CNN based architecture for Sorghum stalk detection and width estimation. They used a Faster RCNN [27] for stalk detection and a Fully Convolutional Network [36] for segmenting the stalk. They report an R squared correlation of 0.88 for stalk counting and an absolute error of 2.77 mm for stalk width estimation.

2.2 Robotic Manipulation

Agricultural manipulation has been an active area of academic research for the past 35 years. The field has focused almost exclusively on horticultural crops such as fruits and vegetables, and is subject to application-specific design constraints such as:

- Highly variable target shapes and sizes
- Outdoor weather conditions including rain, dust, and UV radiation
- Cost

Due to these constraints, most manipulators and end-effectors are highly customized. Bac et al. surveyed 50 peer-reviewed agricultural robotics papers between the years of 1984 and 2012, concluding that 78% (35 out of 45) of manipulators and 98% (46 out of 47) of end-effectors were custom-built [37]. Several representative papers are outlined below.

Baeton et al. tested a VR006L Panasonic industrial manipulator equipped with a suction mechanism for apple harvesting. The authors achieved 80% detection and grasping success, but the system required a custom stabilization unit to accommodate the manipulators high mass [38]. Scarfe et al. developed a custom anthropomorphic manipulator for overhead harvesting of kiwi fruit [39]. The anthropomorphic arm is designed for high speed and low cost, so the manipulator consists of stepper motors located at the base that actuate rotary joints through linkages. The lightweight end-effector consists of two fingers fabricated out of bent round wire. Bac et al. attached a suction mechanism and scissor tool to a Universal Robots UR5 manipulator for autonomous indoor harvesting of sweet peppers [40]. He achieved grasp success ranging from 41% to 61%.

Owing to the ability of neural networks to model complex non-linear functions effectively, there has been a huge body of work focusing on using neural-networks as controller for high DOF robotic manipulators. More recently, Reinforcement Learning(RL) techniques are being used to train these neural network controllers.

Combining model free techniques with model based techniques like trajectory optimization,Guided Policy Search methods, provide guarantee of robust performance, under supervision from an optimal controller during training [41] [42] [43]. On the other hand, model free Deep RL methods do not assume the model of the agent.[44] uses model free Deep Q Networks with Normalized Advantage Functions (DQN-NAF) to learn to open doors.And [45] uses Hindsight Experience Replay to learn various tasks from scratch with sparse binary rewards.Franceschetti et al. [46] preformed a detailed comparison of two model free techniques, TRPO [47] and DQN-NAF for the task of learning jobs like pick and place and goal reaching on a simulated UR5 robot. Owing to theoretic monotonic improvement guarantees of TRPO, Parhar et al. use an improvement over TRPO, viz, PPO [48] for training our simulated UR5 arm to reach goals while avoiding an obstacle in direct line of sight of the end-effector. They further used the trained model to control the real arm to reach buds on a vine.

2.3 Reinforcement Learning for Navigation

There has been substantive recent progress towards solving the long standing goal of autonomous driving [49, 50, 51, 52, 53, 54]. This problem ranges in complexity from learning to navigate in constrained industrial settings, to learning to drive on highways, to navigation in dense urban environments. Autonomous navigation in requires understanding complex multi agent dynamics including tracking multiple actors across scenes, predicting intent, and adjusting agent behavior conditioned on past history. These factors provide a strong impetus for the need of general learning paradigms that are 'complex' enough to take these factors into account.

Current state of the art systems generally use a variant of supervised learning over large data-sets of collected logs to learn driving behavior [53, 54]. These systems typically consists of a modular pipeline with different components responsible for perception, mapping, localization, actor prediction, motion planning, and control [55, 56, 57, 58, 59]. The advantage they offer is the ease of interpretation and ability to optimize subsystem parameters in an understandable way. However in practice, it is extremely hard to tune these subsystems and replicate the intended behavior leading to poor performance in new environments.

Another approach that has recently become popular is exploiting imitation learning where we aim to learn a control policy for driving behavior by observing expert demonstrations [60, 54, 61, 62, 63, 64, 65]. The advantage of these methods are that the agent can be optimized using end to end deep learning to learn the desired control behavior which significantly reduces the effort of tuning each component that is common to more modular systems. The drawback however is that these systems are challenging to scale and generalize to novel situations, since they can never out-

perform the expert agent and it is impractical to obtain expert demonstrations for the all the scenarios that we care about.

There also has been a substantial amount of research in autonomous navigation in agricultural setting. Though less challenging in terms of environmental uncertainty, because of lack of other actors, it comes with its own unique sets of challenges. There have been bodies of work that investigated the use of different sensors for agricultural navigation. [66] investigates the use of monocular RGB camera to navigate between rows of an apple orchard. [67] explores the use of a variable field of view camera for navigation in an early season corn field, while [68] use the 3D point cloud obtained from a LiDAR sensor to localize and navigate inside and apple orchard. [69] make use of an image based particle filter for navigating within maize. They used a downward facing camera and used a binary image indicating the presence of plant material determined by color thresholding in their measurement model. [70] used a planar laser range finder in a push-broom configuration along with a RANSAC line fitting algorithm to successfully detect multiple crop rows that ranged in height from 0.15-0.65 m, in a greenhouse. All these approaches are specific to the particular crop and assume consistent lighting conditions. Hence, they are not easily generalizable to different types of crops.

Deep reinforcement learning (DRL) has recently made large strides towards solving sequential decision making problems, including learning to play Atari games and completing complex robotic manipulation tasks. The superhuman performance attained using this learning paradigm motivates the question of whether it could be leveraged to solve the long standing goal of creating autonomous vehicles. This approach of using reinforcement learning has inspired few recent works [71, 52, 51, 72] that learn control policies for navigation task using high dimensional observations like images. The previous approach using DRL [71] reports poor performance in navigation tasks, while the imitation learning based approach [51] achieves better performance but suffers from poor generalizability. Another work [56] only learns the lane following task for a constrained environment and uses a sparse reward structure that makes it challenging to learn. Moreover, learning policies from high dimensional state spaces still remains challenging due to the poor sample complexity of most model-free reinforcement learning algorithms.

Chapter 3

Plant Phenotyping

With a growing population and increasing pressure on agricultural land to produce more per acre there is a desire to develop technologies that increase agricultural output. Work in plant breeding and plant genomics has advanced many varieties of crop through crossing many varieties and selecting the highest performing. This is done y correlating the plants with desired physical phenotypes with the corresponding genomic markers. Even though the cost of genomic sequencing has come down over the past years, measuring these phenotypes still remains to be a major(Figure 1.1) bottleneck and limits in terms of how many plant varieties can be accurately assessed in a given time frame. Thus there exists a need for systems that can accurately measure plant phenotypes at a high throughput.

The inability to collect physical measurements by humans, at large scale, is what is referred to as the phenotyping bottleneck. Collecting large scale phenotypic data is necessary to alleviate errors due to any sampling bias. Doing this in an outdoor setting is very labor intensive and error prone, as no two humans are guaranteed to produce the same result. This calls for the need to deploy robotic systems that can do this fast, repeatedly and accurately. Unmanned Aerial Vehicles (UAVs) offer an attractive choice, due to the large field of view that they can offer. However, these platforms lack the capacity to produce plan-specific measurements. Moreover, they lack the payload capacity and have low operating times.

In this work we focus on plant phenotyping for bio energy Sorghum crop. Sorghum is a diverse crop with over 40,000 accessions. It is useful for a plethora of purposes including but not limited to bio-fuel, animal fodder and alcohol production. We present a new image processing pipeline the leverages both recent deep convolutional neural networks for object detection and also semantic segmentation networks together to output both stalk count and stalk width. The combination of networks together provides more precise and accurate extraction of stalk contours and therefore more reliable measurement of stalk width.We collect data at two sep-

arate plant breeding sites one in South Carolina, USA and the other in Puerto Vallarta, Mexico and use one data-set for training our networks and the other data-set we ground truth using manually extracted measurements. We study both the efficacy and efficiency of manual measurements by using two humans to independently measure sets of plants and comparing accuracy and also total time taken per plot to measure attributes. We then compare the human measurements to the robot measurements for validation. This research is facilitated by a grant from U.S. Department of Energy Advanced Research Project Agency-Energy (ARPA-E)'s TERRA program.Plant stalk width and stalk count are two of the the most important phenotypes required for the sorghum breeding program.

3.1 Ground Robot

Since, none of the off the shelf robot platforms have the right form factor to enter Sorghum rows and the payload capacity to power and carry a sensor load of 50 Kg, Tim Mueller-Sim, Merritt Jenkins and Justin Abel developed a robotic platform, as reported by [73]. The platform as shown in Figure 3.1 is capable of autonomously navigating rows more that 30 inches wide.

The robot consists of a wide suite of sensors, like LiDARs, Inertial Measurement Units, Time of Flight Sensors and a GPS receiver that gets Real Time Kinematic corrections from a base station for a sub-centimeter location accuracy. It is equipped with a 3 Degree of Freedom manipulator, that is used to measure stalk strength and hyper-spectral reflectance, using a custom built sensor. The robot is also equipped with a 9 Mega Pixel stereo sensor, that can collect stereo images at 15 Hz. The camera has active lighting, that is used to drown out ambient light to get images with consistent lighting. This in turn helps in reducing the amount of training data needed to train the networks. The robot also has 4 NUC i-7 computers on-board. Every sensor and computer is synchronized in a master-slave fashion using Robot Operating System [74].

Figure. 3.2 shows the custom stereo sensor that was used to collect data for this work. The sensor generates high FPS image data, with active lighting, using hardware synchronization between the flashes and the camera shutters. The image data is Geo-stamped via the NAVSAT messages generated through ROS.

3.2 Overview of Processing Pipeline

The motivation behind the work was to come up with a high throughput plant phenotyping computer vision based approach that is agnostic to changes in the field conditions and settings such as varying lighting conditions, occlusions etc. Figure 3.3 shows the overview of the data-processing pipeline, used by our approach. The

Figure 3.1: The Robotanist : The ground based robot, used to collect the data, developed at CMU

faster RCNN takes one of the stereo pair images as its input and produces bounding boxes, each for one stalk. These bounding boxes are extracted from the input image (also called as snips) and fed to the FCN, one at a time. The FCN outputs a binary mask, classifying each pixel as either belonging to stalk or the background. To this mask ellipse are fitted, to the blobs in the binary mask, by minimizing the least-square loss of the pixels in the blob [75]. One snip may have multiple ellipses in case of multiple blobs. The ellipse with the largest minor axis is used for width calculation. The minor axis of this ellipse gives us the pixel width of the shoot in the current snip. The corresponding pixels in the disparity map are used to convert this pixel width into metric units. The whole pipeline takes on an average 0.43 s to process one image, on a GTX 970 GPU. This can make the data-processing on the fly for systems that collect data at 2Hz.

Figure 3.2: Left: The robot navigating in a Hop field. Left: The custom built stereo imager. Right: Sample image of sorghum with active lighting.

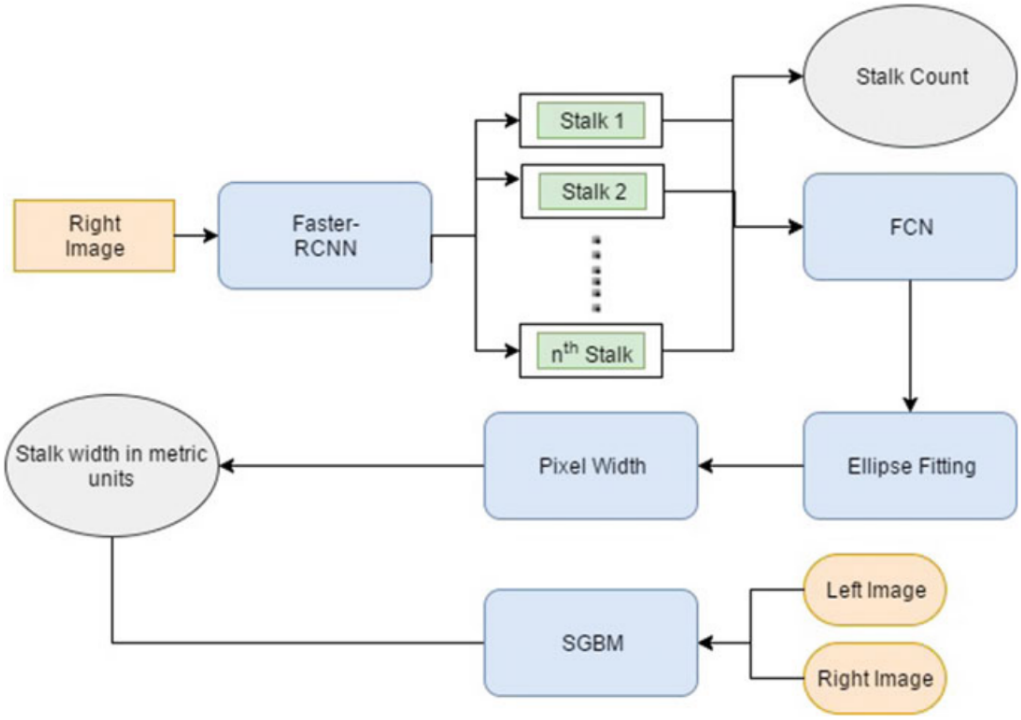

Figure 3.3: The process pipeline for stalk width estimation.

3.3 Detecting and Segmenting Stalks

3.3.1 Detecting: Faster RCNN

Faster-RCNN by Girshick et al [27]. is an improvement over the Fast-RCNN, where there is a separate convolution layer that predicts object proposals based on the features form the activation of the last layer of the VGG-16 network, called Region Proposal Network (RPN). Since the region proposal network is a convolution layer, followed by fully connected layers, it is implemented over GPU, making it almost an order of magnitude faster than Fast-RCNN.One drawback of the Faster RCNN is the use of non-maximal suppression (NMS) over the proposed bounding boxes. Thus, highly overlapping instances of objects might not be detected, due to NMS rejection. This problem is even severe in highly occluding field images. It was overcome by simply rotating the images by 90°so that the erectness of the stalks may be used to draw tightest possible bounding boxes. We fine tuned a pre-trained Faster-RCNN with 2000 bounding boxes. Figure 3.4 shows sample detections.

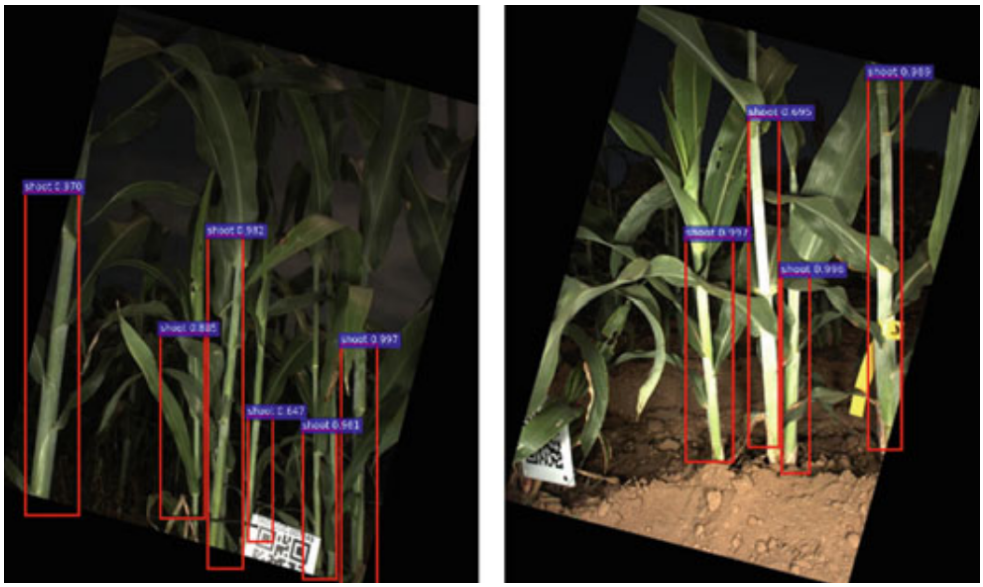

Figure 3.4: Sample Detections from Faster RCNN.

3.3.2 Semantic Segmentation: Fully Convolutional Network (FCN)

FCN (Fully Convolutional Network) is a CNN based end to end architecture that uses down-sampling (Convolutional Network) followed by up-sampling (Deconvolution Network) to take image as input and produce a semantic mask as output. The snips of stalks detected by Faster-RCNN are sent to FCN for semantic segmentation which by virtue of its fully convolutional architecture can account for different sized

incoming image snips. We chose to send Faster-RCNN's output to FCN as input instead of raw image. Output bounding boxes always contain only one stalk and thus FCN is only required to do a binary classification into two classes,namely: stalk and background without having to do instance segmentation also. Our hypothesis was that this would make FCNs job a lot easier and would thus require lesser data to fine tune a pre-trained version. This hypothesis is validated by results presented in a latter section. We fine-tuned a pre-trained FCN with just 100 dense labeled detected stalk outputs from Faster-RCNN. Sample input to output of FCN is shown in Fig.3.5.

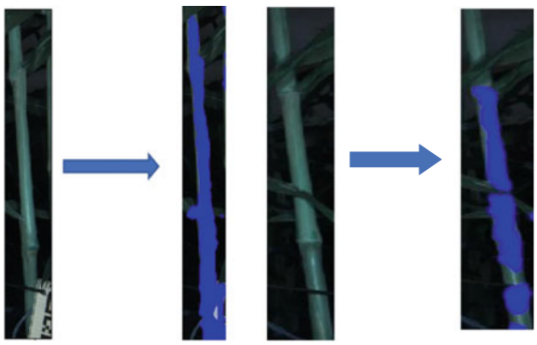

Figure 3.5: Sample snipped bounding box input to segmented stalk output.

3.4 Stalk Width Estimation

Once the masks have been obtained, for each of the snippets, ellipses are fitted to each blob of the connected contours of the mask. The ellipses are fitted to minimize the following objective:

$$\epsilon^2(\theta) = \sum_{i=1}^{n} F(\theta_i, x_i)^2 \tag{3.1}$$

Where, $F(\boldsymbol{\theta}; \boldsymbol{x}) = \theta_{xx}x^2 + \theta_{yy}y^2 + \theta_{xy}xy + \theta_x x + \theta_y y + \theta_0$ is the general equation of conics in 2 dimensions. The objective is to find the optimal value ofsuch that we get the best fitting conic over a given set of points. We use OpenCV's inbuilt optimizer to find best fitting ellipses. Figure7 shows the ellipses fitted to the output mask of the FCN.

Ellipse is fitted to the contours of the blob, so that the minor axis can serve as a starting point for width estimation of the stalk. For the same reason, a simple convex hull fitting was not performed. The minor axes of all the ellipses are then

trimmed to make sure they lie over the FCN mask. From the trimmed line segments, any segment that night have a slope of greater than 30^o is rejected. The remaining line segments are projected on to the disparity map, so that the pixel width can be converted to the width in metric units, as per Algorithm 1. The line segment with the greatest metric width is selected as the width for the stalk in the current snip. The reason behind choosing max width over others is to get rid of the segments proposals that might have leaf occlusions. Fig.3.6 shows the pipeline for stalk width estimation and Fig.3.7 shows the results for stalk width estimation.

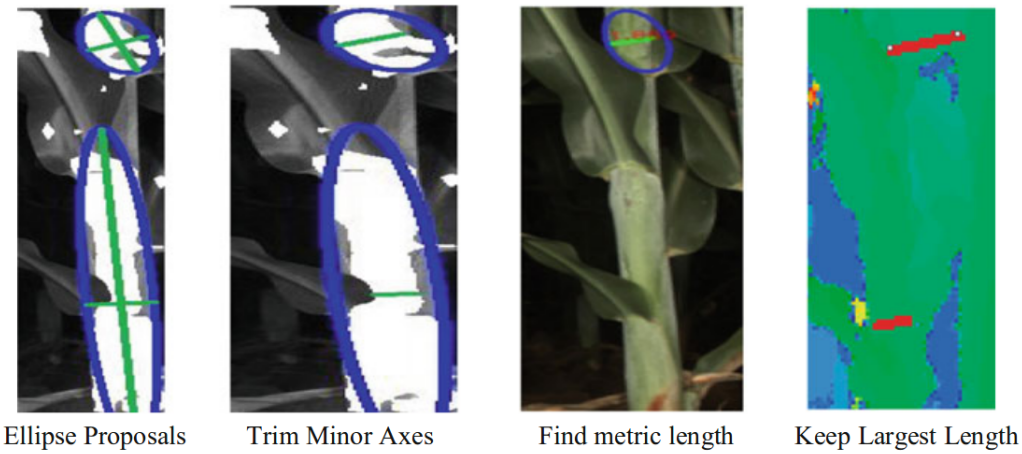

Figure 3.6: Stalk width estimation pipeline

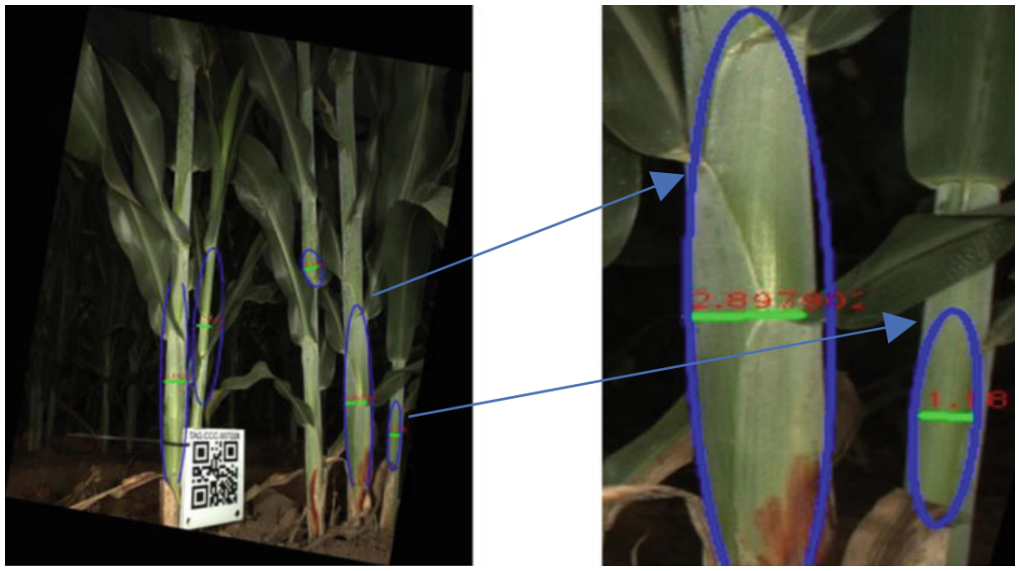

Figure 3.7: Stalk width estimation results

3.5 Results

3.5.1 Data Collection

Image data was collected in Pendleton, South Carolina using the Robotanist platform. The algorithms were developed on this data. To test the algorithm impartially, another round of data collection with extensive ground truthing was done in Cruz Farm, Mexico. The images were collected using a 9MP stereo-camera pair with 8mm focal length, high power flashes triggered at 3 Hz. by Robot Operating System (ROS). The sensor was driven at approximately 0.05 m/s. Distance of approximately 0.8 m was maintained from the plant growth. Each row of plant growth at Cruz Farm is divided into several 7 ft ranges separated by 5 ft alleys. To ground truth stalk count data, all stalks were counted in 29 ranges by two individuals separately. The mean of these counts was taken as the actual ground truth. Similarly, for width calculations, QR tags were attached to randomly chosen stalks for ground truth registration in images. The width of these stalks at height of 12 in. (30.48 cm) and 24 in. (60.96 cm) from the ground was also measured by two individuals separately using Vernier Calipers of 0.01 mm precision. Humans at an average took 210 sec to count the stalks in each range and an average of 55 sec to measure width of each stalk. Each range at an average has 33 stalks, so on an average it takes 33 minutes to measure stalk widths of entire range. Fig.3.8 shows the robot collecting data.

Figure 3.8: The robot collecting data

3.5.2 Results for Stalk Count

Faster-RCNN was trained with 400 images with approximately 2000 bounding boxes using alternate optimization strategy. RPN and regressor are trained for 80000 and 40000 iterations respectively in first stage and 40000 and 20000 iterations in the second stage using base learning rate of 0.001 and a step decay of 0.1 every 60000iterations for RPN and 30000 iterations for regressor. Best test accuracies

were achieved by increasing the number of proposals to 2000 and the number of anchor boxes to 21 using different scaling ratios with NMS threshold of 0.2. Due to inability to get accurate homography for data collected at 3Hz, we resorted to calculating stalk-count/meter using stereo data and do the same for ground truth stalk counts which were collected from ranges, each of constant length 7 ft (2.134 m). Figure 3.9 shows the R-squared correlation for results of 0.88.

To put the results into perspective, attempting to normalize counts using image widths from stereo data may induce some error as this data is sometimes biased towards stalk count towards the start and end of each range where the mount vehicle is slowed down. Also, there is a little inherent uncertainty in the count data. There are tillers (stems produced by grass plant) growing at the side of some stalks which are hard to discern from stalks with stunted growth. To better understand this, we observe in Fig.3.10 that there is a small variation in ground truth stalk counting between two humans as well. The R-squared of Human1's count versus Human2's count should be 1 in an ideal scenario but that is not the case

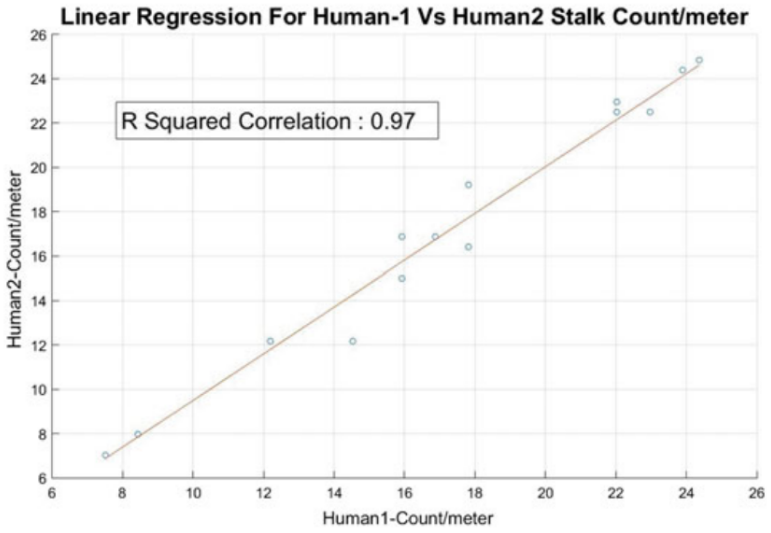

Figure 3.9: Linear regression for human stalk count/meter versus Faster-RCNN's count/meter

3.5.3 Results for Stalk Width Estimation

We plot the stalk width values as measured by Human1, Human2 and our algorithm at approximately 12in. (30.48cm) from the ground. At the time of data collection, it was made sure that a part of ground is visible in every image. This allows us

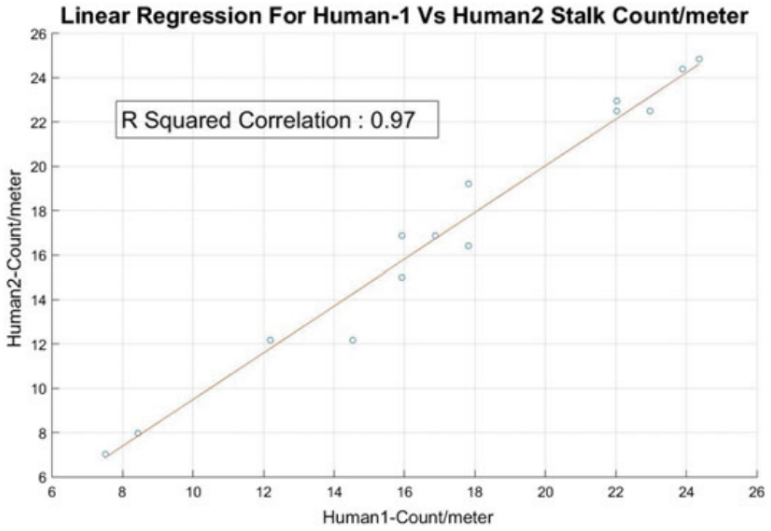

Figure 3.10: Linear regression for Human1's stalk count/meter versus Human2's stalk count/meter

get stalk width at the desired height from the image data. This step is important as there is prevalent tapering in stalk widths as we go higher up from the ground. Figure 3.11 shows the widths of each ground truthed stalk as per both humans and algorithm.

Since there is a discernible difference in measurements of the two humans, we considered the mean of the two readings as actual ground truth. The mean width of stalks as per this ground truth is 14.354 mm. The mean absolute error between readings of Human 1 and Human 2 is 1.639 mm and the mean absolute error between readings from human ground truth and algorithm is 2.76 mm. The error can be attributed to rare occlusions that force algorithm to calculate height at a location other than 12 in.(30.48 cm) from the ground. We suspected this as a possibility and thus measure stalk widths at 2 locations during the ground truthing process: at 12 in. (30.48 cm)and 24 in. (60.96 cm) above the ground. Calculations from this data tell us that there was 0.405 mm/in. mean tapering on the measured stalks as we went up from 12 in. (30.48 cm) to 24 in. (60.96 cm).

To validate our hypothesis that providing faster-RCNN's output bounding boxes as inputs to FCN would require lesser dense labeled data to train it. We trained another FCN on densely labeled complete images. This FCN was trained with more than twice the number of densely labeled stalk data (fine-tuned with approximately 250 densely labeled stalks) than the previous FCN (fine-tuned with on approximately

100 densely labeled stalks). Even after assuming perfect bounding boxes around it for instance segmentation, the mean absolute error of this FCN was 3.868 mm for width calculation, which is higher than its predecessor having a mean absolute error of 2.76 mm.

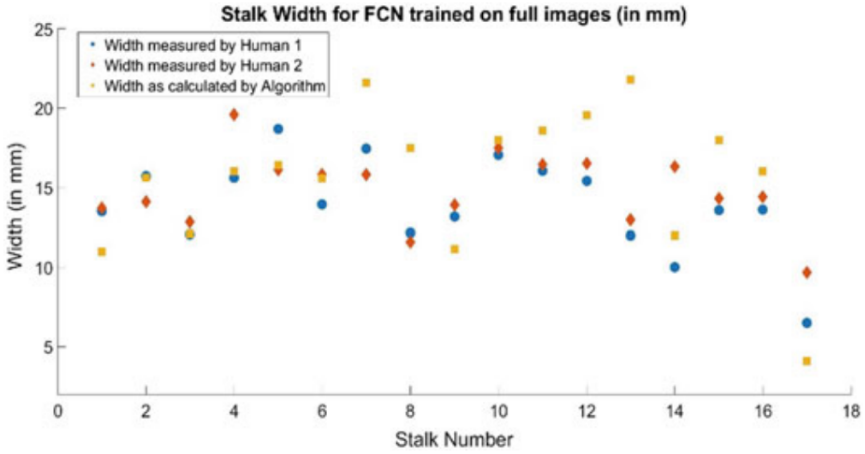

Figure 3.11: Width measured by Human1, Human2 and algorithm

3.5.4 Time Analysis

Table 1 shows the time comparisons of Humans versus algorithm for an average plot. Each plot has approximately 33 stalks and is about 2.133 m in length. We observe that Algorithm is 30 times faster as compared to humans for stalk counting and 270 times faster than human for stalk width calculation

	Human 1 (min)	Human 2 (min)	Robot (s)
Stalk Count	3.33	3.66	6.5
Stalk Width	29	30	
Total	32.33	33.66	6.5

Table 3.1: Time analysis for measuring one experimental plot

3.5.5 Conclusion

We have shown the strength of coupling deep convolutional neural networks together to achieve a high quality pipeline for both object detection and semantic segmentation. With our novel pipeline we have demonstrated accurate measurement of multiple plant attributes.

We find the automated measurements are accurate to within 10% of human validation measurements for stalk count and measure stalk width with 2.76mm on average.Ultimately though, we identify that the human measurements are 30 times slower than the robotic measurements for count and 270 times slower for measuring stalk width over an experimental plot. Moreover, when translating the work to large scale deployments, that instead of 30 experimental plots are 100's or 1000's of plots in size,it is expected that the human measurements become less accurate and logistically tough to measure in timely fashion during tight growth stage time windows.

In future work we plan to integrate more accurate positioning to merge multiple views of the stalks into more accurate measurements of stalk-count and stalk-width.

Chapter 4

Manipulation

4.1 System Overview

In the last section we saw how computer vision techniques can be utilized for estimating metric traits like plant count and plant width. It was a generic pipeline that could be used to measure anything that is visible in the images. However there are several physical traits that cannot be measured by vision alone. Some traits like plant strength and plant's hyper-spectral reflection characteristics. For these reasons this section delineates the development and successful deployment of a 3 Degree of Freedom robotic manipulator for grasping Sorghum stalks. The end-effector of the manipulator is modular and is equipped with either a hyper-spectral sensor or a penetrometer. The manipulator consists of an arm with planar 2 degrees of freedom and a vertical linear stage that adds another degree of freedom. Figure 4.1 shows a closeup rendering of the two DOF planar manipulator.

Figure 4.1: Rendering of two DOF arm mounted on vertical slide of ground robot.

All data collection, processing, and plant manipulation were performed online outdoors on a custom-built unmanned ground robot [73]. The dimensions of the robot are 1.34×0.56×1.83m (L×W×H), and the system weighs approximately 140 kg. The robot is capable of autonomous row navigation using either RTK GPS or a combination of RGB and time-of-flight cameras.

Grasp perception is performed with a Carnegie Robotics MultiSense S7 stereo camera. The MultiSense S7 features an on-board FPGA that performs semi-global block matching at a rate of 15 FPS on 1 MP color images. Point clouds are published over LAN using Robot Operating System (ROS). The MultiSense S7 was mounted to the robot at a height of 1m above the ground and 25 mm in front of the robots center plane. This positions the camera at a distance from stalks greater than its 200 mm minimum range. The camera is synchronized to custom LED flashes visible in Figure 3. The LED flashes provide both consistent lighting and foreground brightening, which significantly improves correspondence matching.

The robot manipulator is designed with three degrees of freedom consisting of a prismatic joint and two rotational joints (PRR). The motivation behind this design is that sorghum stalks are roughly vertical and grasp-based measurements are agnostic to location on a stalk's circumference. A 3-DOF PRR arm can position itself at only two circumferential locations at any height on a plant stalk. The manipulator is actuated by three series-elastic brush-less DC motor modules developed by HEBI Inc. which communicate over LAN.

Computation is performed on a ZOTAC Magnus EN1080K mounted to the robot. The computer communicates on the robot local network over LAN. It is equipped with an Intel i7 processor and an Nvidia 1080 discrete GPU. The computer is a ROS slave running a node consisting of an active session of Tensorflow compute graph. The node subscribes to the rectified left image frame of the Multisense S7 and publishes the processed image with segmented stalks. Image processing takes approximately 0.2 seconds per image.

Operational protocol consists of the robot autonomously driving a total of 0.5m in a sorghum row. A pair of images and the corresponding point cloud is captured every 0.1m. The 5 accumulated point clouds are stitched together to generate a single point cloud in the robot frame. The 2D images are fed to the GAN pipeline to segment stalks in each image. The segmented 2D images are then used to segment stalks in the stitched point cloud. Grasp points are detected in the point cloud with the 3D segmented stalks. The position of the grasp points is used to plan a grasping trajectory, which is then executed by the on-board manipulator. Fig 4.2 depicts the whole system.

4.2 Process Pipeline

4.2.1 Detection Pipeline

A Generative Adversarial Network [76] consists of a generator and a discriminator. The task of the generator is to generate data that closely matches the training data

Figure 4.2: The Robotanist in sorghum breeding plots.

distribution, and the task of the discriminator is to discriminate between the data generated by the generator network and the real data distribution.

In the classic setting for generation of fake, realistic- looking images, the generator would be a convolutional neural network that takes in noise variables $p(z)$ and outputs the high dimensional image data x. Thus it is a differentiable function G(z; g) parameterized by weights g that learn the mapping from $p(z)$ to the real data distribution. Ideally, given an input from $p(z)$, $G(z)$ must resemble data from the real data distribution p_{data}. The discriminator network, $D(x; \theta_d)$, is the second convolutional network that takes input as an image x and outputs the scalar probability of it belonging to the real distribution p_{real}. The training procedure consists of training G to minimize $log(1 - D(G(z)))$ and also training D to maximize the probability of D assigning correct labels to real and generated images. Hence, the two networks play the following mini-max game:

$$min(G)max(D) = \underset{\boldsymbol{x} \in p_{data}(\boldsymbol{x})}{E}[logD(\boldsymbol{x})] + \underset{\boldsymbol{z} \in p_z(\boldsymbol{z})}{E}[log(1 - D(G(\boldsymbol{z})))] \tag{4.1}$$

While this method of training is effective at generating images that appear very similar to the training images, the generated images are entirely new. Conditional GANs come over this shortcoming by generating a new image that is conditioned on an input image rather than a pre-defined noise prior. In this setting the training data consists of pairs of images (x, y). The labeled image is y, and the image on

which the former is conditioned is x. Hence, the objective of the conditional GAN is:

$$L_c(G,D) = \underset{(\boldsymbol{x},\boldsymbol{y})\in p_{data}(\boldsymbol{x},\boldsymbol{y})}{E}[logD(\boldsymbol{x},\boldsymbol{y})] + \underset{\boldsymbol{x}\in p_{data}(\boldsymbol{x});\boldsymbol{z}\in p_z(\boldsymbol{z})}{E}[log(1-D(\boldsymbol{x},G(\boldsymbol{x},\boldsymbol{z})))] \tag{4.2}$$

Other than just fooling the discriminator, the generator also has to produce outputs that resemble the transformed input image data. Hence, an L1 loss term is also incorporated:

$$L_{L1}(G) = \underset{(\boldsymbol{x},\boldsymbol{y})\in p_{data}(\boldsymbol{x},\boldsymbol{y});\boldsymbol{z}\in p_z(\boldsymbol{z})}{E}[|\boldsymbol{y} - G((\boldsymbol{x},\boldsymbol{z})|_1] \tag{4.3}$$

Thus the final training objective is:

$$min(G)max(D)L_c(G,D) + \lambda L_{L1}(G) \tag{4.4}$$

Instead of explicitly giving a noise input $p_z(z)$ to the generator, noise is introduced in the network in the form of dropout. During training, the discriminator is provided real pairs of input and target images $(\boldsymbol{x},\boldsymbol{y})$ and a fake pair consisting of real input image and the conditionally generated output image $(\boldsymbol{x}, G(\boldsymbol{x},\boldsymbol{z}))$. The discriminator's objective remains the same, i.e. to discriminate between real and fake pairs, whereas that of generator is to fool the discriminator by generating an image as close to the transformed input image as possible. For our application using Conditional GANs proved to be be better than pixel-wise segmentation networks like [77] [78], as adversarial training regime allows the network to learn the inherent features of target distribution better. Hence, it produces more realistic segmentation results.

Fig. 4.3 and Fig. 4.4 show the training pipeline and the stalk segmentation results respectively. At deployment, the generator network is run with 256x256 images.

4.2.2 Point Cloud Stitching

The Multisense S7 camera uses a hardware trigger to publish an organized point cloud and a rectified image from the left camera and an organized point cloud. Each point in the organized point cloud has a corresponding pixel in the left camera frame. The image from the left camera frame is sent to the conditional GAN, which returns an image where all of the stalks are labelled red. The pixel locations from the labeled image are copied to the point cloud so that the stalks are labelled red in the point cloud.

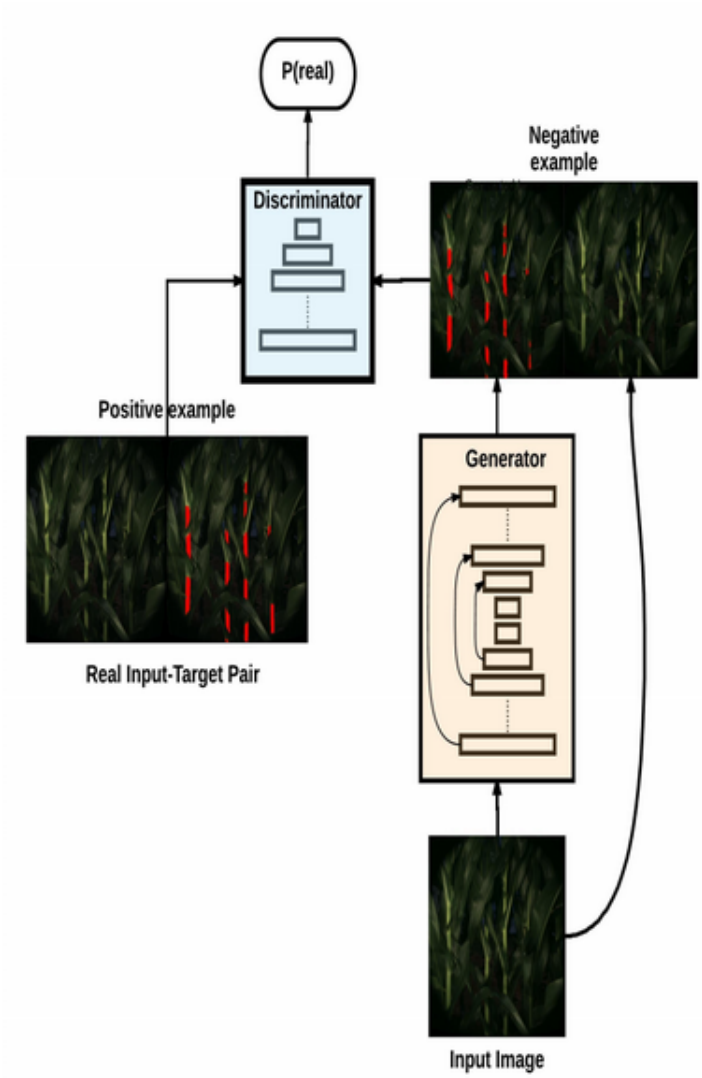

Figure 4.3: Conditional GAN training pipeline shows an input image fed to the generator which generated a segmented image, the discriminator is trained to classify the hand labelled input-target image pair from a generated input-masked-image image pair.

In order to overcome the challenges of visual occlusion, 5 consecutive point clouds are stitched together. The point clouds are stitched using the Iterative Closest Point (ICP) algorithm, which is implemented natively in Point Cloud Library (PCL). The maximum correspondence distance be- tween points was set to 0.15m, and the cloud was down- sampled using a 3 cm x 3 cm x 3 cm voxel grid for faster correspondence search. Once the clouds are stitched, they are thresholded for red color. We then

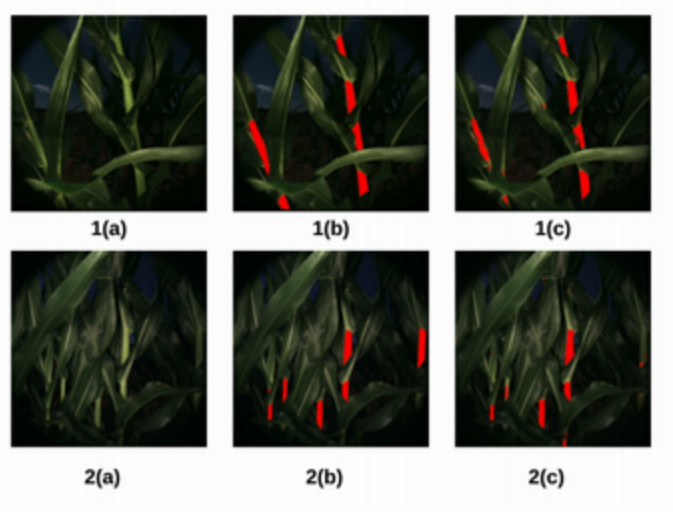

Figure 4.4: 1-2(a): Example input images, 1-2(b): Ground truth labelled images, 1-2(c): Conditional GAN detection.

deploy the technique used by Jenkins et. al. [22] wherein the point cloud is divided into 0.2 m segments and each segment is projected onto the x-y plane. Points on the x-y plane associated with stalks tend to be clustered together due to a stalk's vertical profile. After 2D projection and region growing is applied to every slice, the centroids of each stalk region are reprojected to 3D space in the robot frame. Figure 4.5 illustrates the detection pipeline and Figure 4.6 illustrates the method of slicing a point cloud and region-growing the 2D projection.

4.2.3 Servoing

The 3D centroids described in the previous section represent points on sorghum stalks to which the manipulator can servo. Trajectory optimization takes approximately 12 microseconds and merely consists of parameterizing a 2D line in the manipulator plane between the manipulator's first joint and the target. The arm trajectory is controlled using PID on position error and joint velocity is controlled proportional to position error. Error estimation takes place at approximately 250 Hz and arm movements take an average of 2 s to reach a target. The motor actuating the linear stage is controlled with a PI-controller. This is because the moment of inertia of the carriage on the linear stage is very small and a derivative term is not necessary to prevent overshoot. All three motors were tuned manually using the quarter amplitude decay method. Fig. 4.7 illustrates the planar trajectory of the manipulator.

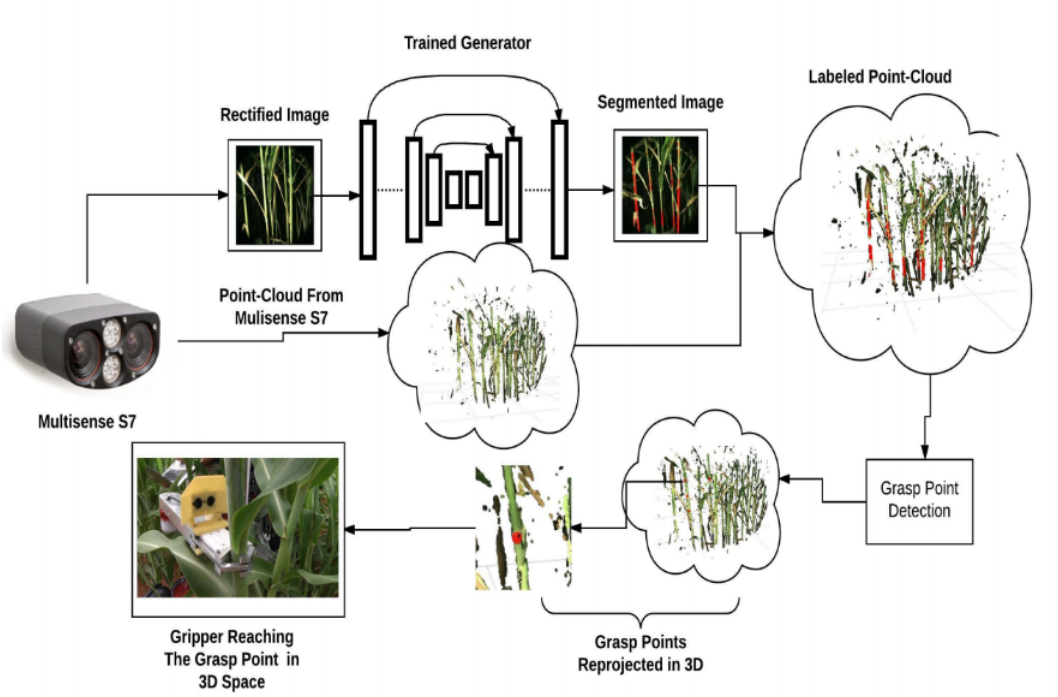

Figure 4.5: The process pipeline: The left image from the Multisense S7 is fed into the trained generator, which masks the stalks in the image with red color. The RGB values of organized point-cloud for the corresponding image is replaced by the output image of generator. Grasp points are detected within this masked point cloud, and the gripper servos to the stalks.

4.3 Results

Network fine-tuning consisted of feeding 60 labeled images into the GAN for pixelwise semantic segmentation pretrained on Cityscapes dataset [31]. This number is surprisingly low for a deep learning application, however our pipeline is a two way classification in a relatively homogeneous environment with consistent camera orientation and lighting conditions. This makes the testing images resemble the training dataset. The Conditional GAN took about 45 minutes to train 200 epochs on a NVIDIA GTX1080 GPU.

The robot was deployed in Florence, South Carolina, USA, on sorghum plants grown for bioenergy. The plants were 90 days old and approximately 2.5m tall. The robot was tested in 16 individual stalk grasping experiments. Each experiment consisted of the robot autonomously driving 0.5m down a sorghum row while running the end-to-end stalk detection and grasping pipeline. While driving down a sorghum row, the system captured five stereo pairs and autonomously servoed the

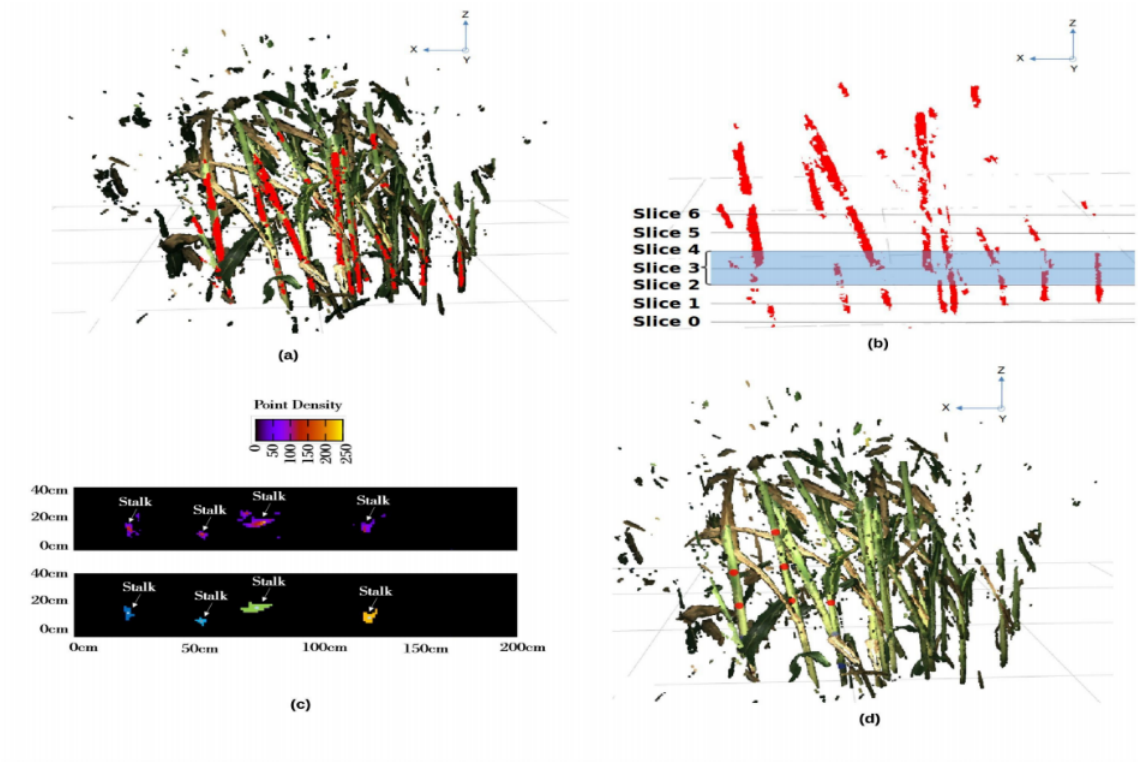

Figure 4.6: (a) Point cloud with masked stalks; (b) Point cloud after thresholding for red color; (c) Result of applying x-y plane projection to a bin in the point cloud. Point density heat-map (top) is filtered and region-growing is applied to segment high density regions (bottom). The centroids of these regions are the prospective grasp points; (d) Shows the detected grasp points (red spheres) within the robot's physical reach, which is only in the left region of the point cloud.

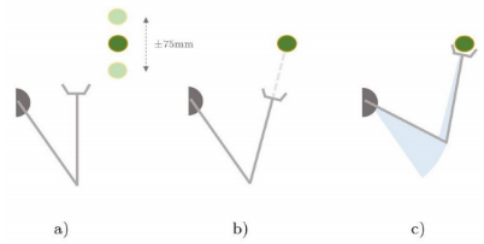

Figure 4.7: Manipulator trajectory outline

manipulator to every grasp point detected by the Conditional GAN pipeline.

In Table 4.1 we show the Precision, Recall and F1-score for pixelwise 2D stalk detection using a Conditional GAN. We achieve a precision of 0.937, recall of 0.872,

Precision	Recall	F1 Score
0.937	0.872	0.903

Table 4.1: Precision, Recall and F1 score for stalk detection accuracy.

and an F1-score of 0.903.

The stereo camera's field of view is wider than the reach space of the manipulator, so not all stalks detected in the camera image are reachable. Therefore we define grasping accuracy as the absolute difference of number of stalks grasped and number of stalks attempted, divided by the number of stalks attempted: $1 - \frac{abs(error)}{numberofattempts}$. The average grasping accuracy over all 16 experimental runs was 74.13%. Fig. 4.8 shows the results for all 16 experiments. We report accuracy for number of stalks attempted vs. number of stalks reached, rather than the number of detected stalks vs. number of stalks grasped. This is because not all detected stalks are in the reachable proximity of the gripper. As observed in Fig. 4.8, it is often the case that the number of stalks grasped is greater than number of grasps attempted, i.e. the gripper often grips more than one stalk in an attempt. This is a result of high stalk density and lack of collision avoidance in the gripper trajectory planning in our current approach.

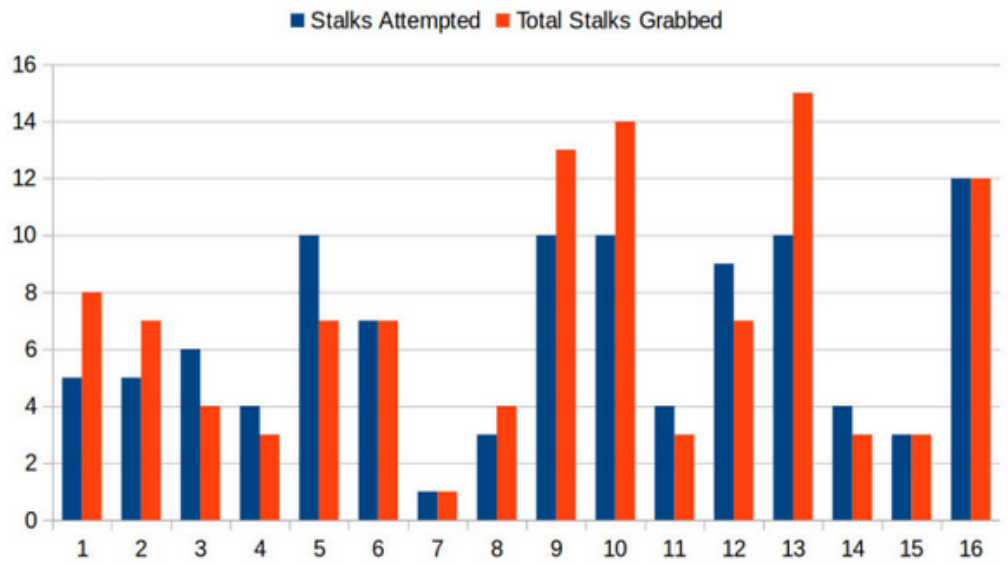

Figure 4.8: Number of stalks grabbed vs stalks number of attempted

Chapter 5

Reinforcement Learning for Autonomous Navigation

Figure 5.1: Clear-path Jackal in a field of hops

The task is to develop a generic algorithm that can learn to navigate in different environments using images as observations. Since images provide the most dense and relevant sensor modality for the task of autonomous navigation. The proposed architecture strives to be generalizable across different agricultural fields and

is tested indoors, on road, in vineyards and in hops. The algorithm learns a simple policy for vision based navigation in less than 2 hours of training from scratch. Figure 5.1 shows the image of the test platform in a field of hops.

5.1 System Diagram Overview

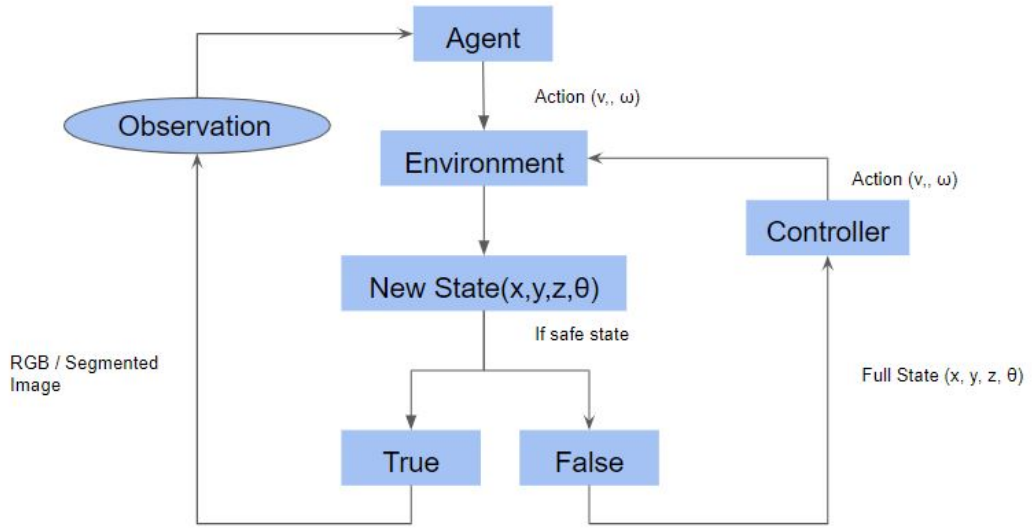

Figure 5.2: System Overview

The architecture is motivated from [52]. Kendall et. al. use DDPG [79] for learning a neural network policy that learns to regress driving commands directly from images, without any ground truth data. They utilize a simple reward function that penalizes the agent if it is in an unsafe state, when a human driver takes over and resets the agent to a safe state. They further report using a VAE for encoding images into a low dimensional state-space decreases the train time.

We propose a similar architecture, but unlike [52], we use a controller that assumes full state observation to reset the robot. This makes the algorithm independent of human involvement. We also propose using Proximal Policy Optimization (PPO) [48] as our learning algorithm, owing to its basis in Trust Region Policy Optimization (TRPO) [47], that guarantees monotonic improvements.

As Figure 5.2 depicts, overview of the learning cycle. The agent observes an observations, which in this case is the encoding of an image from a VAE and takes

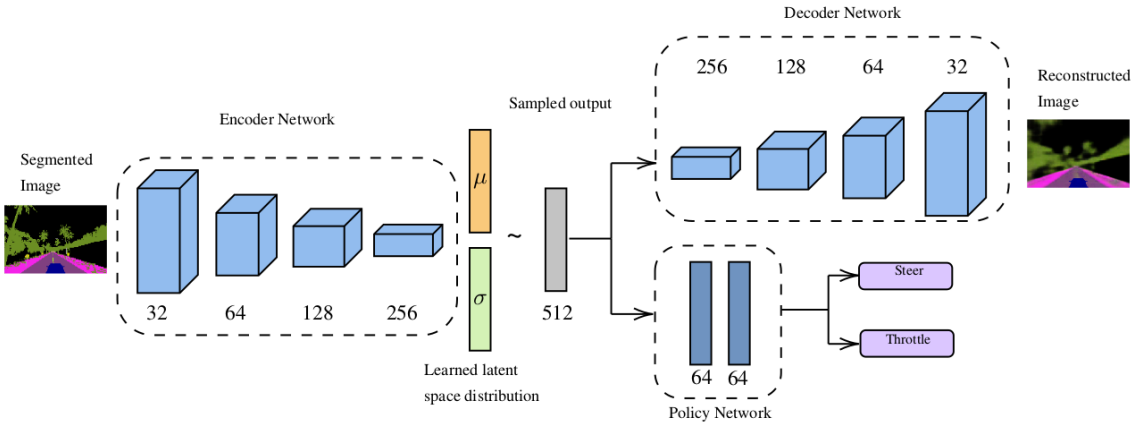

Figure 5.3: Variational autoencoder + policy network architecture diagram

a step in the environment. The agent then ends up at the next state. While Training a full state of the system is tracked. If the robot deviates too far away from a pre-defined trajectory, a controller (DWA Planner) kicks in and resets the robot to a safe state, from where the training resumes. The flow from image observation to actions is depicted by Fig. 5.3

We first train the agent in CARLA, a simulator for urban driving simulation and report training times of less that 30 minutes. Then we test the same algorithm with a reset-controller in a custom built ROS based environment, modeled in Gazebo. The robot model used is that of a Clearpath Jackal. Then we report the same algorithm being deployed on a real robot and report results for navigation tasks in widely different environments like, indoor-hallways, outdoor-roads, vineyards and fields of hops.

5.2 VAE

Policy gradient is known to be noisy. Hence, it is desirable to keep the number of parameters being trained low. Thus, in case the observations are images, it would be desirable to have a meaningful lower dimensional encoding of the input image, to keep the policy network small.

Variational Auto encoders are simply neural networks that take in an input image and produce a smaller representation (encoding), that contain enough relevant information to convert the image into the desired output format. It consists of two parts:

- **Encoder:** Passes the image through a bunch of convolution layers and

produces a small dimensional encoding of the input.

- **Decoder:** Takes the low-dimensional encoding and uses a bunch of Up-Convs to reconstruct the desired output format.

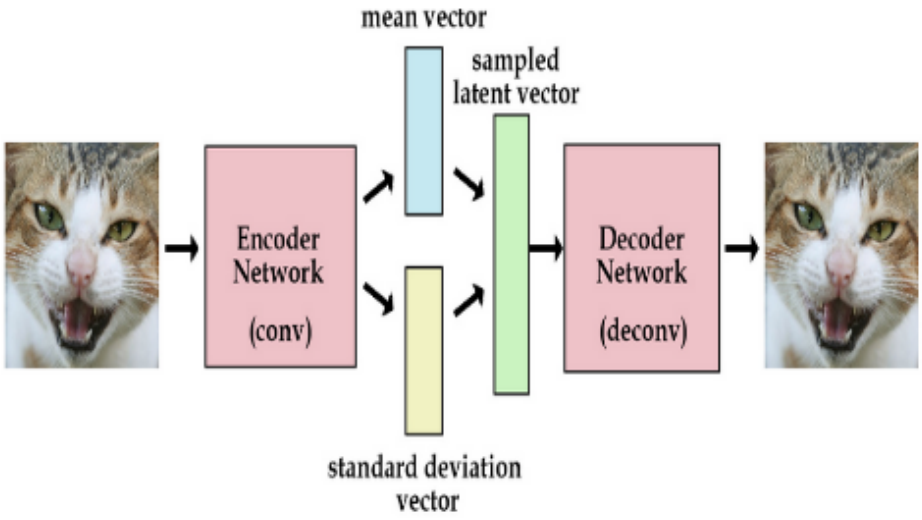

Figure 5.4: Architecture of a Variational Auto Encoder.

Variational Auto-Encoders have one property that makes them better than vanilla Auto-Encoders, viz., rather than learning a discrete encoding for the inputs, they learn the parameters of a Normal Distribution. Intuitively, the mean parameters control the where the encoding of the input is centered around in the encoding space and the variance parameters control how much the encoding can vary. As different encodings might be generated for the same image, the decoder learns top decode not only the one point in the latent space but all the near by point as well.

But, we would like some overlap between the encodings of different classes of images as well. This is done by enforcing a KL divergence constraint between the predicted parameters and a 0 centered unit Gaussian. Also, since the job of the Auto-Encoder here is to construct the encoding of the image, such that, that encoding contains the most amount of information regarding the image, the Auto-Encoder is trained to minimize an L1 reconstruction loss between the input image and the reconstructed image. So the overall loss that the Auto-Encoder tries to minimize is:

$$L_{KL} = \Sigma_{i=1}^{n}(\sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1) \tag{5.1}$$

Where μ and σ are the mean and standard deviation of the Normal Distribution and n is the size (dimension) of the encoding. In our case it is 512.

$$L_{L_1} = \Sigma_{i=1}^{Row}\Sigma_{j=1}^{Col}(abs(X_{i,j} - \hat{X}_{i,j})) \tag{5.2}$$

Here X is the input image and $\hat{X}$ is the reconstructed image of size Row x Col .

$$L_{VAE} = \lambda_1 L_{KL} + \lambda_2 L_{L_1} \tag{5.3}$$

Figure 5.5 shows the sample input image and the respective reconstructed image.

Figure 5.5: Left: input to the VAE. Right: Reconstruction from the produces encoding.

5.3 PPO

Proximal Policy Optimization (PPO) described in [48] is used. PPO belongs to a family of policy optimization algorithms. These algorithms, in general, estimate a policy gradient and take a gradient ascent step in the direction of the gradient, to make better actions more likely.

$$\hat{g} = \hat{E}_t[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{5.4}$$

Intuitively it may seem logical to perform multiple optimization steps with the gradient estimate in eqn.5.4, using a loss function shown by eqn.5.5. However, in practice, it is not stable, because doing so might lead to unstable and too large policy updates [47].

$$L^{PG}(\theta) = \hat{E}_t[\log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{5.5}$$

Proximal Policy Optimization replaces the objective specified in eqn.5.5 with a surrogate objective given by eqn.5.6

$$L^{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta))\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t] \tag{5.6}$$

Here, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the ratio between the action probabilities, under the current and old policy. This ratio, effectively, penalizes bad actions and rewards actions with high advantage, while limiting the change in policy. The objective function in eqn.5.6 is an improvement over TRPO [47], whose objective function maximizes eqn.5.7, subject to a constraint on the KL divergence of the new and old policy, $\hat{E}_t[KL[\pi_{\theta_{old}}(a|s_t), pi_\theta(a|s_t)]] \leq \delta$, where δ is a hyper-parameter.

$$L^{TRPO}(\theta) = \hat{E}_t[\hat{r}_t(\theta)\hat{A}_t] \tag{5.7}$$

Thus, $L^{CLIP(\theta)}$ adds a second term to $L^{TRPO}(\theta)$, that clips the probability ratio, $r_t(\theta)$ to stay between the range $1-\epsilon, 1+\epsilon$. Finally it takes a min over the clipped objective and the un-clipped objective, so the final objective is a lower bound on $L^{TRPO}(\theta)$ Hence, effectively this objective prevents us form performing a huge greedy update, if,

- the advantage value $\hat{A}_t$ is positive and the action probability ratio $r_t(\theta)$ is greater than 1.
- the advantage value $\hat{A}_t$ is negative and the action probability ratio $r_t(\theta)$ is between 0 and 1.

Also, it would allow a large update if the ratio $r_t(\theta)$ is greater than 1 and the advantage value $\hat{A}_t$ is negative. In other words, if the current policy is highly confident about an action that reduces performance, as compared to that of the old policy, the loss L^{CLIP} would allow a huge update away form the current bad policy. Note that [48] also included a KL divergence loss in their objective. But we did not use it in our training and hence, not mentioning it here. Algorithm 1 shows the training algorithm for PPO. Algorithm 2 shows the training regime for training PPO and VAE in conjunction. This can easily be extended to any other RL algorithm as well.

The value of ϵ used is 0.2 as [48] reports best performance for that value.

5.4 Controller

We use ROS's move_base package's DWA planner as the local planner. This package provides a controller that drives the robot with a model predictive controller. The controller's job is to produce d_x, d_y and d_θ velocities to send to the robot, that has its own PID controller running to attain those command velocities.

The basic idea the Dynamic Window Approach (DWA) Planner is as follows:

Algorithm 1 Proximal Policy Optimization

for k=1,2,... **do**

  for l=1,2,...,N **do**

    Run $\pi_{\theta old}$ for H time steps

    Compute the advantage estimates $\hat{A}_1, \hat{A}_2, ...\hat{A}_T$

  Optimize surrogate L^{CLIP} wrt θ, with K epochs

  $\theta \leftarrow \theta_{old}$

Algorithm 2 Proximal Policy Optimization with VAE training

for Iteration=1,2,... **do**

  for Timestep=1,2,...,N **do**

    Observation = VAE.encode(Image)

    $a_t = \pi_{\theta old}(Observation)$

    Compute the advantage estimate $\hat{A}_t$

    Append Image to VAE Buffer

  Optimize surrogate L^{CLIP} wrt θ, with K epochs.

  Optimize VAE weights.

  $\theta \leftarrow \theta_{old}$

- Discretely sample in the robot's control space (dx,dy,d_θ)
- For each sampled velocity, perform forward simulation from the robot's current state to predict what would happen if the sampled velocity were applied for some (short) period of time.
- Evaluate (score) each trajectory resulting from the forward simulation, using a metric that incorporates characteristics such as: proximity to obstacles, proximity to the goal, proximity to the global path, and speed. Discard illegal trajectories (those that collide with obstacles).
- Pick the highest-scoring trajectory and send the associated velocity to the mobile base.
- Rinse and repeat.

5.5 Simulation

We first test the experiment in CARLA simulator. This is an urban simulator, based on Unreal Engine and offers wide varieties of support development, training, and validation of autonomous driving systems. It supports flexible specification of sensor suites, environmental conditions, full control of all static and dynamic actors, maps generation, depth maps, semantic segmented scenes and much more. It comes with an easy to use Python API, that makes coding and developing OpenAI Gym based environments very straight forward. This makes this ideal for reinforcement learning for self driving.

Once the algorithm solved the task of navigation in different scenes in Carla, we built a Gazebo based simulates vineyard environment, so that the algorithm could be more closely tested to the real robot. For this purpose, a custom built vineyard environment was developed, from vine meshes generated from point-clouds obtained from stereo images of real vines in the field. We used Clearpath Jackal's Gazebo model as our learning agent. This also gives us a realistic model of the robot to test the reset controller.

5.5.1 Carla

We train the agent in Carla's Town 2 environment, as shown in Figure. . The environment is 205,59 x 204,48 m. The agent's objective is simply to stay on the road for as long as possible and gets a +1 reward for every time-step an negative reward for every collision, where the robot resets.

Figure. 5.6 shows the image of the map of the town and figure 5.7 shows the image of the semantically segmented scene.

Figure 5.6: Carla's Town 2 map

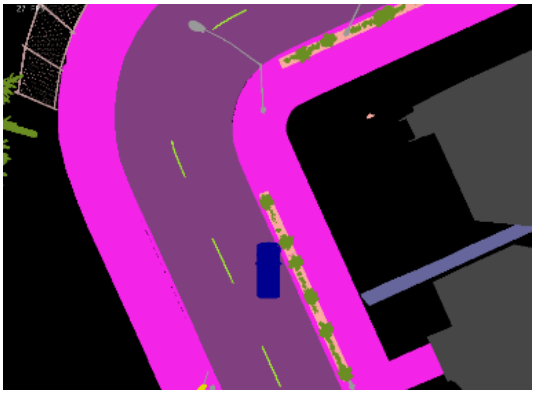

Figure 5.7: Semantically segmented image

We test our algorithm with RGB images and semantically segmented images and find that the algorithm learns to navigate faster with the semantically segmented inputs, as depicted by Fig. 5.8. The agent learns to navigate the environment in just 9 minutes of training on a GTX 1080 GPU. Table 5.1 shows the amount of time required for training.

5.5.2 ROS

Once the algorithm is successfully implemented in Carla, the next step is to implement it over a more realistic implementation of the robot. For this purpose, Gazebo was used. It is a physics simulator that allows to accurately simulate many robot models. It comes with a ROS based wrapper, that allows to develop code that could

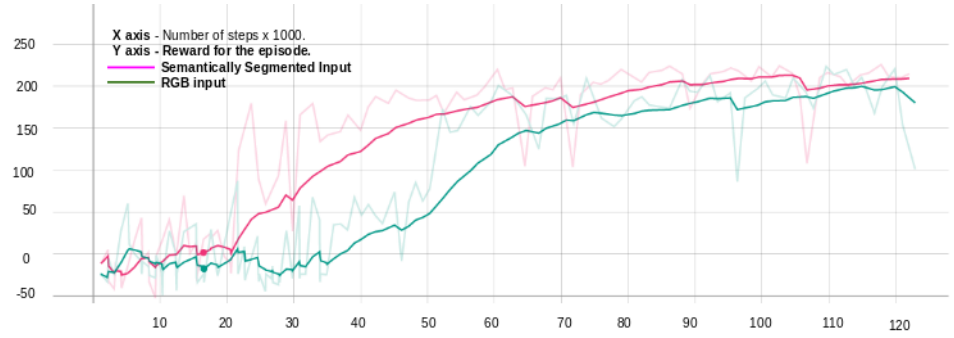

Figure 5.8: Comparison in semantically segmented and RGB input images.

Input image	Number of train steps	Training time (min)	Distance Travelled (m)
Segmented	40.97k	9	818
RGB	121.1k	30	214

Table 5.1: Training time taken for RGB and Semantically segmented images.

be easily translated onto the real robot. We use Clearpath's Jackal's model inside a custom built vineyard environment as our environment. Fig. 5.9 shows the image of the environment.

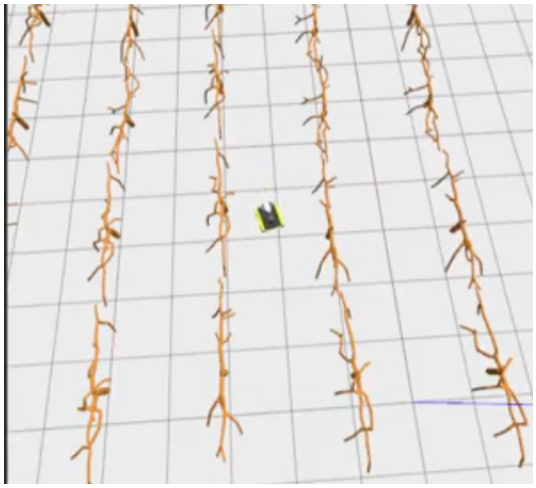

Figure 5.9: Simulated Vineyard Environment in Gazebo

5.6 Hardware

Going from simulation to real robot, we train and test the algorithm on a Clear-path Jackal robot. To demonstrate generalizability, we show experiments on 4 different kinds of environments

- On jogging tracks
- Vineyard
- Hops plantation

We compare the performance of RGB inputs vs semantically segmented inputs and also compare the performance of the system with a human manually resetting the robot vs using a controller to reset the robot. We use a Clear-path Jackal for all the experiments. The Jackal has a GTX-1050 on-board and is quipped with a front-facing stereo pair, a Velodine VLP 16 LiDAR and a Piksi RTK GPS system. The GPS system is used by the controller to reset the robot in case the robot strays too far away from the nominal path and the from facing camera is used as inputs for the VAE that in turn generates observations for the neural-network policy. The platform is shown in Fig. 5.15.

Figure 5.10: Left: Front view of the robot. Right: top view of the robot.

5.7 Experiments

5.7.1 Jogging track

We train the architecture over asphalt jogging track. The overall path is 620m in length. For this experiment, the agent was manually reset, rather than using the

GPS based reset. This was a an experiment to sanity-check the algorithm on the robot. The path chosen is shown in Fig 5.11. The robot ran for a max of 107 meters without manual intervention. The reward curve is shown in Fig 5.12. Table 5.2 shows the performance of the algorithm.

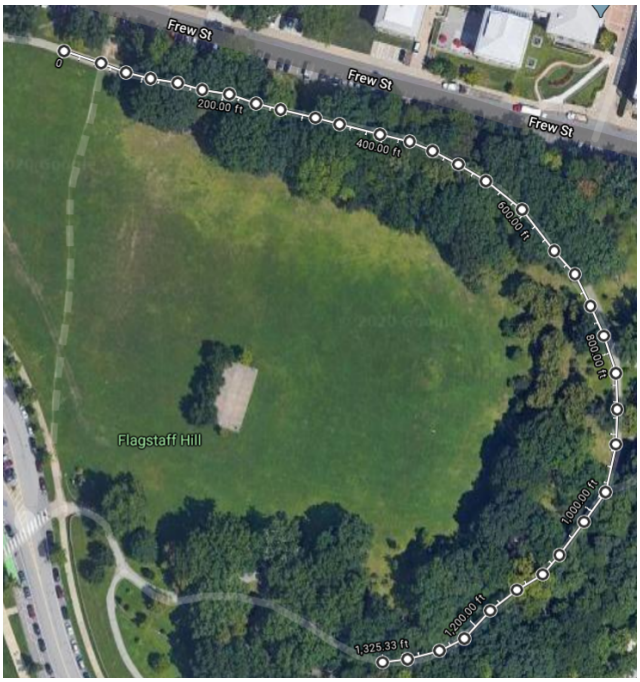

Figure 5.11: Outdoor training track

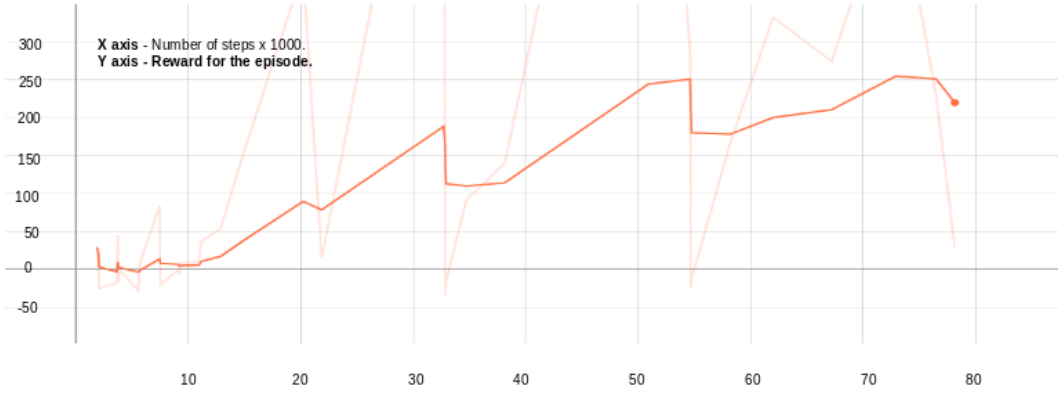

Figure 5.12: Outdoor training reward curve.

Input image	Training time (min)	Distance Travelled (m)	Reset
Segmented	52	107	Manual
RGB	94	57	Manual

Table 5.2: Training time taken for RGB and Semantically segmented images in outdoor jogging tracks.

5.7.2 Vineyard

The robot was trained to follow the rows in a vineyard in Cornell Lake Erie Research Extension at the Geo-location 42.371595, -79.486393. The length of each row is around 148 meters. Prior to training, the robot was manually driven across the rows to collect GPS points at a resolution of 2 meters. During testing the GPS was turned off. In the best case, the robot was able to navigate the row, requiring only 3 manual interventions. Fig. 5.13 shows the location where the agent was trained, Fig. 5.14 shows the reward curve for the agent and Fig. 5.15 shows the input image and the VAE reconstruction. Table 5.3 shows the performance of the algorithm.

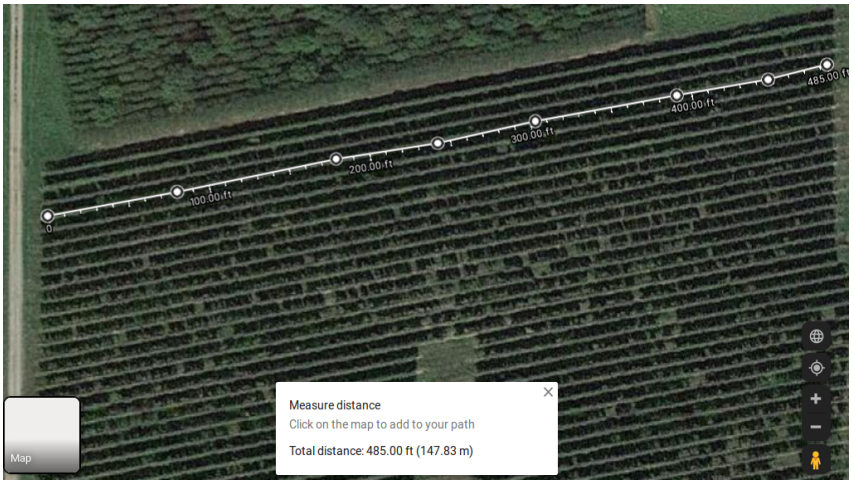

Figure 5.13: The vineyard row in which the robot was trained.

5.7.3 Hops Plantation

To demonstrate generalizability and full system with human free training, the robot was also trained on a field of hops. The system was trained with only RGB images to demonstrate fully human-free training. Each row was about 91 meters long, as shown in Fig. 5.16. The robot manually driven across the field to collect GPS points. The testing was done in the adjacent row. Fig. 5.18 shows the reconstructed image

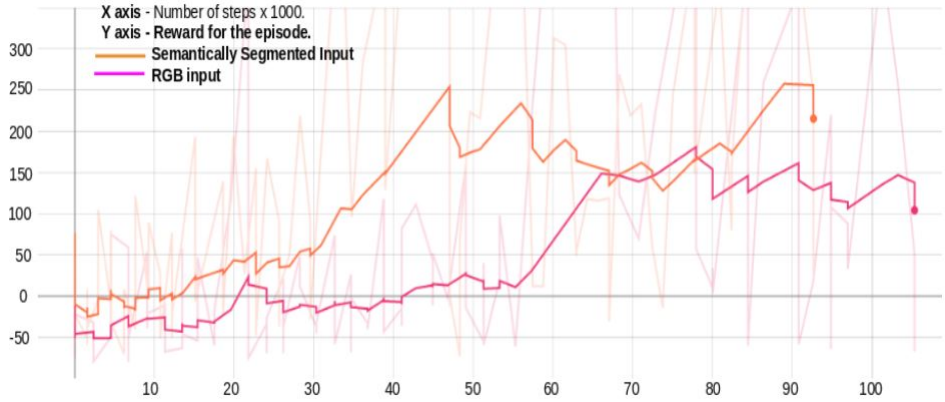

Figure 5.14: The reward curve for the agent trained in vineyard.

Figure 5.15: Left: Input RGB image. Right: VAE reconstruction for vineyards.

Input image	Training time (min)	Distance Travelled (m)	Reset
Segmented	56	112	Manual
RGB	95	110	Manual

Table 5.3: Training time taken for RGB and Semantically segmented images in vineyards.

and Fig. 5.17 shows the training curve for the agent in hops. Table 5.4 shows the results for navigation in hops.

Input image	Training time (min)	Distance Travelled (m)	Reset
RGB	79	87.2	Controller

Table 5.4: Training time taken for RGB and Semantically segmented images in Hops.

Figure 5.16: Hops plantation

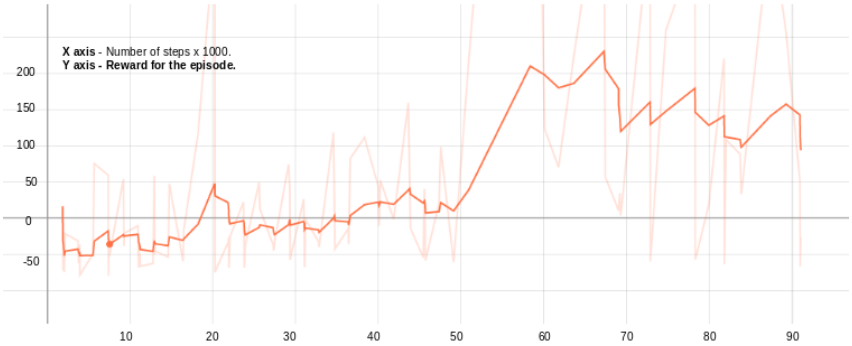

Figure 5.17: Reward Curve for the training in hops.

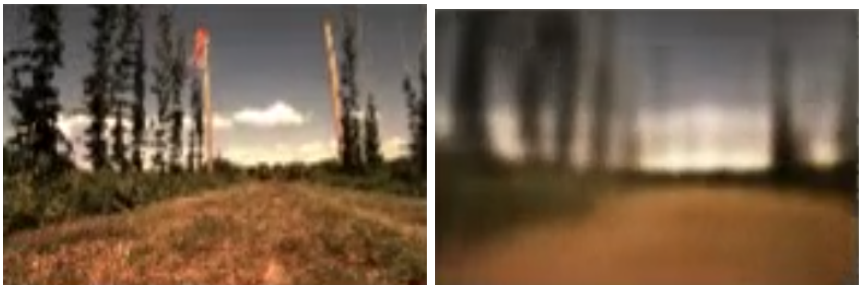

Figure 5.18: Left: Input RGB image. Right: VAE reconstruction for Hops

Chapter 6

Conclusion

This thesis makes the following contribution towards robotics and automation in agriculture.

- An end to end pipeline for high throughput image based plant phenotyping.
- A deep learning based pipeline for grasping stalks for tactile plant phenotyping.
- A deep-reinforcement learning based architecture that can be deployed in different environments and is deployed on a real robot, training from scratch.

We presented that robotics aided by deep-learning can provide high fidelity and generalizable solutions for visual phenotyping, manipulation, and vision based navigation in agricultural setting. We find that vision based measurements are within 2.76 mm of the ground truth for stalk width and within 10% of human count for stalk count. For stalk grasping, we report an accuracy of 74.13% with a stalk detection F1 score of 0.90.

We propose an architecture that uses reinforcement learning with a planer based controller for safely resetting the robot. We propose the use of Variational Auto-Encoder for dimensionality reduction, so that the policy can be trained easier. We demonstrate learning of a simple road following policy in less than 10 minutes of training time. We deploy the algorithm in different environments, on a real robot and report its performance and demonstrate the generalizability of the algorithm. We show that having a consistent input representation like semantically segmented inputs can aid training and show that we can train agents in novel environments with no human inputs and with comparable performance to human aided training.

Chapter 7

Future Work

In future work for visual phenotyping we plan to integrate more accurate positioning to merge multiple views of the plants into more accurate measurements of plant traits. The plant manipulation pipeline could be used to extract several other more sophisticated phenotypes such as leaf angle, leaf necrosis index and light interception. We also plan to incorporate collision avoidance in gripper trajectory planning to avoid grasping multiple stalks. This should increase the grasping accuracy. Another area of investigation is depth estimation learned from plant features in monocular images. Consistent lighting provided by a synchronized flash would improve the viability of this application. Depth estimation from monocular images would not only allow us to replace the stereo camera with a low-cost camera but also eliminate the need to calibrate the stereo sensor. This could prove to be a difficult task, but is a possible area of exploration

For the navigation pipeline, we see that the algorithm learns to navigate in simulation easily, but not as easily on the robot. This is because in outdoor conditions lighting conditions change drastically, hence the quality of observations is low. Using semantically segmented images helps training by making images more consistent, but this introduces a human element to training. We can make images consistent without using semantic segmentation by using active lighting. Hence, we will explore the use of active lighting for consistency in observations.

Bibliography

[1] Nikos Alexandratos and Jelle Bruinsma. World agriculture towards 2030/2050: the 2012 revision. 2012.

[2] Bert Metz, Ogunlade Davidson, Peter Bosch, Rutu Dave, and Leo Meyer. *Climate change 2007: Mitigation of climate change*. Cambridge Univ. Press, 2007.

[3] Sotirios A Tsaftaris, Massimo Minervini, and Hanno Scharr. Machine learning for plant phenotyping needs image processing. *Trends in plant science*, 21(12):989–991, 2016.

[4] Ryo Sugiura, Shogo Tsuda, Seiji Tamiya, Atsushi Itoh, Kentaro Nishiwaki, Noriyuki Murakami, Yukinori Shibuya, Masayuki Hirafuji, and Stephen Nuske. Field phenotyping system for the assessment of potato late blight resistance using rgb imagery from an unmanned aerial vehicle. *Biosystems engineering*, 148:1–10, 2016.

[5] Zania Pothen and Stephen Nuske. Automated assessment and mapping of grape quality through image-based color analysis. *IFAC-PapersOnLine*, 49(16):72–78, 2016.

[6] AR Jimenez, R Ceres, and Jose L Pons. A survey of computer vision methods for locating fruit on trees. *Transactions of the ASAE*, 43(6):1911, 2000.

[7] Gregory M Dunn and Stephen R Martin. Yield prediction from digital image analysis: A technique with potential for vineyard assessments prior to harvest. *Australian Journal of Grape and Wine Research*, 10(3):196–198, 2004.

[8] G Rabatel and C Guizard. Grape berry calibration by computer vision using elliptical model fitting. 2007.

[9] Sheng Xu, Yi Xun, Tingmeng Jia, and Qinghua Yang. Detection method for the buds on winter vines based on computer vision. In *2014 Seventh International Symposium on Computational Intelligence and Design*, volume 2, pages 44–48. IEEE, 2014.

[10] G Alenyà, B Dellen, S Foix, and C Torras. Leaf segmentation from tof data for robotized plant probing. *IEEE Robotics & Automation Magazine*, 2012.

[11] Chunlei Xia, Longtan Wang, Bu-Keun Chung, and Jang-Myung Lee. In situ 3d segmentation of individual plant leaves using a rgb-d camera for agricultural automation. *Sensors*, 15(8):20463–20479, 2015.

[12] Chunming Li, Jundong Liu, and Martin D Fox. Segmentation of edge preserving gradient vector flow: an approach toward automatically initializing and splitting of snakes. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 162–167. IEEE, 2005.

[13] Mario Valerio Giuffrida, Massimo Minervini, and Sotirios A Tsaftaris. Learning to count leaves in rosette plants. 2016.

[14] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[15] Charles Elkan. Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th international conference on Machine Learning (ICML-03)*, pages 147–153, 2003.

[16] Chunlei Xia, Jang-Myung Lee, Yan Li, Yoo-Han Song, Bu-Keun Chung, and Tae-Soo Chon. Plant leaf detection using modified active shape models. *Biosystems engineering*, 116(1):23–35, 2013.

[17] Harjatin Singh Baweja, Tanvir Parhar, and Stephen Nuske. Early-season vineyard shoot and leaf estimation using computer vision. In *2017 ASABE Annual International Meeting*, St. Joseph, Michigan www.asabe.org, November 2017.

[18] Z Mohammed Amean, Tobias Low, Cheryl McCarthy, and Nigel Hancock. Automatic plant branch segmentation and classification using vesselness measure. In *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2013)*, pages 1–9. Australasian Robotics and Automation Association, 2013.

[19] Alejandro F Frangi, Wiro J Niessen, Koen L Vincken, and Max A Viergever. Multiscale vessel enhancement filtering. In *International conference on medical image computing and computer-assisted intervention*, pages 130–137. Springer, 1998.

[20] Zania S Pothen and Stephen Nuske. Texture-based fruit detection via images using the smooth patterns on the fruit. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5171–5176. IEEE, 2016.

[21] Omeed Mirbod, Luke Yoder, and Stephen Nuske. Automated measurement of berry size in images. *IFAC-PapersOnLine*, 49(16):79–84, 2016.

[22] Merritt Jenkins and George Kantor. Online detection of occluded plant stalks for manipulation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5162–5167. IEEE, 2017.

[23] Suchet Bargoti, James P Underwood, Juan I Nieto, and Salah Sukkarieh. A pipeline for trunk detection in trellis structured apple orchards. *Journal of field robotics*, 32(8):1075–1094, 2015.

[24] Paloma Sodhi, Srinivasan Vijayarangan, and David Wettergreen. In-field segmentation and identification of plant structures using 3d imaging. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5180–5187. IEEE, 2017.

[25] Stefan Paulus, Jan Dupuis, Anne-Katrin Mahlein, and Heiner Kuhlmann. Surface feature based classification of plant organs from 3d laserscanned point clouds for plant phenotyping. *BMC bioinformatics*, 14(1):238, 2013.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[28] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):640–651, 2017.

[29] Inkyu Sa, Christopher McCool, Christopher Lehnert, and Tristan Perez. On visual detection of highly-occluded objects for harvesting automation in horticulture. ICRA, 2015.

[30] Calvin Hung, Juan Nieto, Zachary Taylor, James Underwood, and Salah Sukkarieh. Orchard fruit segmentation using multi-spectral feature learning. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5314–5320. IEEE, 2013.

[31] Chris McCool, ZongYuan Ge, and Peter I Corke. Feature learning via mixtures of dcnns for fine-grained plant classification. In *CLEF (Working Notes)*, pages 511–517, 2016.

[32] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

[33] Steven W Chen, Shreyas S Shivakumar, Sandeep Dcunha, Jnaneshwar Das, Edidiong Okon, Chao Qu, Camillo J Taylor, and Vijay Kumar. Counting apples and oranges with deep learning: A data-driven approach. *IEEE Robotics and Automation Letters*, 2(2):781–788, 2017.

[34] Michael P Pound, Jonathan A Atkinson, Alexandra J Townsend, Michael H Wilson, Marcus Griffiths, Aaron S Jackson, Adrian Bulat, Georgios Tzimiropoulos, Darren M Wells, Erik H Murchie, et al. Deep machine learning provides state-of-the-art performance in image-based plant phenotyping. *Gigascience*, 6(10):gix083, 2017.

[35] Harjatin Singh Baweja, Tanvir Parhar, Omeed Mirbod, and Stephen Nuske. Stalknet: A deep learning pipeline for high-throughput measurement of plant stalk count and stalk width. In *Field and Service Robotics*, pages 271–284. Springer, 2018.

[36] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[37] C Wouter Bac, Eldert J van Henten, Jochen Hemming, and Yael Edan. Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, 31(6):888–911, 2014.

[38] Johan Baeten, Kevin Donné, Sven Boedrij, Wim Beckers, and Eric Claesen. Autonomous fruit picking machine: A robotic apple harvester. In *Field and service robotics*, pages 531–539. Springer, 2008.

[39] Alistair John Scarfe. *Development of an autonomous kiwifruit harvester: a thesis presented in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Industrial Automation at Massey University, Manawatu, New Zealand.* PhD thesis, Massey University, 2012.

[40] C Wouter Bac. *Improving obstacle awareness for robotic harvesting of sweet-pepper*. Wageningen University, 2015.

[41] Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

[42] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[43] Gregory Kahn, Tianhao Zhang, Sergey Levine, and Pieter Abbeel. Plato: Policy learning using adaptive trajectory optimization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3342–3349. IEEE, 2017.

[44] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[45] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in neural information processing systems*, pages 5048–5058, 2017.

[46] Andrea Franceschetti, Elisa Tosello, Nicola Castaman, and Stefano Ghidoni. Robotic arm control and task training through deep reinforcement learning. *arXiv preprint arXiv:2005.02632*, 2020.

[47] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[48] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[49] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic mpc for model-based reinforcement learning. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, 2017.

[50] Martin A. Riedmiller, Michael Montemerlo, and Hendrik Dahlkamp. Learning to drive a real car in 20 minutes. *2007 Frontiers in the Convergence of Bioscience and Information Technologies*, pages 645–650, 2007.

[51] Xiaodan Liang, Tairui Wang, Luona Yang, and Eric P. Xing. Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In *ECCV*, 2018.

[52] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. *CoRR*, abs/1807.00412, 2018.

[53] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *I. J. Robotics Res.*, 32:1231–1237, 2013.

[54] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Junbo Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[55] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[56] Takeo Kanade, Charles E. Thorpe, and William Whittaker. Autonomous land vehicle project at cmu. In *ACM Conference on Computer Science*, 1986.

[57] Richard S. Wallace, Anthony Stentz, Charles E. Thorpe, Hans P. Moravec, William Whittaker, and Takeo Kanade. First results in robot road-following. In *IJCAI 1985*, 1985.

[58] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Hähnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. Junior: The stanford entry in the urban challenge. In *The DARPA Urban Challenge*, 2009.

[59] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Sören Kammel, J. Zico Kolter, Dirk Langer, Oliver Pink, Vaughan R. Pratt, Michael Sokolsky, Ganymed Stanek, David Stavens, Alex Teichman, Moritz Werling, and Sebastian Thrun. Towards fully autonomous driving: Systems and algorithms. *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168, 2011.

[60] Dean Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NIPS*, 1988.

[61] Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9, 2018.

[62] Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann L. Cun. Off-road obstacle avoidance through end-to-end learning. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 739–746. MIT Press, 2006.

[63] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. *arXiv preprint arXiv:1810.06544*, 2018.

[64] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016.

[65] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. Agile off-road autonomous driving

using end-to-end deep imitation learning. *arXiv preprint arXiv:1709.07174*, 2, 2017.

[66] Ji Zhang, George Kantor, Marcel Bergerman, and Sanjiv Singh. Monocular visual navigation of an autonomous vehicle in natural scene corridor-like environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3659–3666. IEEE, 2012.

[67] Jinlin Xue, Lei Zhang, and Tony E Grift. Variable field-of-view machine vision based row guidance of an agricultural robot. *Computers and Electronics in Agriculture*, 84:85–91, 2012.

[68] Ji Zhang, Andrew Chambers, Silvio Maeta, Marcel Bergerman, and Sanjiv Singh. 3d perception for accurate row following: Methodology and results. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5306–5313. IEEE, 2013.

[69] Santosh Hiremath, Frits K Van Evert, Cajo ter Braak, Alfred Stein, and Gerie van der Heijden. Image-based particle filtering for navigation in a semi-structured agricultural environment. *Biosystems Engineering*, 121:85–95, 2014.

[70] David Reiser, Garrido Miguel, Manuel Vázquez Arellano, Hans W Griepentrog, and Dimitris S Paraforos. Crop row detection in maize for developing navigation algorithms under changing plant growth stages. In *Robot 2015: Second Iberian Robotics Conference*, pages 371–382. Springer, 2016.

[71] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio López, and Vladlen Koltun. Carla: An open urban driving simulator. In *CoRL*, 2017.

[72] Qadeer Khan, Torsten Schön, and Patrick Wenzel. Latent space reinforcement learning for steering angle prediction. *arXiv preprint arXiv:1902.03765*, 2019.

[73] Tim Mueller-Sim, Merritt Jenkins, Justin Abel, and George Kantor. The robotanist: a ground-based agricultural robot for high-throughput crop phenotyping. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3634–3639. IEEE, 2017.

[74] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[75] Andrew W Fitzgibbon, Robert B Fisher, et al. *A buyer's guide to conic fitting.* University of Edinburgh, Department of Artificial Intelligence, 1996.

[76] Mohamed Chafik Bakkay, Hatem Rashwan, Houssam Salmane, Louahdi Khoudour, Domenec Puig, and Yassine Ruichek. Bscgan: Deep background subtraction with conditional generative adversarial networks. 2018.

[77] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. *CoRR*, abs/1611.07709, 2016.

[78] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[79] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.