Artificial intelligence and image processing applications for high-throughput phenotyping

by

Venkata Siva Kumar Margapuri

B.S., Jawaharlal Nehru Technological University, 2014

M.S., Bradley University, 2015

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science

Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2022

Abstract

The areas of Computer Vision and Scientific Computing have witnessed rapid growth in the last decade with the fields of industrial robotics, automotive and healthcare acting as the primary vehicles for research and advancement. However, related research in other fields, such as agriculture, remains an understudied problem. This dissertation explores the application of Computer Vision and Scientific Computing in an agricultural domain known as High-throughput Phenotyping (HTP). HTP is the assessment of complex seed traits such as growth, development, tolerance, resistance, ecology, yield, and the measurement of parameters that form more complex traits.

The dissertation makes the following contributions: The first contribution is the development of algorithms to estimate morphometric traits such as length, width, area, and seed kernel count using 3-D graphics and static image processing, and the extension of existing algorithms for the same.

The second contribution is the development of lightweight frameworks to aid in synthetic image dataset creation and image cropping for deep neural networks in HTP. Deep neural networks require a plethora of training data to yield results of the highest quality. However, no such training datasets are readily available for HTP research, especially on seed kernels. The proposed synthetic image generation framework helps generate a profusion of training data at will to train neural networks from a meager samples of seed kernels. Besides requiring large quantities of data, deep neural networks require the input to be a certain size. However, not all available data are in the size required by the

deep neural networks. The proposed image cropper helps to resize images without resulting in any distortion, thereby, making image data fit for consumption.

The third contribution is the design and analysis of supervised and self-supervised neural network architectures trained on synthetic images to perform the tasks of seed kernel classification, counting and morphometry. In the area of supervised image classification, state-of-the-art neural network models of VGG-16, VGG-19 and ResNet-101 are investigated. A Simple framework for Contrastive Learning of visual Representations (SimCLR) [133], Momentum Contrast (MoCo) [55] and Bootstrap Your Own Latent (BYOL) [119] are leveraged for self-supervised image classification. The instance-based segmentation deep neural network models of Mask R-CNN and YOLO are utilized to perform the tasks of seed kernel classification, segmentation and counting. The results demonstrate the feasibility of deep neural networks for their respective tasks of classification and instance segmentation. In addition to estimating seed kernel count from static images, algorithms that aid in seed kernel counting from videos are proposed and analyzed. Proposed is an algorithm that creates a slit image which can be analyzed to estimate seed count. Upon the creation of the slit image, the video is no longer required to estimate seed count, thereby, significantly lowering the computational resources required for the estimation.

The fourth contribution is the development of an end-to-end, automated image capture system for single seed kernel analysis. In addition to estimating length and width from 2-D images, the proposed system estimates the volume of a seed kernel from 2-D images using the technique of volume sculpting. The relative standard deviation of the results produced by the proposed technique is lower (better) than the relative standard deviation of the results produced by volumetric estimation using the ellipsoid slicing technique.

The fifth contribution is the development of image processing algorithms to provide feature enhancements to mobile applications to improve upon on-site phenotyping capabilities. Algorithms for two features of high value namely, leaf angle estimation and fractional plant cover estimation are developed. The leaf angle estimation feature estimates the angle between stem and leaf for images captured using mobile phone cameras whereas fractional plant cover is to determine companion plants i.e., plants that are able to co-exist and mutually benefit.

The proposed techniques, frameworks and findings lay a solid foundation for future Computer Vision and Scientific Computing research in the domain of agriculture. The contributions are significant since the dissertation not only proposes techniques, but also develops low-cost end-to-end frameworks to leverage the proposed techniques in a scalable fashion.

Artificial intelligence and image processing applications for high-throughput phenotyping

by

Venkata Siva Kumar Margapuri

B.S., Jawaharlal Nehru Technological University, 2014

M.S., Bradley University, 2015

A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science

Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2022

Approved by:

Major Professor

Dr. Mitchell L. Neilsen

Copyright

Abstract

The areas of Computer Vision and Scientific Computing have witnessed rapid growth in the last decade with the fields of industrial robotics, automotive and healthcare acting as the primary vehicles for research and advancement. However, their research in other fields such as agriculture with abundant potential remains an understudied problem. The dissertation explores the potential of Computer Vision and Scientific Computing in an agricultural domain known as High-throughput Phenotyping (HTP). HTP is the assessment of complex seed traits such as growth, development, tolerance, resistance, ecology, yield, and the measurement of parameters that form more complex traits.

The dissertation makes the following contributions: The first contribution is the development of algorithms to estimate morphometric traits such as length, width, area and seed kernel count using 3-D graphics and static image processing, and extension of the existing Watershed technique for the same.

The second contribution is the development of light weight frameworks to aid in synthetic image dataset creation and image cropping for deep neural networks in HTP. Deep neural networks require a plethora of training data to yield results of the highest quality. However, no such training datasets are readily available for HTP research, especially on seed kernels. The proposed synthetic image generation framework helps generate a profusion of training data at will to train neural networks from a meager sample of seed kernels. Besides requiring a plethora of data, deep neural networks require the input to be a certain size. However, not all available data are in the size required by the deep

neural networks. The proposed image cropper helps to resize images without resulting in any distortion, thereby, making image data fit for consumption.

The third contribution is the design and analysis of supervised and self-supervised neural network architectures trained on synthetic images to perform the tasks of seed kernel classification, counting and morphometry. In the area of supervised image classification, the state-of-the-art neural network models of VGG-16, VGG-19 and ResNet-101 are investigated. A Simple framework for Contrastive Learning of visual Representations (SimCLR) [133], Momentum Contrast (MoCo) [51] and Bootstrap Your Own Latent (BYOL) [119] are leveraged for self-supervised image classification. The instance-based segmentation deep neural network models of Mask R-CNN and YOLO are utilized to perform the tasks of seed kernel classification, segmentation and counting. The results demonstrate the feasibility of deep neural networks for their respective tasks of classification and instance segmentation. In addition to estimating seed kernel count from static images, algorithms that aid in seed kernel counting from videos are proposed and analyzed. Proposed is an algorithm that creates a slit image which can be analyzed to estimate seed count. Upon the creation of the slit image, the video is no longer required to estimate seed count, thereby, significantly lowering the computational resources required for the estimation.

The fourth contribution is the development of an end-to-end, automated image capture system for single seed kernel analysis. In addition to estimating length and width from 2-D images, the proposed system estimates the volume of a seed kernel from 2-D images using the technique of volume sculpting. The relative standard deviation of the results produced by the proposed technique is lower (better) than the relative standard deviation of the results produced by volumetric estimation using the ellipsoid slicing technique.

The fifth contribution is the development of image processing algorithms to provide feature enhancements to mobile applications to improve upon on-site phenotyping capabilities. Algorithms for two features of high value namely, leaf angle estimation and fractional plant cover estimation are developed. The leaf angle estimation feature estimates the angle between stem and leaf for images captured using mobile phone cameras whereas fractional plant cover is to determine companion plants i.e., plants that are able to co-exist and mutually benefit.

The proposed techniques, frameworks and findings lay a solid foundation for future Computer Vision and Scientific Computing research in the domain of agriculture. The contributions are significant since the dissertation not only proposes techniques, but also develops low-cost end-to-end frameworks to leverage the proposed techniques in a scalable fashion.

Table of Contents

List of Figures

List of Tables

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. IOS 1543958 and the U.S. Department of Agriculture under Cooperative Agreement 58-3020-1-013. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

I would like to express gratitude of the highest order to my advisor Dr. Mitchell Neilsen for his unwavering support on every front throughout my doctoral program. His passion for research and learning is the impetus that has propelled my dissertation. None of this would be possible without his support.

I would also like to thank Dr. Torben Amtoft, Dr. Doina Caragea, Dr. Sherry Fleming, and Dr. Naiqian Zhang for serving on my doctoral committee and providing me with valuable input and motivation through my doctoral journey.

My sincere appreciation to the fellow students, faculty, and staff in the department of Computer Science at Kansas State University. I would like to express my gratitude to my research collaborators Mr. Anthony Atkinson, Dr. Paul Armstrong, Dr. Daniel Brabec, Dr. Chaney Courtney, Dr. John Hatcliff, Mr. Friday James, Dr. Ratan Lal, Mr. QA Mendoza, Ms. Niketa Penumajji, Dr. Pavithra Prabhakar, Dr. Trevor Rife, Mr. Robert Stewart, Dr. Brandon Schlautman, Mr. Dan Wagner, Ms. Shanshan Wu, and Mr. Kai Zhao.

I would like to express my gratitude to my parents and immediate family members from the Koripella family, the Chinnam family, the Chirumamilla family, and my friend

Chaitanya Parupalli for their unconditional love, support, patience, and trust during my doctoral endeavor.

Dedication

I dedicate my dissertation to my late grandfather Mr. Rama Rao Koripella who inspired and motivated me to pursue higher learning and research.

Chapter 1 - Introduction

The origins of Artificial Intelligence (AI) date back to the 1950s when Alan Turing, a British mathematician envisioned a world in which computers could think and make decisions based on knowledge acquired. However, the bottleneck at the time was that computers could only execute instructions, but not store them. In other words, the concept of memory for computing systems wasn't prevalent at the time. Over time, the concept of memory in computers has become so prevalent that it is taken for granted. With the explosive growth of memory and the computational ability of computers, modern day machines are put to the task of making educated decisions and predictions based on historical knowledge. The ability of machines to think and act like humans is termed 'Artificial Intelligence'. The applications of AI prevalent in the modern world are speech recognition, language translation and visual perception. The field of AI that deals with visual data is called 'Computer Vision'. Computer Vision describes the computer's ability to process, understand and infer from visual data similar to processing by humans gathering images from their eyes. The fact that machines are able to make sense of visual data means that the data in the form of images and videos can be analyzed. With the abundant availability of mobile cameras, a profusion of visual data is available in different domains across the board. As a result, computer vision has gained popularity amongst researchers in different fields such as industrial automation, healthcare, banking and automotive. The aspects of image processing and convolutional neural networks are great means to apply the idea of computer vision to images and videos to each of the fields mentioned. While computer vision research in the aforementioned fields has been prevalent, computer vision in the field of agriculture offers plenty of avenues for exploration. This dissertation explores the idea of using Computer Vision for High-Throughput Phenotyping (HTP) in agriculture. The general paradigm of agriculture has its foundation in the 'seed'. The analysis of the seed kernel, both genotype and phenotype, provides plant researchers with invaluable information in terms of seed behavior in different environmental conditions. The dissertation specifically focuses on the analysis of phenotypes, and not genotypes. The study of the morphological traits of a seed kernel provides information about its growth, development, yield, and resistance. The morphological traits of a seed kernel such as the length, width, area, volume, and density aid plant scientists in determining and evaluating the behavior of a seed in an ecosystem. The seed kernel length and width are manually measured using a caliper. However, the measurement is subjective to trial and individual on duty. To arrive at estimates that are agnostic of the individual conducting the experiment, image analysis techniques to estimate seed kernel length, width and area are proposed. The proposed image analysis techniques consider three important problems of image processing namely, image skewness, object clusters on image and unclear image background. The proposed techniques leverage low-cost 3-D printed physical components and backlit lightbox to counter the problems.

While the proposed image processing techniques work well, available are state-of-the-art deep neural network models meant for classification and instance segmentation. The use of deep neural network models in the field of HTP is still in a nascent state. Several pre-trained models trained on COCO and ImageNet datasets are available for use. Transfer learning comes into play when pre-trained models are employed on a custom dataset. The training of deep neural network models requires a large volume of training data to yield good results. However, the availability of seed kernel image datasets to train on the neural networks is scarce. An alternative to real seed kernel image datasets is the use of synthetic image datasets. We propose a synthetic image generation framework that applies the principles of Domain Randomization to generate synthetic image datasets that can be used as training data for deep neural networks. The synthetic image generation framework generates synthetic images from a small sample of real seed kernel images. The framework augments each of the real seed kernels to generate the synthetic images. The user is offered flexibility in terms of the augmentations that are applied. The rotation, brightness and shear are the properties that are augmented by default. In addition to a large volume of data, another key requirement of deep neural networks is that the input be of a specific size. For instance, the neural network model of Oxford VGG-16 requires that the input image be of size 224 x 224 px. However, the training data at hand might not always be of that size. While resizing is an option, it usually leads to image distortion leading to loss of image quality. To ensure that the resizing results in an image without distortion, an image cropping tool using OpenCV-Python is developed. The tool helps to resize an image to a size of the user's choice placing the object of interest at the center of the resized image. Such images lend themselves well to training on deep neural networks.

Synthetic image generation is used to create synthetic image datasets on five types of seed kernels namely, canola, sorghum, soy, rough rice and wheat. The synthetic images are used to train deep neural network models meant for image classification. Within the realm of classification, supervised and self-supervised deep neural network architectures are trained and evaluated for their performance. In the space of supervised neural network models, the pre-trained architectures of Oxford's VGG-16, VGG-19 and Microsoft's ResNet-101 are used to build custom models trained on the synthetic image datasets. In the space of self-supervised neural network models, the architectures of SimCLR, Momentum Contrast (MoCo) and Bootstrap Your Own Latent (BYOL) are used to train on the synthetic image datasets. The testing of the supervised and self-supervised models is performed on real seed kernel images but not synthetic images. In addition to classification models, state-of-the-art instance segmentation models are available that use the classification models such as ResNet-50 as backbones. Not only do the instance segmentation neural networks aid in classification, but they are also able to localize each of the objects within an image, unlike the classification models. Localization is done by plotting a bounding box around each of the objects detected within the image. As a result, an analysis over the localized image generated by the instance segmentation neural networks helps determine the object's morphometry and count in the image. The instance segmentation models of Mask R-CNN and YOLO are used to perform instance segmentation on synthetic image datasets generated for each of the seed kernels. Once the seed kernels are segmented by the neural networks and bounding boxes are plotted, the images are analyzed using OpenCV to estimate the morphometry of each of the instances detected. Methods such as using the coordinates of the bounding box, plotting minimum area enclosing rectangle and ellipses aid in the estimation of seed kernel morphometry.

Now that the seed kernel morphometry is computed, the count of seed kernels in the image is estimated as the number of bounding boxes plotted by the instance segmentation neural networks. The count depends heavily on the efficiency of the neural networks in the sense that any missed seed kernels result in an incorrect count. While the estimation of seed morphometry is directly tied to HTP, the count has applications in the seed packaging industry. Conventionally, seed packaging is done based on the weight of a sample of seeds but not on the count of seeds within the seed sample. Since the weight of a random seed sample is influenced by factors such as moisture in the air, weight of the seed sample changes over time. Customers generally purchase seed samples expecting a certain number of seed kernels to be present in the package. Since the packages are made by weight but not seed kernel count, customers don't end up with the exact number of seeds that they expect. Needless to mention the impact of environmental factors on the weight of the seed sample. To better meet the expectations of the customers, techniques to count the seed sample for packaging are developed. Seed counter machines are available in the market for prices ranging from $300 to $2000. However, not all packaging firms are able to afford the machines at such hefty prices. The availability of deep neural networks and image processing techniques aid in the formulation of low-cost seed counting techniques that aid the packaging industry. Proposed is a seed counting framework that estimates seed kernel count from a video captured using mobile devices. The video capture system is a low-cost setup that includes a 3d printed holder that holds a sled to allow seed kernels to roll down as they're captured by the mobile device. The dissertation proposes two techniques to estimate seed count. The first technique is completely based on image processing and analysis using OpenCV whereas the second technique uses an instance segmentation deep neural network and modified Kalman filter.

The estimation of seed morphometry techniques defined thus far estimate different parameters such as length, width, and area but one key parameter that is not estimated is volume. The reason is that volume computation requires much more than a single 2-D image. As a result, a single 2-D image is not sufficient to estimate volume. Conventional techniques for volume measurement include water displacement methods in which the seed is submerged in container with water. The amount of water displaced from the container indicates the volume. The technique is prone to human error while also being tedious and time consuming. Typically, seed kernels absorb moisture when placed in ambient air. Likewise, seed kernels absorb water when dipped in water resulting in a change in seed volume and damage to the seed. As a result, the volume of the seed kernel is altered leading to discrepancy in the volume measurement. Another similar technique is bead displacement in which volume is measured by the displacement of beads placed in a container. The technique is error prone and tends to over-estimate the volume of the seed kernel. To counter the error in estimates, morphometry estimation techniques using image analysis are developed. The new technique builds on work by Cao and Neilsen [24], and estimates volume from a set of images captured at different angles. An automated image capture system that consists of a camera, stepper motor, LEDs and turntable is proposed to capture the images of the seed kernel from different angles. The technique of volume carving for 3-D reconstruction is used to estimate the volume of the seed kernel. While volume carving is an existing technique, an end-to-end automated framework that uses volume carving for 3-D reconstruction is not available. High accuracy in volume estimation is achieved using the framework.

Data collection is a key aspect that helps advance emerging fields of research such as high-throughput phenotyping in agriculture. The availability of data helps scientists develop systems that are able to extract and analyze information from data to make defining discoveries. Techniques of the yesteryears required researchers to have a laboratory for research. However, the onset of mobile development shifted the paradigm toward on-site research wherein phenotyping applications run on modest mobile devices such as mobile phones and tablets. One such application in the field of high-throughput phenotyping is Fieldbook, an Android application developed by the Poland Lab at Kansas State University, KS. The application is geared towards plant breeders and researchers who aim to collect, record and process information to better conduct field experiments. The application has the potential to cater to the needs of the larger research community in the domain of agriculture. To enhance the feature stack of the existing mobile applications, algorithms for two new features are proposed and developed. The first feature is leaf angle estimation wherein the angle between the stem and leaf of a given plant image is estimated using a combination of neural networks and image processing techniques. Plant studies [36] have shown that plants with upright leaf angles contribute to better hybrid plant varieties. However, the determination of leaf angle by manual means is tedious and subject to bias. Image processing tools such as ImageJ that consist of an angle estimation tool are able to estimate the leaf angle from images. While tools such as ImageJ provide a sense of reproducibility of results across different individuals, the problem of tedium still remains. The proposed algorithm for automated leaf angle estimation is perhaps one of the very few fully vetted algorithms in the realm. The second feature is percentage plant cover estimation on the field for the use case of companion planting. Companion planting refers the idea of growing multiple species of plants in close proximity so that they reap mutual benefits, such as improved crop yield, soil quality and pest control. The development of the algorithm is performed in conjunction with researchers from the Land Institute in Salina, KS whose goal is to grow different grasses together in close proximity to make discoveries pertinent to grasses that make good companions. The known techniques for plant cover estimation such as Daubenmire technique are ocular-based and therefore, subject to bias. The proposed algorithm is able to estimate the amount of plant cover in metric units along with percentage measure. The literature survey conducted as part of the research shows no other application that estimates the amount of plant cover in metric units.

1.1 Contribution and Overview

In summary, this dissertation focuses on pushing the boundaries of Computer Vision in the field of agriculture, primarily High-Throughput Phenotyping. The overview of each of the chapters of the dissertation is as follows:

1. Chapter 2 proposes a low-cost, high-throughput technique for the estimation of seed kernel morphometry using 3-D printed components that is applicable to a wide variety of seeds. Elaborate analysis on the factors that impact the performance of the proposed algorithm is performed and results are presented.
2. Chapter 3 investigates the use case of seed kernel morphometry estimation on different classes of neural networks such as supervised, self-supervised and instance segmentation models. The technique of domain randomization is applied to generate synthetic image datasets to train the neural networks. The results obtained demonstrate the feasibility of domain randomization for high-throughput phenotyping.
3. Chapter 4 proposes a near real-time seed kernel counting algorithm that aids in seed packaging. The idea is to track a group of seeds as they flow down a platform in a video and estimate the count of seeds. The technique of slit imaging is used to capture and count the seeds as they flow down the platform.

4. Chapter 5 describes the current iteration of the low-throughput seed kernel volume estimation framework. The framework in the current iteration consists of a user interface to capture input and the image capture is completely automated based on input received from the user.

5. Chapter 6 proposes two algorithms to improve upon the feature stack of the mobile applications for high-throughput phenotyping. The first algorithm performs the task of estimating angles between leaf and stem from images. The second algorithm is to estimate the amount of plant cover within multiple bounded regions wherein the use case is to identify plants that are able to develop and co-exist in a given environment. Both algorithms are at the stage where they may be implemented as mobile and desktop applications.

6. Chapter 7 sheds light on the image processing tools developed as part of each of the chapters of the dissertation. The tools are handy in terms of image segmentation and cropping.

Chapter 2 - Image Processing and 3-D Components for Seed Morphometry Estimation

Image processing is defined as the analysis and augmentation of images to draw meaningful insights that lead to novel discovery. In other words, image processing helps to discover patterns and aspects in images that are otherwise not apparent. The applications of image processing are prevalent in industrial robotics, automotive and healthcare industries. The learnings from image processing applied in other industries serves as the basis for applications in the field of agriculture. Computer Vision relies on the inputs provided in terms of videos and images to make inferences and image processing complements computer vision by providing inputs that are feasible to infer upon. In addition to image processing, another field that has revolutionized the world in the last couple of decades is 3-D graphics. The 3-D models can be used to prototype complex models and conduct inexpensive experiments. For instance, in the field of construction and architecture, prototypes of large structures, such as buildings and dams, are constructed to study and evaluate the strength and aesthetics of the structures. Image processing coupled with 3-D graphics opens many unexplored opportunities in the domain of agriculture. This chapter explores the potential of the combination of Image Processing and 3-D graphics for High-throughput Phenotyping (HTP) of seeds. HTP of seeds, also known as Seed Phenotyping, is the comprehensive assessment of complex seed traits such as growth, development, tolerance, resistance, ecology, yield, and the measurement of parameters that form more complex traits [78]. HTP increases the accuracy of measurements while reducing costs through the application of automation, remote sensing, data integration, and experimental design. Multiple studies [116][105][184][68] emphasize the importance of seed morphometric estimations in predicting the behavior of seeds in different environmental settings. The morphometric estimation of seed length, width, area and volume may be performed using tools such as Vernier caliper. However, such a procedure is time consuming and tedious resulting in low-throughput. A high-throughput and low-cost solution to estimate seed morphometry is proposed using 2-D imagery and 3-D graphics. The goal of the work is to address some of the key problems in image processing namely, object skew on images, object clusters on images, poor image quality due to improperly lit backgrounds.

The chapter proposes algorithms for seed morphometry estimation with the goal to address use cases such as seed kernel counting, length and width estimation, and area and perimeter estimation using OpenCV for static image processing and compares against the existing Android applications in the realm of seed morphometry estimation.

2.1 Related Work

The foundation for the algorithmic development using OpenCV is the Watershed algorithm, as discussed by F. Meyer and S. Beucher in the work, ‘The Morphological Approach to Segmentation: The Watershed Transformation’ [15]. The work discusses in detail the intricacies of the algorithm, including the tools, transformations, uses, and the application of watershed algorithm to images.

Tanabata et al. proposed SmartGrain [164], a high-throughput phenotyping software tool to measure seed shape through image analysis. The tool uses a technique where outlines of seed kernels are automatically recognized from digital images and several seed shape parameters such as length, width, area, and perimeter length are calculated. The software is validated on rice seed by employing quantitative trait locus (QTL) analysis. SmartGrain removes areas of awns and pedicels automatically from images as it processed them.

What et al. proposed GrainScan [182], a low-cost, fast method for grain size and color measurements of seed kernels. The software tool identifies each seed kernel in an image independently and assigns them a unique color. In case of varieties such as wheat that have a crease at the bottom, the software is also able to identify the crease precisely and draw a line along it.

Schrader et al. proposed LeafIT [150], an android application for the measurement of leaf area. Leaf traits are some of the most important traits because they describe key dimensions of a plant's life history strategy. Furthermore, leaf area correlates with leaf chemical composition, photosynthetic rate, leaf longevity, and carbon investment. The application's precision and accuracy are compared against commercial software tools such as WinFOLIA [185] which uses the Altman-Bland method wherein the results showed that results are similar to a high degree.

Komyshev, et al. [71] proposed SeedCounter, a mobile application for grain phenotyping. The app analyses grain morphometry in cereals which is an important step in selecting new high-yielding plants. It was developed under the premise that manual assessment of parameters such as the number of grains per ear and grain size is laborious. The application was evaluated under six different light conditions and three mobile devices wherein the application demonstrated that the lighting has an impact on the outcomes.

2.2 Mesh Algorithm

The Mesh algorithm refers to the static image processing algorithm and framework including the 3D stand, 3D mesh, and lightbox, as shown in Figure 2.1. Each of the components has a specific use case. The algorithm solves multiple problems in the following ways:

1. **3D Stand:** The 3D stand solves the problem of skew that most image processing algorithms encounter. The stand ensures that the image capture device such as a phone is always parallel to the surface and directly above the seeds to be captured. This ensures the image is not skewed. The stand used for the experiment is 3d printed using a white filament with a height of 110 mm (11 cm).
2. **3D Mesh:** The 3D mesh addresses the problem of the object touching on images. The hexagonal boundaries act as barriers ensuring that the seeds do not touch each other. The mesh, printed using a white filament contains hexagons of side 5 mm. Meshes for different sized seeds can be easily produced.
3. **Lightbox:** The lightbox ensures that the images which are captured have a bright background. A bright background makes it easier to identify the objects on the image and eliminate any noise such as tiny dirt particles that may be present on the image.

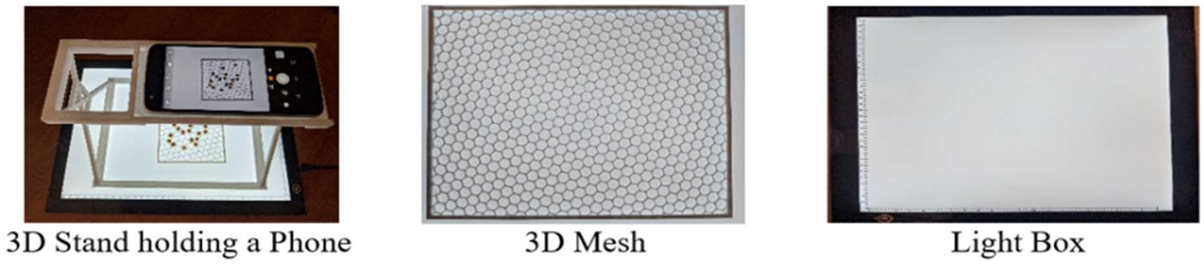


Figure 2.1: Components for Mesh Algorithm

The algorithm can be generalized as a two-step process, described as follows:

1. Detect the mesh on the image and estimate the morphometry. The hexagons within the mesh are saved as ground truth values.
2. Detect the seeds on the image and use the estimated morphometry of the mesh to infer the size of the seeds.

The key concepts that aid in the development of algorithms are RGB color space and HSV color space. Broadly, any color space is a mathematical model that describes colors as a list of numbers. RGB is one of the most common color spaces used in image processing [103].

RGB Color Space: RGB stands for Red, Green, Blue color space. The color space is defined by the numerical values assigned to the colors of red, green, and blue. For a 24-bit image, the values assigned for each color are stored using 8-bits for a range between 0 and 255. In the context of the RGB color space, the colors of red, green, and blue are referred to as primary colors, and every other color is referred to as non-primary. All non-primary colors generated in the RGB color space are a result of the combination of the primary colors in different intensities. Hence, the color space may be classified as an additive color space and is graphically described as shown in Figure 2.2.

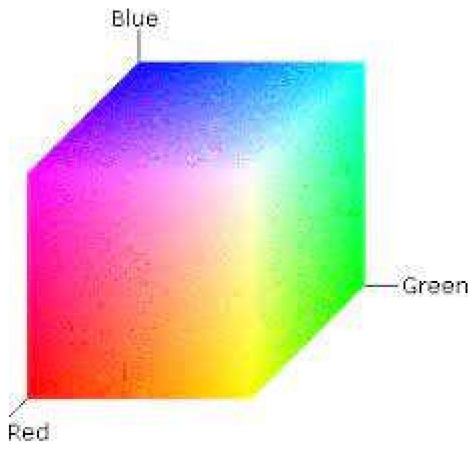


Figure 2.2: RGB Color Space [103]

HSV Color Space: HSV refers to Hue, Saturation and Value which make up the coordinates for the color space as shown in Figure 2.3. It is a cylindrical color space where the radius represents Saturation, the vertical axis represents Value, and the angle represents Hue. Intuitively, Hue

is the dominant color visible to an observer, Saturation is the amount of white light mixed with a hue and Value is the chromic notion of intensity. As Value decreases, the color gets

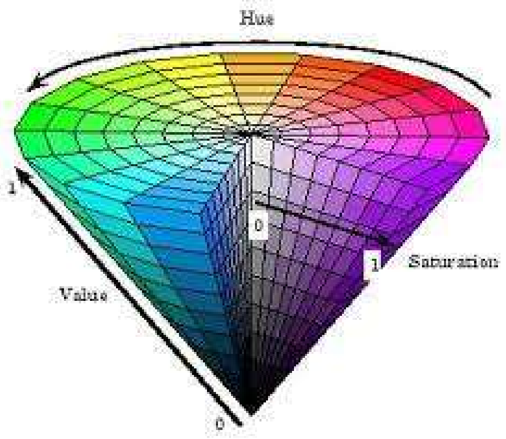


Figure 2.3: HSV Color Space [37]

closer to black whereas the intensity of the color increases as Value increases[A1]. Each of Hue, Saturation, and Value have their own significance in the processing of images. Hue helps to classify the colors whereas Value determines the intensity of the color. Saturation is the property that determines the amount of white light mixed with hue.

2.2.1 Mesh Detection and Estimation using Pixel Color

A key point worth reiterating here is that the filament used to build the mesh and the light emitted by the lightbox are white. From Figure 2.4, the mesh is merely a shade darker than the light emitted by the lightbox.

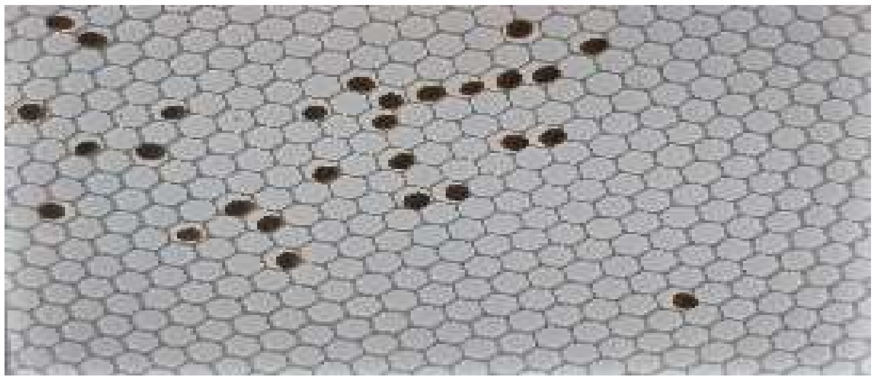


Figure 2.4: Soy Seeds in a 3-D Grid

1. For reproducibility, the mesh should have pixel values between 170 and 255 for each color channel.

2. Perform a median blur and apply Canny edge detection to detect the edges on the image.

3. Detect the contours of the edges identified by Canny edge detection.

4. (Observation/ Assumption) Find all contours which occupy an area greater than an individual hexagon of the mesh in pixels.

5. Compute the median area occupied by the mesh in pixels and consider it as the area occupied by each hexagon in the mesh. Likewise, compute the median perimeter of the mesh.

A relationship between the number of pixels to the area in metric units is now established. The area of a hexagon in mm^2 is given by $\frac{3\sqrt{3}}{2}\ a^2$ where a is the length of a side of the hexagon in mm, and the perimeter is given by $6 * a$ .where a is the side of the hexagon in mm. Figure 2.5 shows the different steps involved in the process of mesh detection.

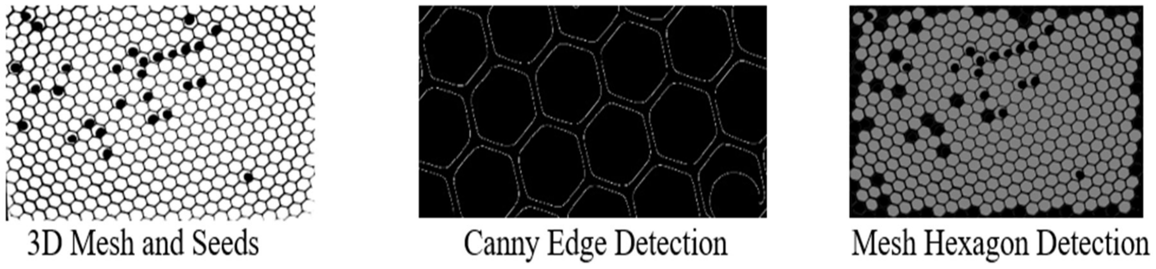


Figure 2.5: Mesh Detection using Pixel Color

2.2.2 Mesh Identification and Estimation using HSV Color Thresholding

1. Convert the source image from RGB to HSV color space.
2. Create a mask which identifies H, S and V values of a desired range[A2]. The values of H, S, and V need to be programmed since each of H, S, and V has its own significance in the context of image processing. Augmenting each of H, S and V includes or eliminates part of the image.
3. Perform a bitwise-AND operation on the mask and the source image.
4. Apply a Median blur followed by Canny edge detection.

Once the Canny edges of the image are identified, the process to estimate the mesh is the same as that described in steps 4 and 5 of section 2.2.2. Figure 2.6 shows the detection of the mesh from the input image.

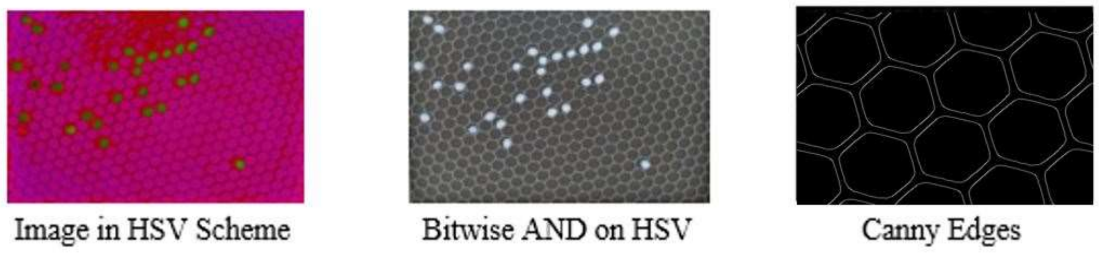


Figure 2.6: Mesh Detection using HSV Thresholding

2.2.3 Seed Detection and Estimation

1. Apply binary thresholding on the source image to filter out all of the white pixels and retain only the black pixels, i.e., pixels that represent seeds on the image.

2. Median blur the image and perform Canny edge detection on the image.

3. Detect seeds on the image and compute the area occupied by each of the seed contours.

4. (**Seed Count**) The number of contours deemed seeds is the estimated number of seeds on the image[A3].

5. (**Area and Perimeter Estimation**) The area of seeds in metric units is estimated by taking the median area occupied by the mesh's hexagons in pixels and metric units as ground truth values, and scaling, i.e., $seed\ area\ in\ mm^2 = (seed\ area\ in\ pixels\ *\ hexagon\ area\ in\ mm)/(median\ hexagon\ area\ in\ pixels)$. Likewise, a simple scaling is applied to obtain the average perimeter of the seeds in mm.

6. (**Length and Width Estimation**) The lengths and widths of each contour are estimated using minimum area rotated rectangles. Rectangles are fit to the contour samples. The longest axis returned is considered the length, while the shorter axis is the width. Besides, connecting the left-most to the right-most and top-most to the bottom-most points, and computing the length of the connecting lines is a means to estimate the length and width of the seeds[A4]. The orientation of the seed kernels has an impact on the estimation of length and width of the seed kernels. The orientation of the seed kernels can be derived as the minimum area rotated rectangles are plotted using the OpenCV function from the orientation of the sides of the rectangle. The knowledge of seed kernel orientation helps users identify the differences of length and width of seed kernels at different orientations.

Figure 2.7 shows the seed counts, estimated seed areas in mm^2, and length-width estimations in mm for a soy seed sample.

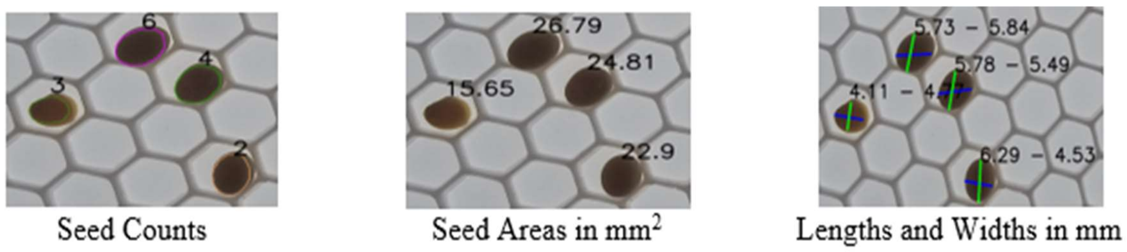


Figure 2.7: Seed Morphometry (Image Zoomed In)

As mentioned, the experiment is performed on six different types of seeds. The algorithm performs well consistently on five of the six types of seeds in question failing on white rice. The reason is that the light emitted by the lightbox is the same color as the seed of white rice. As a result, the algorithm fails to distinguish between the mesh and seeds. While the proposed algorithm still holds, a different experimental setup involving a contrasting background is essential for white rice. The app in Python-OpenCV is available at https://github.com/VenkatMargapuri/MeshAlgorithm.

2.2.4 Morphometry Estimations on Different Mesh Types

The algorithm is tested on three different geometries of meshes i.e., hexagon, triangle and rhombus with different infill rates and boundary thickness. Table 2.1 shows characteristics of the meshes used as part of the experiment[A5]. In table 2.1, the side refers to the side of each of the geometric shapes within the mesh whereas area refers to the area of the geometry with the given side. Infill rate indicates the fullness of the inside of a 3-D printed component. An infill rate of 0% indicates that the component is hollow whereas an infill rate of 100% indicates that the component is completely filled.

Table 2.1: Geometry and Characteristics of the Meshes

Mesh Geometry	**Side (mm) – Area(mm^2)**	**Infill Rates**	**Thickness**
Hexagon	4 – 41.57, 5 - 64.95,6.5 – 109.77	18%, 15%, 12%, 10%,8%	5 mm, 1.5 mm, 0.5 mm
Rhombus	7.7 – 60.5, 6.3 – 40.5, 10.6 – 112.5	15%, 12%, 10%	5 mm, 0.5 mm
Triangle	13 – 84.5, 10 – 50	10%, 12%	5 mm, 0.5 mm

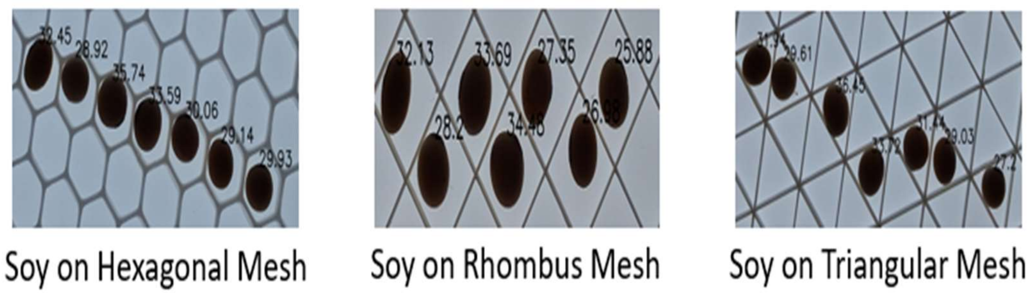


Figure 2.8: Seed Soy Sample Area Estimations

As the morphometry of the seeds is estimated on different mesh types, the observation is that the estimations are similar, although not exactly the same on all of the mesh types. The difference in the estimations is better explained in section 2.5. In order to demonstrate the performance of the algorithm, a sample of seven soy seeds is considered and estimated using different mesh types. The seeds laid on different meshes is shown in Figure 2.8 and the readings are as shown in Table 2.2.

Table 2.2: Seed Area (mm^2) Estimations for Seven Seed Sample of Soy

Mesh Geometry	**Seed 1**	**Seed 2**	**Seed 3**	**Seed 4**	**Seed 5**	**Seed 6**	**Seed 7**
Hexagonal	32.45	28.92	35.74	33.59	30.06	29.14	29.93
Rhombus	32.13	28.2	33.69	34.45	27.35	26.1	25.88
Triangular	31.91	29.61	36.45	33.72	31.44	29.03	27.2

2.3 No-Mesh Algorithm

The No-Mesh algorithm is a static image processing algorithm that does not use the 3D printed mesh, unlike the algorithm discussed in Section 2.2 Instead, the algorithm leverages the popular Watershed Algorithm to segment seed clusters on images. The other components, 3D stand, and the lightbox are optional, but recommended to achieve the proper performance of the algorithm. The algorithm is explained as a three-step process, described as follows:

1. Detect segmented seeds on the image i.e., seeds not in contact with other seeds on the image.
2. Apply the Watershed Algorithm on the image and segment the seeds to perform morphometric estimations.

3. Estimate the morphometry of seeds using a ground truth object.

2.3.1 Detection of Segmented Seeds

The detection of segmented seeds makes use of the interquartile range, convexity defects, and the convex hull. The **Interquartile Range (IQR)** is a measure of the middle 50% of the values in a dataset. It is a proven mathematical technique that aids in the detection of outliers. In Figure 2.9, the IQR is $Q3 - Q1$. Q1 represents a quarter of the way through the list, and Q3 represents three-quarters of the way through the list. The IQR rule is that all values that are not between $Q1 - 1.5\ x\ IQR$ and $Q3 + 1.5\ x\ IQR$ are outliers.

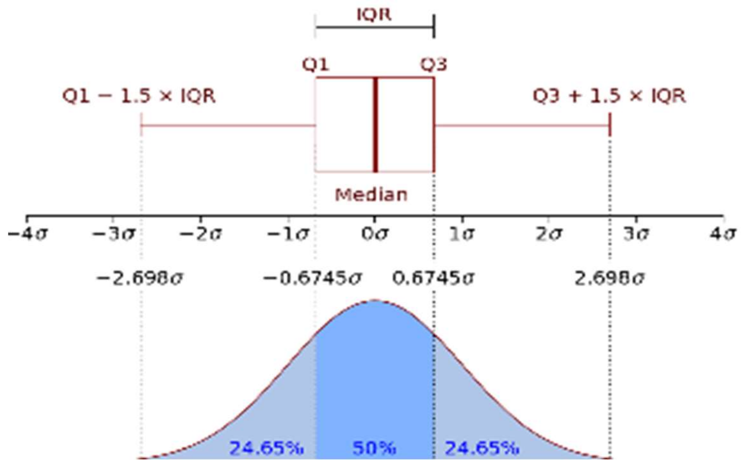


Figure 2.9: Interquartile Range Calculation [153]

Convex hull and convexity defects: A convex object is one where none of the interior angles is greater than 180 degrees. The convex hull is a tight-fitting boundary around the object. As shown in Figure 2.10, the image on the left is convex. It has a convex hull that wraps entirely around the boundary, whereas the image on the right is not convex. The convex hull does not wrap around correctly. The imperfections in the convex hull are known as convexity defects.

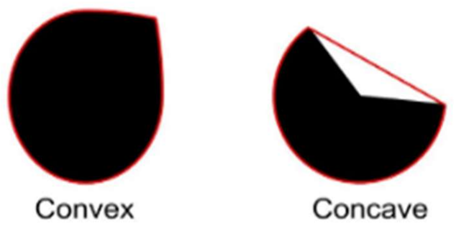


Figure 2.10: Convexity Defects [183]

1. Convert the image to grayscale and gaussian blur the image using a $3\ x\ 3$ kernel.
2. To threshold the seeds from the background, apply inverted binary Otsu thresholding on the image.
3. Detect the contours of the seeds along with the area occupied by each of the contours. Also, find the convex hull of each of the seed contours.
4. Detect a segmented seed by evaluating the convexity of a contour, i.e., $if\ length(contour)/\ length(convexhull)\ <=\ 3$, the contour is identified as a potential segmented collection of seeds.
5. Find the area occupied by each of the contours of the potential segmented seeds.
6. Remove the outliers from the potential segmented seeds by applying the IQR rule for outliers, stated above.
7. **(Case for Touching Seeds)** An edge case is that a few false positives might satisfy the step-four check. To eliminate them, filter any contours where the

center is relatively white, i.e., with an intensity of about 170 or higher.

After segmentation, compute the morphometries, including the farthest point from the convex hull to the seed. Results are utilized further in section 2.3.2. The steps in the process of segmented seed detection are as shown in Figure 2.11.

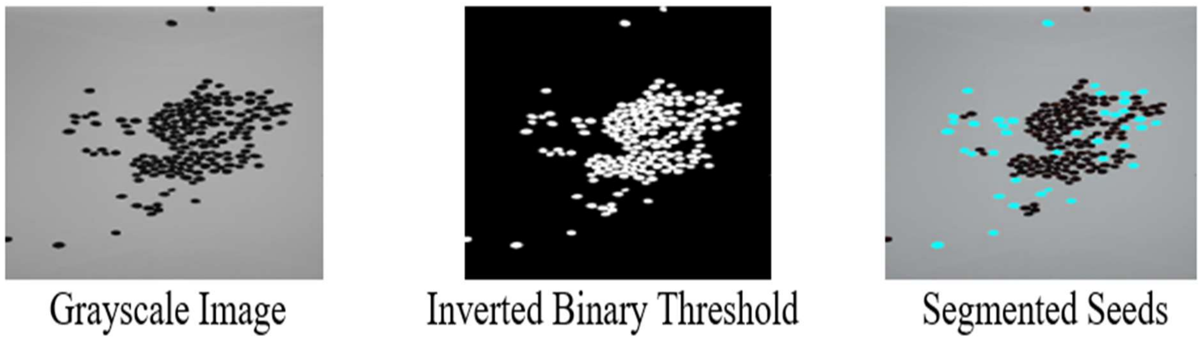


Figure 2.11: Detection of Segmented Seeds

2.3.2 Application of Watershed Algorithm

The Watershed algorithm's application relies on the concepts of morphological operations, erosion, dilation, closing, and opening, shown in Figure 2.12.

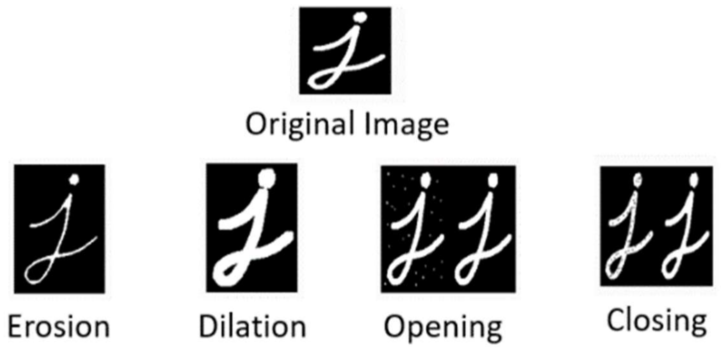


Figure 2.12: Morphological Operations

The principle behind morphological operations is the idea that 'on a grayscale image, black (0) and white (255) pixels can be identified with good precision'.

Erosion: Erosion is a means to pare the image. A kernel of any size is convolved across the image. If one of the pixels along the kernel is black, all pixels on the image with respect to the kernel, are set to black (0).

Dilation: Dilation is a means to enhance the size of the image. A kernel is convolved across the image. If one of the pixels on the kernel is white, all pixels on the kernel are set to white (255).

Opening: Opening is a means to eliminate noise from an image. It is an erosion followed by a dilation.

Closing: Closing is a means to fill out the gaps in the image. It is a dilation followed by an erosion.

2.3.2.1 Pre-processing and Application of Watershed Algorithm[A6]

Watershed algorithm is a popular image segmentation algorithm that has the ability to segment clustered objects in an image. In use cases such as this where counting objects in the image is one of the goals, object clustering leads to erroneous counts. Employing Watershed algorithm leverages different transforms of the original image for segmentation. The preliminary steps applied to the original image are explained as follows.

1. Convert the image to grayscale and apply binary thresholding to the image.
2. Perform a morphological opening to remove noise and a morphological closing to remove any holes in the image.
3. **(Background Identification)** Sure background area on the image is identified by performing a dilation.

4. **(Foreground Identification)** Compute the distance transform on the image and apply a threshold on the distance transform. The threshold values depend on the image.

5. **(Unknown/ Border)** The regions which are neither foreground nor background are the unknown/ border areas. They are computed by subtracting the foreground area from the background area.

6. **(Label Assignment)** Assign labels to the identified foreground, background, and unknown areas. If using OpenCV, the cv.ConnectedComponents() function can be used for this purpose. It assigns 0 to the background and random numbers starting from 1 to the objects in the foreground. However, the Watershed algorithm assumes all regions with a value of 0 as unknown areas. To avoid this scenario, increment all values by one, so the background has a value of 1 and assign 0 to all pixels of the unknown area.

7. Apply a gaussian blur to the image to reduce the inner contour noise and avoid over-segmentation.

8. Apply the Watershed algorithm to the blurred image and capture the output markers.

The pre-processing steps applied for the Watershed algorithm is as shown in Figure

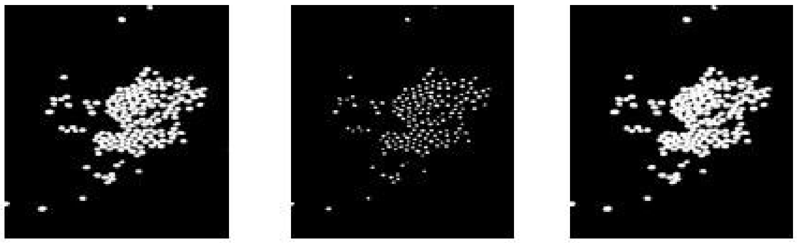


Figure 2.13: Preprocessing and Application of Watershed

2.13[A7].

2.3.2.2 Image Segmentation Post-Watershed Application

1. Draw the boundaries of the watershed segments on the markers produced by applying the Watershed algorithm (The Watershed algorithm denotes boundaries by -1).

2. Change the boundary markings from -1 to background, i.e., 0.

3. Invert the background and the foreground, i.e., set all 0's to 255's and the others to 0's, to identify boundaries in white since markers is a grayscale image.

4. Find contours on markers that also return the hierarchy of the contours.

Note: Contour hierarchy is an OpenCV concept. The hierarchy is a data structure that allows contours to access four different relative values, next contour, previous contour, first child, and parent contour.

5. Use the hierarchy to find all contours which are not considered noise. Identify child contours larger than a quarter of the average segmented seed contour area and mark them as 'parent'.

2.3.2.3 Counting Seeds[A8]

Seeds in images are often overlapping or clustered; therefore, counting the number of seeds on the image is a tricky task. The segmentation performed by the Watershed algorithm is used for counting. The algorithm, however, provides two types of counting techniques, namely, area-based count and contour-based count.

Area-based Count: The area-based counting technique uses the average segmented seed areas to estimate the number of seeds in a given contour containing a cluster of seeds. For example, if the average pixel area of segmented seed contours is 10000 and the contour with a cluster of seeds has an area of 50000, the area based counting technique estimates the number of seeds in the cluster to be five.

Contour-based Count: The contour-based counting technique uses the results computed in section 2.3.1, i.e., the mean (SS_Mean), median (SS_Median), and standard deviation (SS_SD) of the segmented seeds. The count also makes use of the contour area (PC_CA), and the fixed-point depths (PC_FPD) of the contours identified as 'parent' contours in section 3.2.2.2. The estimation of the seed count is defined as the following:

$$If\ Mean\ (PC\_FPD)\ >\ SS\_Median\ +\ 3\ *\ SS\_SD:$$

$$count\ =\ rounded\ value\ of\ \left(\frac{PC_{CA}}{SS_{Mean}}\right)$$

$$Else: count\ =\ 1$$

The algorithm is used to count the number of seeds on images of different seeds. The results are shown below in Figure 2.14.

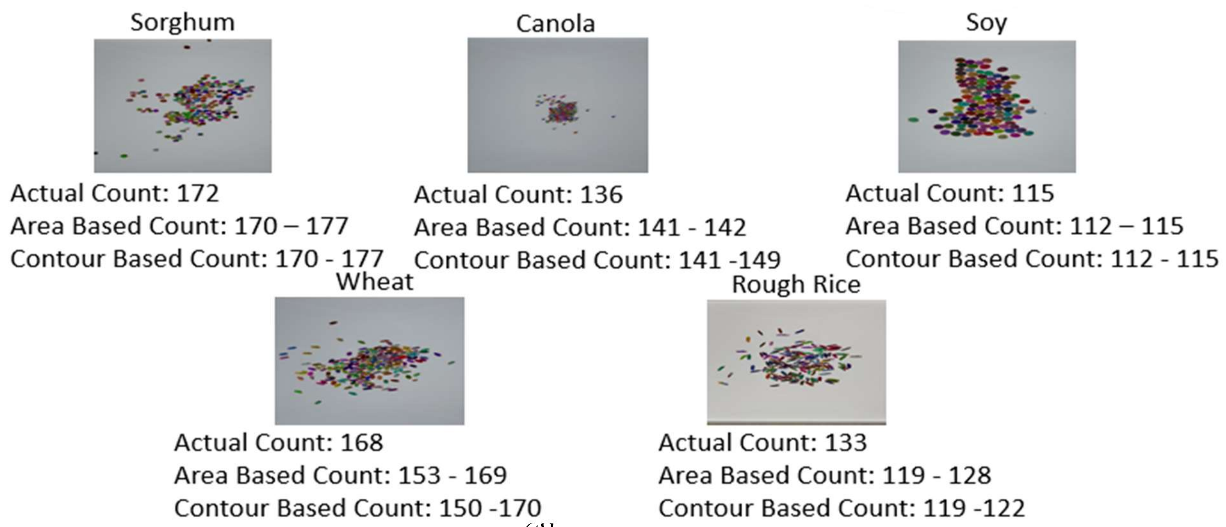

Sorghum
Actual Count: 172
Area Based Count: 170 – 177
Contour Based Count: 170 - 177
Canola
Actual Count: 136
Area Based Count: 141 - 142
Contour Based Count: 141 -149
Soy
Actual Count: 115
Area Based Count: 112 – 115
Contour Based Count: 112 - 115
Wheat
Actual Count: 168
Area Based Count: 153 - 169
Contour Based Count: 150 -170
Rough Rice
Actual Count: 133
Area Based Count: 119 - 128
Contour Based Count: 119 -122

Figure 2.14: Seed Count using No-Mesh Algorithm

As observed from the results, the algorithm performs well on the images of sorghum, soy, and wheat, where the range of estimated count is within the actual number of seeds on the image. However, on the seeds of canola and rough rice, the algorithm overcounts and undercounts, respectively. The results for white rice are not presented because the algorithm fails to detect the seeds precisely. Note that a change in the arrangement, be it the position or the orientation of seeds on the image, leads to the detection of different contours and might influence the seed count since it is entirely dependent upon the detected contours.

2.4.3 Seed Morphometry Estimation

Unlike the Mesh algorithm, the No-Mesh algorithm has no ground truth value. Hence, the idea is to capture the image with objects of known morphometry and base the estimates on the ground truth. For this experiment, the penny, a US coin, is used. A total of four pennies are placed in each of the four corners of the lightbox when the image is captured. The pennies on the image are identified based on the size of the contour, i.e., the four most massive contours found on the image are assumed to be those of coins and circularity computed as $4 * pi * (area/perimeter^2)$ where area and perimeter are pertinent to the contour in question. sample of seven soy seeds whose areas (mm^2) are

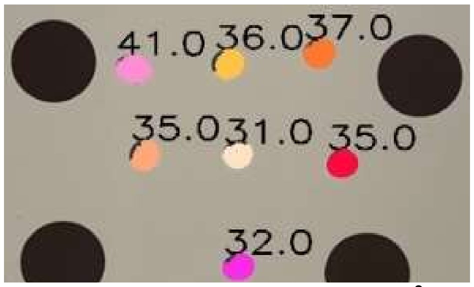


Figure 2.15: Estimated Seed Areas in mm^2 using Pennies

estimated using pennies is as shown in Figure 2.15.

2.5 Analysis of Outcome Impacting Parameters

Analysis on the impacts of parameters such as height of image capture, size of image and seed orientation is performed to recommend the best parameters for image capture. The observations are as follows.

2.5.1 Analysis of Outcome Impacting Parameters

In order to study the impact of height and image size, three random samples of seven seeds are taken. 3 different heights of 11cm, 17cm and 21cm are the choice for image capture. The size of the images is varied between dimensions of *3456 x 4608* px and *1920 x 1080* px. For this experiment, the heights of image capture are varied while image size is kept constant. The results observed for the three samples of seeds are similar. The results of one of the samples are shown in Tables 2.3 and 2.4 for reference.

Table 2.3: Seed Areas with Varying Image Heights and Image Size of 3456 x 4608 px

Image Height	**Seed 1**	**Seed 2**	**Seed 3**	**Seed 4**	**Seed 5**	**Seed 6**	**Seed 7**
11cm	32.99	32.49	29.48	34.53	28.07	28.43	27.71
17cm	33.04	32.84	29.71	32.18	27.98	28.16	27.97
21cm	33.13	33.03	29.85	31.98	27.79	28.07	28.27

From the readings in Table 2.3, it can be inferred that the impact of height is minimal when the size of the image is *3456 x 4608* pixels (px[A9]).

Table 2.4: Seed Areas with Varying Image Heights and Image Size of 1920 x 1080 px

Image Height	**Seed 1**	**Seed 2**	**Seed 3**	**Seed 4**	**Seed 5**	**Seed 6**	**Seed 7**
11cm	44.0	42.0	37.0	42.0	35.0	37.0	37.0
17cm	41.0	36.0	37.0	35.0	31.0	35.0	32.0
21cm	0	0	0	0	0	0	0

From the readings of the values in Table 2.4, it is inferred that the size of the image and the height of image capture have an inverse relationship[A10] i.e., the height of the image has minimal impact when the size of the image is large. Figure 2.16, which contains seeds and pennies on the image together help to reason the behavior. Green contours are supposed to be drawn only around seeds but when the height of image capture is 21 cm, the pennies also appear small enough that they are also considered seeds.

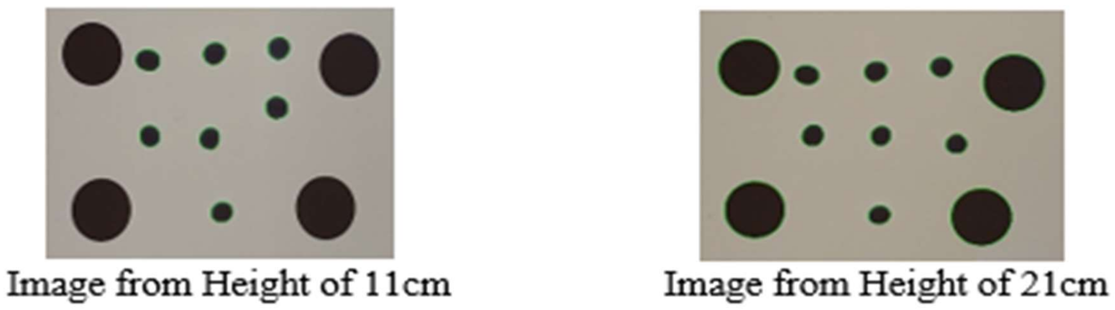


Figure 2.16: Impact of Height of Image Capture on Estimation

2.5.2 Analysis of Outcome Impacting Parameters

The impact of seed orientation is investigated by using a sample of seven seeds laid on a mesh. The hypothesis behind investigating orientation is that the algorithm gives out different results depending upon the orientation of the seeds since it affects the detected contours which is ultimately the basis for the estimations. Turns out that the hypothesis is true and is explained further from observing Figure 2.17 which shows a sample of seven seeds each laid in a different orientation.

Figure 2.17: Impact of Seed Orientation on Estimation

As observed from the results in Table 2.5, the lengths and the widths of the seeds differ depending upon the orientation of the seeds. It could also impact the area estimations if they were based on the length and width of the seeds.

Table 2.5: Lengths and Widths of Seeds in Different Orientations (Length - Width in mm)

Orientation	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5	Seed 6	Seed 7
#1	7.44 - 5.96	6.85-6.17	6.38-6.15	6.62-6.16	6.06-5.9	6.14-5.87	6.36-5.68
#2	7.19-6.15	6.76-6.1	6.39-6.18	6.56-6.23	6.04-5.85	6.11-5.85	6.56-5.53

[A11]

2.5.2 Analysis of Outcome Impacting Parameters

The thickness of the mesh has a direct correlation to the portion of the seed that is overlaid by the mesh. The more the overlay, the higher the deviation of the estimated morphometry from the actual. Figure 2.18 shows an example where a sample of seven canola seeds are laid on a thicker hexagonal mesh (3 mm) and a thinner rhombus mesh (0.5 mm) and Table 2.6 shows the area estimates for seed kernels placed within meshes of varying thickness.

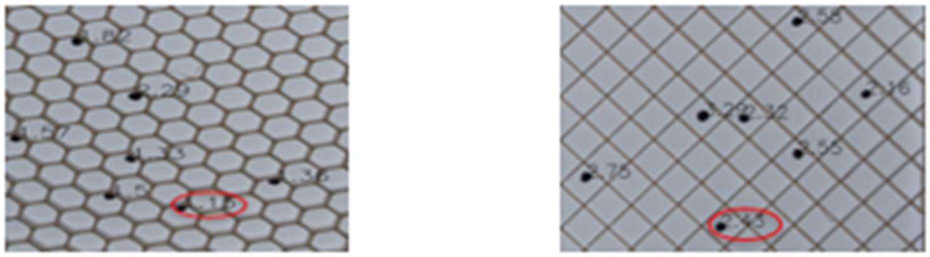


Figure 2.18: Impact of Mesh Thickness on Estimation

Table 2.6: Seed Areas using Meshes of Different Thickness (Area in mm^2)

Thickness	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5	Seed 6	Seed 7
3mm	1.57	1.82	1.33	2.29	1.5	1.36	1.15
0.5mm	2.75	2.55	2.32	3.29	2.16	2.55	2.43

The readings in Table 2.6 show that the estimates for seed areas are significantly different when using meshes of different thickness. The thicker mesh overlays more and hides more part of the seed compared to the thinner mesh which does not overlay as much. As a result, the contours detected are not the same even with all other parameters being the same, causing the results between the two images to be deviant. For instance, consider the seed circled in red in Figure 2.18.

The contours of the seed when placed on the thicker and lighter meshes are as shown in Figure 2.19. As observed, the thicker mesh results in a part of the seed being hidden yielding a partial contour. The impact of mesh thickness is felt more on seeds that are relatively smaller in size such as canola.

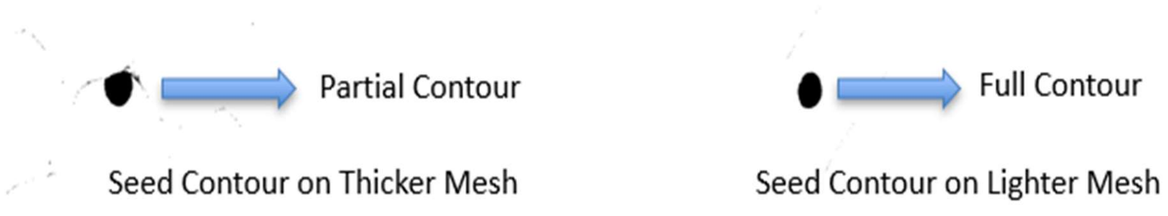


Figure 2.19: Detected Seed Contour on Thicker and Lighter Meshes

2.5.3 Comparison to Other Applications

The areas estimated by the proposed algorithms are compared against the estimates of other applications. Seed Counter, LeafIT, SmartGrain, and a manual estimation performed using a vernier caliper are compared. A sample of seven soy seeds is used for the comparison. The reason for the soy seed sample is that some of the applications in consideration are built to estimate specific types of seeds and do not necessarily pick up

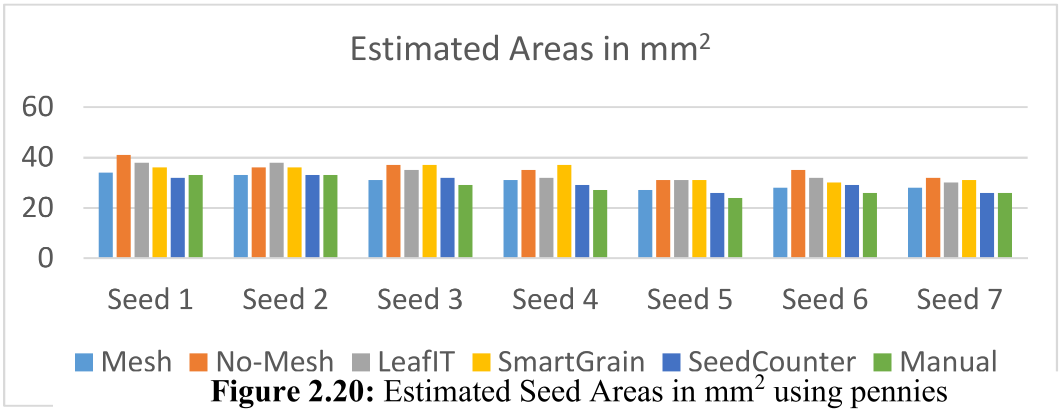


Figure 2.20: Estimated Seed Areas in mm^2 using pennies

seeds smaller or larger than a specific size. Soy is one of the seeds which is estimated by all of the applications in question and the results of estimation are shown in Figure 2.20.

From[A12] the results[A13]:

1. Manual estimations using a vernier caliper are consistently lower than those provided by any of the applications and proposed algorithms.

2. The results provided by the Mesh algorithm are most similar to those of the results provided by the android app, Seed Counter with the most considerable difference in estimates being two mm^2 for seeds one, four, and seven.

3. The results provided by the No-Mesh Algorithm are most similar to that of LeafIT's, with the most considerable difference in estimations being three mm^2. Not surprisingly, both algorithms use ground truth reference objects. The difference, though, is that LeafIT estimates morphometry of leaves, one at a time. In contrast, the No-Mesh algorithm estimates morphometry of seeds, multiple at a time.

2.6 Future Work and Conclusion

The proposed algorithms are a formidable addition to the repertoire of image processing algorithms for seed kernel morphometry estimation. The algorithms demonstrate the estimation of seed morphometry using low-cost hardware components. While the algorithm is demonstrated on smaller seed varieties such as soy and canola, the performance of the algorithm on larger seed varieties such as cassava remains to be seen. A sturdy hardware setup that is shippable to customers that wish to leverage the technology is to be developed if the algorithms are to be widely adopted. The implementation of the algorithms as part of an android application, OneKK is currently in progress. The app helps plant scientists leverage the power of the algorithms using handheld devices such as mobile phones and estimate seed morphometry of a large volume of seeds in a matter of seconds.

Chapter 3 - Deep Learning using Domain Randomization for Image-based High-throughput Phenotyping

Computer Vision is a subset of Artificial Intelligence (AI) that aids computers in extracting meaningful information from different input sources such as images and videos. In other words, a human-like understanding and interpretation from input sources may be deemed Computer Vision. Deep Learning, a subset of the broader family of Machine Learning, is the idea of utilizing neural networks to learn from a set of data. The basic unit of a neural network is the perceptron which takes an input x_i and outputs $f(w^T x + b)$ where w and b are the parameters controlled by a set of hyperparameters and f is a non-linear activation function. Several neurons are stacked together leading to a neural network. The use of Computer Vision and Deep Learning in High-Throughput Phenotyping provides invaluable knowledge to plant scientists in specific and farming community in general. Precisely, Seed Phenotyping is the area of High-Throughput Phenotyping that deals with the comprehensive assessment of complex seed traits such as growth, development, tolerance, resistance, ecology, yield, and the measurement of parameters that form more complex traits [78]. The goal of Seed Phenotyping is the analysis of seed quality using morphological traits, eventually aiding in Seed Certification. Seed Certification is a mechanism to ensure high quality of the seed and propagate materials of superior adapted crop varieties is the identification of seed type [172]. The state-of-the-art computer vision techniques rely on neural networks for tasks of visual perception such as image classification, object detection and object recognition. Chapter-3 investigates the performance of the state-of-the-art neural networks on synthetic image datasets generated for various types of seed kernels using a technique known as Domain Randomization. The use of the technique enables plant scientists and enthusiasts to train neural networks without the availability of a profusion of training data, a key bottleneck that hinders the use of neural networks in the area of High-throughput Phenotyping.

3.1 Domain Randomization Framework for Synthetic Image Datasets

In the field of AI in general and deep learning in specific, data is key to the development of models that are efficient and robust. However, the availability of data remains a key bottleneck in many domains, high-throughput phenotyping (HTP) included. In the area of HTP, one of the critical tasks is seed certification i.e., the process by which a state seed certifying agency gives official recognition to seeds produced of a cultivar or named variety under a limited generation system which ensures genetic purity, identity, and a given minimum level of quality. A key bottleneck with regards to using neural network models for HTP is the lack of quality image datasets that can be used to train the neural networks. As a result, HTP still lags behind in being able to leverage state-of-the-art neural network models. None of the popular image datasets used to pre-train state-of-the-art image classification and instance segmentation models such as ImageNet and COCO contain seed kernels of any variety as one of their classes. In order to generate quality image datasets to train the state-of-the-art neural networks, one could either capture a multitude of seed kernel images manually or leverage Domain Randomization. The capture of a profusion of imagery requires access to a plethora of seed kernels which is not always feasible, especially considering the growing number of citizen scientists who don't always enjoy the luxury of a lab setting with resources available. Even for plant scientists who have access to resources, the capture of a large volume of images proves tedious and time consuming. Considering the aforementioned problems, an efficient way to create datasets is to leverage Domain Randomization (DR), a technique that bridges the gap between simulation and real-world experiments. In the context of applying DR to HTP for neural networks, the idea is to generate synthetic images that resemble seed kernels in the real-world to train on the neural networks.

3.1.1 Related Work

In the work done by Rozantsev, et. al [140], an approach to estimate the rendering parameters required to synthesize similar images given a coarse 3D model of the target object was proposed. The identified parameters could be reused to generate an unlimited number of training images of the object of interest in arbitrary 3D poses.

Silva et. al [33] provide an approach to estimate soybean leaf defoliation using convolutional neural networks and synthetic images. The captured imagery is put through a pre-processing pipeline where the image is converted to grayscale and rotated to make the image rotation invariant. The rotated image is put through polygonal or circular defoliation i.e., multiple polygonal or circular areas of a certain dimension are cropped out of the image resulting in augmentation.

Rahnemoonfar and Sheppard [132] use synthetic images for fruit counting using deep simulated learning. In order to generate the synthetic images, a blank image of size 128 x 128 pixels is created followed by filling the entire image with green and brown circles to simulate the background and the tomato plant, which are later blurred by a Gaussian filter. Tomatoes of different sizes are created by drawing several circles of random size in red at random positions on the image.

3.1.2 Procedure

The synthetic image generation framework is developed for the seed kernel types of canola, rough rice, sorghum, soy, and wheat. In addition, synthetic images for the US Penny coin are also developed. The coin, in the current context, is used to generate images that the framework may use as the ground truth for the estimation of morphometry pertinent to each of the seed kernels. The framework involves three steps primarily:

1. Acquisition of a sample of real images to be synthesized.
2. Generation of synthetic images with desired foregrounds and backgrounds.
3. Generation of image annotations for the neural networks to consume the synthetic images.

3.1.2.1 Image Acquisition

Initially, the images for each of the seeds in the considered sample are required to be acquired. In order to develop the framework, 40 seeds are considered for each seed type of

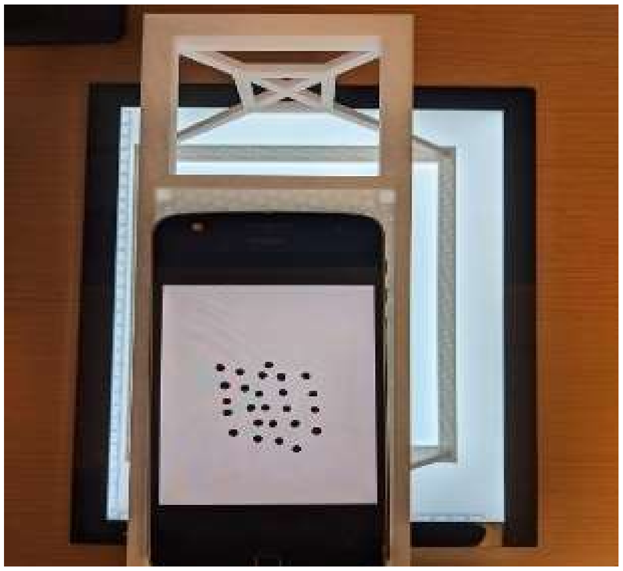


Figure 3.1: Image Capture of Sorghum Seed Kernels

which 25 are used for training/ validation and 15 for testing. For the coins, 10 pennies are considered of which seven are used for training and three for testing. Fewer pennies are selected since pennies are uniform as opposed to seed kernels that have random shapes. The images are captured using a Motorola Moto G6 mobile phone placed on a 3-D stand that holds the phone orthogonal to the ground so as to eliminate any skew that might result from holding the phone free-hand. For the background, a lightbox emitting white light is chosen. The resolution of the captured images is 3456 x 4608 px. Figure 3.1 shows the capture of the 25 seed sample of soy selected for training. Likewise, images of the others are captured.

3.1.2.2 Creation of Synthetic Images

1. Capture images of each of the samples as shown in Figure 3.1 and extract out the seeds (or coins) on the image. Typically, the seeds (or coins) are the foreground on the image and are extracted by separating the alpha channel from the image. Such an extraction can be performed using tools such as Gimp or OpenCV GrabCut. For this work, Gimp is used.
2. Select a canvas onto which the extracted samples can be laid. The canvas functions as the background of the image and it is recommended that a canvas that resembles the real-world test site be picked. If there are different test sites, it is best to have at least one image of each of the test sites.
3. Now that foregrounds and backgrounds are available, select all samples belonging to a particular foreground and lay them on each of the backgrounds with varying

sizes and orientations. Other kinds of augmentation such as brightness are also be performed as needed.

4. As the images are being created in step 3, concurrently create a black canvas and color each of the regions where the seed (or coin) is placed on the image. Use a unique color for each of the objects on the image so as to identify them later.

Figure 3.2 shows the synthetic images of the seed kernels of canola, rough rice, sorghum, soy, and wheat, along with the US Penny coin.

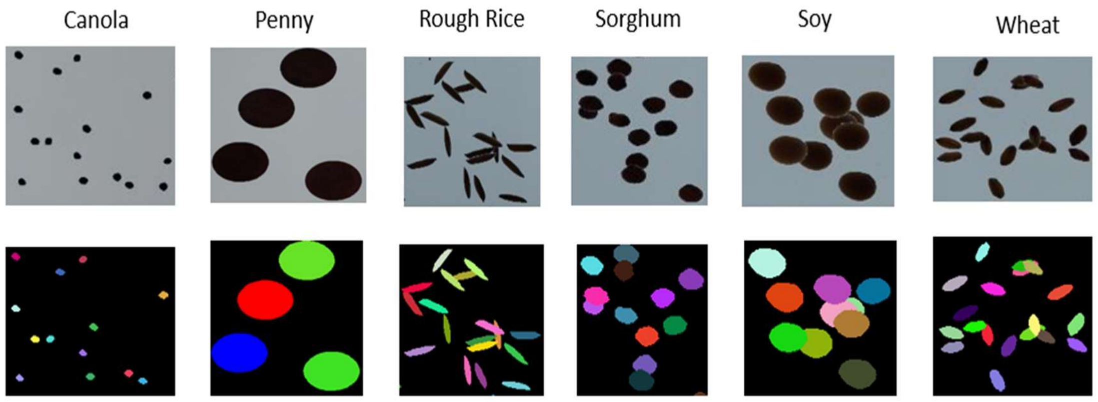


Figure 3.2: Sample of Seeds and Corresponding Masks for Synthetic Data Sets

3.1.2.3 Image Annotations

In order for the neural networks to consume the images generated by the framework, the images are required to be annotated. Annotations are files that contain information about the image ID, boundary, class, area and any other characteristic of each of the objects in the image. Different neural networks consume annotations in different formats. For instance, Mask R-CNN consumes annotations in the COCO file format whereas YOLO consumes annotations in the TXT file format although either of them performs the task of instance segmentation. Currently, frameworks that convert annotations in the COCO file format to TXT file format are available. However, a framework that auto-generates the annotations in COCO file format alongside synthetic images is not available. Hence, the proposed framework auto-generates annotations in the COCO file format which may later be used as-is on Mask R-CNN or converted to a different file format such as TXT using an available translation framework.

COCO Annotation Format

The COCO annotation file is formatted in JSON and consists of five sections:

1. Info
2. Licenses
3. Images
4. Annotations
5. Categories

Info: The info section consists of basic information about a dataset and may comprise of any information that the user/ creator sees fit. Typically, information such as description, date, author and version are included.

Licenses: The licenses section consists of licenses applicable to each of the images in the dataset. The author/ creator of an image dataset may choose to make their images open-source under a specific license that the users are required to cite. Image ID, URL and the name of the license are the fields commonly present in the section.

Images: The Images section consists of all the images in the dataset. The section primarily consists of the high-level information about each of the images such as name, image ID, height, width and date of capture. The image IDs are required to be unique to the image for ease of identification. Note that information related to the labels, bounding boxes or segmentation is not present in the section.

COCO has annotations meant for five different use cases:

1. Object Detection
2. Key Point Detection
3. Stuff Segmentation
4. Panoptic Segmentation
5. Image Captioning

The proposed framework aids in generating annotations for object detection wherein the purpose is to detect each of the objects belonging to a certain category in a given image. The object detection use case is required to have a list of categories and annotations to define each of the objects in the image. The categories section is intuitive and consists of each of the broader categories of objects in the image such as person, car, table, chair and so on. The annotations section contains a list of properties to define each of the objects belonging to every category on the image. The annotations section consists of the following properties as described below:

1. Segmentations: The segmentations are a list of coordinates of polygons drawn around each of the objects in the image.
2. Area: The area is measured in pixels and represents the area of the polygonal segmentation.

3. IsCrowd: The IsCrowd property specifies if the annotation refers a single object or a cluster of objects such as the crowd at a game.
4. Image_ID: The Image_ID property refers to a specific image in the dataset.
5. Bbox: The Bbox property specifies the coordinates of the bounding box around an object in the image. The format is [top left x, top left y, width, height].
6. Category ID: The category ID attributes a category from the list of categories defined in the categories section.
7. Annotation ID: The annotation ID specifies a unique ID for each annotation.

3.2 Seed Kernel Counting and Morphometric Estimation using Domain Randomization and Transfer Learning

In the area of computer vision, the topics of object detection and semantic segmentation lay the foundation for instance segmentation. Object detection refers to the task of detecting an object on an image and plotting a bounding box around the object whereas semantic segmentation refers to the task of associating each pixel of an image to a class label. All the instances that belong to a certain class are treated the same. Instance segmentation applies a layer of precision on top of semantic segmentation and distinguishes between each of the instances belonging to a certain class. For instance, assume an image that consists of two humans and two apples. When semantic segmentation is applied on the image, the humans and apples are identified distinctly but the two humans are considered the same entity. Likewise, the apples. However, when instance segmentation is applied, each of the humans is considered their own entity and likewise, the apples. In other words, instance segmentation is the process of applying a distinct mask to each of the entities on the image and assigning each mask a class label. Applying the principle of instance segmentation to neural networks, the number of seed kernels on an image may be estimated. In addition, the morphometry of each of the detected seed kernels may be estimated by processing the mask drawn around each of the seed kernels. In order to apply instance segmentation, the state-of-the-art neural networks of YOLOv5 and Mask R-CNN are considered.

3.2.1 Related Work

Toda et al. [170] applied synthetic image datasets to train instance segmentation neural networks for crop seed phenotyping. The model was trained on the seed kernels of rice, wheat, lettuce, and oat. The work demonstrated that utilizing synthetic data is a powerful method to alleviate human labor costs for deploying deep learning-based analysis in the agricultural domain.

Kuznichov et al. [75] proposed a technique named collage, an image analysis technique to augment data for leaf segmentation and counting tasks in Rosette plants. The idea is to create a set of segmented leaf images on a transparent background, a single leaf per image, using manual annotations or an automatic procedure. The single leaves are made to undergo different geometric transformations with random parameters in a fixed range and pasted in random locations over selected backgrounds. The application of the technique provides diverse datasets for training on neural networks.

Tanabata et al. [164] proposed SmartGrain, a software tool to measure seed shape through image analysis. The software uses an image analysis method to reduce the time taken in the preparation of seeds and in image capture. Outlines of seeds are automatically recognized from digital images, and several shape parameters, such as seed length, width, area, and perimeter length, are calculated. The software removes areas of awns and pedicels automatically, and several QTLs were detected for six shape parameters.

Schrader [150] proposed LeafIT, an android application to measure leaf area. Leaf area is a key parameter with relevance for other traits such as specific leaf area, which in turn correlates with leaf chemical composition, photosynthetic rate, and carbon investment. The android application is free to use and runs on Android 4 or higher. The area measurements made by the applications are compared to the measurements made by WinFOLIA [185], a commercial software that uses the Altman-Bland method for measurements.

Komyshev, et al. [71] proposed SeedCounter, a mobile application for grain phenotyping. The application solves the problem of grain phenotyping analysis using image processing. The application was evaluated under six different lighting conditions and three mobile devices at the time of development. The finding was that lighting had a significant impact on the outcomes produced by the application.

3.2.2 YOLOv5

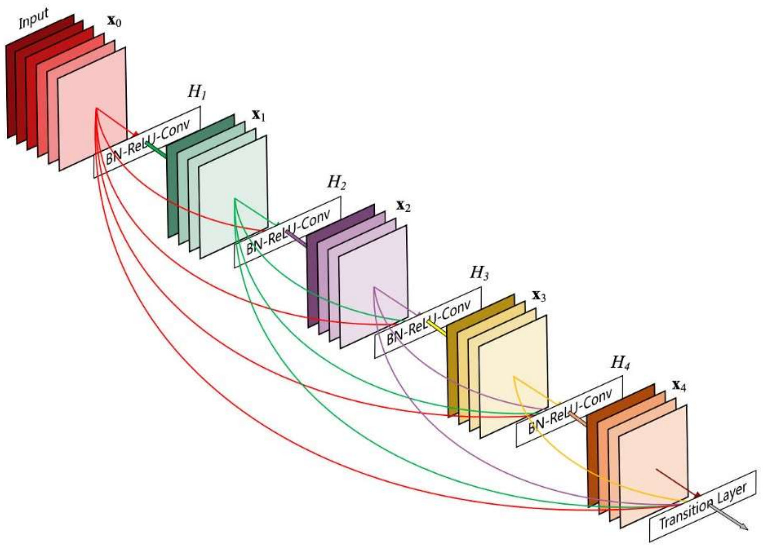
You Only Look Once (YOLO) is one of the first object detection models that introduced the idea of combining bounding box prediction and object classification into a single network. The architecture got introduced as part of a framework called Darknet. YOLO has come out with timely iterations with the latest iteration being five. Each of the iterations, except the fifth iteration has architectural changes done to improve upon the performance of the model. YOLOv5 consists of four different architectures, YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x. Each of the architectures differ in the number of neural network layers with 5s, 5m, 5l and 5x containing 283, 391, 499 and 607 layers respectively. All four of the official models made available are trained from scratch on the COCO dataset. YOLOv5 is a one-stage object detection framework that has fast inference and optimization for parallel computations. The architecture of YOLOv4 primarily consists of three components:

1. Backbone

2. Neck

3. Head

Backbone: The backbone of the object detector is a pre-trained neural network such as VGG-16, ResNet-50, CSPDarknet53 or ShuffleNet. The YOLOv5 model considers the use

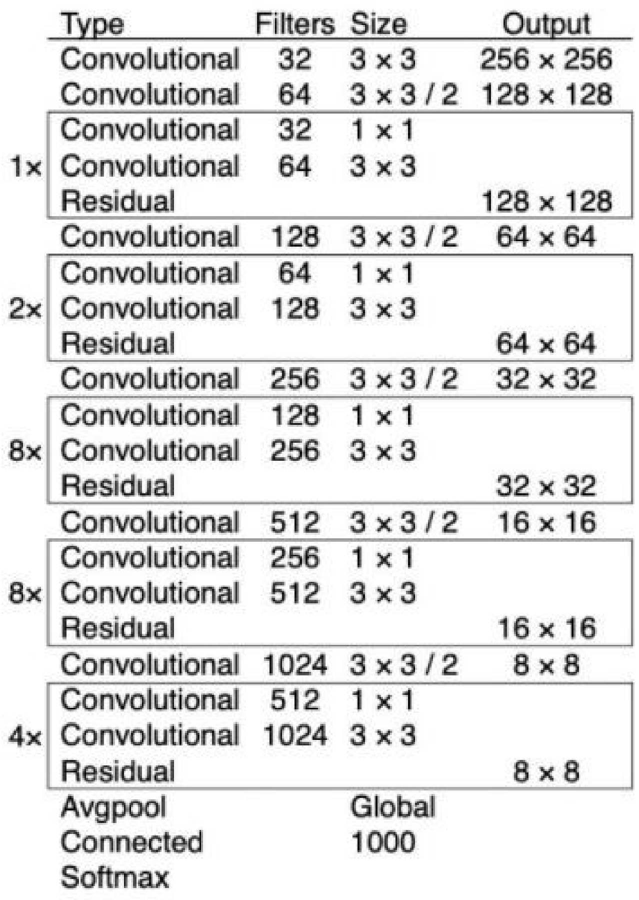


of a Focus layer [117] in a DenseNet coupled with Cross-Stage-Partial (CSP) connections,

Figure 3.3: DenseNet Architecture [56]

and CSPDarknet53 for the backbone. DenseNet is made up of multiple dense blocks, as shown in Figure 3.3.

The Focus layer provides the advantage of requiring reduced CUDA memory, increased forward propagation and backpropagation. A dense block contains multiple convolutional layers with each layer H_i composed of batch normalization, ReLU, and followed by convolution. Each layer H_i consumes as input the output of each of the previous layers as well as the original input. Each layer produces four feature maps i.e., the number of feature maps increases by four upon passing through a dense block. Each of the dense layers consists of a transition layer containing convolution and pooling layers. However, the DenseNet architecture is computationally expensive. Cross-Stage-Partial (CSP) Connections increase the computational efficiency of the convolution. Each layer H_i consumes as input the output of each of the previous layers as well as the original input. Each layer produces four feature maps i.e., the number of feature maps increases by four upon passing through a dense block. Each of the dense layers consists of a transition layer containing convolution and pooling layers. However, the DenseNet architecture is computationally expensive. Cross-Stage-Partial (CSP) Connections increase the computational efficiency of the design by separating the input feature maps of the DenseNet architecture into two parts. The first part $x_o^{'}$ bypasses the dense block and becomes part of the input to the next transition layer whereas the second part $x_o^{''}$ passes through the dense block. The CSP connections along with Darknet-53 whose architecture is as shown in Figure 3.4 for feature extraction.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3.4: CSPDarknet-53 [136]

Neck: Object detection models insert additional layers between the backbone and head to collect feature maps from different stages. The neck typically consists of several bottom-up paths and top-down paths. Examples include Feature Pyramid Network (FPN), Path Aggregation Network (PAN) and Bi-FPN. PAN is the choice of neck used in the current experiment. In conventional deep learning frameworks, each layer consumes input from

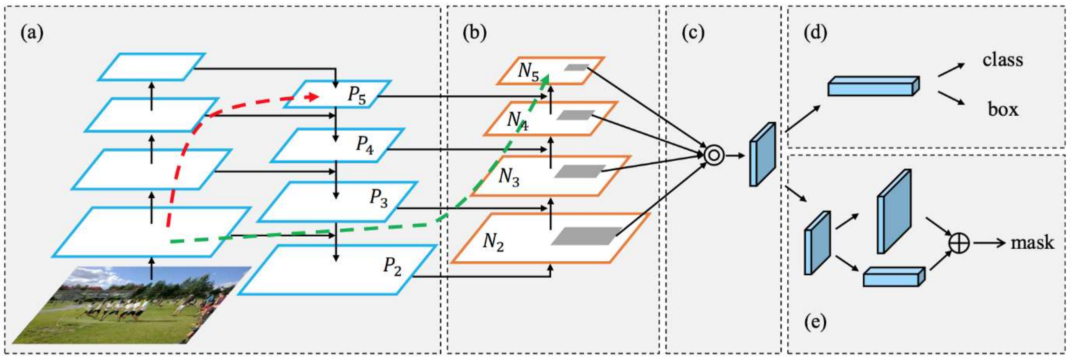


Figure 3.5: Path Aggregation Network [83]

the previous layer. The early layers in the network extract localized texture and pattern to build up the semantic information required by the later layers in the network. However, as the information is repeatedly convolved, the knowledge required to localize is lost. PAN is a kind of an architectural system that facilitates the availability of fine-grain information to the top layers in the network. Figure 3.5 shows an FPN with PAN.

The bottom-up path, shown in Figure 3.5 (b), is augmented to make the information from lower layers percolate to the layers at the top. In a conventional FPN, the spatial information travels to the upper layers using the path shown by the red arrow in (a) whereas PAN introduces the path shown by the green arrow to percolate information. In the original implementation of PaNet, the current layer and information from previous layer are added together to form a new vector whereas in YOLOv4 implementation, a modified paradigm wherein a new vector is created by concatenating the input and vector from a previous layer.

Head: The head of the object detector predicts classes and bounding boxes of objects. In case of a one-stage detector, the primary responsibility of the head is to perform dense prediction. The dense prediction is the final prediction which is composed of a vector containing the coordinates of the predicted bounding box (center, height, width), the confidence score of the prediction and label. The prediction is based on anchor boxes that help to predict objects of a certain aspect ratio and size. Multiple anchor boxes may be assigned to each of the grid cells of the image.

Figure 3.6 shows the detections made by YOLOv5 on the different entities that the model is trained.

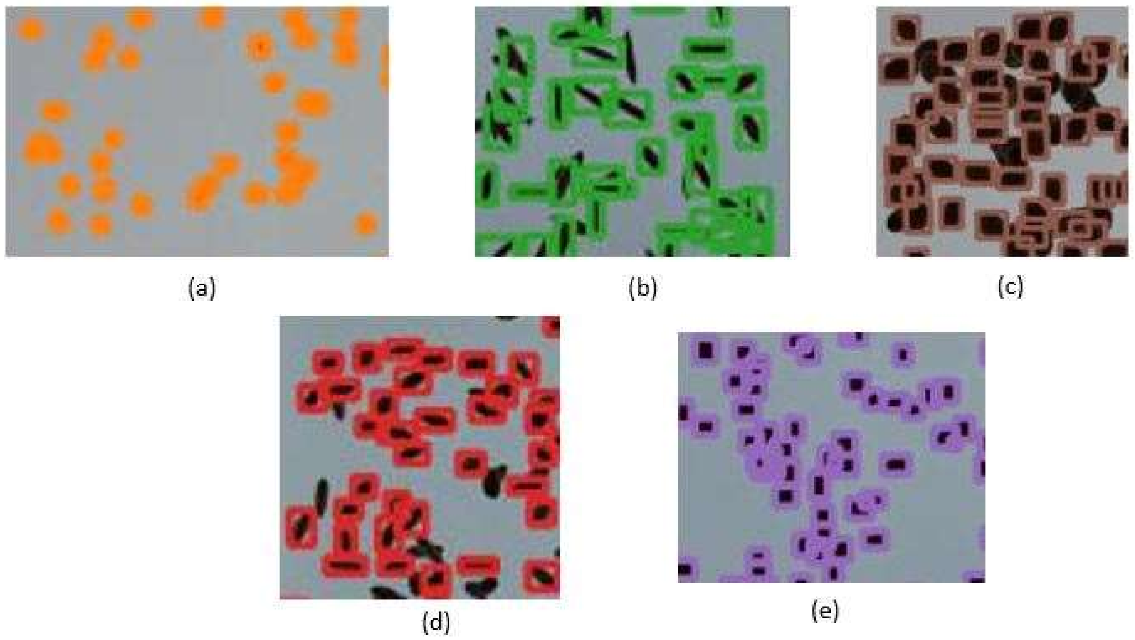


Figure 3.6: YOLOv5x Instance Detections on Clustered Seeds of (a) Canola (b) Rough Rice (c) Soy (d) Wheat (e) Sorghum

3.2.2.1 Tiny YOLO

The models created using YOLOv5 are too complex and large to deploy natively as mobile apps. Tiny-YOLO models are a variation of the YOLO models but are much smaller in comparison to the YOLO models. However, the models are not accurate as their bigger counterparts. For the purposes of this work, YOLOv4-Tiny is considered since it is closest to the YOLOv5 models at the time of experimentation. The average precision (AP) at 50% mask-IOU of YOLOv4-Tiny is 40.2% in comparison to YOLOv4's 64.9% on the COCO data set. In exchange for accuracy, the speed of the models has a multifold increase. The Tiny-YOLOv4 achieves about 371 FPS during inference on a single GPU (GTX 1080 Ti) in comparison to YOLOv4's 62 FPS on a single GPU (TeslaV100) on the Darknet framework and is about six times faster. While there is a significant decrease in the accuracy of the Tiny-YOLO models in comparison to YOLO models on the complex COCO data set, it is worth noting that Tiny-YOLO models function nearly as efficiently as the larger YOLO models when trained on custom data sets that are not as complex, like the ones in this work. A key difference between the larger models and Tiny-YOLOv4 is that the input image resolution of Tiny-YOLOv4 is 416 x416 px as opposed to the larger models' 768 x768 px.

3.2.3 Mask R-CNN

3.2.3.1 Mask R-CNN Architecture

The architecture of Mask R-CNN is explained further and is as shown in Figure 3.7.

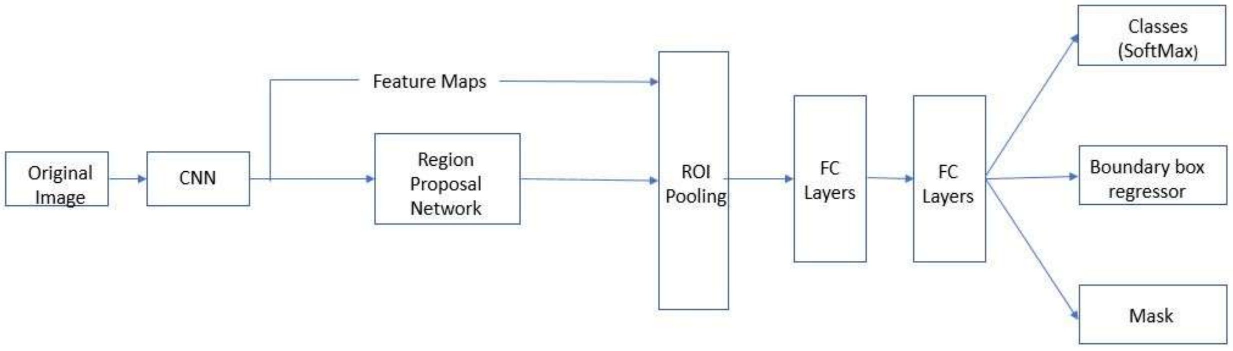


Figure 3.7: Mask R-CNN Architecture

The architecture of Mask R-CNN is divided into two stages. In the first stage, the network generates proposals about the regions where an object may be present based on the input image whereas the second stage predicts the class of the object, refines the bounding box and generates a mask at the pixel level of the object. The network consumes the input image and passes it through a CNN such as VGG-16, ResNet-50, and Inceptionv3. Typically, CNN applies a series of convolutions on the input image to generate feature maps. The last stage of CNN is the fully connected layer that predicts the class of the image from the feature maps. However, in case on Mask R-CNN, the fully connected layer of the network is not leveraged. Instead, only the feature maps are collected as output and passed to the Region Proposal Network (RPN). RPN is a fully convolutional network that predicts the object bounds and “objectness” score at each position in a feature map. Objectness score refers to the likelihood of an object being present at a certain location. The architecture of RPN is shown below in Figure 3.8.

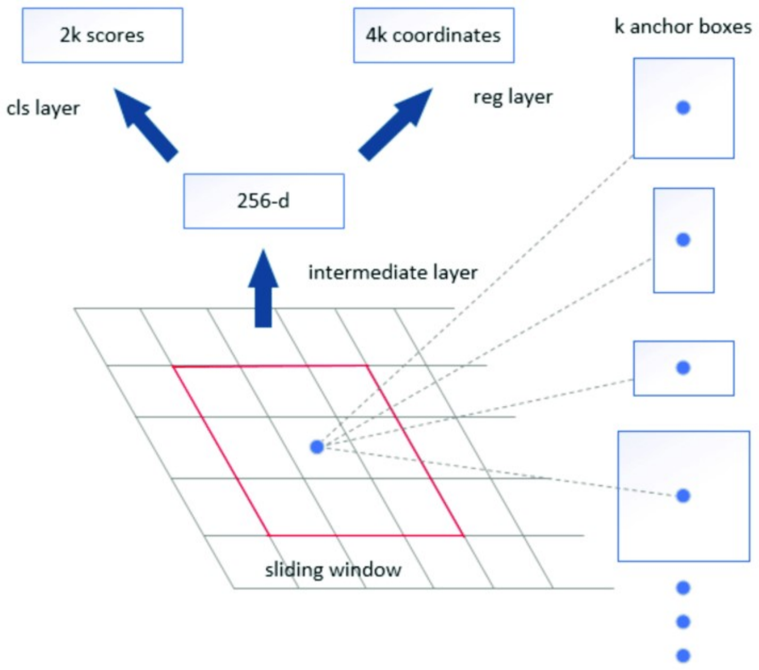


Figure 3.8: RPN Classifier, Regressor and Anchor Boxes [137]

In order to generate the proposals, the RPN slides a small network over the convolutional feature map that is output by the last convolutional layer (indicated as intermediate layer in Figure 3.8) of the CNN. The network uses n x n spatial window as input from the feature map. Each sliding window is mapped to a lower dimensional feature. RPN has two components – classifier and regressor.

Classifier determines the probability that a proposal contains the target object. Regressor regresses the coordinates of the proposal, so they are better refined and fit the target precisely. The key idea that aids the regressor and classifier is Anchor Boxes or Anchors. Anchors are a set of predefined locations and scales relative to the image, typically centered at the sliding window of the RPN. Binary ground-truth classes i.e., object and background, and bounding boxes are assigned to individual anchors according to a predefined Intersection-over-Union (IoU) score. In case of Mask R-CNN, anchors with three different aspect-ratios and three scales are proposed. Hence, a total of nine anchor boxes are present for each pixel of the image. Hence, there a W x H x K anchors in an image of width W, height H, and K anchors per pixel. The classifier and regressor are fully convolutional layers that typically apply 1 x 1 sized filters. The classifier is binary in the sense it merely predicts the presence of an object but not the class of the object. The regressor predicts the bounding box around the object detected by the classifier. Assuming an image of height H, width W, N classes of objects, and K anchors per pixel, the output of the classification head has the shape H x W x 2.K whereas that of the regression head is H x W x 4.K. The idea behind having multiple anchor boxes per pixel location is so that the network is able to predict multiple objects of different sizes per location. Finally, the loss function for RPN is given by $L(\{p_i\}, \{t_i\}) = (1/N_{cls}) \times \Sigma L_{cls}(p_i, p_i^*) + (\lambda/N_{reg}) \times \Sigma\ p_i^* Lreg(t_i, t_i^*)$ [125] where i is the index of anchor, p is the probability of being an object, t is the vector of four parameterized coordinates of predicted bounding box, * represents ground truth box, L for cls represents Log loss over two classes. $p^*$with regression term in loss function counts if and only if an object is identified, else, it is zero. N_{cls} and N_{reg} are normalizations. λ is set to a default value of 10 in order to scale classifier and regressor by a common factor.

After the RPN is the ROI Align i.e., Region of Interest Align phase. ROI refers to a region proposed by the RPN that corresponds to the original image. ROI Align is a process that is put in place to improve upon the technique of ROI Pooling. The process of ROI Pooling is described first to better describe the process of ROI Align. The goal of ROI pooling (and ROI Align) is to convert the feature maps to a standard size, so they are able to be consumed by the fully connected layers (FC Layers shown in Figure 27). The process of ROI Pooling is explained using Figure 3.9 which shows the image of a wheat plant that is run through a CNN resulting in a convolved feature map. Assume that the size of the original image is

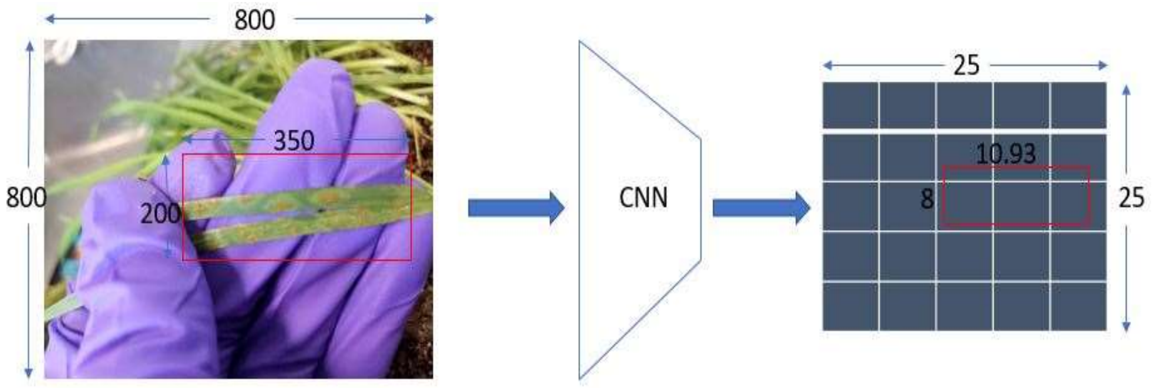


Figure 3.9: Original Image Convolved to Feature Map

800 x 800 px and the size of the ROI is 350 x 200 px. As the image is run through the CNN and convolutions are applied, consider that the resulting feature map is of dimensionality 25 x 25 px i.e., a reduction of 32 times from the original size. Along with the size of image, the size of the ROI is also scaled down by 32 times which results in a size of 10.93 x 8 px. ROI Pooling applies quantization on the feature map as it consumes the feature map as input. Quantization is the process of constraining an input from a set of real values to a set of discrete values, such as integers. As quantization is applied, only the integral values are

considered as input for processing. In other words, ROI of 10 x 8 px is considered instead of 10.93 x 8 px. While 0.93 px might not seem like much of a displacement, it is important to realize that each pixel in the feature map corresponds to 32 pixels in the original image. As a result, portion of the ROI is lost as it is consumed as the input. In addition to loss of ROI in the input phase, information is also lost in the output as ROI Pooling attempts to alter the dimensionality of the feature maps. Assume the required output dimensionality of the feature map is 7 x 7 px. In order to convert a feature map of size 10 x 8 px to 7 x 7 px, ROI pooling gets the ratio of the width and height dimensions i.e., 10/7 and 8/7 respectively. The results are 1.42 and 1.14 respectively, both float values. Once again, ROI Pooling applies quantization which considers only the integral value of the float numbers. A 1 x 1 filter is applied along the feature map to retrieve a 7 x7 output using either max pooling or average pooling. Max pooling refers to the technique of selecting the maximum value within the filter whereas average pooling refers to the technique of selecting the average of the values within the filter. It doesn't matter in a 1 x 1 filter if max pooling or average pooling is applied. Since only seven rows and seven columns are in the output, three columns out of the ten rows and one row out of the eight is ignored, as shown in Figure 3.10. In summary, ROI pooling applies quantization twice, once during the mapping of image coordinates to feature map

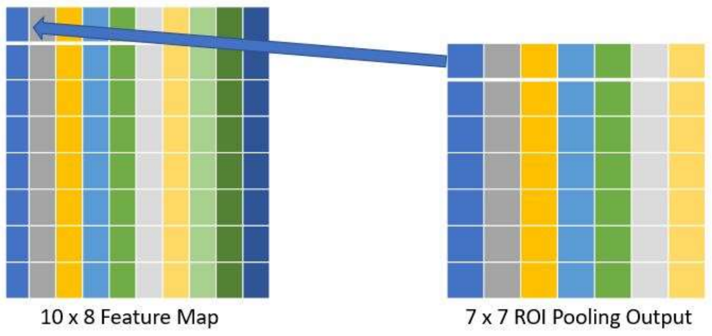


Figure 3.10: Quantized Feature Map using ROI Pooling

coordinates and again during the mapping of feature map coordinates to ROI Pooling output feature map. Overall, ROI Pooling is a lossy transformation wherein key features of the image are ignored in the process of quantization.

ROI Align helps mitigate the pitfalls of ROI Pooling by eliminating the application of quantization for data pooling. Instead, it divides the ROI into a certain number of equal sized boxes and applies the process of bi-linear interpolation for data pooling as shown in Figure 3.11.

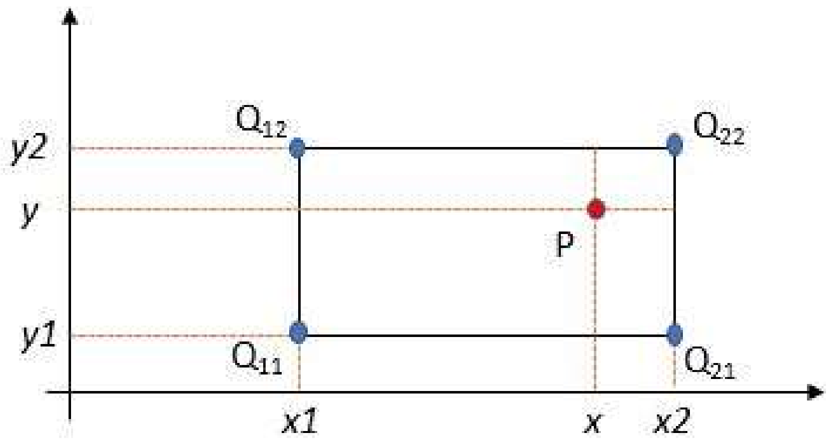


Figure 3.11: Bi-linear Interpolation

Bi-linear interpolation is a technique for two-dimensional interpolation on a rectangle. It works under the assumption that the values of some unknown function at points that form a rectangle are known. In Figure 31, the four tuples formed by the cross-join of the two points (x_1, y_1) and (x_2, y_2) make up the vertices that form the rectangle. The values at (x_1, y_1), (x_1, y_2), (x_2, y_1) and (x_2, y_2) are Q_{11}, Q_{12}, Q_{21}, and Q_{22} respectively. The point whose value is to be determined is (x, y) indicated by P. The size of each box for bi-linear interpolation is determined by the size of the mapped ROI and size of the pooling layer.

Assuming a pooling layer of size 7 x 7 as shown in Figure 3.12 and considering the feature map ROI shown in Figure 3.12 whose width and height are 10.93 x 8 respectively, the width and height of each box are 1.14 and 1.56 respectively.

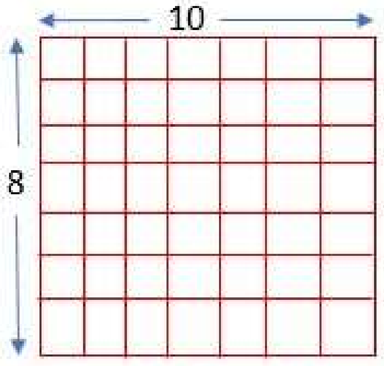


Figure 3.12: ROI Segmented into 7 x 7 Boxes

Four sampling points are created within each of the cells, one at the top-left, one at bottom-left, one at top-right and another at bottom-right of the cell. The relevant equation to derive the values for each of the points is as follows:

Top-Left Point:

$X = X\_box + (width/3) * 1$

$Y = Y\_box + (height/3) * 1$

Bottom-Left Point:

$X = X\_box + (width/3) * 1$

$Y = Y\_box + (height/3) * 2$

Top-Right Point:

$X = X\_box + (width/3) * 2$

$Y = Y\_box + (height/3) * 1$

Bottom-Right Point:

$X = X\_box + (width/3) * 2$

$Y = Y\_box + (height/3) * 2$

Where X_box and Y_box indicate the x and y coordinates that belong to the top-left corner of the ROI. A point from the derived list of points is considered and connected to the middle of the closest neighboring cells, unless they are connected already. Figure 3.13 shows one segment from each of the segments of the 7 x 7 ROI grid wherein the segment overlaps six cells in the grid. As a hypothetical instance, the point (9.5, 10.5) is assumed to be one of

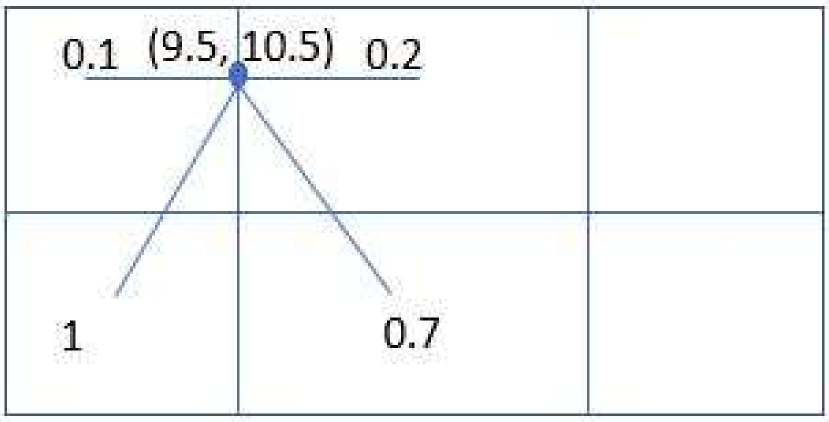


Figure 3.13: Bi-linear Interpolation on a Grid Cell in ROI

the points from the derived list. Following the procedure described earlier, the point is connected to the midpoint of the closest neighboring cells. The values at each of the midpoints of the cells to the top-left, top-right, bottom-left, and bottom-right are assumed to be 0.1, 0.2, 0.7 and 1 respectively. Bi-linear interpolation is applied using the formula,

$P = (y2 - y)/(y2 - y1)((x2 - x)/(x2 - x1)\, x\, Q11 + (x - x1)/$

$(x2 - x1)\, x\, Q21) + (y - y1)/(y2 - y1)((x2 - x)/(x2 - x1)\, x\, Q12 + (x - x1)/$

$(x2 - x1)\ x\ Q22)$. Upon the computation of bi-linear interpolation values for the four sampling points in the segment, max pooling or average pooling is applied (as in ROI Pooling). Since each of the cells in the 7 x 7 grid is used, all the information within the ROI is leveraged. Warped features from the ROI Align phase are fed into the fully connected layers of the network. The network uses the bounding box regressor to predict the bounding box around the object, mask stage to plot the mask around the object and SoftMax layer to predict the class of the object that is detected by the network.

3.2.3.2 Mask R-CNN Implementation

The Mask RCNN implementation used is the popular implementation by Matterport that runs on Tensorflow. Mask RCNN comes pretrained on 80 classes that belong to the COCO data set. For a majority of the applications, the default parameters shipped out by the Matterport implementation are good enough for object detection. However, for the object detection of the seeds and pennies, one key change that is made is the 'anchor scales' and 'anchor ratios' of the Region Proposal Network. Anchor scales and anchor ratios play a crucial role in instance detection. For instance, say the network is trained on canola with images of size 768 x 768 px. If the anchor scales were selected to be [32, 64] and anchor ratios, which are essentially aspect ratios, were [1, 2], four anchor boxes would be generated at each position with dimensions 32:32 px, ~22:44 px, 64:64 px, ~90:45 px. However, from experimentation, it is observed that even the largest of canola seeds don't occupy nearly as close to the smallest of the anchors 32:32 px (or 22:44 px). As a result, the network fails to train well yielding a higher overall loss and sloppy detection. In order to ensure this does not happen, different anchor scales and ratios are experimented with and the values of [2, 4, 16, 32, 64] and [1.5, 2, 3] are settled on for anchor scales and ratios respectively. For this work, two different models are created using the pre-trained weights of COCO and ImageNet with the backbone of ResNet101 architecture. While ImageNet is a data set that comprises of 1000 classes in comparison to COCO's 80, it is worth noting that neither ImageNet nor COCO was trained on any kind of seeds originally. While the performance of both models is similar, there is one key fallacy that they possess pertinent to seed shape. Amongst the five seeds in question, the seeds of soy and sorghum are circular and similar in size on occasion. Likewise, wheat and rough rice, except that they are oblong in shape. As a consequence, both models tend to commonly mistake rough rice for wheat and vice-versa, and soy for sorghum and vice-versa. The seeds of canola, although circular like soy and sorghum are significantly smaller in comparison leading to a lower error rate while detecting them. Figure 3.14 shows the detections of seeds on each of the images in question. While the classification of seeds may not be accurate, a point worth noting is that the instance segmentation performed by the networks is beneficial by itself because the majority of applications, desktop or mobile, are catered to work for a certain type of seed. As a result, classification of seeds isn't as critical as instance segmentation.

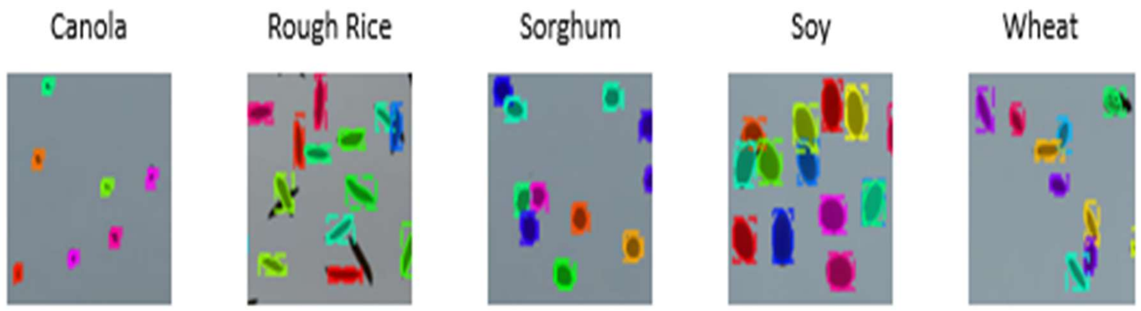


Figure 3.14: Mask RCNN Instance Detections on Different Seeds

3.2.4 Results and Discussion

The project starts off with work on Mask RCNN where the network, once with pre-trained weights of COCO and another time with pre-trained weights of ImageNet, is trained on 31 images of soy where each image contains multiple (about 20 - 30) seeds. The images are manually annotated using Oxford's VGG Annotator tool. Although the network trains well and achieves a decent loss of ~2, the performance of the model on the test data set is sub-par. It is after the experience that the use of domain randomization is employed. The trained models are tested on two test datasets, one that contains real-world images of seeds and another of synthesized seed images. One of the criticisms of the work by Toda et al. [170] is that the metric scores represented for the synthetic data set could be biased since the synthetic test data set is created using the same seed samples used for training. In order to test out the alternative, the synthetic test data set is created using different seed samples. Overall, 150 images of size 768 x 768 px (416 x 416 px for YOLO-Tinyv4) are created where each type of seed contains 15 synthetic images and 15 real-world images containing 50 - 100 seeds. For the real-world test data set on each seed, about 5 of the images do not contain entities that touch each other but the remainder of the images do. The reason for such a setup is to observe the performance of the network on touching and non-touching entities. It is common knowledge that object detection networks struggle to perform well when entities on the image touch. The hypothesis is proven correct by both Mask RCNN and YOLO as the metric scores are significantly lower for images with touching entities in comparison to the ones that don't. Similar behavior is observed on both the synthetic and real seed image test data sets. It is also worth pointing out that the networks perform phenomenally on images with only non-touching entities achieving recall and precision of

~1.0. It is also observed that the performance of the models on the real-world data sets is OK but not close to that of the performance on synthetic datasets that they are trained on.

The metrics to evaluate the models are Recall and Average Precision (AP) which are the standard evaluation metrics for object detection and localization models such as Mask RCNN and YOLO. Recall is defined as the true-positive rate, or the ratio of true positives detected by the model to the total number of objects present. In object detection and localization networks, recall is generally defined over different thresholds of a parameter called Intersection-over-Union (IOU). For the purposes of the current work, recall is defined on the IOU of the bounding rectangles predicted by the network and the ground-truth bounding rectangles. The AP of each class is computed using IOU of masks predicted by the network and the ground-truth masks. As pointed out in the work by Toda et al. [170], the use of bounding rectangles is not an accurate measure since the rectangles are not the minimum area bounding rectangles and don't tightly bind around the instances in the image. The use of IOU over masks helps alleviate the problem since the masks resemble the original instance to a higher degree as they represent the orientation of the instance in the image.

The performance of the models along with pre-trained weights used to train the model on each of the networks is shown in Table 3.1. The image datasets used for training and testing contained seed kernels that are both clustered and segmented.

Table 3.1: Performance of Models on Synthetic and Real Seed Datasets[A14]

Seed	Model	Synthetic Data Set	Real Seed Data Set

	Network	**Weights**	**Recall$_{50}$**	**AP$_{50}$**	**AP@[.5 : .95]**	**Recall$_{50}$**	**AP$_{50}$**	**AP@[.5 : .95]**
Canola	M-RCNN	ImageNet	0.88	0.90	0.76	0.78	0.77	0.62
	M-RCNN	COCO	0.90	0.92	0.77	0.74	0.78	0.64
	YOLOv5x	COCO	0.92	0.94	0.80	0.81	0.77	0.65
	YOLO5l	COCO	0.90	0.92	0.78	0.78	0.74	0.62
	YOLO5m	COCO	0.89	0.88	0.74	0.71	0.73	0.62
	YOLO5s	COCO	0.82	0.83	0.69	0.69	0.71	0.56
	T-YOLOv4	T-YOLOv4	0.71	0.73	0.65	0.56	0.61	0.49
Rough Rice	M-RCNN	ImageNet	0.90	0.91	0.79	0.85	0.74	0.61
	M-RCNN	COCO	0.89	0.89	0.75	0.81	0.74	0.60
	YOLOv5x	COCO	0.94	0.97	0.82	0.83	0.81	0.68
	YOLO5l	COCO	0.92	0.92	0.80	0.77	0.77	0.61
	YOLO5m	COCO	0.90	0.90	0.76	0.72	0.73	0.58
	YOLO5s	COCO	0.84	0.87	0.71	0.64	0.67	0.51
	T-YOLOv4	T-YOLOv4	0.76	0.81	0.72	0.68	0.63	0.52
Sorghum	M-RCNN	ImageNet	0.89	0.90	0.77	0.80	0.79	0.66
	M-RCNN	COCO	0.87	0.89	0.75	0.78	0.80	0.67
	YOLOv5x	COCO	0.93	0.93	0.80	0.84	0.80	0.72
	YOLOv5l	COCO	0.91	0.90	0.78	0.81	0.77	0.64
	YOLOv5m	COCO	0.89	0.88	0.75	0.74	0.72	0.60

	YOLOv5s	COCO	0.89	0.84	0.74	0.68	0.66	0.55
	T-YOLOv4	T-YOLOv4	0.73	0.78	0.66	0.67	0.70	0.56
Soy	M-RCNN	ImageNet	0.87	0.86	0.74	0.86	0.73	0.68
	M-RCNN	COCO	0.88	0.89	0.76	0.84	0.76	0.63
	YOLOv5x	COCO	0.94	0.91	0.77	0.88	0.77	0.66
	YOLOv5l	COCO	0.88	0.88	0.73	0.81	0.71	0.62
	YOLOv5m	COCO	0.87	0.88	0.71	0.82	0.66	0.63
	YOLOv5s	COCO	0.83	0.81	0.65	0.74	0.64	0.53
	T-YOLOv4	T-YOLOv4	0.81	0.84	0.69	0.76	0.77	0.63
Wheat	M-RCNN	ImageNet	0.89	0.88	0.76	0.84	0.75	0.61
	M-RCNN	COCO	0.89	0.86	0.74	0.81	0.71	0.64
	YOLOv5x	COCO	0.91	0.93	0.79	0.82	0.74	0.63
	YOLOv5l	COCO	0.89	0.92	0.77	0.80	0.72	0.62
	YOLOv5m	COCO	0.86	0.90	0.75	0.82	0.67	0.58
	YOLOv5s	COCO	0.81	0.82	0.71	0.76	0.66	0.57
	T-YOLOv4	T-YOLOv4	0.73	0.77	0.61	0.61	0.65	0.51

Recall$_{50}$ – Recall values at the bounding box IOU threshold of 50% **AP$_{50}$** – Average precision values at the mask IOU threshold of 50% **AP@[.5:.95]** – Mean of AP value from IOU of 50% to 90% with a step size of 5%

T-YOLOv4 – Tiny-YOLOv4

From the results, it is observed:

1. Although the synthetic test data set is created from seeds entirely different than those used for training, the metrics for recall and average precision are significantly higher for the synthetic data set in comparison to the real seed data set.

2. The models trained on both the COCO and ImageNet for Mask R-CNN data sets produce similar results on both the synthetic and the real seed test data sets showing that both the pre-trained weights are as good as each other[A15].

3. Of the YOLO models, YOLO5x outperforms the others in both recall and average precision for each one of the seeds. It is inferred that the accuracy of the network is directly influenced by the number of layers in the network.

4. In terms of the experiment, only synthetic images are used for training. However, all the synthetic images are generated from images of real seeds extracted using the proposed synthetic image generation framework. The number of real seeds used in the generation of synthetic images for each seed type is 30 to 40. The synthetic images are generated by putting the images of real seeds through different image augmentation procedures before pasting them on the desired canvas. Each synthetic image consists of 50 to 100 seeds whose count is assigned at random by the synthetic image generation framework. A large training dataset where each seed type consists of ~150 images are used for training. Based on the results on the test dataset, one might argue that more data is required. However, the training results are satisfactory considering the network exhibits good Recall and Precision on all the seed types. The scope for improvement

perhaps lies in the synthetic image generation framework wherein the seed kernels generated might need to reflect the real-world seeds more closely.

The performance of Tiny-YOLOv4 lags behind the larger Mask R-CNN and YOLO models, except for YOLOv5s on occasion. In all fairness, it is not reasonable to compare it against the larger models considering it is smaller in size architecturally. However, it is worth reiterating that they are fit for mobile applications where the models are required to be compact.

3.2.5 Morphometry Estimation

The focus of the work from this point shifts towards morphometric estimations of the detected seeds. As part of the morphometric measurements, a method which closely follows the No-Mesh algorithm described in chapter-2 and standards laid out by the International Seed Morphology Association, an organization that promotes research in the areas of seed morphology and identification, to estimate the length, width and area of each of the seeds along with the count of seeds on the image is proposed. The proposed technique is dependent upon drawing smallest rotated rectangles around observed contours and works well on both Mask R-CNN and YOLO models for all of the seed types in question. The caveat is that the estimations are dependent solely upon the quality of instance detections made by the networks. In case the network doesn't detect an instance or predicts contours incorrectly, which is not uncommon from the results shown in Table 7, the estimates suffer a lack of accuracy. In order to estimate the morphometry i.e., the area, length and width of each of the seeds in question, the following strategy using OpenCV2 is proposed:

1. Capture an image with multiple US pennies. For this experiment, four pennies are used. It is important to have multiple pennies because it is observed from prior research that the position of the objects on the image impacts their perceived area.

2. Input the image to the trained neural network model and detect the contours of each of the coins on the image and determine the average area of all of the detected coins in metric units. The computation of average area is performed by plotting a rectangle around each of the coin contours that are detected and applying the area of the rectangle in pixels to the area of the coin in mm^2. It is known that the diameter of the US penny is 19.05 mm resulting in an area of 284.87 mm^2 approximating it as a circle. The coins detected and labeled using YOLOv5 are shown in Figure 3.15.

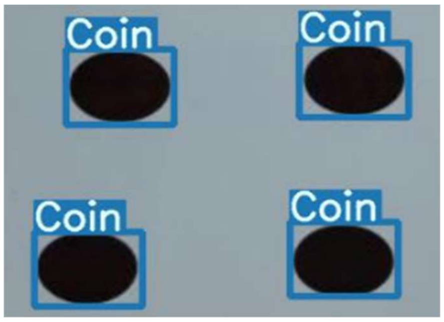


Figure 3.15: Coin Detection by YOLO5x

3. **(Length and Width Estimation)** Likewise, on images with seeds, detect the contours around each of the seeds in the image and plot the smallest rotated rectangle that encloses each of the contours. The longer axis returned is deemed length and the shorter axis, width.

4. **(Area Estimation)** The area of each of contours is given by a cross-multiplication using the contour area of the detected seed in pixels, contour area

of the penny in pixels and area of the penny in mm^2 as $seed\ area\ in\ mm^2 = (seed\ area\ in\ pixels * coin\ area\ in\ mm^2)/\ (median\ coin\ area\ in\ pixels)$.

Figure 3.16 shows the detected instances of sorghum seeds on Mask RCNN along with the smallest enclosing rectangles plotted for size estimation using OpenCV2. Likewise, Figure 3.16 shows the detected instances of wheat amongst four pennies placed at each of the corners of the image.

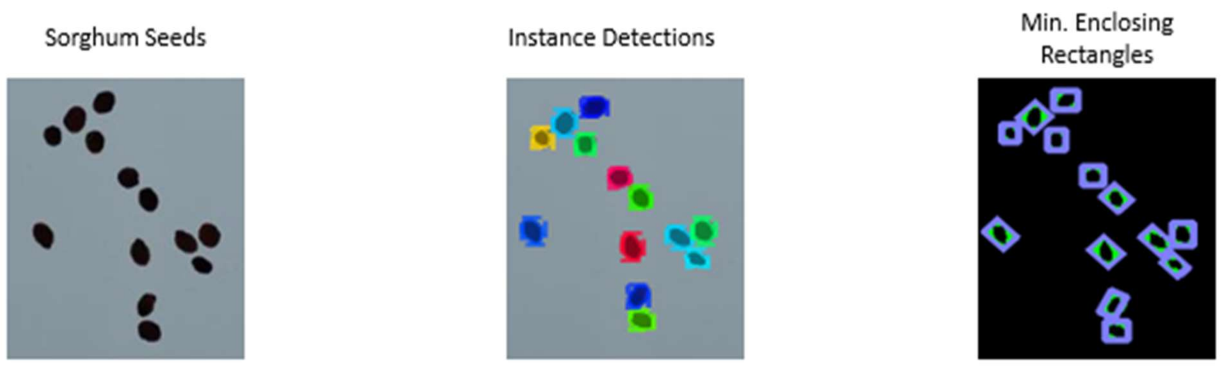


Figure 3.16: Size Estimation Operations on Sorghum Seeds on Mask R-CNN

3.2.6 Estimated Morphometry in Comparison to Other Applications

The morphometric estimations made by the neural networks are compared against the estimations made mobile applications Leaf-IT and Seed Counter which are also in the realm of morphometry estimation. In addition, the estimates are compared against the estimations made by the Mesh algorithm and manual measurements using a vernier caliper. Soy is considered to evaluate for the reason that not all of the seed types in question are

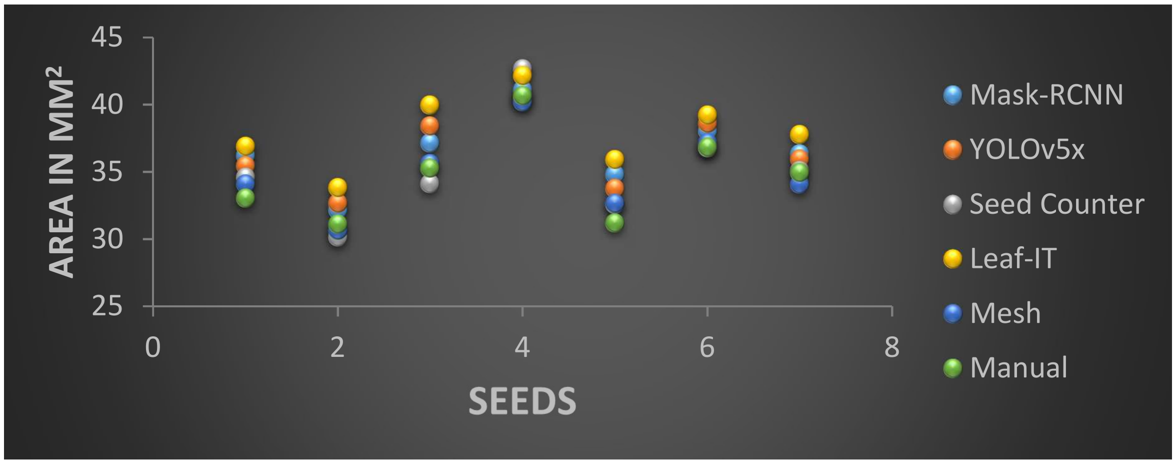


Figure 3.17: Seed Area Estimates using Morphometry Estimation Applications

able to be estimated by all of the applications in consideration. The areas estimated by each of them on a sample of seven soy seeds is as shown in Figure 3.17. Please note that the experiment is repeated on all of the applications with the seed maintained at the same orientation in order to cut out any bias due to orientation and that the seeds do not touch on the images. The type of mesh used for the application of the Mesh algorithm is hexagonal. The results show a high degree of cadence[A16] amongst the estimates made by each of the applications. It shows that the estimates made using neural network detections correlate to the applications that currently prevail[A17].

3.3 Seed Classification on Supervised Neural Networks

Supervised Learning is the technique of training neural networks on labeled data to learn a mapping from an input to a certain output. The technique is popular in the realm of deep learning and is widely leveraged to train neural networks. While the technique may be applied to a neural network of any kind, the current work focuses on the application of supervised learning to convolutional neural networks (CNN) for the classification of seed kernels of different varieties. The CNNs considered for the experiment are the state-of-the-art networks of Oxford's VGG-16, VGG-16 and Microsoft's ResNet-101. The architectures of the networks are described as follows:

3.3.1 VGG-16

VGG-16 is a convolutional neural network developed by researchers Karen Simonyan and Andrew Zimmerman at the University of Oxford. The 16 in the VGG-16 stands for the number of the layers that the architecture consists of. The model achieved 92.7% top-5 test accuracy in ImageNet Large Scale Visual Recognition Challenge (ISLRVC) on ImageNet, a dataset that consists over 14 million images belonging to 1000 classes. The model improved over AlexNet which was the state-of-the-art model at the time. The architecture comprises a stack of five convolutional layers followed by three fully connected layers, as shown in Figure 3.18.

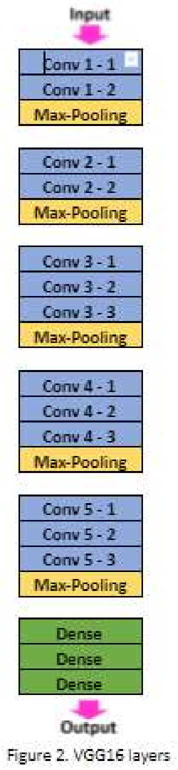


Figure 3.18: Layers of VGG-16 [168]

The input to the first convolutional layer is an RGB image of size 224 x 224 px. Each of the convolutional layers comprises a filter with a receptive field of 3 x 3. Max pooling layers are present after each of the convolutional layers to perform spatial pooling. Max pooling is performed over a 2 x 2-pixel window with a stride of 2. Of the three fully connected layers, the first two have a total of 4096 channels each, and the third dense layer has as many channels as the number of output categories that the network is expected to classify. The final layer is the softmax layer that outputs a probability of an image belonging to a certain class. ReLU is used as the activation function for each of the hidden layers. Figure 3.19 shows the feature map sizes of VGG-16.

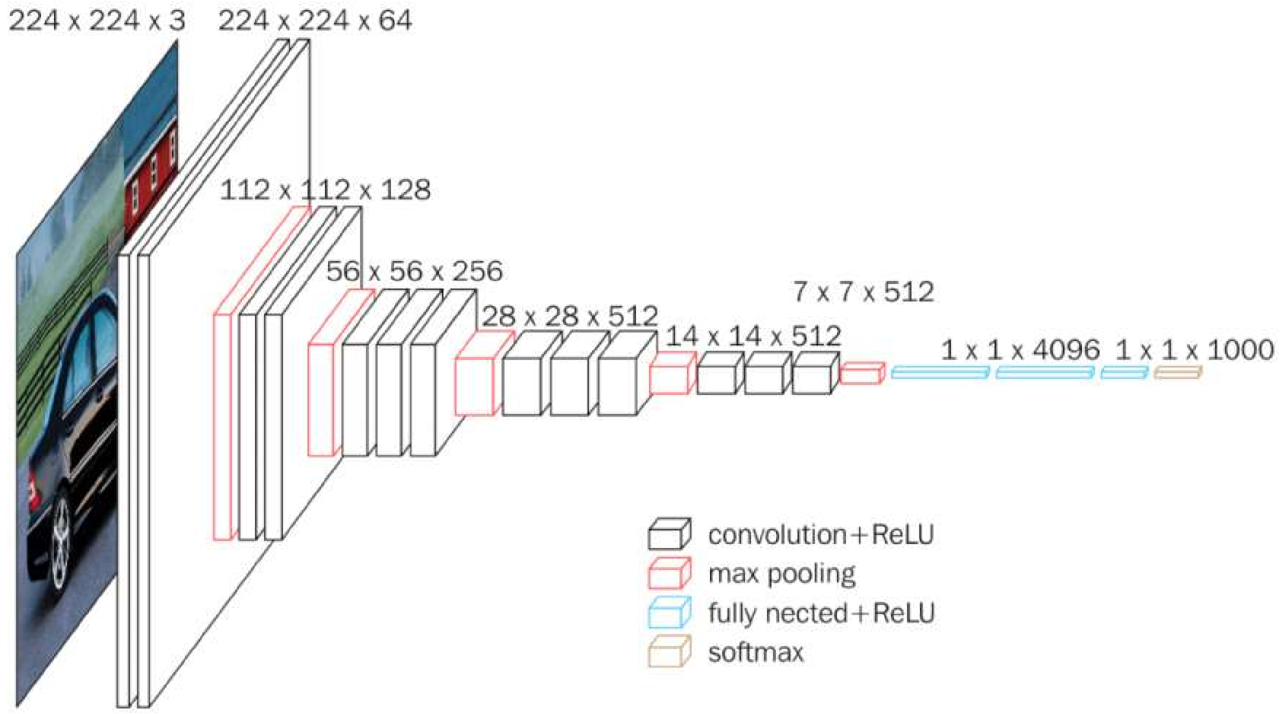


Figure 3.19: Architecture of VGG-16 [168]

3.3.2 VGG-19

The architecture of VGG-19's is similar to that of VGG-16's. The architecture comprises a total of 19 layers in comparison to VGG-16's 16. Like VGG-16, the architecture of VGG-19 comprises a stack of five convolutional layers with max-pooling layers to reduce the spatial dimensions of the output volume. However, a total of 13 convolutional layers are observed in VGG-16 whereas 16 convolutional layers are present in VGG-19. The architecture follows VGG-16 in every other aspect.

3.3.3 ResNet-101

The ResNet architecture relies on the idea of using a skip connection process that performs a convolution transformation F(X) on an incoming feature X and adds the result

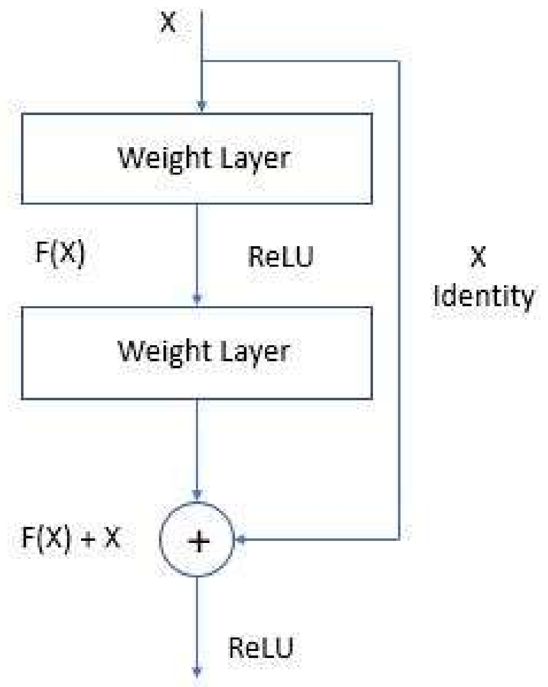


Figure 3.20: Skip Connection in ResNet Architecture

to the original feature X. Figure 3.20 shows a skip connection used in the ResNet architecture.

The modified feature is then served as the input to the next layer. It primarily solves the problem of vanishing gradient i.e. the condition where the loss function shrinks to zero after repeated applications of the chain rule. It is generally the case when the network architectures are overly deep. With ResNets, the network learns the residual elements. There are two main designs of residual elements, skip connection and identity mapping [92]. The training of the residual network may be explained as learning the residual function $Y = F(X) + X$ where the goal is to minimize $F(X)$ so it reaches 0. Applying L2 regularization or weight decay is a way to achieve $F(X) = 0$ since it incentivizes the network weights to be as small as possible. At that point, $Y = X$ i.e. an identity mapping. As a result, the addition of more layers to the network does not hurt the performance of the network. In other words, the presence of multiple residual blocks in the network allows the structure of the residual network to be self-regulated through skip connections thereby achieving the deepening effect of the network [92].

3.3.4 Related Work

The work by Toda et. al. [170] applies the technique of domain randomization to train Mask R-CNN, an instance segmentation neural network on a synthetic image dataset of 20 different cultivars belonging to barley, four cultivars of wheat, and one cultivar each of rice, oat and lettuce. The average precision (AP) values computed based on mask regions at the varying intersection over union (IoU) thresholds achieved comparable AP50 values of 96% and 95% for the synthetic and real-world datasets respectively.

Ward, Moghadam, and Hudson [179] conducted similar experiments to perform automated segmentation of individual leaves of a plant for high-throughput phenotyping. The proposed technique leveraged synthetic images of plants generated from real plant images. The Mask R-CNN architecture was trained with a combination of real and synthetic images of Arabidopsis plants. The proposed technique achieved a 90% leaf segmentation score on the A1 test set and outperformed the state-of-the-art approaches for the CVPPP Leaf Segmentation Challenge (LSC).

Gulzar, Hamid, Soomro, Alwan and Journaux [47] developed a seed classification system for 14 different types of seeds employing the concept of transfer learning on a pre-trained model of VGG-16. The results showed a classification accuracy of 99% on a test image dataset of 234 images.

Buters, Belton, and Cross in [21] conducted experiments to perform seed and seedling detection and classification using unmanned aerial vehicles. The technique of object-based image analysis (OBIA) with eRecognition software used as part of the experiments. The results showed the feasibility of low-cost commercially available UAVs in monitoring ecological recovery.

Coulibaly, Kamsu-Foguem, Kamissako, and Traore [31] proposed a deep learning technique with transfer learning in millet crop images using VGG-16. The goal of the study was to identify mildew disease in pearl millet. Fine-tuning was performed on a pre-trained model of VGG-16 by adding two fully connected layers and training the model on the image dataset. An accuracy of 95%, precision of 90.50%, recall of 94.5% and f1score of 91.75%.

In the study conducted by Mukti and Biswas [110], transfer learning-based plant disease detection was performed using ResNet-50. The dataset included 38 different classes of plant leaf images where in 70295 training images and 17572 validation images were used. Fine-tuning was performed to enhance the performance of the ResNet-50 model that eventually yielded a training accuracy of 99.80%. 33 images belonging to the 38 classes of leaves were used for testing. The proposed model yielded an accuracy of 100% on the test dataset.

Muneer and Fati [111] proposed a deep learning technique to classify and recognize the desired herb from thousands of herbs using shape and texture features. The proposed system employed two classifiers, Support Vector Machine (SVM) and Deep Learning Neural Network (DLNN). The models were tested on a dataset containing 1000 herbs. The results showed that the SVM model achieved a recognition accuracy of 74.63% whereas the DLNN achieved 93% accuracy. Furthermore, the processing time was four seconds for SVM and five seconds for DLNN.

A deep-learning classification system for identifying weeds using high-resolution UAV imagery was proposed by Bah, Dericquebourg, Hafiane, and Canals in [14]. The proposed system was applied to images of vegetables captured about 20m above the soil using a UAV. The results showed that weed detection was effective in different crop fields with overall precisions of 93%, 81%, and 69% obtained for beet, spinach, and bean respectively.

Bristeau, Vissière, Callou and Petit [19] discuss the navigation and control technology embedded in the Parrot AR Drone. The article sheds light on the low-cost inertial sensors, computer vision techniques, attitude and velocity estimation, and control architecture used in the UAV.

3.3.5 Experimental Setup

Each of the neural network architectures is trained on synthetic images [A18]generated using the Domain Randomization framework described in section 3.1. One of the observations from chapter 2 is that the height of image capture impacts seed classification i.e., an object in question appears different from different heights as the field of view of the camera changes. In order to avoid the issue of misclassification due to height of image capture, the neural networks are trained on images captured from varying heights. However, the manual capture of images from different heights is a tedious and error prone. An image captured freehand results in skew since it is hard to hold one's hand perfectly orthogonal to the ground. Also, from experimentation, it is realized that is more than a one-person job. Hence, the use of an unmanned aerial vehicle (UAV) aka drone is considered to capture the images.

3.3.5.1 Parrot AR Drone 2.0

The Parrot AR Drone 2.0 is a quadcopter manufactured by the French company Parrot SA. The drone uses a 3630 OMAP CPU, a processor that is based upon a 32-bit ARM cortex and runs with 1 GHz. The drone uses 4 brushless motors running at 28.5 revolutions/minute which are controlled by an 8 MIPS AVR CPU on each motor controller. The drone has a front and a bottom camera to capture images and videos. The front camera provides an HD image resolution (720p-30fps) whereas the bottom camera provides QVGA (320 x 240) at 60fps. The drone may be controlled via a mobile application supported on Android and iOS provided by Parrot SA. Besides the mobile app, the quadcopter has support for autonomous navigation through libraries supported in Robot Operating System (ROS) and nodeJS. It turns out that JavaScript is a good fit to control drones autonomously for the work since it is inherently event driven. The nodeJS library, node-ar-drone is leveraged to operate the drone autonomously in the breeding environment. While the node-ar-drone library provides a plethora of functions to operate the quadcopter, the ones used for the current work are take-off, land, spin, and video capture and stop. The behavior of the functions is self-intuitive from the names of the functions. The video captured by the drone is output in .h264 format when the video capture function of the node-ar-drone library is used. Videos in raw .h264 format don't carry rate information. However, different frames of the video are required to be extracted for the creation of synthetic images. As a work-around, the FFmpeg library is used to convert the video in .H264 format to .mp4 format rendering it conducive for frame extraction using the Python-OpenCV framework. The UAV is flown across a prototypical seed phenotyping environment where in a sample of each of the seed varieties is placed on a lightbox. Figure 3.21 shows the Parrot AR Drone 2.0 flying across the prototypical seed phenotyping environment.

3.3.5.2 Synthetic Image Dataset

The seeds of canola, rough rice, sorghum, soy, and wheat are used as part of the experiment. 30 seeds belonging to each of the seed types in question i.e., canola, rough rice, sorghum, soy, and wheat are randomly selected from a large pool of available seeds belonging to each of the seed types and placed in the prototypical seed phenotyping environment. The Parrot AR Drone 2.0 is flown over the phenotyping environment autonomously and made to capture a video of the seeds as it moves across the seed phenotyping environment using its bottom camera. The drone is made to fly at three different heights of 0.3m, 0.5m, and 0.7m to capture three different videos. Four images of the white light-emitting lightbox are used as the canvas (background) for the synthetic images. While it is fair to assume that the lightbox looks the same across the board while emitting white light, it is not the case and minor discrepancies in the intensity are observed amongst lightboxes. As a result, using the images of different lightboxes ensures that minor discrepancies are captured during the creation of the synthetic dataset. Each of the videos

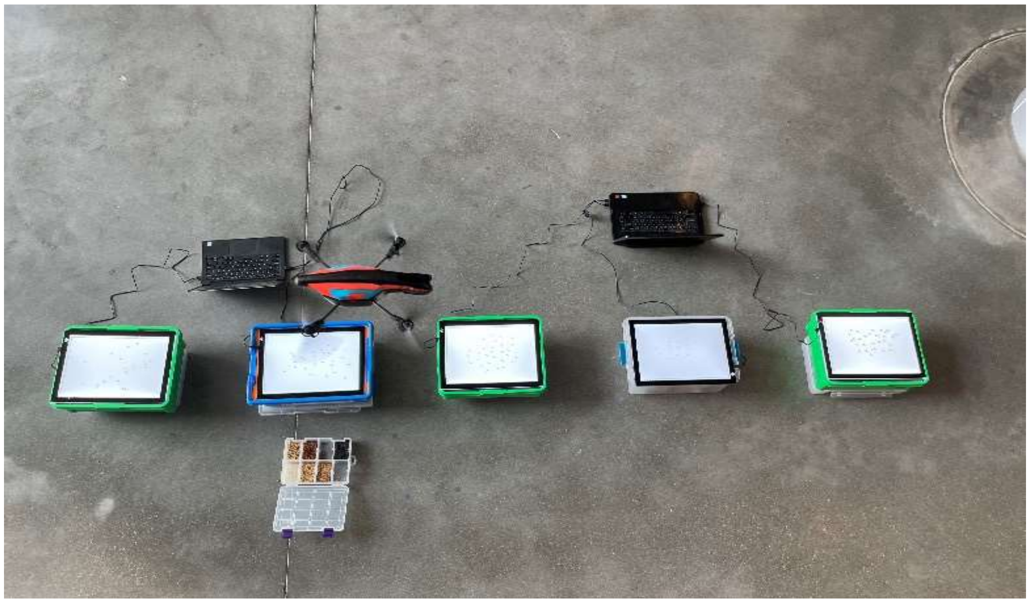


Figure 3.21: Image Capture of Seeds in Phenotyping Environment using Parrot AR Drone 2.0

is processed using the Python-OpenCV framework described in section 3.1 to create synthetic images for each of the seeds in question. The synthetic images generated are of the size 224 x 224 x3 to match the input size for the neural networks used to train on the dataset. For each of the seeds, 1000 synthetic images are generated of which the images consist of 20 - 50 seeds each. The dataset is made diverse by the fact that the images consist of seeds captured from different heights i.e. the 1000 images of a single seed comprise 333 images where the seeds are captured from a height of 0.3m, 333 images where the seeds are captured from a height of 0.5m and 334 images where the seeds are captured from a height of 0.7m. Overall, a total of 5000 images are generated amongst all of the seeds. One of the struggles in classifying multiple classes of seeds is the appearance of seeds at different heights. It is not an issue in cases where the images are captured from a fixed height. However, it is not always feasible whilst using drones since they are mobile and are influenced by environmental factors such as wind and obstacles. It is observed that a given seed appears smaller from larger heights and vice-versa. Owing to this, the training dataset consisting of all images from a fixed height may not be a representative sample of the real-world test set that the neural network might encounter. Hence, the drone is programmed to capture images from varying heights of 0.3m, 0.5m, and 0.7m to ensure that a real-world representative sample of seeds is obtained. Figure 3.22 shows the synthetic images generated for the seed types of canola, rough rice, sorghum, soy and wheat.

3.3.6 Experiment and Results

The three neural network architectures are imported from the canned architectures provided by Keras as part of their Applications module. The imported models are models pre-trained on the ImageNet dataset that comprises 1000 classes. The key idea behind the use of pre-trained models trained on ImageNet is that the notion of Transfer Learning may apply. Briefly, Transfer Learning is the ability of a neural network to apply the knowledge gained by training on a dataset to a different dataset where there is a presence of common domains between the source and target datasets. However, the ImageNet dataset does not consist of the five seeds or anything remotely similar that are of interest for the experiment. In cases where the source and target domains don't share domains, the sheer transfer of knowledge is usually unsuccessful and often results in poor performance. Since the neural network architectures are large, training the entire neural network from scratch requires an exorbitant amount of data. As a means to better train the network whilst preserving the learned features from the ImageNet dataset, the architecture of the neural networks is

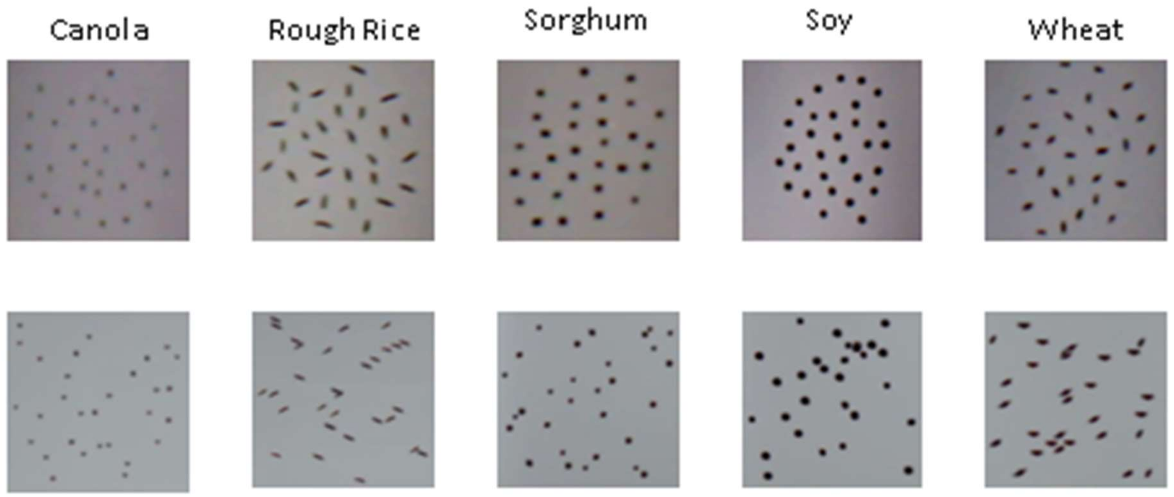


Figure 3.22: Top row shows real seeds captured by the UAV; Bottom row shows synthetic images generated by the DR Framework using the real seeds

augmented. As for augmented architectures, three Sequential models (one for each of VGG-16, VGG-19, and ResNet-101) are built from the imported models wherein all of the layers except the fully connected layers and softmax layer are copied to the Sequential models. The VGG models consist of three fully connected layers whereas ResNet-101 has one. Three fully connected layers are added to the Sequential models followed by a softmax layer for classification. All of the transferred layers from the pre-trained models are rendered frozen for training and only the three fully connected layers added to the model are trained. The dataset used for the experiment is the synthetic image dataset described in section 3.3.2.2. 80% of the dataset is used for training and 20% of the dataset is used for validation. Experiments with different hyperparameters are conducted in the training phase. Upon training each model with a different set of hyperparameters, the models that yield the best accuracy and loss are saved. The hyperparameters used for each of the best models are as shown in Table 3.2.

Table 3.2: Hyperparameter Values for Training

Hyperparameter	**VGG-16**	**VGG-19**	**ResNet-101**
Nodes per trained layer	512	512	1024
Learning Rate	1 x e^{-3}	1 x e^{-3}	1 x e^{-2}
Learning Rate Decay	100 steps @0.96	100 steps @0.96	No Decay
Dropout	0.4	0.5	0.5
Batch Size	32	32	32
Optimizer	Adam	SGD	Adam

The training accuracies and losses of the best models for the fine-tuned neural networks are as shown in Figure 3.23. Overfitting is commonly observed across all three of the models. Overfitting refers to the situation where the neural network learns the training data too well to the point it does not generalize to other data during validation. The use of Dropout helped with reducing the overfitting of the models. However, it results in the validation accuracy being higher than training accuracy in parts[A19]. Overall, the validation accuracy for each of the fine-tuned models of VGG-16, VGG-19, and ResNet-101 at the end of the training phase is 96.43%, 90.03%, and 97.81% respectively.

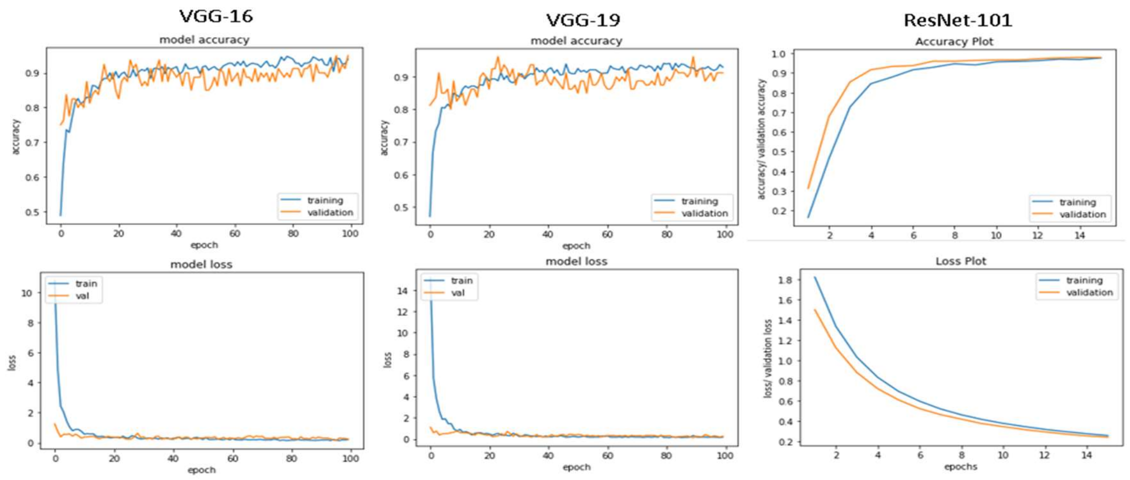


Figure 3.23: Training and Validation Accuracies and Losses for Fine-tuned VGG-16, VGG-19 and ResNet-101

The images for the test dataset consist of 75 images containing 20 – 50 seeds of a certain seed type captured from varying heights. Overall, a test dataset consisting of 450 images is used to evaluate the performance of each of the networks. The results of evaluation based on the metrics of Accuracy, Precision, and Recall are as shown in Table 3.3. The metrics are briefly described below:

Accuracy: Accuracy is defined as the fraction of the samples in the dataset correctly classified by the classifier. It is given by $(true\ positives + true\ negatives)/ (true\ positives + true\ negatives + false\ positives + false\ negatives)$.

Precision: Precision is given by $true\ positives/\ (true\ positives + false\ positives)$. Precision indicates the number of positives correctly classified by the classifier.

Recall: Recall is defined as the True Positive Rate of a classifier and given by $true\ positives/\ (true\ positives + false\ negatives)$. Recall indicates the number of positives detected by the classifier of all the positives in the dataset.

The results show that ResNet-101 lends itself best to the task of classification achieving an overall accuracy of 92% while VGG-16 and VGG-19 lag behind at 89% and 86% respectively. However,

the performance of all of the models is sub-par in comparison to the accuracy on the validation set during training.

3.3.7 Ensemble Model

Ensemble Learning refers to the generation and combination of multiple inducers to solve a machine learning task [144]. In an attempt to better the classification

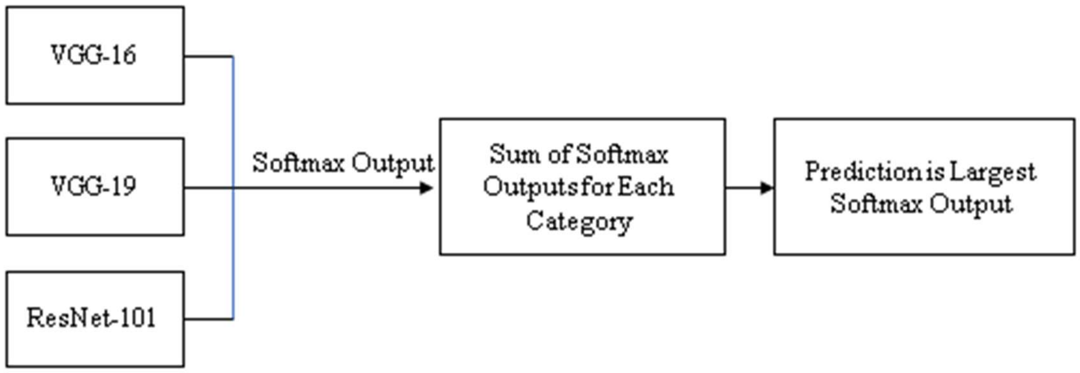

VGG-16
VGG-19
ResNet-101
Softmax Output
Sum of Softmax Outputs for Each Category
Prediction is Largest Softmax Output

Figure 3.24: Workflow of Ensemble Model

performance, an ensemble whose predictions are based on the individual predictions made by each of the models is developed.

Table 3.3: Evaluation Results on the Fine-Tuned Models of VGG-16, VGG-19, and ResNet-101

	VGG-16			**VGG-19**			**ResNet-101**		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Canola	0.91	0.76	0.79	0.86	0.67	0.67	0.91	0.78	0.78
Rough Rice	0.91	0.78	0.78	0.90	0.77	0.80	0.93	0.81	0.87
Sorghum	0.90	0.70	0.77	0.75	0.66	0.60	0.91	0.78	0.77
Soy	0.90	0.75	0.80	0.90	0.74	0.77	0.93	0.88	0.81
Wheat	0.84	0.78	0.76	0.89	0.73	0.75	0.94	0.85	0.87
Overall	0.89			0.86			0.92		

The ensemble model works by summing the categorical softmax outputs of each of the models for a given sample. The prediction made by the model is the largest resultant categorical value. The workflow of the ensemble model is as shown in Figure 3.24 and the results obtained are as shown in table 3.4. The results show an improvement in overall accuracy to 94.6% showing a 2.6% improvement in accuracy compared to the best individual model of ResNet-101. Besides, a significant improvement in the precision and recall for each of the individual classes is also observed which indicates better quality of predictions.

Table 3.4: Evaluation Results on Ensemble Model

Seed Type	**Ensemble Model**		
	Accuracy	Precision	Recall
Canola	0.94	0.84	0.90
Rough Rice	0.95	0.88	0.89
Sorghum	0.94	0.91	0.82
Soy	0.95	0.83	0.96
Wheat	0.95	0.88	0.90
Overall	0.94		

3.4 Seed Classification on Self-Supervised Learning Frameworks

Self-Supervised learning refers to the technique of training neural networks where the training data is labeled automatically [18]. Self- supervised learning consists of the traits of both supervised and unsupervised learnings. It is supervised in the sense that the model is still trained to learn a function from pairs of inputs and labeled outputs. It is unsupervised in the sense that the model learns without being provided with labels. Contrastive Self-Supervised learning falls under the broader umbrella of self-supervised learning where the intuition is to learn the features of a dataset by a measure of distinctiveness. While labels are absent, the model learns the similarity between the datapoints in the dataset and groups them together. This process results in obtaining groups of similar datapoints, yielding in classification.

Three frameworks namely, SimCLR, Momentum Contrast (MoCo) and Build Your Own Latent (BYOL) are used as part of the experiments. The architecture and working of each of the frameworks is as described further.

A. SimCLR

SimCLR functions by maximizing the similarity measure between two augmented views obtained from the same image. The framework is described as a three-step process:

1. **Data Augmentation:** Given an input image, two correlated views of the image are generated using three different transformations namely, random crop and resize, horizontal flip and color distortion involving jitter and grayscale conversion. This process helps in generating different views of the same image aiding the network in its ability to be transformation independent.
2. **Neural Network Encoder:** The augmented images are put through an encoder such as ResNet-18 or ResNet-50 and vector encodings of the images are extracted.
3. **Projection Head:** A neural network encoder such as ResNet-50 generates encodings that are 2048 dimensional. In order to reduce the dimensionality of the vector encodings, a multi-layer perceptron (MLP) with one hidden layer is used. The encodings in high dimensional space are mapped to 128-dimensional latent space upon which contrastive loss is applied. The activation function used in the MLP is ReLU. The architecture of SimCLR is as shown in Figure 3.25.

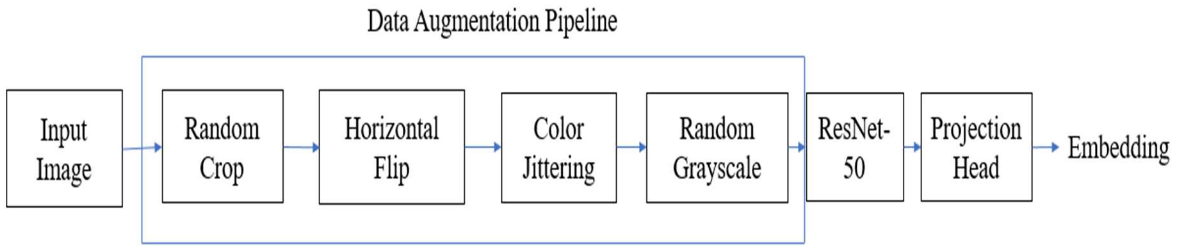


Figure 3.25: SimCLR Architecture

Contrastive Loss: Given a set of augmented images that contains two correlated views for each of the images in the set, the contrastive loss functions aims to identify each pair of correlated views that belong to the same image. The vector encodings in 128-dimensional latent space are run through the contrastive loss function that takes similarity of the images into account. The similarity between two images is computed using a measure named Cosine Similarity. It is defined as $sim(x,y) = (x^t y)/|x||y|$ where x and y are two different vector encodings. The contrastive loss function that takes cosine similarity into account is defined as:

$$L_{i,j} = -log \frac{e^{\frac{sim(z_i,z_j)}{\tau}}}{\sum_{k=1}^{2N} e^{\frac{sim(z_i,z_k)}{\tau}}_{k \neq i}}$$

$Z = \{z_1, z_2, \ldots, z_k\} \in \mathbb{R}^k$ are output vectors from the projection head. In the event that two vectors are similar, the function yields a result of zero which is the optimal loss. τ is the temperature parameter used to scale the cosine similarities. Reference [14] found that the optimal temperature parameter helps the model learn from hard negatives. It is worth noting that the value of τ depends on the number of epochs and batch size during training. Given a batch of N images, the augmentation results in each image having two representations yielding a total of 2N augmented images. The positive pair is the pair of images that are a result of augmentation from the same image and every other pair in the set is considered a negative pair. The augmented image set consists of one positive pair and 2(N – 1) negative pairs. The framework relies on the presence of large models and batch sizes to achieve better accuracy. This requirement is also deemed a bottleneck for the framework because the increase in batch and model sizes have a direct impact on the number of computational resources required to train the network.

B. MoCo

Momentum Contrast (MoCo) is a technique similar to SimCLR with the key improvement being that it eliminates the need for large models and batch sizes employed by SimCLR. There is one key difference that sets SimCLR and MoCo apart from each other; the use of two neural network encoders. MoCo functions on the principle of matching queries to keys where key and query refer to the encodings of an augmented image. As opposed to the idea of having a single neural network encoder in SimCLR, a second neural network encoder, similar to the first is introduced where one of them generates the encodings for the key and the other, query. MoCo constructs a dictionary built and operated as a queue that keeps a history of the encoded keys. The dictionary in turn, acts as the resource pool for positive and negative pairs of images. A positive pair refers to the instance where a query matches the key. Every other sample in the dictionary that doesn't correspond to the query acts as a negative sample. One of the key challenges of the approach is that, learning the parameters of the key encoder requires calculating the gradients of each of the samples in the queue. The larger the number of samples, the greater the number of computational resources required. In order to address the issue, MoCo updates the key encoder with the momentum-based average of the query encoder. It is defined as $\theta_k \leftarrow m\theta_k + (1-m)\theta_q$ where θ_k and θ_q are the parameters of key and query

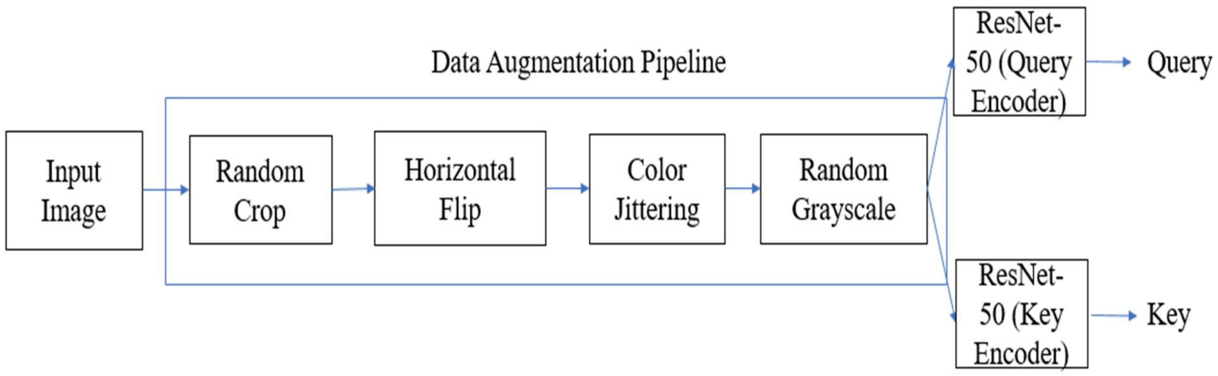


Figure 3.26: Momentum Contrast Architecture

encoders respectively, m is the momentum whose value is kept close to one. The architecture of MoCo is as shown in Figure 3.26.

C. Bootstrap Your Own Latent (BYOL)

BYOL is a technique similar to SimCLR and MoCo with one key difference. BYOL does not take negative samples into account. Instead, BYOL focuses solely on ensuring that similar samples have similar representations. While it might not be apparent at first glance as to the reason it is significant, it is important to understand that BYOL avoids the collapsed representation problem without negative samples. Collapsed representation is the state wherein a network trained only on similar pairs learns a constant function since the loss output over similar pairs is always a constant, such as zero. No discriminative features are learnt rendering the network unfit for prediction on a test dataset or fine-tuning on a different dataset. The working of BYOL is described as follows:

1. **Target and Online Networks:** Consider two encoders with the same architecture, generally ResNet-50, where in one of them is randomly initialized to be the 'target' network and another set to be trainable known as the 'online' network.
2. **Data Augmentation:** Pass an input image 'I' through the data augmentation pipeline to generate two stochastically augmented views 'I1' and 'I2'.
3. **Vector Encoding Extraction:** Run the augmented views 'I1' and 'I2' through the 'target' and 'online' networks respectively and extract the vector representations of the views
4. **Projection Head:** Use an MLP to reduce the vector encodings to a lower dimensional latent space, usually 256.

5. **Prediction:** The vector encoding from the 'online' network is used to predict the vector encoding from the 'target' network i.e., the distance between the vector encodings is minimized using the normalized mean squared error loss function.
6. **Update Target Network:** The 'target' network is updated at the end of each training step as the exponential moving average of the parameters of the 'online' network. The 'target' network closely follows yet lags behind the 'online' network at all times.

In essence, BYOL attempts to leverage the 'online' network in one training step as the 'target' network in the subsequent training step. Doing so avoids the collapsed representation problem and also betters the performance of the network. The architecture is as shown in Figure 3.27.

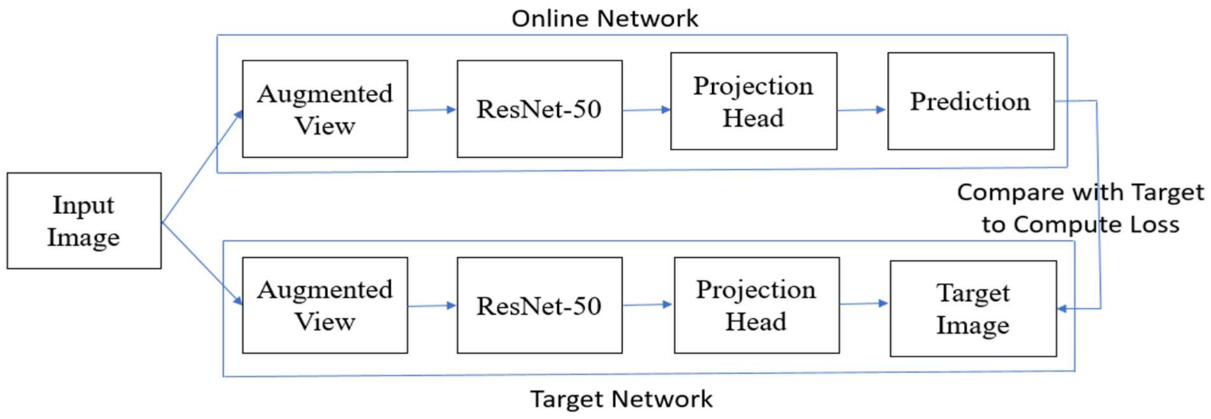


Figure 3.27: BYOL Architecture

3.4.1 Related Work

Relevant research involving a combination of domain randomization and self-supervised learning is not abundant. However, works that employ domain randomization and self-supervised learning independently are available. In terms of training neural network models using synthetic datasets, Toda et. al [170] applied synthetic datasets to

Mask RCNN, an instance detection framework, to perform instance segmentation and detection of seeds. Gulzar et. al [47] developed a seed classification system for 14 different types of seeds by applying transfer learning on VGG-16, a convolutional neural network. In the realm of contrastive self-supervised learning frameworks, Chen et. al [68] proposed the SimCLR framework, a major breakthrough in terms of contrastive self-supervised learning frameworks in the sense that don't require a memory bank. The framework relies on the presence of large batches of training data. The work demonstrated the critical role played by the composition of data augmentations on contrastive learning. He et. al [150] proposed the Momentum Contrast (MoCo) framework for unsupervised visual representation learning that approached contrastive learning as a dictionary look-up problem. A dynamic dictionary with a queue and a moving-averaged encoder is built on-the-fly to facilitate contrastive unsupervised learning. Chen et. al [26] proposed a modified framework to MoCo by using a multi-layer perceptron projection head and more data augmentation that outperformed SimCLR without the need for large training batches. Grill et. al [46] introduced the Bootstrap Your Own Latent (BYOL) approach to self-supervised image representation learning. BYOL works by the use of an online and a target neural network. The principle is to train the online network on an augmented view of an image to predict the target network representation of the image that is a different augmented view.

3.4.2 Dataset and Method

A. Dataset

The seeds of canola, rough rice, sorghum, soy and wheat are used as part of the experiment. Since the work attempts to study the feasibility of domain randomization on self-supervised learning frameworks, the images used for the training and validation datasets are synthetic images generated using a representative sample of each of the seeds. A large seed pool of the order of several hundred is available considering the nature of research conducted in the laboratory is primarily focused on agriculture. 30 seeds belonging to each of the seed types in consideration are chosen at random and photographed individually resulting in a sample of 150 images. In accordance with the technique outlined in section 3.1, images containing seeds overlaid on a common background are generated where in each image contains 50 seeds of a certain seed type as shown in Figure 3.28.

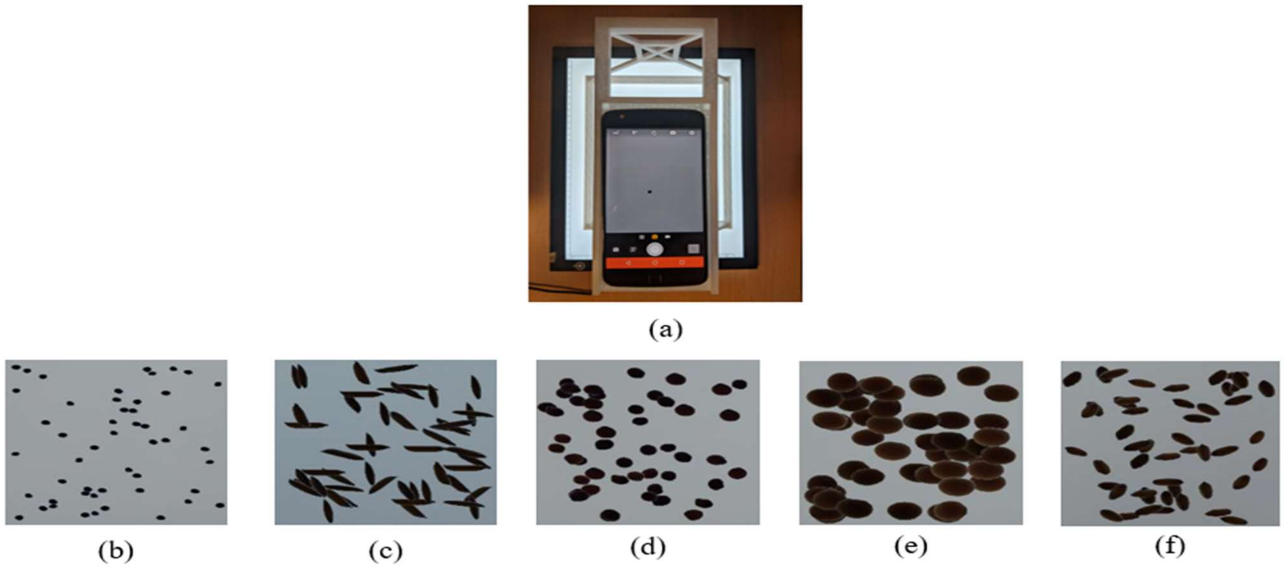


Figure 3.28: (a) Image capture of soy seed on lightbox; Bottom row shows synthetic images of (b) canola (c) rough rice (d) sorghum (e) soy (f) wheat

1000 images per seed type, each 224 x 224 x 3 in size, are generated of which 800 are used for training and 200 for validation. As for the test dataset, 40 images of each seed type are photographed where in each of the images contains ~50 seeds of the same seed type placed on a lightbox. It is ensured that the seeds in the test dataset are not the seeds used to compile the synthetic image dataset. Overall, the training dataset comprises of 5000 images while the test dataset is made up of 200 images.

B. Experimental Setup

ResNet-50 is used as the encoder for each of the self-supervised learning frameworks in consideration, SimCLR, MoCo and BYOL. The implementation of the frameworks is done in PyTorch and executed on a Tesla V100 GPU provided by Google Colab Pro. The lr-finder module is used to find the learning rate that best fits a model for fine-tuning.

C. SimCLR, MoCo and BYOL

Each of the frameworks, SimCLR, MoCo and BYOL, is implemented as described in section 3.3 (A – C) using a ResNet-50 model pre-trained on the ImageNet dataset as encoder. While there are several variants of ResNet such as ResNet-34, ResNet-50, ResNet-101 and ResNet-152, the reason behind selecting ResNet-50 as the encoder is its performance on the ImageNet validation dataset. The error rates (%) of single-model results on the ImageNet validation set are as shown in Table 3.5 [64].

Table 3.5: Error Rates (%) of ResNet Models on ImageNet Validation Set

Model	Top-1 Error	Top-5 Error
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	19.38	4.49

The results show that an increase in the number of layers has a direct improvement on the error rate. ResNet-50 provides a happy medium between error rate and network size. Hence, ResNet-50 is the network of choice to compare against the self-supervised neural networks. Each of the frameworks is trained for a total of 50 epochs at a learning rate of 1e^{-3} on the training dataset. Weight Decay, a regularization technique, set to 1e^{-4} is added to the models to prevent overfitting. It is worth noting that the training at this stage does not involve any labels and is solely to minimize the loss functions of each of the frameworks. The implementation details specific to each of the frameworks are described further.

SimCLR: A projection head with Linear-ReLU-Linear layers that yields 128-dimensional latent space vectors as output is added on top of the ResNet-50 model because [26] shows that the framework delivers its best performance under a non-linear projection head. A batch size of 192 is used while the network is trained. Upon training the network, the non-linear projection head is removed and each of the layers in the network is frozen. A linear classifier with one layer that predicts the class of an image is added to the network.

MoCo: The projection head of MoCo's is similar to that of SimCLR's and contains Linear-ReLU-Linear layers. The projection head outputs 128-dimensional latent space vectors. A dictionary of size 256 is used to keep track of the encoded keys in the 'target' network. Upon training the model, the projection head is removed, and a linear classifier is added on top of the model.

BYOL: The projection head is made up of two linear layers where in an input vector of 2048 dimensions is taken, projected to 4096 dimensions and then, a vector reduced to 256 dimensions is output.

3.4.3 Results and Discussion

Predictions are made on SimCLR, MoCo and BYOL upon training the networks as described earlier. The accuracies obtained by the models is as shown in Table 3.6.

Table 3.6: Self-Supervised Framework Accuracies

Framework	Test Accuracy
SimCLR	33%
MoCo	39%
BYOL	26%

It is apparent from the results that the models do not generalize well to the real-world images in the test dataset. In order to enhance the performance of the models, linear classifiers are built on top of the self-supervised models and trained for 100 epochs on 5%

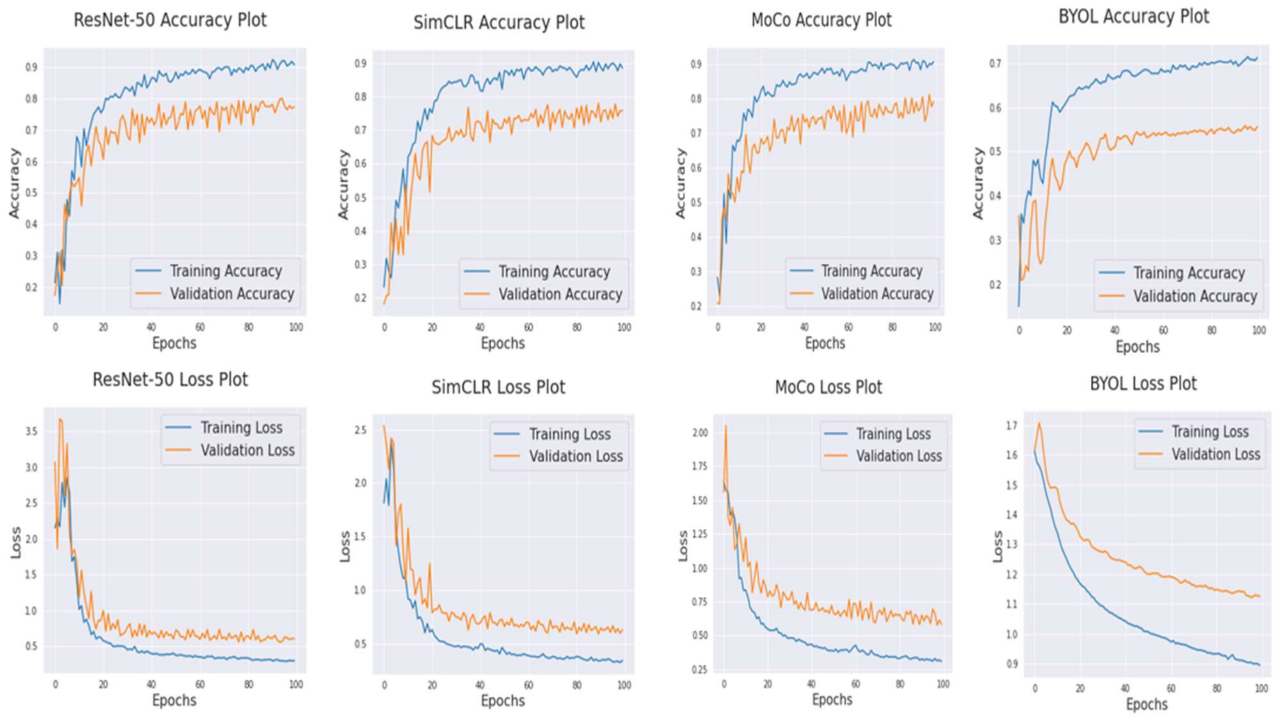


Figure 3.29: Training and Validation Accuracies and Losses of ResNet-50, SimCLR, MoCo and BYOL

of the labels i.e., 250 images comprising of 50 images of each seed type. 40 images of each seed type are used for training and 10 for validation. Supervised training is performed on ResNet-50 on the entire 5000 image training dataset for a total of 100 epochs. The accuracies and losses obtained by each of the models during training and validation are as shown in Figure 3.29. Amongst the self-supervised learning frameworks, SimCLR and MoCo perform almost identically. It isn’t too surprising considering the architectures of the models are similar in nature. SimCLR and MoCo attain a validation accuracy of ~80% and loss of ~0.6. It is interesting to observe that their validation statistics are highly coherent with that of ResNet-50’s. ResNet-50 also achieves a validation accuracy of ~80% and loss of ~0.6. However, the same is not true of BYOL. The model achieves a validation accuracy of ~55% and loss of ~1.1, performing well below SimCLR and MoCo. The total lack of negative examples might be the factor that contributes heavily to such a behavior. The models trained on linear classifiers are evaluated on the test dataset.

The classification report containing precision, recall and F-1 score for each of the classes generated using Scikit are as tabulated in Table 3.7. The terms Precision, Recall, F1-Score, Accuracy and Macro Average are briefly described below:

Precision: Given by $true\ positives/\ (true\ positives + false\ positives)$, precision indicates the number of positives correctly classified by the classifier. It is also called False Positive Rate.

Recall: Given by $true\ positives/\ (true\ positives + false\ negatives)$, recall indicates the number of positives identified by the model of all the positives in the dataset. It is also called True Positive Rate.

Table 3.7: **Classification Reports of ResNet-50, SimCLR, MoCo and BYOL**

	ResNet-50			**SimCLR**			**MoCo**			**BYOL**		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Scor
Canola	0.96	1.00	0.98	0.60	1.00	0.75	0.83	1.00	0.91	0.65	0.95	0.77
Rough Rice	0.85	0.94	0.89	0.83	0.39	0.53	0.98	0.75	0.85	0.62	0.46	0.52
Sorghum	0.79	0.87	0.83	1.00	0.04	0.08	0.79	0.43	0.56	0.29	0.04	0.07
Soy	1.00	0.98	0.99	0.77	0.82	0.80	0.98	0.91	0.95	0.77	0.89	0.82
Wheat	0.83	0.62	0.71	0.32	0.63	0.43	0.43	0.78	0.55	0.43	0.64	0.51
Accuracy			0.90			0.56			0.77			0.58
Macro Avg.	0.89	0.88	0.88	0.70	0.58	0.51	0.80	0.78	0.76	0.55	0.60	0.54

F1-Score: Considering both precision and recall, F1-Score is the harmonic mean of precision and recall. It is given by the formula, $2 * \frac{precision * recall}{precision + recall}$.

Accuracy: Accuracy is given by the sum of true positives and true negatives to the sum of true positives, true negatives, false positives and false negatives.

Macro Average: Macro Average is simply the arithmetic mean of the values for each of the classes in the dataset. It treats the contribution of each of the classes in the dataset equally.

Unsurprisingly, ResNet-50 achieves the best performance of the four models in question and establishes a steep benchmark for the self-supervised learning models to meet. Amongst the self-supervised models, MoCo is the standout winner with a macro average F1-Score that is significantly higher than SimCLR's and BYOL's. A high macro average

F1-Score indicates that the model delivers good precision and recall across the five classes in the dataset. The performance of the models on the test dataset is in-line with the performance on the training dataset except for SimCLR. SimCLR’s validation accuracy closely follows that of MoCo’s. However, the performance does not translate onto the test dataset with a macro average F1-Score of 0.51 in comparison to BYOL’s 0.54, the model with the lowest validation accuracy. While achieving a macro average precision of 0.70 that is not too far from MoCo’s 0.80, the macro average recall is a mere 0.58 in comparison to MoCo’s 0.78 pulling down the F1-Score significantly. As for the individual seed types, the best F1-Scores are observed on canola and soy across the board with sorghum and wheat lying at the tail end. This is interesting considering the shape of the seeds. The seeds of canola, sorghum and soy are circular in nature while those of rough rice and wheat are oblong. Considering the models of SimCLR and BYOL on sorghum, a high precision and low recall is observed which indicates that the model does a good job of identifying sorghum when it does but does not think a lot of the sorghum seeds are actually sorghum. The majority of sorghum either get classified as canola or soy. While better than the performance on sorghum, a similar characteristic is observed between rough rice and wheat. Self-supervised learning frameworks have a sense of reliance on color histograms. The intuition is that the color histograms of random crops of the same image are similar and different images are dissimilar. However, random color jittering applied to images occasionally results in multiple images from different classes having similar histograms. Stark similarities between histograms of images belonging to different classes leads to error in classification. Figure 3.30 shows one such case where similar histograms are obtained on color jittered images of canola and rough rice. While the images belong to two entirely different classes, the root mean square difference between the two histograms is 35.6, low enough to group both histograms in the same category. It is hard to predict the frequency of occurrence of such similarities since the data augmentation pipeline is random, but it is almost certain that they will occur and hinder classification. Considering the fact that every class in the dataset is a seed type and the seeds are similar to a degree, the dataset may be considered tough for self-supervised learning. In addition, the aspect of using synthetic datasets for training means that the models are supposed to learn from datasets that are not real-world. Owing to such convolutions, MoCo, the best model observed, achieves an

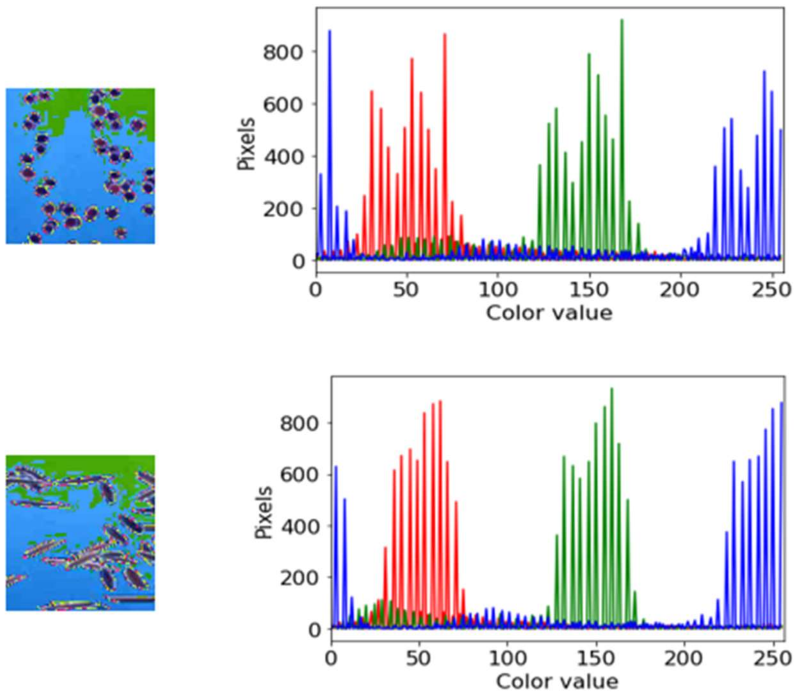


Figure 3.30: Similar Color Histograms of Canola and Rough Rice

accuracy of 77% which is only 13% shy of ResNet-50's 90% whilst being trained on a dataset that is 1/20th the size.

3.5 Future Work and Conclusions

Overall, the performance of the domain randomization framework demonstrates that synthetic image datasets are a great alternative to high-throughput phenotyping applications in the absence of real-world datasets. One of the self-criticisms of the work is that the experiments conducted result in the neural networks being overly reliant on the size and shape of the seed kernel but doesn't consider the color of the seed kernel. For instance, the seed kernels of soy are yellow whereas the seed kernels of wheat are reddish-brown. However, the current experiment captures the images of seed kernels placed on a light box to ensure an illuminous background. The downside to the practice is that the seed varieties appear to be the same color. Hence, the network relies on the size and shape of the seed kernels primarily to distinguish one type from another. As the size of broken seed kernels of a certain variety could resemble a different seed type, the models could fail to correctly identify broken seed kernels. Moving forward, the experimental setup which is illuminous and captures the color of the seed kernel properly will be developed to efficiently leverage color information. More experiments to determine the feasibility of the neural network models on broken seeds will be conducted as the color scheme of the seed kernels is also utilized.

Section 3.1 presents a randomized synthetic image generation framework using image processing that applies different image transforms to generate transformed variants of the object in the image. The use of simple image transforms means that the framework is light weight and is able to function efficiently on a CPU unlike large scale neural network models that require a GPU for efficient functioning. In addition to images, the framework also generates annotation files in the COCO format, so the images are able to train on the neural networks.

Section 3.2 describes training instance segmentation neural networks such as Mask R-CNN and YOLO on the synthetic images generated by the framework described in section 3.1. The trained network models are evaluated for their performance on images containing real seeds. The evaluation demonstrates that the proposed random image generation framework produces synthetic images that are able to transfer to real seed images.

Sections 3.3 and 3.4 describe the application of synthetic images to train classification neural network models in the supervised and self-supervised domain. The models of VGG-16, VGG-19 and ResNet-101 are the candidates from the supervised domain whereas SimCLR, Momentum Contrast and Bootstrap Your Own Latent (BYOL) are the candidates from the self-supervised space. The obtained results show that randomized synthetic images generated by the proposed framework are able to achieve good performance on the classification models as well.

Overall, the chapter demonstrates the feasibility of domain randomization and transfer learning for seed phenotyping. The ability to train neural network models on synthetic images eliminates the need for large real-world datasets which are often unavailable to researchers.

Chapter 4 - Real-time Seed Kernel Tracking and Counting

One of the key aspects of High-throughput Phenotyping is cereal yield estimation. The two important characteristics that determine cereal yield are cereal size and number of grains per ear. While leveraging images to count the number of grains as mentioned in previous sections works, it has the pitfall of being tedious and time-consuming. Imagine having a pile of seeds where they have to be arranged and photographed on a common background so that they aren’t clustered or overlapped. It requires significant time and man-power. Proposed as part of the chapter is a fleshed-out image processing algorithm to conduct the task of seed kernel counting from videos. A video is a multitude of frames (images) put together and played in sequence. The image processing solution is based on a technique known as ‘slit imaging’ wherein only a tiny sliver from each frame of the video is used for seed kernel detection and counting. The application is significant since the seed counter machines available in the market cost upwards of $500 on average, a hefty price tag. The development of mobile applications that estimate seed count from videos significantly reduces the cost associated with seed counting. In addition, the solution acts as a vehicle to drive phenotyping from the conventional lab setting to the field setting that the majority of plant scientists prefer.

4.1 Related Work

Neilsen et al. [114] proposed an algorithm to conduct seed kernel counting from videos. The working of the algorithm is based on tracking each of the seed kernels as they flow down a backlit platform. A seed kernel is considered a valid detection and counted if the seed kernel is detected a predefined number of times (threshold). The algorithm serves as the basis for the work in Chapter 4.

GridFree [55] is a python package for image analysis of interactive grain counting and measurement. GridFree uses an unsupervised machine learning approach, K-Means, to segment kernels from the background by using principal component analysis (PCA) on both raw image channels and their color indices. The package incorporates users' experiences as a dynamic criterion to set thresholds for a divide-and-combine strategy that effectively segments adjacent kernels. When adjacent multiple kernels are incorrectly segmented as a single object, they form an outlier on the distribution plot of kernel area, length, and width. The software exhibits great performance on multiple crop types such as alfalfa, canola, lentil, wheat, chickpea, and soybean.

Abto Software developed a counting technique for moving objects using the technique of slit imaging. Reference [4] demonstrates the technique developed by the company to count cars in a video. The algorithm developed by the company serves as the basis for the proposed algorithm using slit imaging in chapter 4.

Parico et al. [122] performed real-time pear fruit detection and counting using YOLOv4 models and Deep SORT algorithm. The study provides a systematic and pragmatic methodology to choose the most suitable neural network model for a desired application in the field of agriculture. The region-of-interest (ROI) line technique was used by the study to estimate the number of pear fruits detected by the neural network model.

Santos et al. [148] conducted grape detection, segmentation and tracking using deep neural networks and three-dimensional association on wine grape clusters. Different neural networks such as Mask R-CNN, YOLOv2, and YOLOv3 were investigated as part of the work, and it was determined that the Mask R-CNN model produced the best outcomes. An F1-Score up to 0.91 for instance segmentation was reached on 408 grape clusters where the images were captured using a trellis-system based vineyard.

Hajar et al. [48] performed vision-based moving obstacle detection and tracking in paddy field using YOLOv3 and Deep SORT. The center point positions of the obstacles were used to track the objects as they moved through the paddy field. The augmented YOLOv3 architecture consisted of 23 residual blocks and up-sampled only once. The augmented architecture obtained a mean intersection over union score of 0.779 and was 27.3% faster in processing speed than standard YOLOv3.

Chen et al. [25] performed citrus detection in orchard environment using the YOLOv4 neural network model. The motivation for the work was that detecting fruits in natural environments was a challenge due to occlusion among leaves and fruits. A Kinect V2 camera was used to collect RGB images of citrus trees. The Canopy algorithm and K-Means++ algorithm were applied to automatically select a pre-defined number of frames from the RGB images. Finally, a fine-tuned YOLOv4 model was applied on the images to detect the citrus fruits. The experimental results delivered an accuracy of 96.04%.

4.2 Seed Kernel Counting using Slit Imaging

The proposed algorithm to count seed kernels leverages a technique named Slit Image Generation. Slit image is an image constructed by joining pixels belonging to each frame in the video corresponding to a singular location. For instance, in a video consisting of 1000 frames where each frame is 200 x 200, a slit image may be generated by extracting pixels between the coordinates (0, 0) and (10, 10) and vertically stacking the extracted regions together to form a single image. The regions at which the pixels are extracted are known as regions of interest (ROI). The chosen ROI for a video is subjective and depends on the task at hand. Processing and analyzing a video are tasks that require plenty of time and resources. It is a computationally expensive operation that generally calls for the use of Graphical Processing Unit (GPU). Even with the availability of a high number of resources, it is important to note that not every point in a video (or image) is equally informative. It could be the case that a subset of the frames in the video contain only the background or noise. In such cases, the resources are essentially wasted. The use of Slit Imaging technique alleviates some of the problems associated with video processing. Since a slit image is a representation of the entire video, the video is no longer required upon the generation of the slit image. The slit image may then be processed to gain a complete insight into the specifics of the video. Not only does the technique save time, but it also lowers the resources required since a video only needs to be processed once to build the slit image and the image may be processed on resource constrained mobile devices. Performing inference on the slit image is relatively easy compared to performing inference on the entire video which once again is a collection of images.

4.2.1 Video Capture for Seed Kernel Counting

Inspired from the video capture mechanism described in [114], a video capture mechanism where seed kernels are rolled down a white light emitting light box by a mechanical hopper is designed. Figure 4.1 shows the seed kernel image capture setup designed for the experiment. The mechanical hopper helps to deliver seeds at a constant rate unlike free-hand delivery that tends to be erratic. The mobile phone is placed on a 3-

D printed stand to ensure that the camera is always held orthogonal to the surface. It helps to eliminate any skew that may result during the capture of the video. The stand is fitted

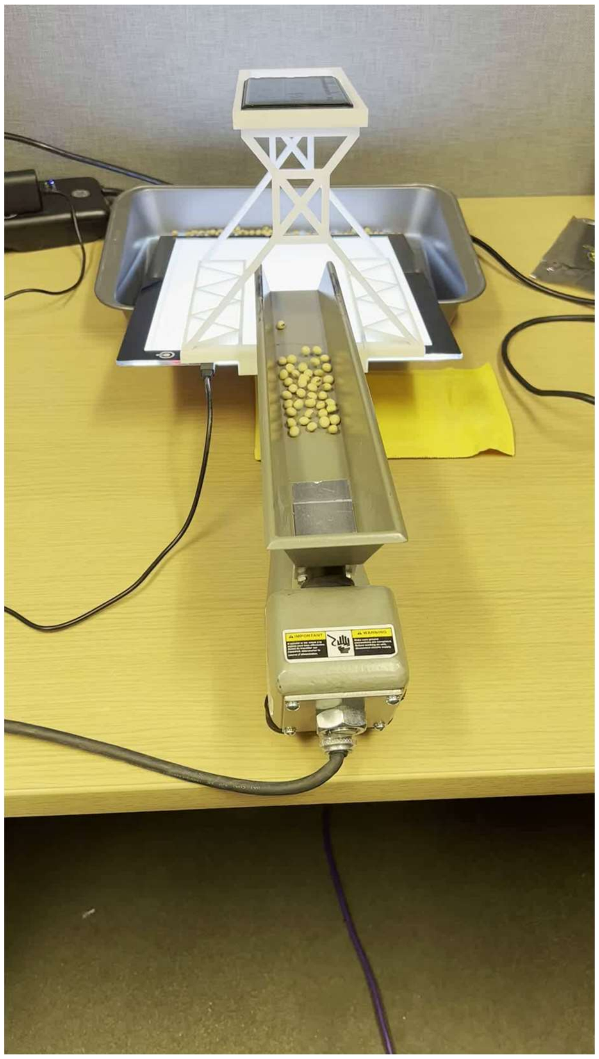


Figure 4.1: Mechanical Hopper Delivering Soy Seed Kernels

with a 3-D printed platform at the bottom. The platform at the bottom channels the seed kernels ensures that the seed kernels remain in the field of view of the camera as they roll down the light box. In the absence of the platform, it is observed that seed kernels often drift to the side and fall off the light box prematurely hindering their detection. The mobile phone used for image capture is a Motorola Moto G6 mobile phone whose capture frame rate is 60 fps for HD quality video.

4.2.1.1 Influencing Factors

The slit image is required to be representative of the entire video. In the event the slit image is not constructed adequately, key information might fall through the cracks leading to sub-par results upon analysis. In the context of seed kernel counting, the factors that require careful consideration are as follows:

1. **Frame Rate:** The frame rate of the video, also known as frames per second (fps), is key to information extraction and retrieval from the video. Since the slit image is a group of pixels extracted from the same location in every image of the video, high frame rate is preferred. A high frame rate ensures that the action in the video is highly continuous with few breaks. It is essential since the ROI used for slit image creation is typically a small window. In the event objects of interest aren't captured in a window, the end count of seed kernels is thrown off.

2. **Object Clusters:** In a video where a multitude of seed kernels are made to flow down a sled at the same time, it is almost always the case that the seed kernels touch each other while in motion. In case the seed kernels are clustered at the time of extraction for slit image, the count is incorrect if the cluster is considered to be one seed kernel since it consists of at least two or more kernels. Figure 4.2 shows a frame in a video where seeds are clustered.

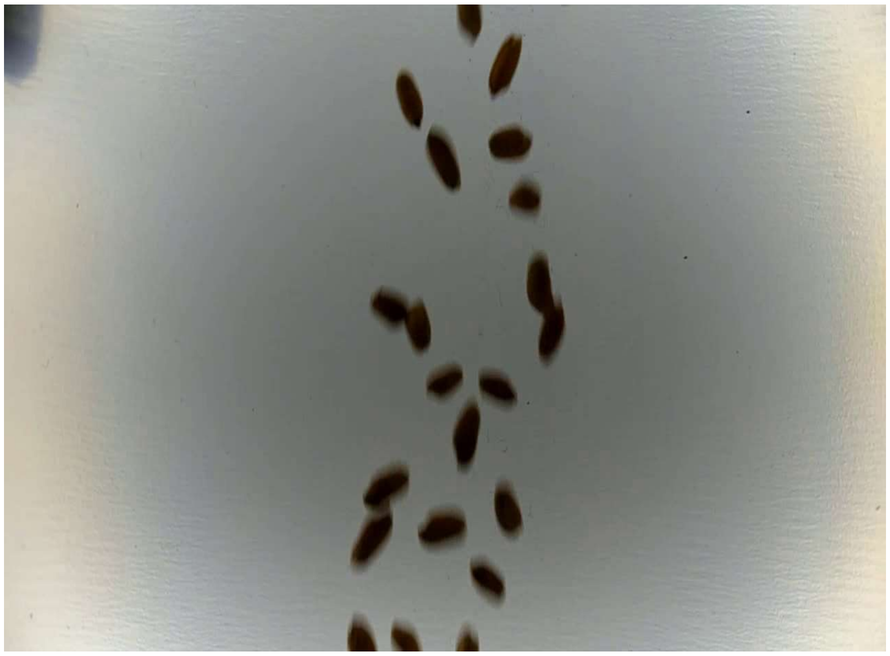


Figure 4.2: Frame with Seed Clusters

4.2.1.2 Algorithm

The algorithm is defined as a three-step process:

1. Capture of seed kernel video
2. Generation of slit image
3. Analysis of slit image to count seed kernels

1. **Capture of Seed Kernel Video**

Frame rate is key to the efficient functioning of the algorithm. The typical capture rate of mobile phone cameras over the years is 15 – 30 frames per second. Recent models of smartphones are able to capture slow-motion and super slow-motion videos at 60 frames per second and 120 frames per second respectively. The work [114] analyzed the impact of frame rate on seed kernel counting and realized that frame rate is directly proportional to the accuracy i.e., the higher the frame rate, the more accurate the seed kernel count produced by the algorithm. In order to perform a well-rounded evaluation of the algorithm, videos of seed kernels flowing down the light box are captured at normal, slow-motion and super slow-motion rates i.e., 30 fps, 60 fps, and 120 fps respectively. Regardless of the frame rate at which videos are captured, video players render videos at 30 fps. The captured videos are put through a minimal amount of pre-processing before applying the algorithm. The pre-processing is primarily to remove any unnecessary frames in the video. For instance, in the majority of videos captured, the initial (about five) seconds of the video does not contain any seed kernels since it takes a few seconds for the mechanical hopper to deliver the seed kernels onto the lightbox. Pruning such non-informative parts of the video helps in improving the runtime of the algorithm. MoviePy is a python module for video editing operations such as clipping, concatenation, and title insertion. The MoviePy's subclip function is used to clip the video. The subclip function takes in two arguments – start and end, and outputs a clip of the video. The input arguments are used as the endpoints to clip the video.

2. Generation of Slit Image

The first step in the generation of slit image is the selection of region of interest. In theory, any part of the image could be selected to as the region of interest. However, in practice, it is observed that the seed kernels being delivered by the mechanical hopper are clustered at the point of delivery. Multiple seed kernels appear as one when they are clustered. Algorithmic analysis on clustered seeds is complex and often leads to erroneous results. Another observation is that the seed kernels bounce off the lightbox since they are delivered from a height of about two inches from the lightbox by the mechanical hopper. Seed kernels which have circular morphometry, such as soy, are observed to bounce more than ones with elliptical morphometry, such as wheat. In order to best avoid seed kernel clusters and bounce off the lightbox, it is empirically determined that any part of the image that lies in the bottom-half of the image is best suited to be region of interest. Upon selecting the region of interest, the video generated in the previous step is put through a video processer that runs through every frame of the video and extracts the pixels belonging to the region of interest. All the regions of interest extracted from the video are vertically concatenated with each other resulting in the slit image. For instance, if the ROI extracted from each frame is of size w x h pixels where w and h are number of width and height respectively, the resulting slit image is of size (k x h) x w pixels where k is the number of frames in the video. Figure 4.3 shows the slit image generated from a video of wheat seed kernels. The video consists of 1500 frames and the region of interest spans three pixels in height and the entirety of the width i.e., 1809 pixels, of the image. The resulting slit image is of dimensionality is given by (3 x 1500) x 1809 i.e., 4500 x 1809 pixels. The region of interest for a given video depends on multiple factors such as frame rate, render rate and height of image capture. The influencing parameters are further discussed in 4.4.3. The biggest challenge in the generation of the slit image is to configure the dimensionality of the region of interest in a manner so that every seed kernel that flows down the lightbox is captured. The dimensionality of the region of interest is primarily influenced by three factors – seed kernel velocity, frame rate of image capture[A20] and height of image capture. The velocity of the seed kernel is influenced by the texture of the seed kernel and the angle of orientation of the lightbox along with the amount of friction it offers. Seed kernels with circular morphometry such as soy exhibit a higher velocity in comparison to seed kernels such as wheat that have elliptical morphometry. Generally, wheat seed kernels have a flat

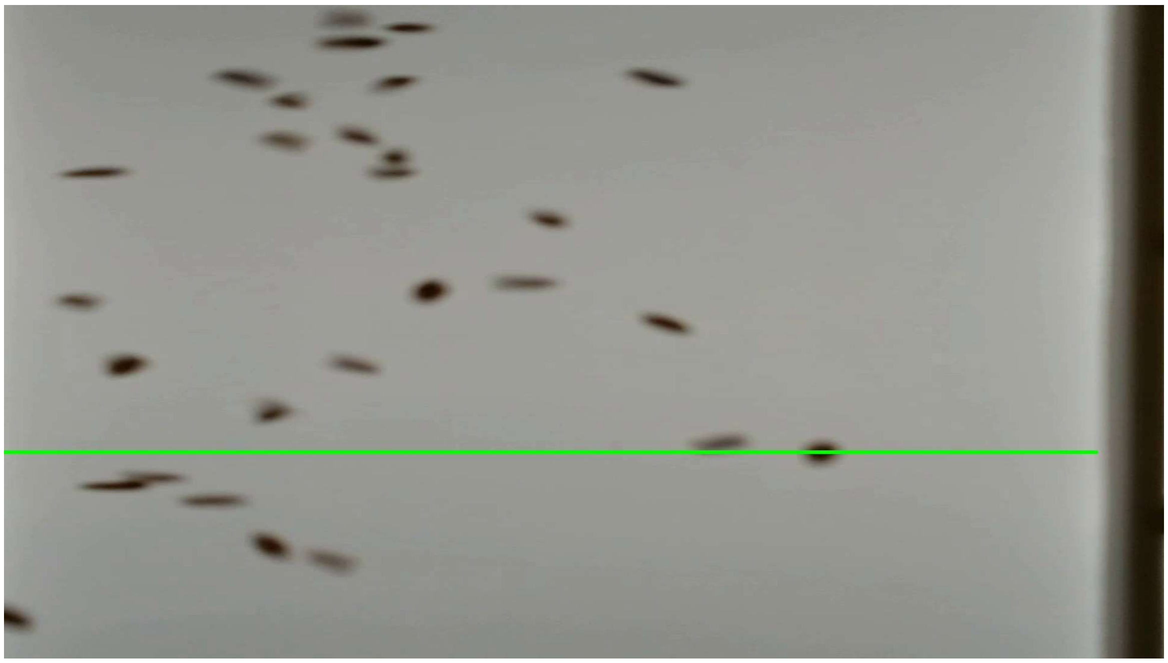


Figure 4.3: A Frame showing Two Wheat Seed Kernels at Region of Interest

bottom with a crease in-between which slows their flow on the lightbox. The higher the velocity, the faster the seed kernels moves in and out of the region of interest. The frame rate of image capture is critical to ensure appropriate collection of information. A higher frame ensures that a larger amount of information is captured. The height from which the images are captured significantly impact the size of the seed kernels in the video. For instance, the seed kernels appear larger in a video where the camera is placed at 10 cm from the ground compared to a video where the camera is placed at 30 cm from the ground. Hence, it is important to strike a balance among the three aforementioned parameters to ensure the generation of a quality slit image that captures all the information. Figure 4.4 shows the slit image generated for wheat seed kernels flowing down the lightbox oriented at 10° where the camera is held 18 cm from the ground and the region of interest is set to a height of ten pixels. Figure 4.4 is further analyzed to estimate the number of seed kernels in the slit image.

3. **Seed Kernel Counting using Slit Image**

The steps involved in estimating the count of seed kernels are as follows:

1. Convert the slit image from RGB color space to grayscale.

Figure 4.4: Slit Image of Each ROI stacked Vertically for Wheat Seeds

2. Gaussian blur[A21] the image to reduce noise and apply inverted binary thresholding on the grayscale image. The result is an image where the seed kernels are in white and background is black, as shown in Figure 4.5.

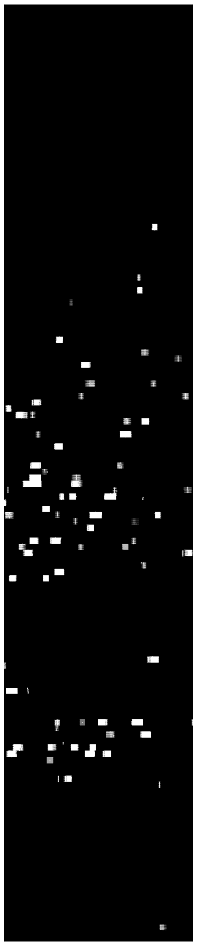


Figure 4.5: Contours on Grayscale Slit Image for Wheat

3. Perform a dilation on the image and find contours on the image to identify the contours of the detected seed kernels.

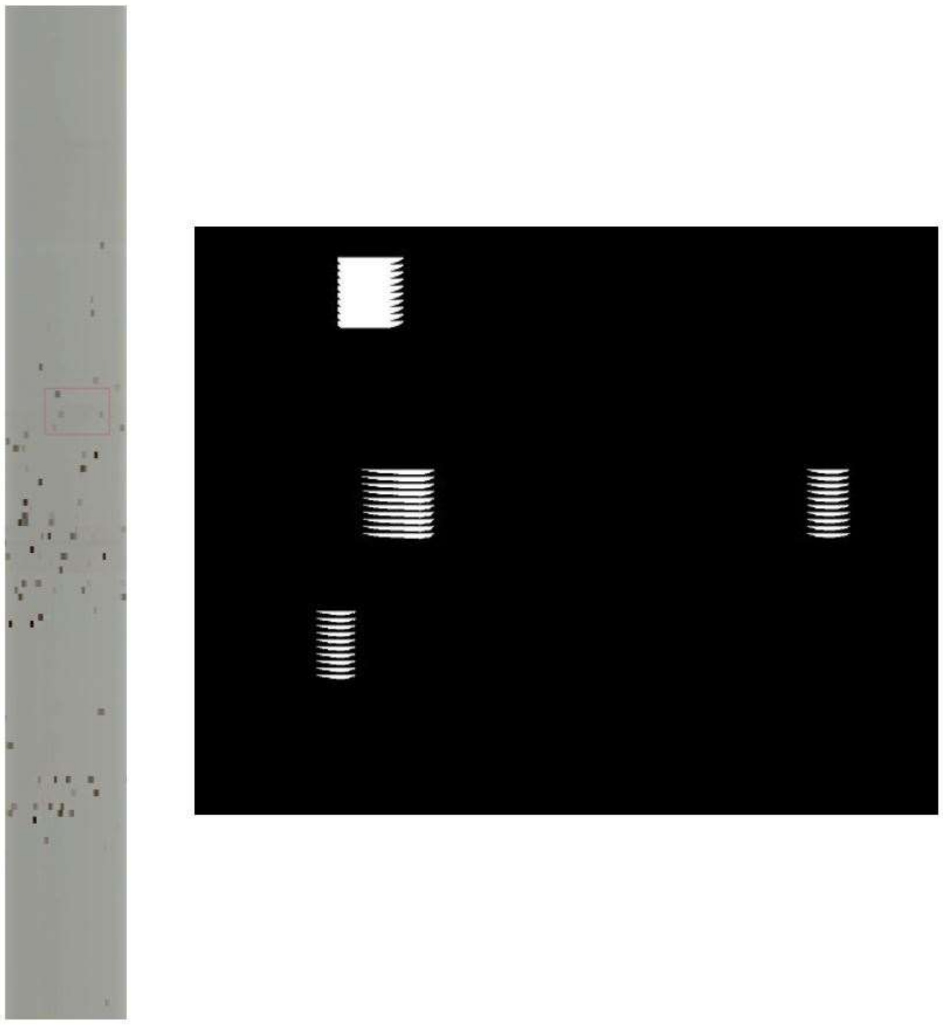


Figure 4.6: Zoomed-in Detections within Red Rectangle of Wheat Seed Kernel Slit Image

Essence of Dilation: The importance of dilating the image before the detection of contours is paramount. The key point to note with the detected contours in the absence of dilation is that they are slightly disjointed although they appear connected all along. Figure 4.6 shows four contours (from the red rectangle) that are identified from the slit image for analysis. They are disjointed because the region of interest is much larger than the seed kernels and captures the seed kernels multiple times as they flow through it. The region of interest is set to 10 pixels. The frame rate of 30 fps is high enough to capture the seed kernels multiple times as they flow through the region of interest. Hence, the contours are almost identical to each other albeit being different contours vertically stacked on top of each other. In a perfect world, the region of interest is precise enough and frame rate is high enough that the region of interest precisely captures a seed kernel exactly once and the number of contours identified in the image is the number of seed kernels present in the image. However, configuring a region of interest that captures a seed kernel exactly once is complex. The downside to multiple contour detections for a given seed when relying on contour count to estimate seed kernel count is that it leads to inaccurate seed count.

Dilation helps to ignore duplicate contours of the seed kernel and accurately estimate the count because dilation closes the gap and connects the disjointed contours. It leads to well-formed singular contours where the scope for multiple contour detection is significantly reduced although not completely eliminated. Figure 4.7 shows the result of dilating contours in Figure 4.6.

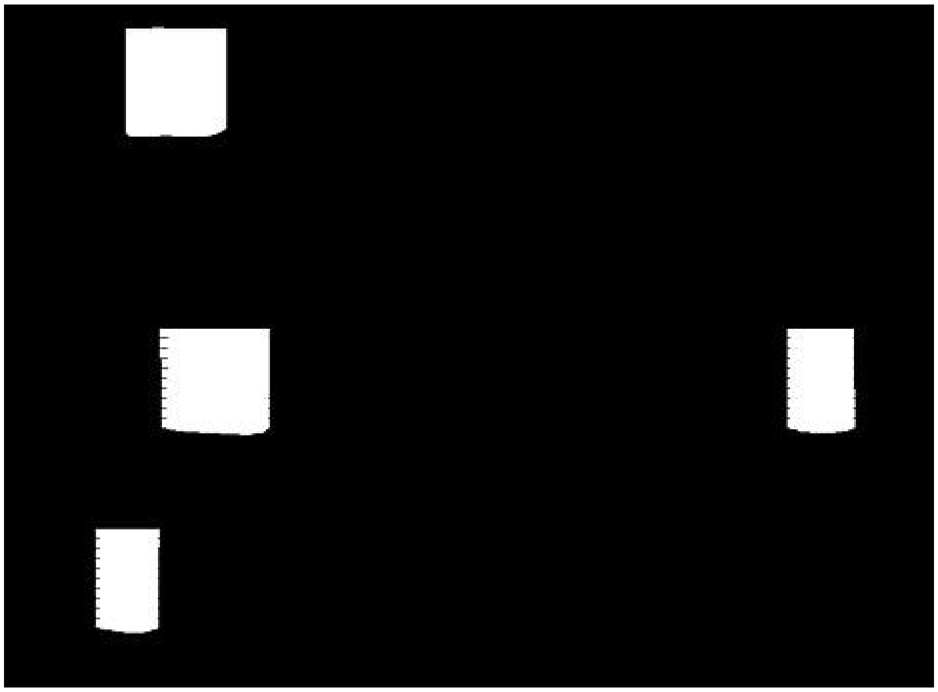


Figure 4.7: Result of Dilating Detections in Figure 4.6

Figure 4.8 shows a comparison of contours identified on the original and dilated slit images, as shown in Figure 4.6 and 4.7 respectively. The gray dots within the contour indicate the center of the contours.

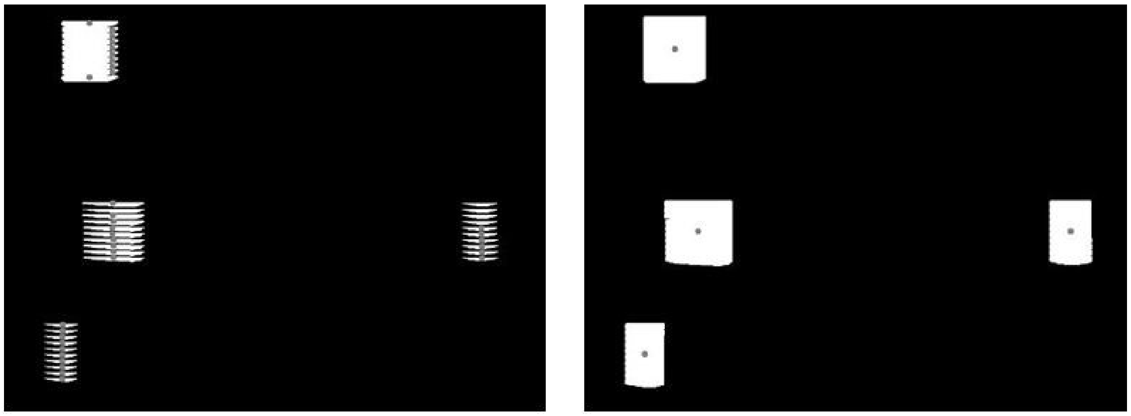


Figure 4.8: (**left**) Detection of Contours on Original Slit Image (**right**) Detection of Contours on Dilated Slit Image with Contour Centers shown by Gray Dots

As observed from Figure 4.8, multiple contours are detected (as indicated by multiple gray dots representing contour centers) for the same seed kernel on the original slit image whereas exactly one contour per seed kernel is detected on the dilated slit image. While multiple contours may be filtered to arrive at the accurate count estimate, it leads to additional overhead in terms of computational resource requirement and run-time of the algorithm. Hence, dilation is paramount to the efficient functioning of the algorithm.

4. Filter the contours by size to ensure that any stray contours that are detected on the dilated image. Figure 4.9(a) shows multiple small contours detected at the coarse edges of the contour. The stray contours observed are generally tiny abrasions that get picked up as contours by OpenCV. Applying a threshold on the size ensures that the tiny contours are ignored, as shown in Figure 4.9(b)

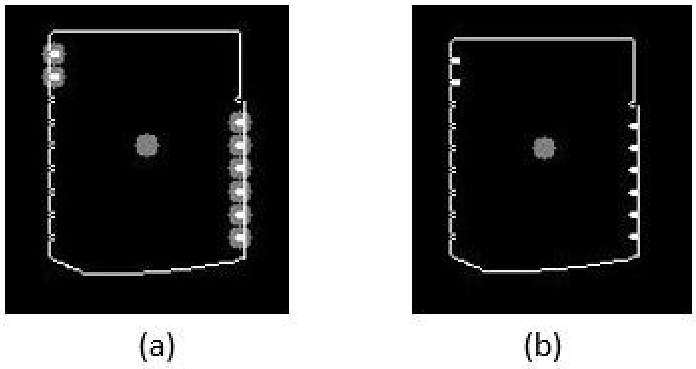


Figure 4.9: (a) Tiny Contours Detected in the absence of Size Thresholding (b) Tiny Contours Ignored by Size Thresholding

5. One of the downsides to dilation is that multiple seed kernels could appear to be one seed kernel. Figure 4.10 shows the case where dilation causes grouping of contours belonging to multiple seeds.

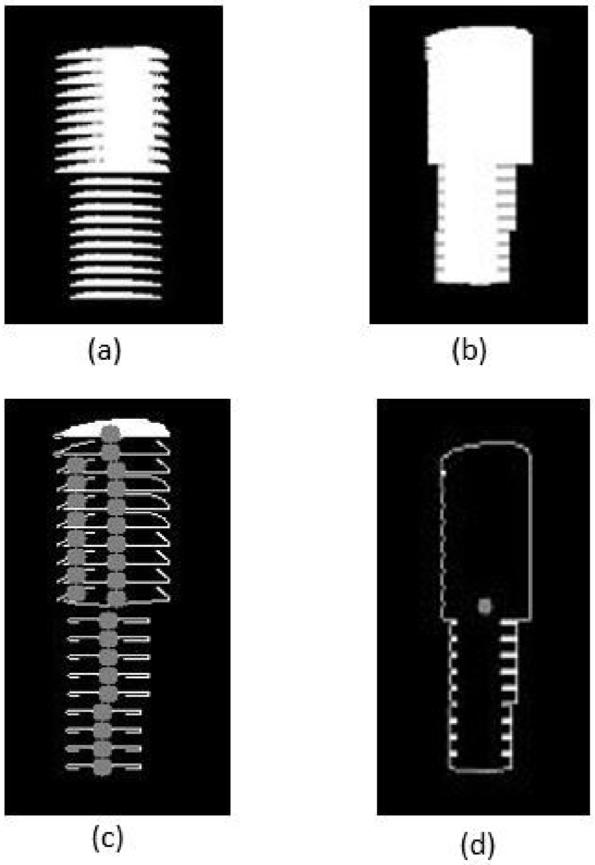


Figure 4.10: (a) Original Slit Image showing Two Seed Kernel Contours (b) Dilated Slit Image where Two Seed Kernels appear to be One (c) Contour Detection on Original Slit Image (d) Contour Detection on Dilated Slit Image

Figure 4.10(a) shows the recordings for two seed kernels wherein they passed one after the other. While the distinction is evident from the original image in Figure 4.10(a), the same cannot be said of Figure 4.10(b) which is the dilated version of Figure 4.10(a). The recordings of both seed kernels are connected offering the impression that they are one seed kernel. The same is observed upon detecting contours. Figure 4.10(c) shows the contours detected on Figure 4.10(a). While multiple contours per seed kernel are detected, it is still possible to observe the presence of two seed kernels whereas in Figure 4.10(d) that shows the contours detected on Figure 4.10(b), only one contour is detected for the two seed kernels. Dilation has the ill effect of clubbing multiple seed contours leading to inaccuracies. In order to negate the inaccuracies caused by clubbing contours, the metric of contour size is used. Another observation that is made from Figure 4.10(d) is that clubbed contours drawn around multiple seed kernels are significantly larger than contours drawn around single seed kernels. The observation is leveraged to arrive at the true seed kernel count and negate inaccuracies.

6. **Elimination of Duplicate Contours:** It is possible that multiple contours of the same size be detected around the same seed kernel. The contours be large enough to pass the minimum size threshold and perhaps identical to the contour that wraps around the entire seed kernel. Figure 4.11 shows four contours detected around the same seed kernels wherein all the contours are deemed valid. The numbers next to the contours are labels. The presence of overlapping numbers indicates that multiple valid contours are detected around the seed kernel. The detection of multiple contours is a known behavior of OpenCV when contours have abrupt edges or contours don't have a perfectly closed boundary. The detection of multiple valid contours leads to inaccuracies in the count due to a contour being accounted multiple times. While multiple contours are present, it may also be observed that there is only one gray dot in each of the contours. The gray dot indicates contour

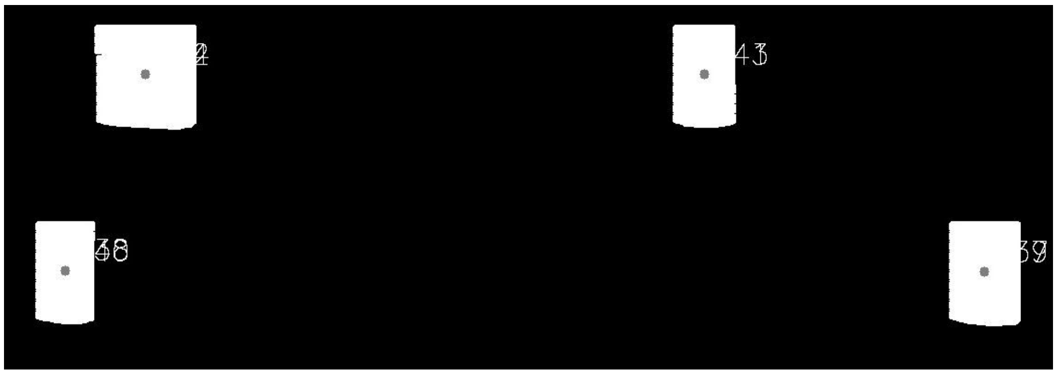


Figure 4.11: Multiple Valid Contours around Seed Kernels

center. The presence of a single gray dot indicates that the centers of both contours are at the same point. Hence, multiple contours centered at the same location are considered to represent the same seed kernel. In order to improve filtering, a threshold on the distance between the two contours may be established to filter out multiple contours that represent the same seed kernel. The experiments on multiple videos as part of the work showed that the occurrence of multiple contours centered at different locations is a rarity although the occurrence of multiple contours centered at the same location is common.

7. Compute the average area of the contours identified in the image. OpenCV provides the cv2.ContourArea function to estimate the area of contours in pixels.

8. Find the contours that have an area greater than 1.5 times the average contour area. The threshold of 1.5 is arrived upon repeated experiments providing accurate results. However, it is a tunable parameter that is required to be tuned to the task at hand. The idea behind the step is to identify contours that could potentially belong to multiple seeds. All the contours with an area under the threshold are assumed to represent one seed kernel[A22].

9. Find the number of seed kernels within each of the contours identified in step 6 by calculating the ratio of individual contour area and average contour area. The resulting value (upon rounding) is the count of seeds present within the contours. For instance, consider the average contour area is 400 pixels and a random contour has an area of 700 pixels, the number of seeds within the contour is *round (700/400)* i.e., 2.

10. The number of seed kernels is given by the appropriately dissecting the contours detected in the image as mentioned in steps 7, 8, and 9[A23].

4.4.3 Experiment and Results

The algorithm is applied on seed kernel videos of wheat and soy for experimentation. The reason behind the choice of wheat and soy is their distinct morphometry. As mentioned earlier, the experiment includes the seed kernels flowing down a lightbox. The circular morphometry of soy seed kernels means that they flow down the lightbox seamlessly even when the lightbox is oriented at less than 10 degrees whereas the seed kernels of wheat do not flow down smoothly at smaller orientations of the lightbox owing to their elliptical morphometry. The minimum inclination of the lightbox for smooth rundown of the soy seed kernels is 5° whereas the wheat seed kernels require at least 12° to 15°. It is best to orient the lightbox at 15° from the surface to ensure that all seed kernels flow down seamlessly. However, it is to be noted that the orientation of the lightbox has a visible impact on the velocity of the seed kernels as they flow down the lightbox. Higher the orientation of the lightbox, larger the velocity of the seed kernels. The frame rate of image capture is to be increased with increase in seed velocity to ensure optimal recording of the seed kernels as they flow down the lightbox. The size of the region of interest is also required to increase with increase in velocity of the seed kernels. Hence, the orientation of the lightbox impacts multiple other parameters such as velocity, frame rate, and region of interest. The experiment is conducted on different videos captured at frame rates of 30 fps i.e., normal mobile phone frame rate, 60 fps i.e., slow-motion frame rate, and 120 fps i.e., super slow-motion frame rate. The height of image capture and the size of region of interest are also altered as the videos are captured. Videos are captured from heights of 18cm, 12cm and 9cm and regions of interest of sizes 10 pixels, 15 pixels and 20 pixels are configured for the experiment. Table 4.1 shows the results of the algorithm applied to wheat seed kernels.

Table 4.1: Algorithmic Results on Wheat Seed Kernels

Frame Rate	Height (cm)	Region of Interest (pixels)	Actual Count	Algorithmic Count	Actual – Algorithmic Count	Count Estimate Accuracy (%)
30	18	5	275	186	89	67.63
		10	275	193	82	70.18
		15	275	191	84	69.45
	12	5	275	179	96	65.09
		10	275	184	91	66.90
		15	275	187	88	68

	9	5	275	178	97	64.72
		10	275	182	93	66.18
		15	275	185	90	67.27
60	18	5	275	262	13	95.27
		10	275	265	10	96.36
		15	275	273	2	99.27
	12	5	275	266	9	96.72
		10	275	270	5	98.18
		15	275	268	3	97.45
	9	5	275	262	13	95.27
		10	275	266	9	96.72
		15	275	268	7	97.45
120	18	5	275	271	4	98.54
		10	275	271	4	98.54
		15	275	273	2	99.27
	12	5	275	270	5	98.18
		10	275	269	3	98.9

		15	275	271	3	98.54
	9	5	275	269	6	97.8
		10	275	270	5	98.1
		15	275	272	3	98.9

As evident from the results, the performance of the algorithm on videos captured at 30 fps is sub-par. Regardless of the height of video capture and size of region of interest, the best performance of the algorithm is 70.18%. Upon investigating the slit image for the mediocre performance, it is observed that the fallacy lies in the capture of seed kernels within the region of interest. The frame rate of 30 is so low that a portion of the seed kernels is never captured as it passes through the region of interest. As a result, the slit image does not contain the seed kernels. However, it is worth mentioning that the algorithm does a good job of precisely accounting for the seed kernels that are captured in the slit image. The performance of the algorithm receives a significant boost as the frame rate of the video is increased. At 60 fps and 120 fps, the algorithm achieves accuracies of over 95%. The best accuracy of 99.27% is achieved on the video captured at 120 fps. The reason for such high performance at higher frame rates is that the algorithm captures all the seed kernels as they pass through the region of interest. As the amount of information in the slit image increases, the performance of the algorithm also significantly improves. The size of region of interest does not have a significant impact when the frame rate is high enough, as evident from the results. However, the factor that leads to the algorithm still undercounting by a few seeds is seed clusters. As explained in the algorithm, as the seed kernels pass through the region of interest, they could be in contact. The clusters of seed kernels produce larger contours than singular seed kernels. While the algorithm establishes a mechanism to account for seed kernel clusters, a few clusters still escape the filter. Table 4.2 shows the results from the application of the algorithm to soy.

Table 4.2: Algorithmic Results on Soy Seed Kernels

Frame Rate	Height (cm)	Region of Interest (pixels)	Actual Count	Algorithmic Count	Actual – Algorithmic Count	Count Estimate Accuracy (%)
30	18	5	236	141	95	59.7
		10	236	144	92	61.01
		15	236	148	88	62.71
	12	5	236	139	97	58.89
		10	236	147	89	62.28
		15	236	149	87	63.13
	9	5	236	137	99	58.05
		10	236	140	96	59.32
		15	236	141	95	59.74
60	18	5	236	226	10	95.76

		10	236	225	11	95.33
		15	236	227	9	96.18
	12	5	236	226	10	95.76
		10	236	226	10	95.76
		15	236	223	13	94.49
	9	5	236	225	11	95.33
		10	236	226	10	95.76
		15	236	224	12	94.91
120	18	5	236	227	9	96.18
		10	236	228	8	95.39
		15	236	229	7	97.03
	12	5	236	224	12	94.91
		10	236	226	10	95.76
		15	236	227	9	96.18
	9	5	236	226	10	95.76
		10	236	228	8	96.61
		15	236	228	8	96.61

A pattern similar to wheat is observed for soy as well wherein the performance of the algorithm is mediocre on videos captured at 30 fps regardless of the height of image capture. The performance of the algorithm on videos captured at 60 fps and 120 fps also follows that for wheat wherein the algorithm performs exceedingly well. The reason for the lack luster performance of the algorithm at 30 fps is that the seed kernels are not captured as they pass through the region of interest due to low frame rate.

4.4.4 Comparison with Neilsen et al. [114]

The proposed algorithm is inspired by the earlier work by Neilsen et al. [114]. The algorithm put forth by Neilsen et al. [114] works on the basis of tracking a seed kernel using the technique of background subtraction as it flows down the lightbox. A seed kernel is deemed valid and counted when it is detected a certain number of times defined by the average flow rate. The average flow rate is a scalar that indicates the rate at which the seed kernels in the video flow down. It is used to estimate the location of the seed kernel at a given point and search for it. In the algorithm put forth by Neilsen et al. [114], assuming that the current position x of a seed kernel is known, the seed kernel is expected to be found in the subsequent frames at or within $4 * average\ flow\ rate$. Hence, the average flow rate plays a critical role in determining the position of a seed kernel in subsequent frames. In addition to flow rate, another parameter that the algorithm relies on is average area. The average area parameter is indicative of the average area occupied by each of the contours detected in the video. The algorithm initially assumes a value of 0 for both average flow rate and average area. While the algorithm updates the values of average flow rate and average area, it takes a few frames to do so. Prior to the update to average flow rate and average area, the count made by the algorithm tends to be inaccurate. Determining the appropriate values for average flow rate and average area requires meticulous observation. Neilsen et. al determined that the most optimal frame rate for efficient functioning of the algorithm is 60 fps. The work by Neilsen et al. also mentioned that the error rate for the algorithm exploded when the frame rate was less than 60 fps. As a means to compare the proposed algorithm to the one proposed by Neilsen et al., the video of wheat seed kernels captured at 60 fps is used. The results are as shown in Table 4.3.

Table 4.3: Comparison between Proposed Algorithm and Neilsen et al.

Seed Type	Frame Rate	Height (cm)	Actual Count	Algorithmic Count	Neilsen et al. Count	Algorithmic Count Accuracy (%)	Neilsen et al. Count Estimate Accuracy (%)
Soy	60	18	236	226	228	95.76	97.02
		12	236	225	229	95.33	97.03
		9	236	227	228	96.18	97.02
Wheat	60	18	275	266	270	96.72	98.18
		12	275	270	270	98.18	98.18
		9	275	268	266	97.45	96.72

The proposed algorithm exhibits a high degree of correlation to that of the algorithm proposed by Neilsen et al. Overall, the experiments conducted on the seed varieties of soy and wheat show that the algorithm exhibits quality performance on slit images provided that the slit image is constructed well. However, the construction of a well informing slit image depends greatly on the frame rate at which the video is captured. The algorithm is shown to provide inferior results on videos captured at 30 fps in comparison to the videos captured at 60 fps and 120 fps. At such high frame rates, the size of the region of interest does not impact the outcome much.

4.4.5 Conclusions and Future Work

The algorithm demonstrates the feasibility of seed kernel counting using region of interest segmentation and slit imaging. The algorithm gracefully handles the issue of duplicate contours within the detected contours. The pitfall of occlusion within centroid tracking algorithm is handled using the slit imaging technique. Since the slit image is created by stitching multiple regions of interest together, the likelihood that the seed kernel is not occluded increases, thereby, reducing the chance of missing out on seed kernels. However, the proposed algorithm's reliance on an informational slit image can be stated as its pitfall. Especially at lower frame rates where the slit image tends to miss recording information, the algorithm exhibits poor performance. Moving forward, an algorithm which is the combination of centroid tracking and slit imaging will be implemented. Currently, counting using centroid tracking relies on observing the seed kernel for a certain number of times before it is deemed a valid seed kernel that is counted. However, occlusion in certain cases causes the centroid tracking algorithm to lose track of the seed kernel. In order to avoid such a scenario, the seed kernel is tracked all through the region of interest to ensure that the seed kernel is detected in case it is not blocked (or tied).

Chapter 5 - Automated Phenotyping of Single Seeds using a Novel Volume Sculpting Framework

Seed and plant breeding applications require the ability to estimate the morphometric characteristics, otherwise known as phenotypes, of agricultural entities and products such as seeds, leaves and fruits in order to establish a correlation between morphometry and behavior. Plant phenotyping is the assessment of complex plant traits such as growth, development, tolerance, resistance, architecture, physiology, ecology, yield, and the basic measurement of individual quantitative parameters that form the basis for complex trait assessment [79]. Reliable techniques to estimate morphological traits such as plant biomass [102][44], root morphology [73][178], leaf morphology [62][12] and fruit traits [30][109] are available to aid in plant phenotyping. Besides the aforementioned traits, another key morphological trait that aids in phenotyping is seed volume. The estimation of seed volume is complex because seeds are irregular shaped entities, in general. A seed is the most basic agricultural entity from which the complex root and shoot systems develop. The research on seed-functional ecology and seed-trait correlates is heavily dependent upon seed volume, which is overwhelmingly represented by the mean seed mass [143]. In order to integrate the ecological and functional correlates of seed size distributions in soil seed bank studies, seeds are required to be sorted by size. Conventionally, seeds are sorted by passing them through a series of sieves of diminishing mesh sizes. The pitfall with the technique is that sieves only separate the seeds according to a linear dimension rather than volume. In addition, it is laborious, time consuming and may damage the seeds at times [146]. Other popular techniques to estimate seed volume include water displacement, volume slicing and silhouette-based volume sculpting. The technique of water displacement attributes the volume of an object to the amount of water displaced when the object is dropped in the body of water. However, the technique is detrimental to the structural integrity of the seeds since they tend to absorb water and bloat in size. Volume slicing refers to the technique of dividing the object in the image into a certain number of cross-sections and computing the volume of each of the cross-sections. The aggregate of cross-sectional volumes is deemed the total volume. The silhouette-based volume estimation is the technique of constructing the 3-D model of the object from multi-view imagery. While no volume estimation technique is 100% accurate, the image-based techniques of volume slicing and silhouette-based estimation preserve the structural integrity and chemical composition of the seeds. Another key benefit to image-based analysis is that the results are reproducible across different experiments eliminating subjective bias. The estimation of volume, a three-dimensional quantity from images, two- dimensional entities, needs a meticulously curated setup wherein the 3-D reconstruction of the seed from images is achieved. The article extends upon the silhouette-based volume estimation technique and presents a low-cost, end-to-end silhouette-based volume sculpting framework that reconstructs the 3-D model of a seed using multi-view imagery that captures the seed from different perspectives. The proposed system is low-throughput that processes one seed at a time.

5.1 Related Work

Koc [70] demonstrated the use of ellipsoid slicing and image processing techniques to estimate the volume of watermelon. The dimensions such as the length, major and minor diameters of the watermelon were used as the parameters in the ellipsoid approximation method. In case of image processing, the grayscale contour i.e., boundary of the watermelon was extracted using image processing. The boundary image was approximated for volume using the disk method [138]. The outcomes of the two techniques were compared to those by water displacement. The comparison showed that the outcomes of the water displacement and image processing techniques were more similar to each other than those between water displacement and ellipsoid slicing.

Roussel, Geiger, Fischbach, Jahnke and Scharr [60] described a method to perform 3D reconstruction of plant seeds surfaces with diameters as small as 200 μm using shape-from-silhouette approach. The robotized system developed as part of the work was low-throughput and handled one seed at a time. One of the noteworthy accomplishments of the work was the manner in which camera pose variations were handled. The work served as the basis for Cao et al. [24].

Cao and Neilsen [24] extended the work by Roussel et al. [60] and presented an affordable 3-D single seed volume measurement system. The method involved the use of 3-D printed components to act as the scaffolding onto which cameras were mounted. The seed was placed on a turntable with a black background. Images of the seed were captured by the camera as the seed rotated on the turntable. The captured imagery was processed using silhouette-based volume sculpting to estimate volume. The system was verified against a ceramic ball of known geometry. The volume estimated by the system showed less than 3% difference to the actual. Cao et al. [24] is the basis for the current article.

Pound, French, Murchie and Pridmore [126] proposed an approach for automated recovery of three-dimensional models of plant shoots from multiple color images using a single low-cost camera. The algorithm for reconstruction used an initial point cloud estimate as a basis for the growth of plant surfaces in three dimensions. The reconstructed plants were represented as a series of small planar section that together model the more complex architecture of leaf surfaces. The boundary of each leaf patch was refined using the level-set method, optimizing the model based on image information, curvature constraints, and the position of neighboring surfaces. The proposed approach was tested on the datasets of wheat, rice and a virtual dataset that allowed for the measurement of reconstruction accuracy.

Yang and Cho [191] performed 3-D crop reconstruction and automatic analysis of phenotyping index using machine learning. In the experiment, a system was configured and implemented for the 3-D image reconstruction of red pepper plant, as well as its automatic analysis. A Kinect v2 with a depth sensor and a high-resolution RGB camera were used to obtain more accurate reconstructed 3-D images. The reconstructed 3-D images were compared with conventional reconstructed images, and the data of the reconstructed images were analyzed with respect to their directly measured features and accuracy, such as leaf number, width, and plant height. The results showed that the proposed method showed an error of about 5 mm or less when reconstructing and analyzing 3-D images, and was suitable for phenotypic analysis.

Potmesil [89] presented a method to generate octree models for 3-D solid objects from their silhouettes obtained in a sequence of images. The silhouettes of objects were projected into an image and the center of projection generated 3-D conic volumes. The 3-D model of the objects is constructed by intersecting such conic volumes obtained from a sequence of images. In order to process 3-D volume data efficiently, hierarchical octree structures were used. The volumes of individual objects were labelled by a connectivity-labeling algorithm, and surface-normal vectors were added to their surface volume elements.

Sankaran, Wang and Vandemark [146] proposed an image-based rapid phenotyping technique to estimate the size of chickpeas. The experiment was conducted on samples collected from 72 plots from two different locations. The experiment involved the use of images where chickpeas were clustered together and also separated from each other. The Watershed algorithm was used to segment the clusters of chickpeas and a macro developed in ImageJ was used to estimate seed count and seed size. The reference object to estimate chickpea size was the US Penny. The results exhibited a high correlation between the seed size estimated by the image processing technique and ground-truth data.

Zhang, Zhang, Wang, and Yang [196] developed an information management architecture for distributed wireless sensor networks to deploy in distributed regions. Different types of sensors, environmental factors, and control devices were considered during the development of the architecture. The architecture consisted of four layers: sensor, gateway, central server, and domain application. The developed architecture was applied to a melon greenhouse production base and apple tree cultivation base. The integrated bases were able to interact with the information management platform well.

Golbach, Kootstra, Damjanovic, Otten and Zedde [43] proposed a high-throughput system for 3-D reconstruction of plants. The 3-D reconstruction was based on the use of silhouette-based technique. In order to make the process high-throughput, varying number of cameras were used to capture the image of the plants from different perspectives. While the quality increased with the increase in the number of cameras, it also led to an increase in the computational complexity of the system. In the end, ten cameras were chosen to balance the trade-off. The results indicated a high correlation between the results obtained by the algorithm and ground-truth measurements obtained by manual measurement.

5.2 Materials and Methods

The concepts that help to understand the estimation of volume using the silhouette-based approach are explained as follows. Please note that the underlying mathematical derivations are out of scope of the article.

Voxel: A voxel is the unit of information that defines a point in 3-D space. A pixel in 2-D space is analogous to a voxel in 3-D space. Simply, they are virtual cubic blocks that are representative of 3-D space.

World Coordinate System: It is the basic 3-D cartesian coordinate system with an origin assigned by means of arbitrary assumption. In other words, it may be defined as any point in the 3-D space.

Camera Coordinate System: It is the 3-D coordinate system that measures relative to the camera's origin and orientation. It is possible to apply the operations of rotation and translation to convert a point in the world coordinate system to the camera coordinate system. There exists a 4 x 4 transformation matrix known as Camera Extrinsic Matrix which applies rotation and translation to convert a point from world coordinate system to camera coordinate system. The camera extrinsic matrix changes with the physical location of the camera.

Image Coordinate System: It is the coordinate system that projects the 3-D point from camera coordinate system to the 2-D plane. Only the height and width of the 3-D point are captured but there is no sense of depth in the image coordinate system. Hence, it is a lossy transformation that cannot be reversed.

Pixel Coordinate System: In order to discretize the image coordinate system, it is divided into pixels. The pixel coordinates of an image are discrete values within a range computed by dividing the coordinates in the image coordinate system by pixel width and pixel height.

Intrinsic Parameters: The parameters that define the relationship between the image coordinates and camera coordinates are called Intrinsic parameters. Intrinsic parameters are specific to the camera in use and a meticulous calibration of the intrinsic parameters is required to avoid image distortion. The intrinsic parameters typically considered are:

1. Focal Length: Focal Length is defined as the distance between the image plane and origin of the camera coordinate system.

2. Principal Point: The Principal Point is the point where the Optical axis i.e., the Z-axis of the camera coordinate system, intersects the image.

Figure 5.1 presents a graphical representation of the intrinsic parameters of focal length and principal point.

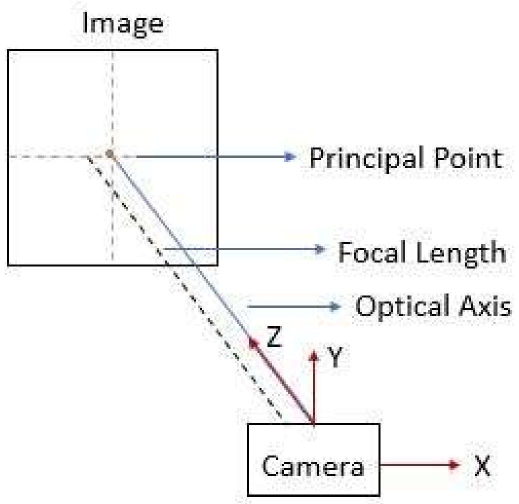


Figure 5.1: Intrinsic Parameters

5.2.1 Shape-from-Silhouette Method for Seed Kernel Reconstruction

The silhouette of an object in an image refers to the contour that separates the object from the background. The shape from silhouette method requires multiple images captured from different angles to perform 3-D reconstruction of a seed kernel. For each of the images captured, the silhouettes are segmented using a method known as background subtraction. The silhouettes are then back-projected onto a common 3-D space with projection centers equal to the camera locations. The back-projection onto a common space requires the intrinsic camera matrix K and distance between the origin of the working volume and camera center. The origin of the working volume is considered to be the intersection point (IP) of the seed bottom horizontal line and the line drawn through the center of the image. A total of N images equidistant from each other spaced at rotation angles a_i where $i \in \{1\ldots, N\}$ are acquired. The rotation is around the vertical axis through the IP and parallel to the Y-axis of the camera. A grayscale threshold on each of the acquired images is applied and segmented into a binary mask M_i where $i \in \{1\ldots, N\}$. For each image, the camera projection matrix P_i is calculated from the rotation angle a_i by $Pi = K(Ri \mid Ti)$ where R_i is the rotation matrix and T_i is the translation matrix corresponding to the given angle a_i. Then, an equidistantly spaced cubic voxel grid around the world origin is defined around the world origin. The size of each voxel is set to 1 mm^3. Each voxel center with homogenous world coordinates $\vec{X}$ is projected to a point, $\vec{x}_i$ in each mask M_i by $\vec{x}_i = P_i\vec{X}$. If a voxel belongs to the foreground object, its value $V(\vec{X})$ is set to 1. If the voxel does not belong to the foreground object, its value $V(\vec{X})$ is set to 0. The mask M_i in volume carving is sensitive to misalignment of the object volume and needs to be carefully adjusted.

5.2.2 Hardware Framework

3-D reconstruction of seeds requires multiple images captured from different angles at regular angle intervals. In order to ensure that the images are captured precisely, a low-cost hardware setup is proposed. Different parts of the hardware setup are described as follows and shown in Figure 5.2.

a) **Seed Station:** The seed station shown in Figure 5.2 carries the seed as it is rotated by a NEMA 23 stepper motor. The seed station is just big enough to ensure that the seed comfortably fits on it. The stepper motor is powered by a 12V power supply and controlled by an Arduino micro-controller using a Synthetic gShield stepper driver.

b) **Image Capture:** The images of the seed are captured using a color camera DFK 37BUX287 manufactured by ImagingSource IC. Two cameras that are placed orthogonal to each other are used in the setup. The image capture is automated using the API provided by ImagingSource IC software.

c) **Stepper Motor:** The stepper motor is controlled by an Arduino UNO R3 board fitted with a gShield v5 board. The stepper motor is controlled by G-Code strings sent to it from a program written in Python. The turn rate in degrees is customizable by the user. For the experiment, a 10-degree rotation is used as the default setting.

d) **3-D Graphics and Legos:** The mounds that hold the cameras are printed using 3-D graphics. In order to print the parts, a MakerGear M2 3-D printer is used. A software tool named Cubit is used to create the STL models for the 3-D mounds. The model is converted to gcode for printing using another software tool named slic3r. The gcode is input to the MakerGear M2 3-D printer for printing. The camera mounds are rested on a base built using Lego pieces. The Legos are sturdy enough to hold the lightweight 3-D camera mounds and cameras themselves.

e) **Light Source:** Two gooseneck LED light sources are set next to the seed station to focus on the seed kernel, so the seed kernel appears bright at the time of image capture.

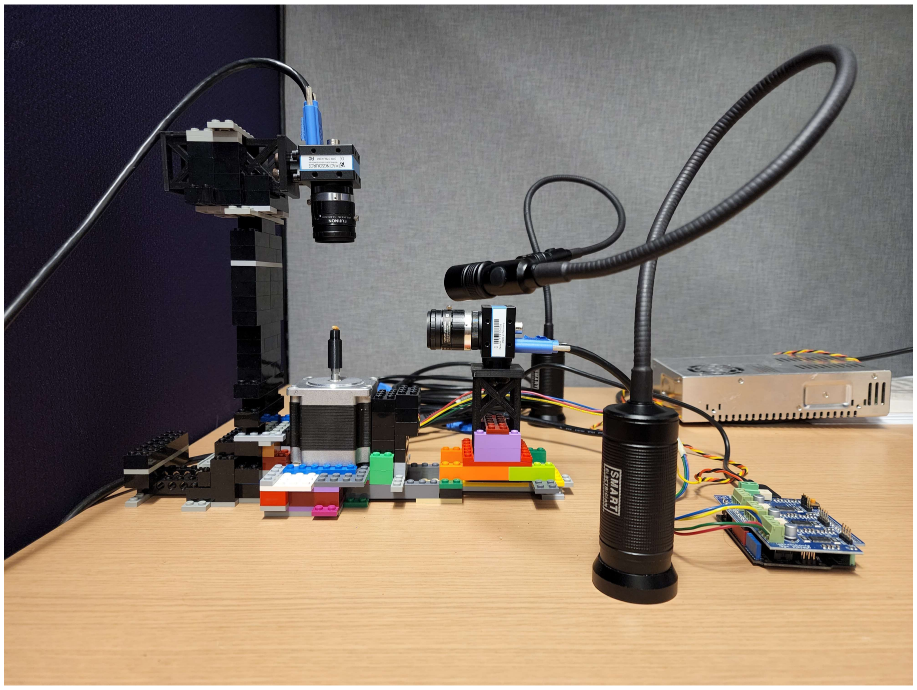


Figure 5.2: Hardware Setup for Single Seed Reconstruction

5.2.3 Software Framework

The software implementation to control the hardware framework and 3-D reconstruction is done in Python and C#. While C# is predominantly used to control the image capture of the camera, Python is used to perform every other task including the 3-D seed reconstruction. The different software packages and their versions used in the application are as shown in Table 5.1.

Table 5.1: Packages and their Versions for Single Seed Reconstruction Applications

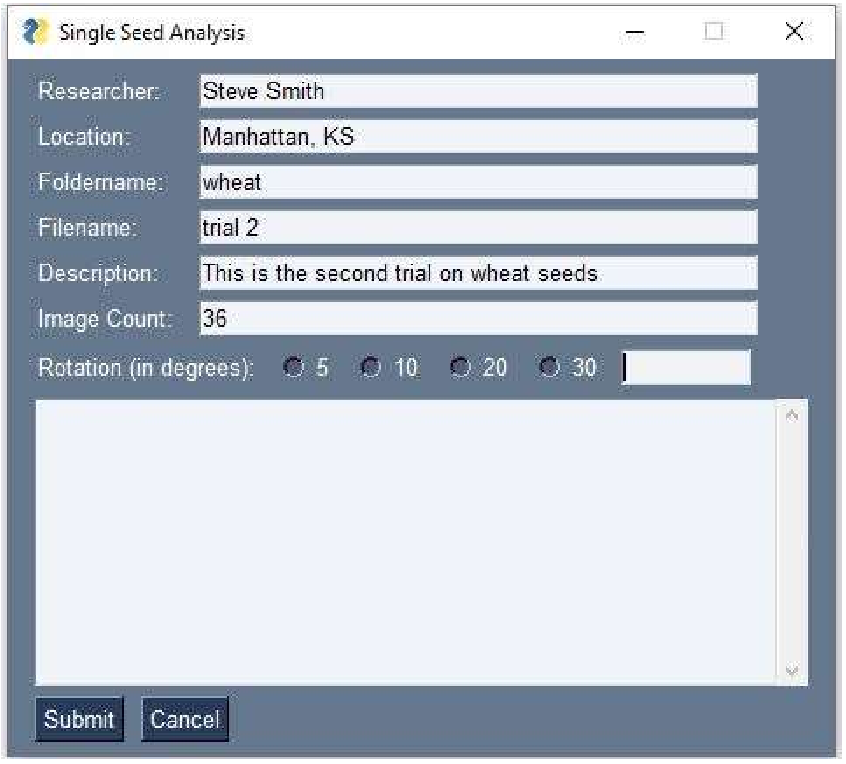
Package	**Version**
Python	3.7
Numpy	1.20.1
Matplotlib	3.3.3
Mayavi	4.7.2
SciPy	1.2.3
OpenCV	4.5.1.48

The steps in the application for seed reconstruction are described as follows.

1. The user interface shown in Figure 5.3 is provided as part of the framework and requires the user to enter basic information such as username, location, folder to save results, description and rotation angle of stepper motor. The rotation angle determines the number of images that are to be captured as part of the experiment. For instance, a rotation angle of 10-degrees means that the experiment will capture 360/10 = 36 images. Either image count or rotation only is required to be entered the user. The application calculates the other from the entry. In case 30 images is entered by the user, the application determines the rotation to as 360/30 = 12 degrees and sends G-Code commands to the stepper motor accordingly.

2. Crop the images such that the object of interest i.e., the seed kernel is at the center of the image.

3. Convert the images to HSV color space from RGB to aid in better thresholding. Image thresholding is applied on the HSV image to extract the foreground from background i.e., the seed kernel.



Researcher:	Steve Smith
Location:	Manhattan, KS
Foldername:	wheat
Filename:	trial 2
Description:	This is the second trial on wheat seeds
Image Count:	36
Rotation (in degrees):	○ 5 ○ 10 ○ 20 ○ 30

Figure 5.3: User Interface for Single Seed Reconstruction Application

4. Apply binary thresholding on the extracted seed kernel to obtain a binary mask.

Perform volume carving using each of the binary masks obtained to perform 3-D seed reconstruction.

5. Apply binary thresholding on the extracted seed kernel to obtain a binary mask.

6. Perform volume carving using each of the binary masks obtained to perform 3-D seed reconstruction.

- The center of the reconstruction cuboid is identified by using the IP of the seed bottom horizontal line and the centerline in the image. A cuboid is constructed outside of the object defined by voxels where each voxel has a size of 1 x 1 x 1.
- For every binary mask obtained, the cuboid is carved i.e., the locations where the seed exists is kept whereas the other locations are removed. The process requires that a projection matrix [99] be constructed and back-projected.

7. The total volume of 3-D reconstructed voxels is $\sum_{i=1}^{n} v_i$ where $v_i = 1 \times 1 \times 1$. The total volume of seed kernel is computed by multiplying by a factor which is the cubic volume of a pixel per mm in camera properties.

5.3 Improvements from the Previous Version

The current work is an iterative improvement to Cao et al., 2020 [24]. The current iteration is a significant improvement from a hardware and software perspective. The improvements are described as follows.

5.3.1 Hardware Improvements

a) **Seed Station:** Figure 5.4 shows the single seed reconstruction hardware proposed by Cao et al., 2020 [24]. The hardware proposed by the current work (shown in Figure 5.2) alters the size of the seed station and proposes one that is only large enough to hold the seed kernel. The benefit to the smaller seed station is that the seed kernel could be placed back at the location of image capture in case it gets dislodged as the images are captured. The exact location of image capture for the seed kernel is hard to determine on the larger seed station making it practically impossible to place the seed at the same location in case it got dislodged. Another improvement in the proposed hardware is the use of double-sided tape to firmly attach the seed kernel to the seed station. In the previous iteration, the seed kernel freely rests on the seed station. As a result, the seed kernel tends the move by a tiny fraction during image capture leading to misaligned images. The use of the double-sided tape ensures that the seed kernel is firmly attached to the seed station resulting in fully aligned images.

b) **Stepper Motor Automation:** In the previous iteration, the stepper motor was operated manually using Universal G-Code Sender, a desktop application that connected to the stepper motor. The operation of the stepper motor is automated in

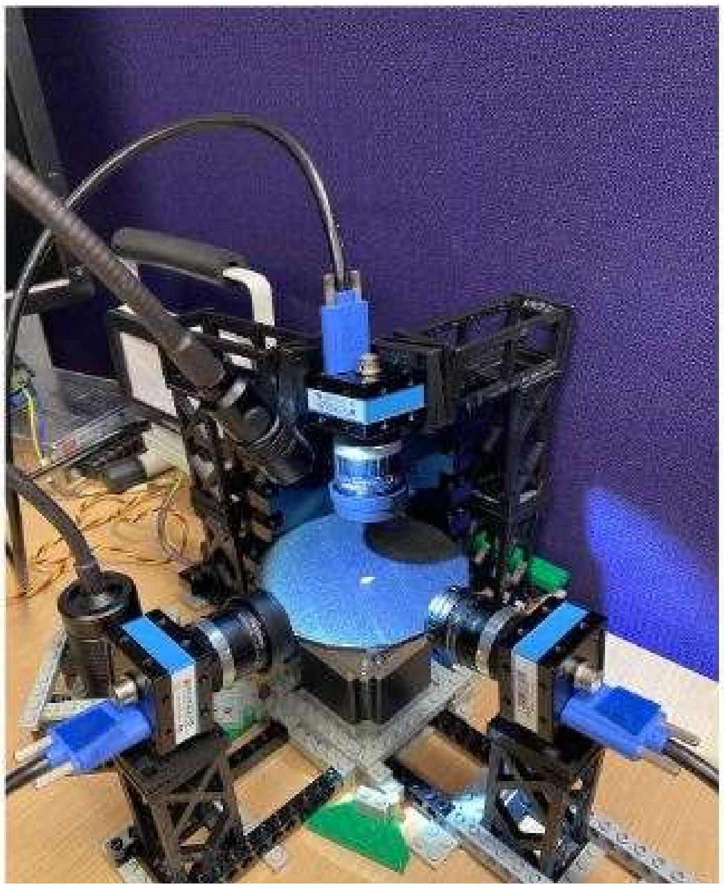


Figure 5.4: Single Seed Reconstruction Framework [24]

the current iteration wherein G-Code corresponding to a specific amount of rotation is sent to the stepper motor using Python. The input provided by the user (using the user interface shown in Figure 5.3) is processed to determine the G-Code to be sent to the stepper motor. The NEMA 23 stepper motor takes 200 steps per rotation. Hence, each step amounts to a rotation of 1.8°. It is worth noting that only rotation in multiples of 1.8° is achievable by the stepper motor.

c) **Camera Orientation:** The camera was fixed to the mount at an orientation of 10° in the previous iteration which led to an image wherein the bottom surface was not captured. The camera orientation is adjusted so that the camera is held orthogonal to the seed kernel. As a result, the bottom surface of the seed kernel is captured appropriately. Figure 5.5 shows a graphical representation of the difference in camera setup between the two iterations.

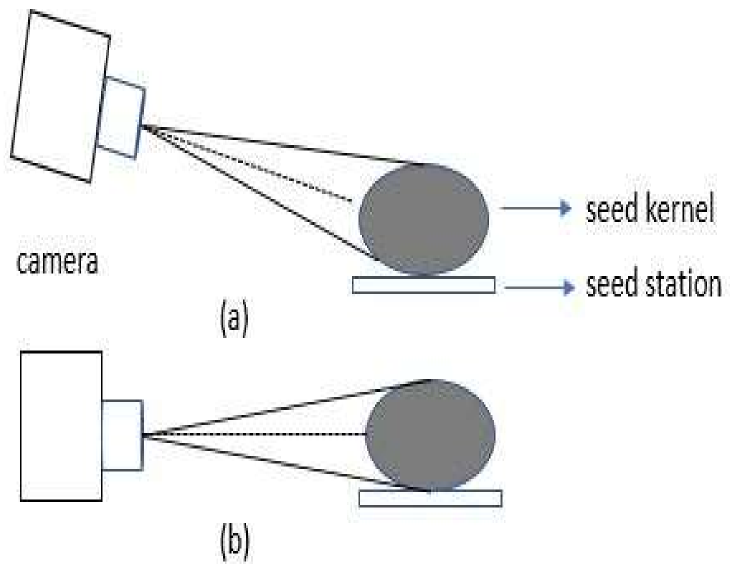


Figure 5.5: Camera Orientation in (a) Previous Iteration (b) Current Iteration

5.3.2 Software Improvements

Automated Seed Kernel Detection: In the previous iteration of the application, the user was expected to determine pixel coordinates of the most likely location of the seed kernel. As part of it, the user had to provide three rectangular regions to identify the seed kernel. They are search rectangle, template rectangle and crop rectangle. The crop rectangle is the rectangular region that closely encompasses the seed kernel. The Template rectangle is the rectangular region that encompasses the crop rectangle. The Template and Crop rectangle together help crop the seed kernel in the sense the offset between the two rectangular regions is always fixed and the Crop rectangle moves along with the Template rectangle as and when its position is changed. The search rectangle encompasses the Template rectangle and is the region within all the images where the seed is definitely found. For instance, if the image dataset contains 36 images where the seed kernel may be present at different regions of the image, the search rectangle is the rectangular region that contains the seed kernel in any image when searched. The rectangular regions are shown in Figure 5.6.

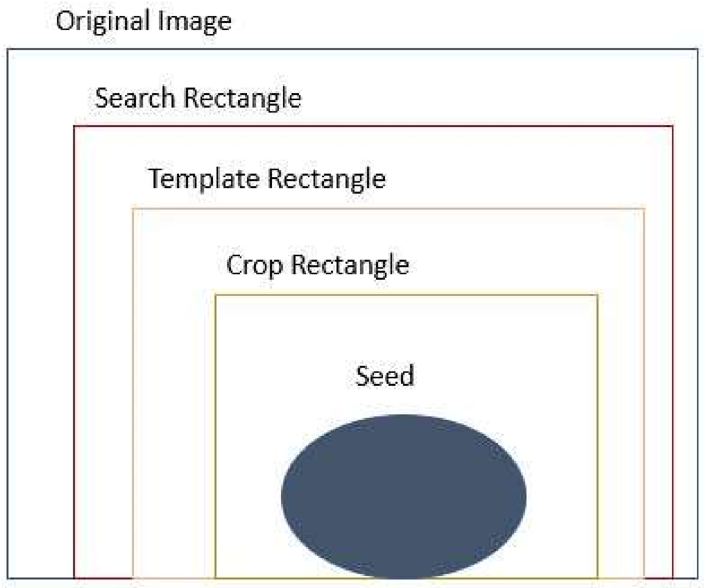


Figure 5.6: Rectangular Regions to crop Seed Kernel within Image

The three rectangular regions are specific to the experiment and anytime the location of the seed on the seed station changes, the regions are to be determined once again. Determining the criteria for the three rectangular regions is cumbersome, error prone and the expectation is unrealistic from a user perspective. The current iteration improves upon the aspect wherein it automatically determines the location of the seed kernel using image processing. The steps to determine the location of the seed kernel are as follows.

1. In each of the images acquired, detect the contour of the seed kernel and crop the rectangular region around the contour, as shown in Figure 5.7.

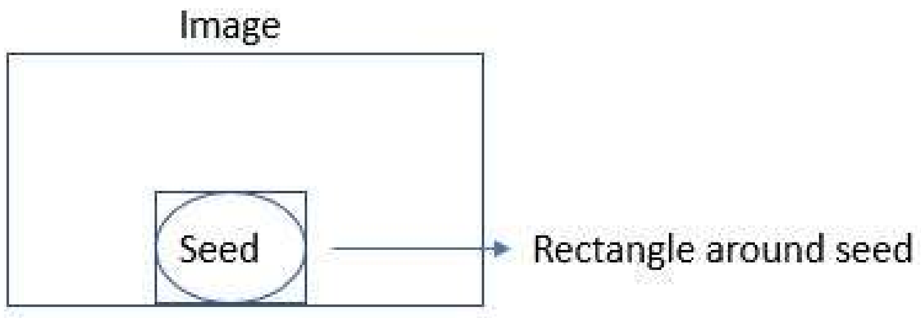


Figure 5.7: Detected Rectangle around Seed Kernel Contour

Note: A key point to note is that the size of the contour of the seed kernel within each of the images is slightly different due to the position of the seed kernel as it rotates on the seed station. Since the seed kernel rotates circularly on the seed station, assuming that the diameter of the circle is d and distance between camera and center of the seed station is x, the seed kernel is at a distance of x – d and x + d at two distinct points during the rotation. The distance x – d is the closest that the seed kernel comes to the camera and x + d is the farthest that the seed kernel moves from the camera. The seed kernel appears larger when it is positioned closer to the camera than farther. In order to align all the image on a level playing field, it is required to normalize the size of the cropped images and make their size uniform.

2. Resize all the images in the dataset to the mean size of the images containing the largest and smallest seed kernel contours.

3. Detect the contour of the seed kernel in each of the resized images (similar to step 1).

5.4 Verification Mechanism

The volume of each of the seed kernels is estimated based on the proposed 3-D reconstruction technique. However, there is no known standard technique to estimate the volume of a seed kernel. Manual measurement using a caliper is error prone since the measurements are not reproducible across different individuals and subject to bias. As a means to verify and validate that the volumetric estimation made by the 3-D reconstruction technique is within reason, the seed kernel is approximated to a mathematical shape whose volume may be estimated. While the likelihood that the shape of any seed kernel fully conforms to a mathematical shape is minimal, the approximation acts as a means to verify that the results output by the 3-D reconstruction technique are reasonable. In general, seed kernels have varying shapes which makes developing a universal model that works for all seed types. As part of the experiments for the current work, the seeds of wheat and soy are considered and are as shown in Figure 5.8.

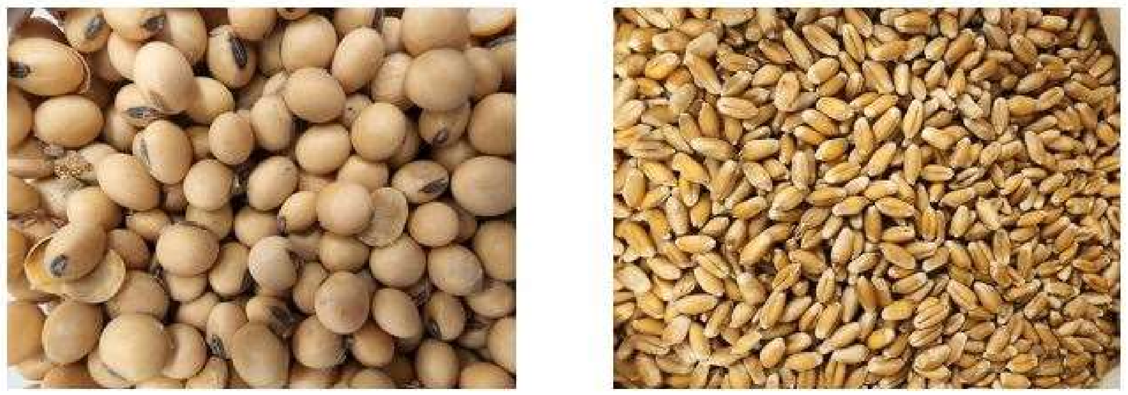


Figure 5.8: (left) Soy Seed Kernels (right) Wheat Seed Kernels

The closest mathematical shape that the seeds of soy and wheat may assume is that of the ellipsoid since it captures the properties of both spherical (soy) and oblong (wheat) seed types. The volume of a seed kernel upon being assumed to be an ellipsoid is the aggregate of the individual cross-sectional volumes. Each cross-section of the ellipsoid is an elliptic cylinder, as shown in Figure 5.9. The volume of the ith cross-section is given by $v_i = \pi.A_i.B_i.h_i$ where v_i is the volume of the ith cross-section of an ellipsoid (elliptic cylinder), A_i is the radius of the major axis and B_i is the radius of the minor axis of the ith cross-section of the ellipsoid and h_i is the height of the ith cross-section of the ellipsoid.

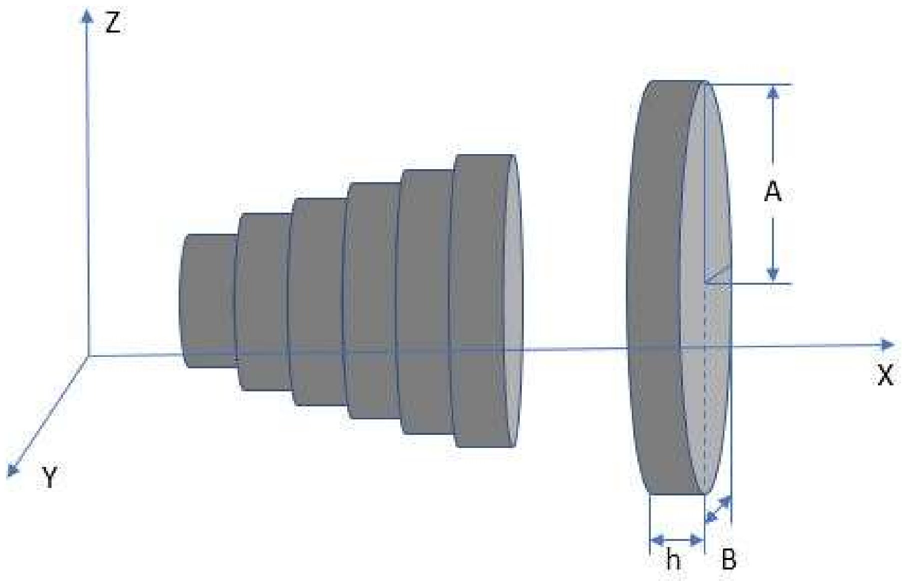


Figure 5.9: Cross-sections of Ellipsoid

The algorithm for volumetric estimation using ellipsoidal assumption is inspired by Cao, 2020 [23] and assimilated to the current work. The algorithm is described as follows:

1. Convert the original image of the seed kernel in RGB color space to HSV color space.
2. Filter out the background from the seed and color threshold the HSV image.

3. Detect the seed kernel in the image and plot the contour that encompasses it.

4. Plot a rotated (minimum area) rectangle around the seed contour. The length of the rectangle is assumed to be the length of the seed kernel.

5. Segment the seed kernel contour into n horizontal segments where n is the length of the bounding rectangle.

6. Plot a line joining the left most and right most points on the seed kernel contour.

7. Compute the mid-point on the line and determine points on the contour that are above and below the line joining left most and right most points. Store the points belonging to the top and bottom of the line in two different lists.

8. Calculate the average of the five adjacent points that belong to the top and bottom lists to make a rectangular box for each segment (from step 5).

9. Estimate the volume of each segment (cross-section) by approximating it as an elliptic cylinder i.e., $v_i = \pi.A_i.B_i.h_i$.

Finally, compute the aggregate of the volumes of each of the segments to estimate the volume of the seed kernel, i.e., $V = \Sigma^n_{i=1} \pi.A_i.B_i.h_i$.

5.4.1 Image Capture

The goal of image acquisition is to capture the seed kernel without ignoring even the slightest detail. The estimation of volume using the ellipsoid slicing technique requires the top view and side view of the seed kernel. The length and width of the seed kernel are obtained from the top view whereas the side view is required to obtain the thickness of the seed kernel. The acquisition of seed kernel images is performed using two different techniques that capture the seed kernel from the top view and side view. The hardware setup for the 3-D reconstruction technique shown in Figure 5.2 is used for image capture. The techniques are described as follows.

a) Two-Camera Image Capture: As evident from Figure 5.2, the hardware setup consists of two cameras, one at the top and another at the side, held equidistant from the seed station. The seed kernel is placed on the seed station and the image of the seed kernel is captured by both the top and side cameras. The image captured by the side camera gives the side view of the seed kernel whereas the image captured by the top camera gives the

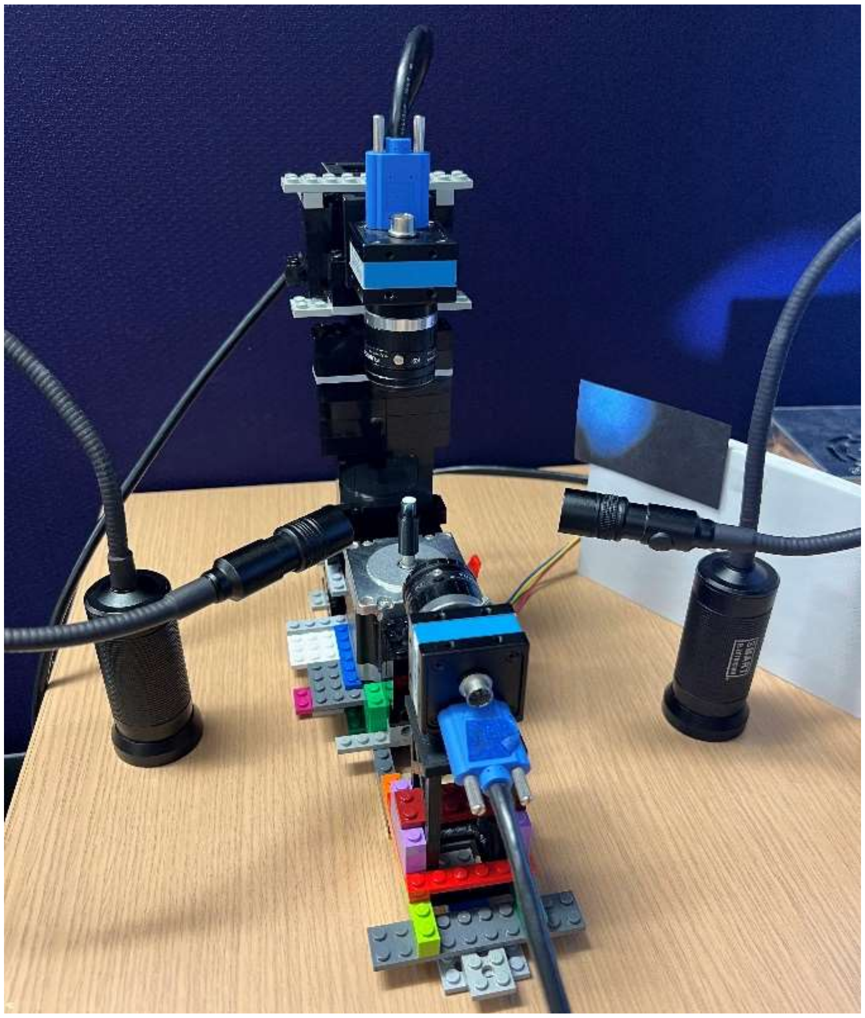


Figure 5.10: Two-Camera Image Capture of Wheat Seed

top view of the seed kernel. Figure 5.10 shows the capture of a wheat seed kernel using the two-camera image capture technique.

The images captured using the two-camera technique are put through the silhouette-based 3-D reconstruction technique to reconstruct the 3-D model of the seed kernel from images. Figure 5.11 shows the side view of the 3-D reconstruction model at 0°, 90°, and 180°. Likewise, Figure 5.12 shows the top-view of the 3-D reconstruction model at 0°, 90°, and

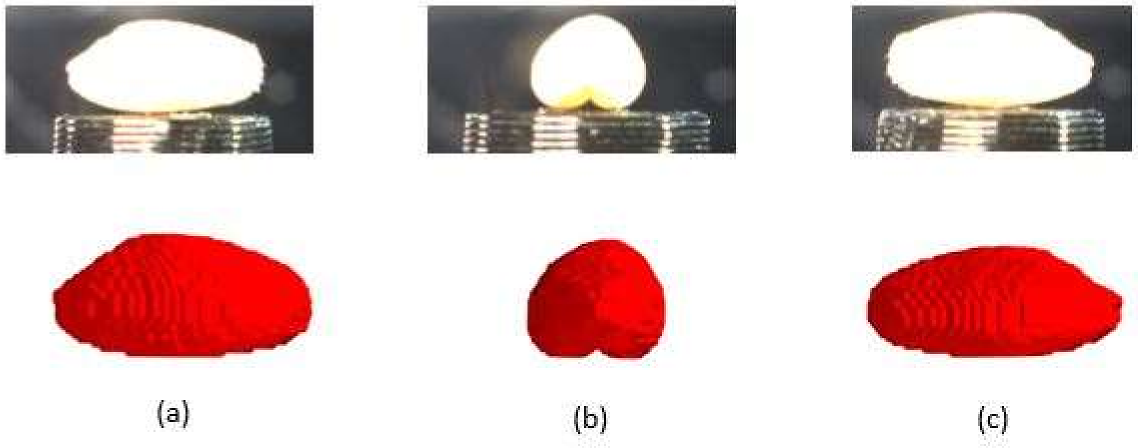


Figure 5.12: Side-view of 3-D Seed Kernel Reconstruction at (a) 0° (b) 90° (c) 180°

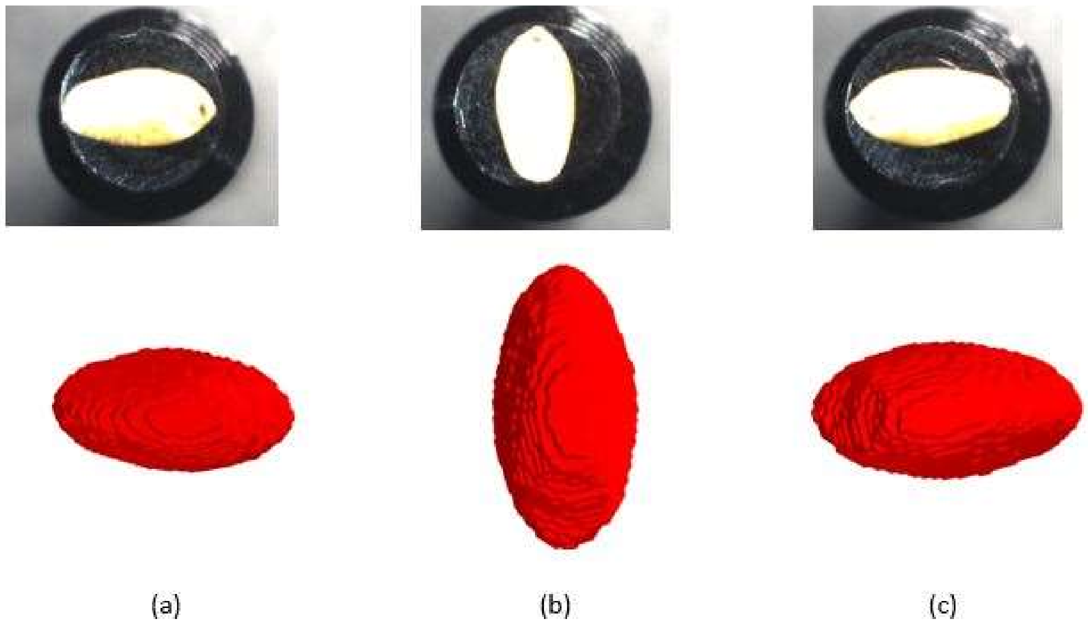


Figure 5.11: Top-view of 3-D Seed Kernel Reconstruction at (a) 0° (b) 90° (c) 180°

180°. The reconstructed model is viewed in a 3-D model viewing tool known as Mayavi. The obtained top view and side view of the wheat seed kernel are as shown in Figure 5.13. The obtained images are processed using the algorithm described in section 5.4 to estimate the volume of the seed kernel. Figure 5.14 shows the top view and side view of the seed kernel sliced into different segments of the ellipsoid.

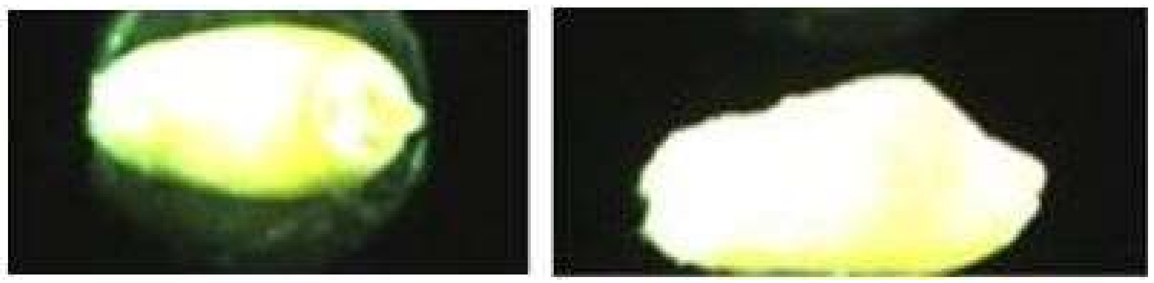


Figure 5.13: (left) Top View Wheat Seed Kernel (right) Side View of Wheat Seed Kernel

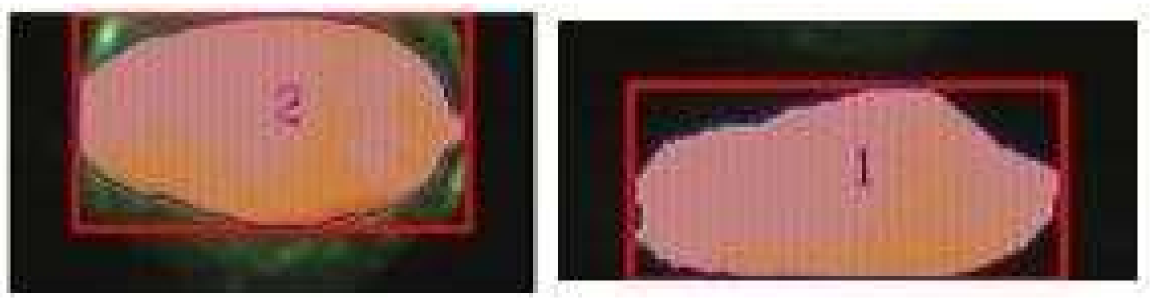


Figure 5.14: (left) Top View Sliced into Elliptic Cylinder Segments (right) Side View into Elliptic Cylinder Segments

While the detail captured by the two-camera technique is excellent, the illumination around the seed kernel causes part of the seed station to show prominently. It is observed in Figure 5.13 (left) where an illuminated circle of light is detected around the seed kernel. It requires additional processing to ensure that the circle of light is ignored from the volumetric estimation. The inclusion of the light around the seed results in inaccurate computation of the volume.

b) Mirror-based Image Capture: The mirror-based image capture technique captures the top view and side view of the seed kernel using a camera and mirror. The idea is to hold a mirror at a 45° to the seed station and capture the image of the seed kernel placed on the seed station using the side camera. Since the mirror is at a 45° to the seed station, the reflection of the seed kernel in the mirror captures the top view of the seed kernel. Figure 5.15 shows the image capture using the mirror-based technique.

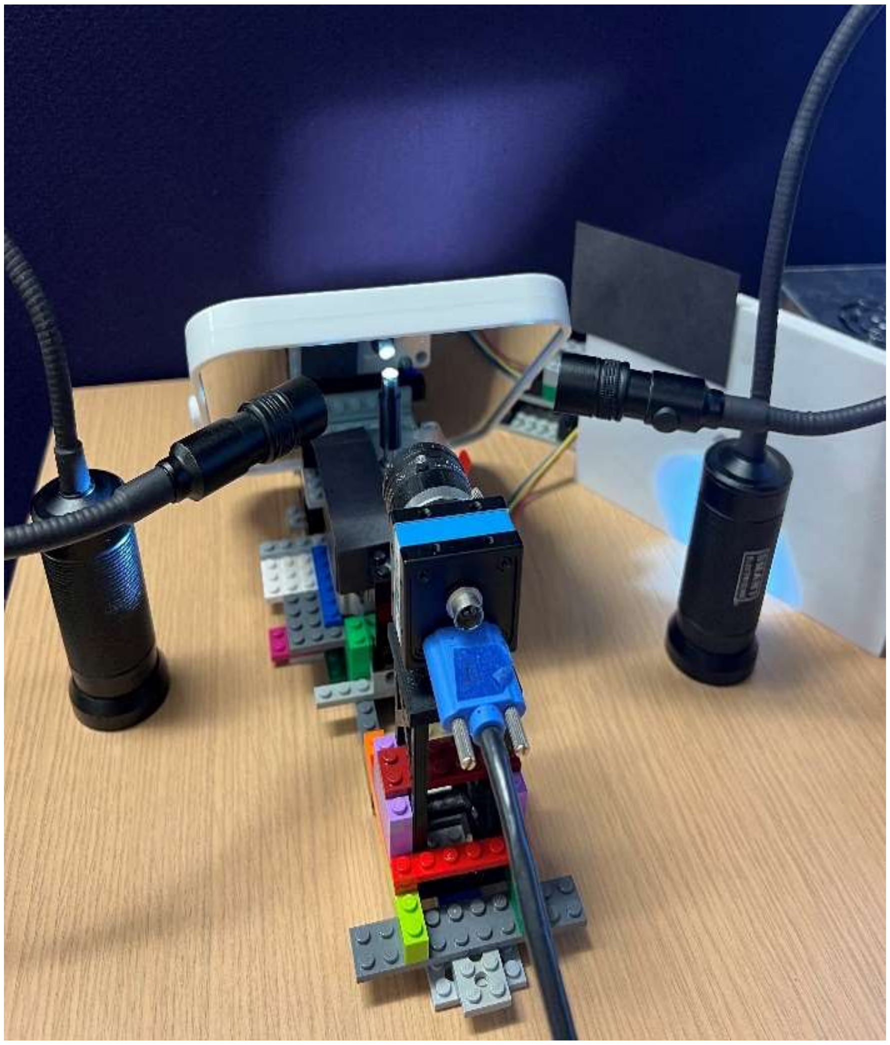


Figure 5.15: Mirror-based Image Capture of Wheat Seed Kernel

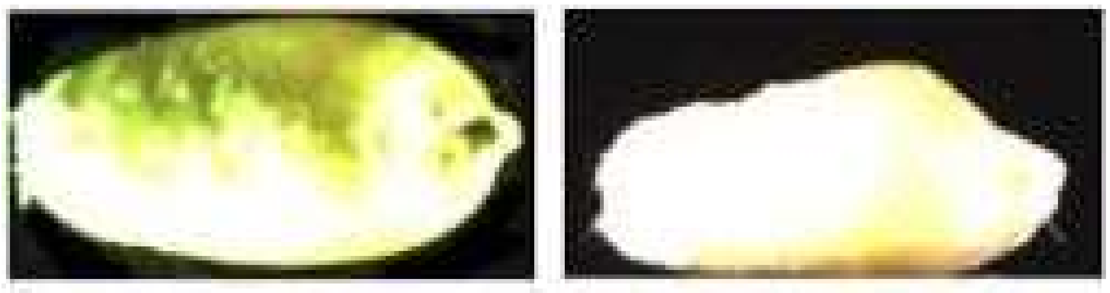


Figure 5.16: (left) Top View of Wheat Seed Kernel (right) Side View of Wheat Seed Kernel

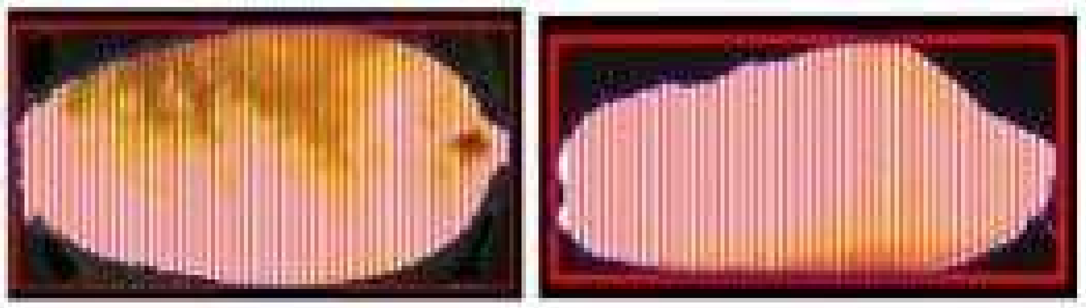


Figure 5.17: (left) Top View Sliced into Elliptic Cylinder Segments (right) Side View Sliced into Elliptic Cylinder Segments

The captured images for the top view and side view are as shown in Figure 5.16. The images are put through the algorithm described in section 5.4 to estimate the volume of the seed kernel. Figure 5.17 shows the images of the top and side views sliced into multiple segments.

5.5 Results

Experiments using the proposed 3-D reconstruction framework and ellipsoid slicing techniques are conducted on a random sample of five soy seed kernels and five wheat seed kernels. The experiment using each of the techniques is carried out for a total of 20 times to also detect any variability in the estimations. The images used for the ellipsoid slicing technique are captured using the mirror-based approach and two-camera based approach. In order to compare the volumetric estimations between the two techniques, Relative Standard Deviation (RSD) is used as the metric. The underlying concepts to understand RSD are as described further.

Mean: Mean is given the ratio of the sum of all the datapoints in the dataset and number of data samples. Assuming d = [1, 2, 3, 8] as being the dataset and n = 4 is the length of the dataset, mean is given by $\frac{\sum_{i=1}^{n} \mathrm{d_i}}{n}$ i.e. (1 + 2 + 3 + 8)/ 4 = 3.5. It gives the average of the values in the dataset.

Variance: Variance indicates the dispersion of data points around the mean. It is given the by the squared difference between each datapoint d_i of dataset d and mean, M divided by the degrees of freedom i.e., n – 1 where n is the number of datapoints in the dataset d. It is expressed as $s^2 = \frac{\sum_{i=1}^{n} (\mathrm{d_i} - \mathrm{M})^2}{n-1}$.

Standard Deviation: The square root of Variance is known as Standard Deviation. Variance being the squared value tends to be large at times being hard to calculate. Therefore, standard deviation is often considered.

Relative Standard Deviation (RSD): RSD, otherwise known as Co-efficient of Variation, is the measure of standard deviation relative to the mean of the dataset. It is a unitless measure of dispersion obtained by dividing the standard deviation by the mean of the sample. Its use is preferred in cases where a comparison on the dispersion of two datasets of different units is made. Mathematically, $RSD = s/M$ where s is the standard deviation and M is the mean of the dataset. A comparison of the RSD for each of the five wheat seeds using the mirror-based approach and 3-D reconstruction approach is shown in Table 5.2. Similarly, a comparison for five soy seed kernels is shown in Table 5.3.

Table 5.2: Standard Deviation, Average Volume and RSD for Wheat using Mirror-based and 3-D Reconstruction Approaches

Wheat						
ID	Mirror-based Approach			3-D Reconstruction Approach		
	Standard Deviation	Avg. Volume (mm^3)	Relative Standard Deviation	Standard Deviation	Avg. Volume (mm^3)	Relative Standard Deviation
1	0.318	27.602	0.011	0.13	27.912	0.004
2	0.244	29.401	0.008	0.102	30.670	0.003
3	0.443	31.642	0.014	0.142	31.645	0.004
4	0.356	27.409	0.012	0.127	28.560	0.004
5	0.296	25.897	0.011	0.108	26.742	0.004

Table 5.3: Standard Deviation, Average Volume and RSD for Soy using Mirror-based and 3-D Reconstruction Approaches

Soy						
ID	Mirror-based Approach			3-D Reconstruction Approach		
	Standard Deviation	Avg. Volume (mm^3)	Relative Standard Deviation	Standard Deviation	Avg. Volume (mm^3)	Relative Standard Deviation
1	0.638	147.918	0.004	0.3845	146.2841	0.002
2	0.605	106.057	0.005	0.3448	104.1521	0.003
3	0.973	132.855	0.007	0.375	130.1393	0.002
4	1.347	151.763	0.008	0.5594	150.6711	0.003
5	0.468	135.773	0.003	0.3283	133.4657	0.002

A comparison of the RSD for each of the five wheat seeds using the two camera-based approach and 3-D reconstruction approach is shown in Table 5.4. Similarly, a comparison for five soy seed kernels is shown in Table 5.5.

Table 5.4: Standard Deviation, Average Volume and RSD for Wheat using Two Camera-based and 3-D Reconstruction Approaches

Wheat						
ID	Two Camera-based Approach			3-D Reconstruction Approach		
	Standard Deviation	Avg. Volume (mm^3)	Relative Standard Deviation	Standard Deviation	Avg. Volume (mm^3)	Relative Standard Deviation
1	0.473	28.727	0.016	0.195	28.47	0.006
2	0.408	30.332	0.013	0.229	32.615	0.007
3	0.356	32.368	0.011	0.207	33.437	0.006
4	0.344	27.845	0.012	0.109	29.042	0.003
5	0.254	26.582	0.009	0.196	27.093	0.007

Table 5.5: Standard Deviation, Average Volume and RSD for Soy using Two Camera-based and 3-D Reconstruction Approaches

Soy						
ID	Two Camera-based Approach			3-D Reconstruction Approach		
	Standard Deviation	Avg. Volume (mm^3)	Relative Standard Deviation	Standard Deviation	Avg. Volume (mm^3)	Relative Standard Deviation
1	1.406	147.209	0.009	0.369	147.517	0.002
2	0.768	106.897	0.007	0.372	105.779	0.003
3	0.705	132.406	0.005	0.313	132.135	0.002
4	0.813	152.475	0.005	0.392	151.990	0.002

5	0.636	135.205	0.004	0.423	133.147	0.003

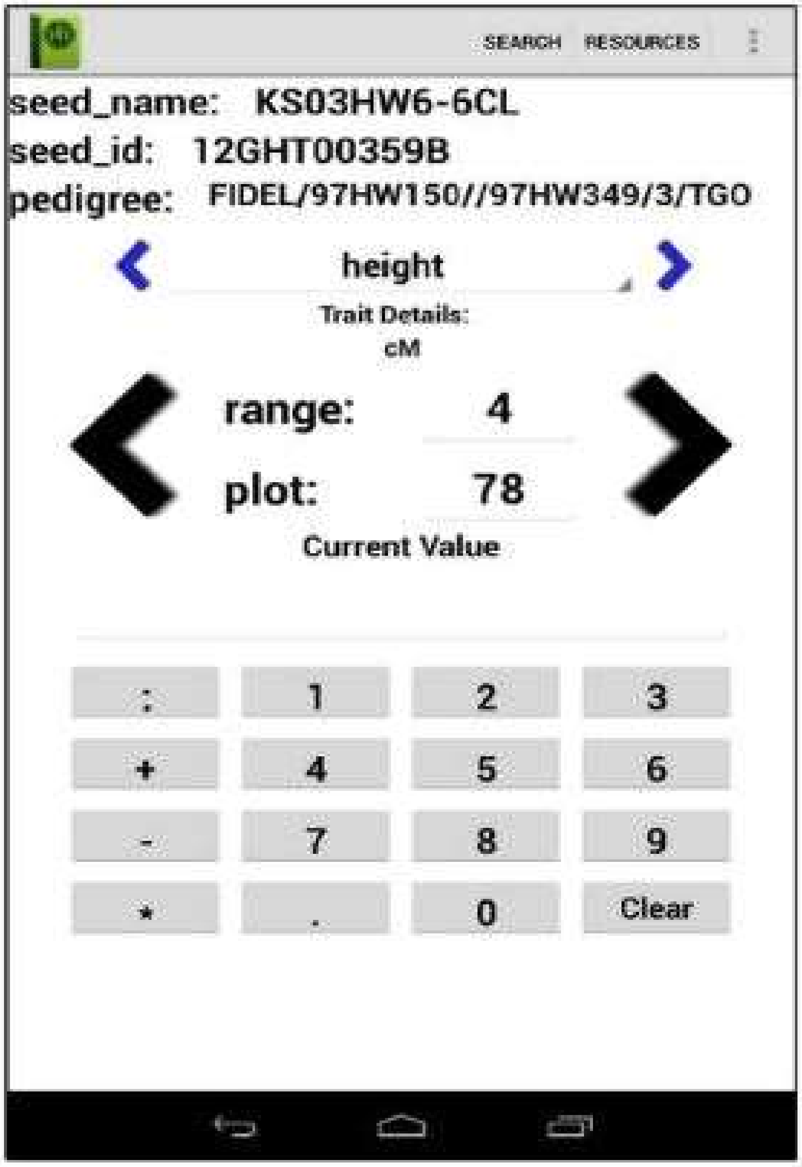
Please note that the five seeds each of wheat and soy used in the experiment are the same. However, there is a minor variation in the results of the 3-D reconstruction approach between the experiment due to variation in light intensity and position of the seed kernel on the seed station. The position and light intensity are maintained constant in a given experiment. The obtained results demonstrate superior RSD with the 3-D reconstruction approach in comparison to the two-camera and mirror-based approaches. The RSD of all seed kernels, soy or wheat, is lower for the proposed 3-D reconstruction technique in every experiment. The lower RSD indicates that lower variance and higher level of reliability for the 3-D reconstruction technique in comparison to the others.

5.6 Future Work and Conclusion

The proposed technique demonstrates an end-to-end 3-D reconstruction technique for volumetric estimation of seed kernels and its improved performance over the previous setup. In addition, the proposed technique also provides better results in comparison to volume slicing using ellipsoid approximation technique where images are captured using the two-camera and mirror-based setups. In upcoming iterations of the 3-D reconstruction setup, one of the ideas is to build a seed station with non-reflective material. Since the currently used seed station is 3-D printed using a material that reflects light, it shows in the captured images. As a result, the image processing algorithm used to process the images struggles to precisely identify the seed kernel in the image. Along the same lines, the use of a light intensity that is constant from start-to-end of the experiment is required. The gooseneck LED that is currently used is battery powered. Hence, the intensity of the light emitted changes over time as the battery is drained. Other methods to initially locate the target’s contour are to be developed in addition to the one proposed. In addition, the impact of stepper motor variation on volume estimation needs investigated. The NEMA stepper motor used in the experiment does not yield an exact ten-degree rotation. It is because each step of the stepper motor yields a 1.8-degree rotation. As a result, the rotation obtained is 10.8-degrees (1.8 x 6) but not ten which has the potential to lead to inaccuracies.

Chapter 6 - Phenotyping Algorithms for Mobile Applications

Accurate data collection and analysis is key to advance the field of digital image processing. The availability of mobile phones and portable devices with cameras is feasible means to collect image data at will. However, analysis of the collected data to make meaningful inferences requires mobile applications that have the ability to extract, process and analyze the image data. A plethora of phenotyping applications are available in the market for free as mobile applications. The applications are a boon to the agricultural community since they enable farmers and plant researchers to conduct phenotyping on the field. One such phenotyping application is FieldBook, an open-source android application that is under active development by the Poland Lab at Kansas State University. The goal of the application is to increase the speed of data collection and analysis to increase the size and accuracy of agricultural field experiments. Fieldbook moves towards the vision of one handheld device per breeder, giving every breeder access to robust data collection and management that will facilitate the development of improved varieties to enable needed gains in agricultural productivity [139]. Fieldbook in its current state displays data at an individual entry level with the capacity to navigate independently between traits and entries as shown in Figure. 6.1 [139]. The features currently available in Fieldbook include visualization of the state of data collection in the field with a field map that indicates which entries have and have not been collected for a specific trait. In addition, the map allows the user to perform a visual analysis on the data to identify outliers.



seed_name: KS03HW6-6CL
seed_id: 12GHT00359B
pedigree: FIDEL/97HW150//97HW349/3/TGO

height
Trait Details:
cM
range: 4
plot: 78
Current Value

Figure 6.1: Layout and Trait Information Displayed by FieldBook [139]

While Fieldbook is promising in its current state, there is a need for additional features to make it widely adopted in the agricultural industry. The chapter proposes two fully developed algorithms to improve upon the feature stack of Fieldbook or other mobile applications in the realm of phenotyping. They are:

1. Estimation of angle between leaf and stem

2. Estimation of plant cover.

6.1 Estimation of Angle between Leaf and Stem

The yields of maize and planting densities in the United States have increased concurrently in the past 50 years [36]. A comparative analysis of U.S. commercial maize hybrids released since the 1960s revealed that by selecting high yielding hybrids under high planting densities, breeders indirectly selected hybrids with upright leaf angles (LAs) [36]. The discovery was that upright LAs combined with higher planting densities improve light distribution within the canopy. For instance, modern hybrids intercept 14% more light than older hybrids [36]. Leaf Angle Distribution (LAD) is a key parameter that describes the structure of horizontally homogeneous vegetation canopies. It is defined as the probability of a leaf element of unit size to have its normal within a specified unit solid angle [199]. LAD affects the manner in which incident photosynthetically active radiation is distributed on plant leaves, thus directly affecting plant productivity. The traditional technique to measure LAD involves the use of mechanical inclinometer, a precision instrument that measures the angle of slope of an object with respect to its gravity by creating an artificial horizon. The use of the mechanical inclinometer makes the process of determining LAD laborious and time-consuming. Other techniques such as 3-D digitizing of individual plant elements using specialized instrumentation [154] and laser scanning [50] are available. However, the techniques are resource demanding. A feasible low-cost alternative is use of digital imagery to estimate LA since image processing preserves the state of the canopy while providing enough insight into the morphometry. Image processing tools such as ImageJ and Adobe Photoshop have built-in angle estimation tools that aid in the estimation of LA. The pitfall is that the operation of the tools is manual which means that the estimations may be subject to user bias and users end up spending extended periods of time to estimate LA in case of a large dataset. In order to overcome the pitfalls of the manual setup, the current chapter proposes an automated technique for LA estimation using convolutional neural networks (CNN) and image processing.

6.1.1 Related Work

Dzievit [36] conducted genetic mapping and a meta-analysis to dissect genetic factors controlling LA variation on maize. Genetic mapping populations were developed using inbred lines B73 (Iowa Still Stalk Synthetic), PHW30 (lodent, expired plant variety protection inbred), and Mo17 (Non-Stiff Stalk) that have distinct LA architectures. The leaf angles were estimated using the ImageJ image processing tool.

Herbert [53] described a technique for the estimation of leaf angles using stereo-photogrammetry. The technique permitted accurate measurement of leaf angle and position from several meters away and had sufficient resolution to permit the analysis of complex phenomena such as the effect of leaf shape upon interception of light and photosynthesis.

Sinoquet et al. [155] proposed a method to measure light interception by vegetation canopies using a 3-D digitizer and image processing software. The 3-D digitizer allowed for simultaneous acquisition of the spatial coordinates of leaf locations and orientations. The software for image synthesis also had the ability to make virtual photographs of the real canopy. The information on light interception was derived from virtual images by using the simple features of image analysis software.

Hosoi et al. [50] estimated the LAD of wheat canopy at different growth stages such as tillering, stem elongation, flowering, and ripening stages by using a high-resolution portable scanning lidar. The canopy was scanned three-dimensionally by optimally inclined laser beams emitted from several measuring points surrounding the canopy and 3-D point cloud images in each stage were obtained. After co-registration of lidar images between different measurement positions, leaves were extracted from the images and each leaf was divided into small pieces along the leaf-length direction. Each of the pieces was approximated as a plane, to which normal were estimated. The distribution of the leaf inclination angles was derived from the angles of the normal with respect to the zenith.

Ryu et al. [135] examined the feasibility of seven techniques such as litterfall, allometry, LAI-2000, TRAC, digital hemispheric photography, digital cover photography, and traversing radiometer system to determine leaf area index across a 9-ha domain in an oak-savanna ecosystem in California, USA. It was shown that the combination of digital cover photography and LAI-2000 could provide spatially representative leaf area index, gap fraction and element clumping index.

6.1.2 Materials and Methods

Researchers at Iowa State University captured the images of several plants in a field using the Fieldbook android application. As for the current work, two datasets namely, Summer_2015-Ames_ULA and Summer_2015-Ames_MLA wherein the former consists of 924 images and latter, 969 images, are used as the experimental datasets. Each of the images in either dataset shows the intersection of the leaf and stem. A sample image is a shown in Figure 6.2.

The proposed leaf angle estimation procedure contains two broad steps:

1. Extraction of region of interest (ROI)
2. Estimation of Leaf Angle

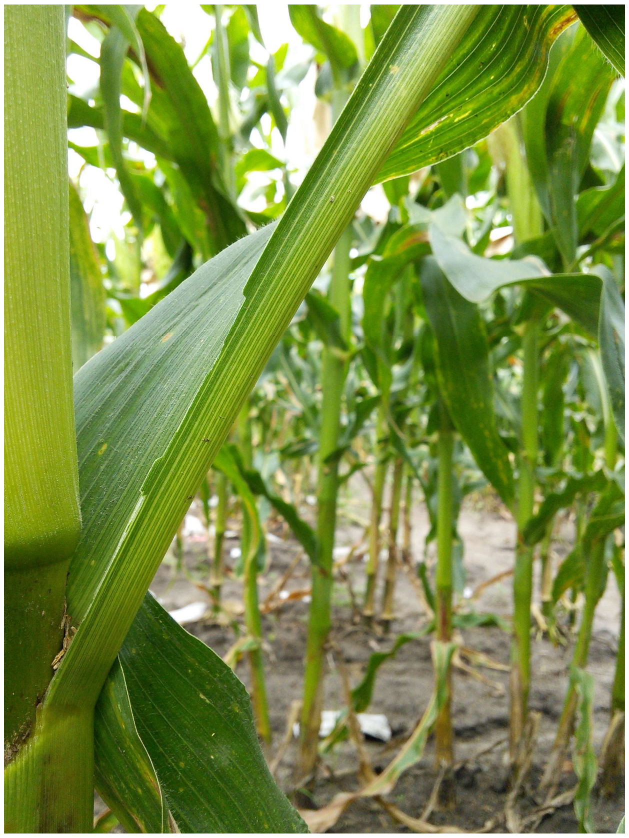


Figure 6.2: Leaf-Stem Image Captured using Fieldbook

6.1.3.1 Extraction of Region of Interest (ROI)

The first step in the extraction of ROI is the detection of leaf and stem within the image. The images captured in the field contain numerous plants that look similar. Hence, it is important to determine the plant of interest and perform a curated extraction of it for the estimation of LA. Such an extraction may be performed using the semi-automated image cropper described in Chapter 4 by plotting a polygon around the leaf-and-stem portion of the image. However, such an extraction works well in the case of relatively small sized datasets but isn't feasible for larger datasets such as the one at hand. It is determined from empirical observation of the dataset that the plant of interest is always the foreground object present either to the left or right of the image. Another feasible alternative is the use of foreground extraction techniques such as OpenCV's GrabCut algorithm. However, the results obtained upon the application of the GrabCut algorithm to each of the datasets are not satisfactory. While the algorithm captures the foreground, it also captures part of the plants in the background. The captured plants in the background are essentially noise and detrimental to LA estimation. Figure 6.3 shows the botched foreground extraction performed on a plant image from the Summer_2015-Ames_MLA dataset.

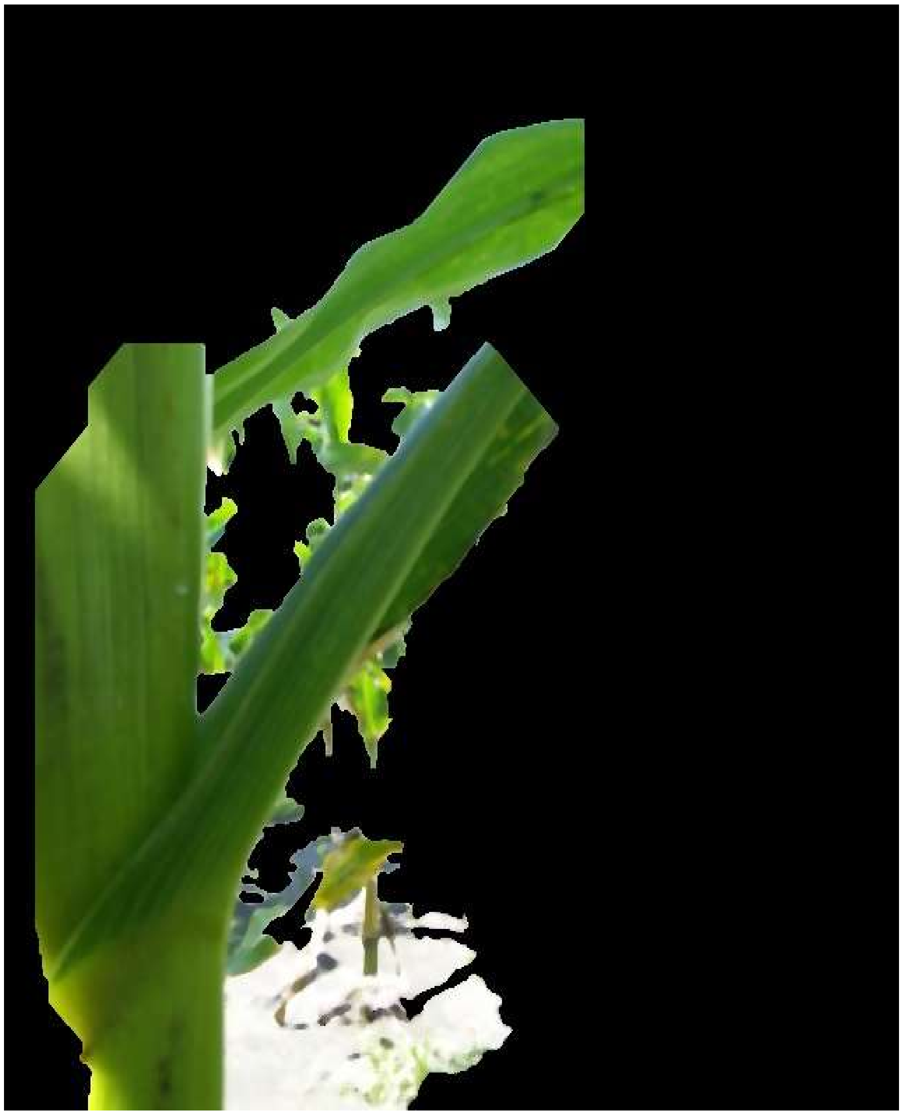


Figure 6.3: Foreground Extraction using GrabCut Algorithm

In order to better perform foreground extraction, the CNN model of Mask R-CNN is employed to extract foreground and identify the region of interest appropriately. Mask R-CNN is a deep neural network aimed to solve the problem of instance segmentation in computer vision. Instance Segmentation is the task of precisely identifying the pixels of each of the objects in the image. It is perhaps the hardest and most precise of the vision tasks of classification, semantic segmentation, object detection, and instance segmentation. In order to better understand the vision tasks, consider the image with balloons shown in Figure 6.4. Classification is the task of merely detecting that the image contains balloons. Semantic segmentation is the task of detecting all the balloon pixels within the image.

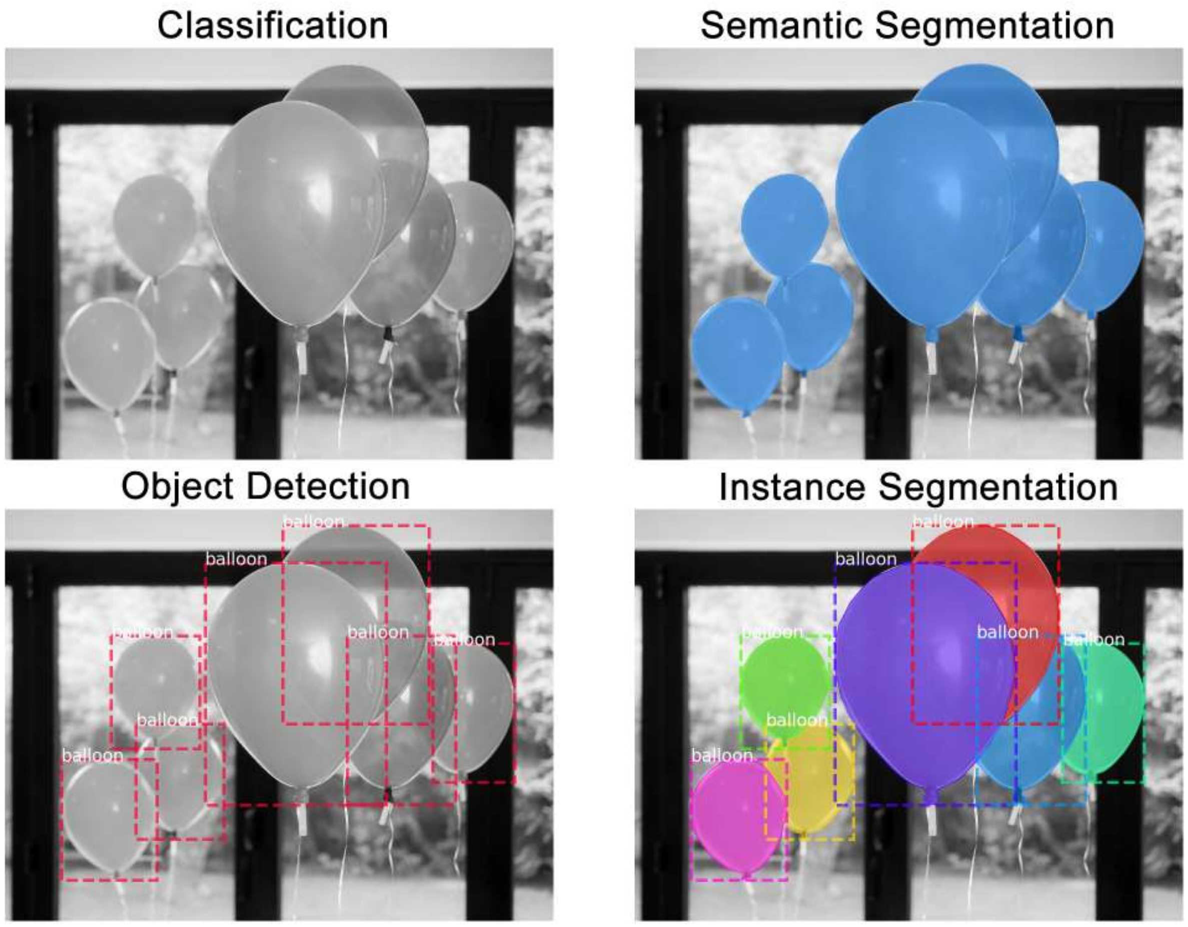


Figure 6.4: Different Computer Vision Tasks [135]

Object detection is the task of detecting the number of entities of a certain kind. In other words, it is the task of identifying that the image contains seven balloons. Instance segmentation combines the tasks of semantic segmentation and object detection wherein it precisely identifies the pixels that belong to each of the entities within the image. Hence the reason, each of the balloon images is applied a unique color. The architecture of Mask R-CNN is explained in section 3.2.3.1 elaborately. Please refer to it for an in-depth understanding of the architecture.

1. 175 images belonging to each of the Summer_2015-Ames_MLA and Summer_2015-Ames_ULA datasets are manually annotated using makesense.ai tool. The tool is free to use and generates annotations in the COCO format that Mask R-CNN is able to consume. A polygonal annotation is drawn where the stem and leaf meet each other. Figure 6.5 shows a polygonal annotation drawn using makesense.ai on an image with leaf and stem.

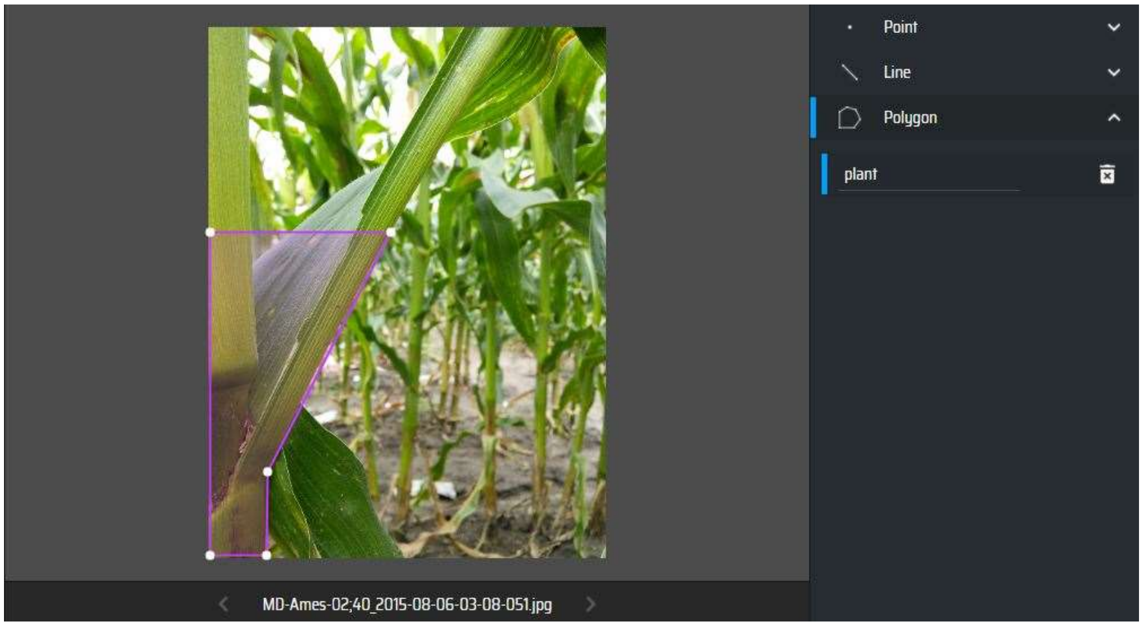


Figure 6.5: Polygonal Annotation Plotted using MakeSense.ai

2. The annotated images are trained on the Mask R-CNN network for a total of 20 epochs. The loss of after 20 annotations is 0.034. However, the loss is not significant since Mask R-CNN is being used to extract the portion of the image where the leaf and stem meet but not to extract masks that are precise in shape and size. All the annotated images are used for training only. The dataset is not split into training, validation and test datasets since the goal is not to evaluate the performance of Mask R-CNN but instead to extract regions of interest in the image that are later processed to estimate the angle between leaf and stem. Upon training, the trained weights are used to extract the masks of each of the images in the Summer_2015-Ames_MLA and Summer_2015-Ames_ULA datasets. Figure 6.6 shows the mask extracted by Mask R-CNN for the plant image shown in Figure 6.5.

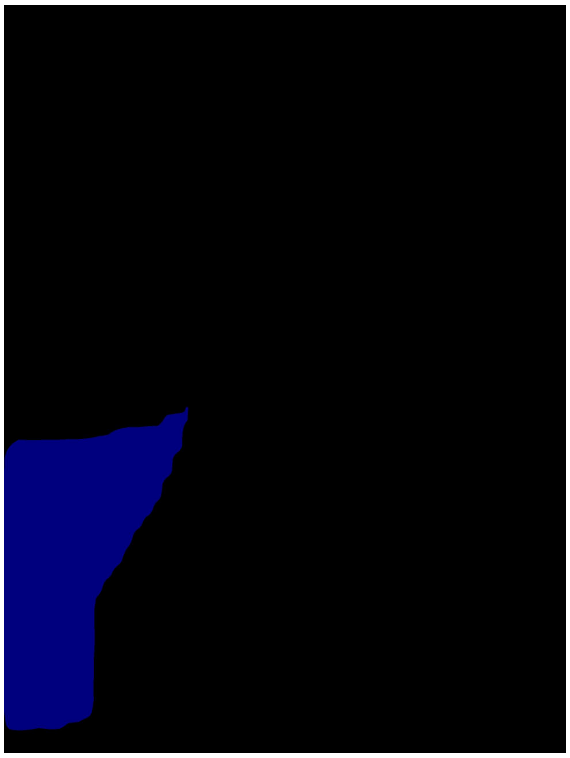


Figure 6.6: Mask Predicted by Mask R-CNN

3. The predicted mask indicates the region of interest that corresponds to the leaf and stem. In order to extract the region from the original image, the bitwise and operation is applied on the mask predicted by Mask R-CNN and original image. The resulting image gives the region of interest in the original image. Figure 6.7 shows the region of interest extracted from the original image.

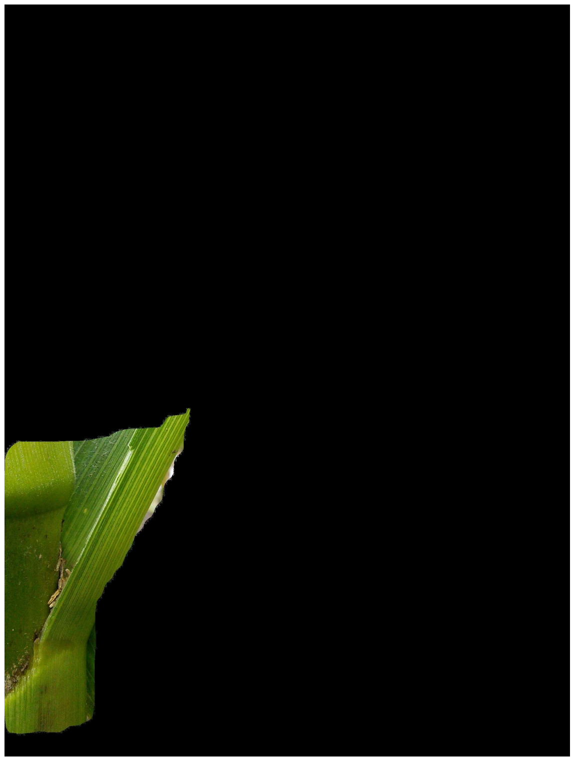


Figure 6.7: Extracted Region of Interest from Original Image

6.1.3.2 Estimation of Leaf Angle

The extracted region of interest provides enough information for further processing to estimate the angle between leaf and stem. The basis for the estimation of leaf angle is provided by [36]. The work describes the above ear leaf and below ear leaf angles that are measured by plant scientists. Figure 38, an excerpt from [36], shows the exact position at which the angle between leaf and stem is to be measured. While leaves are typically long and tilt in different directions owing to environmental factors, the angle between leaf and stem is measured exactly where the leaf and stem connect to each other.

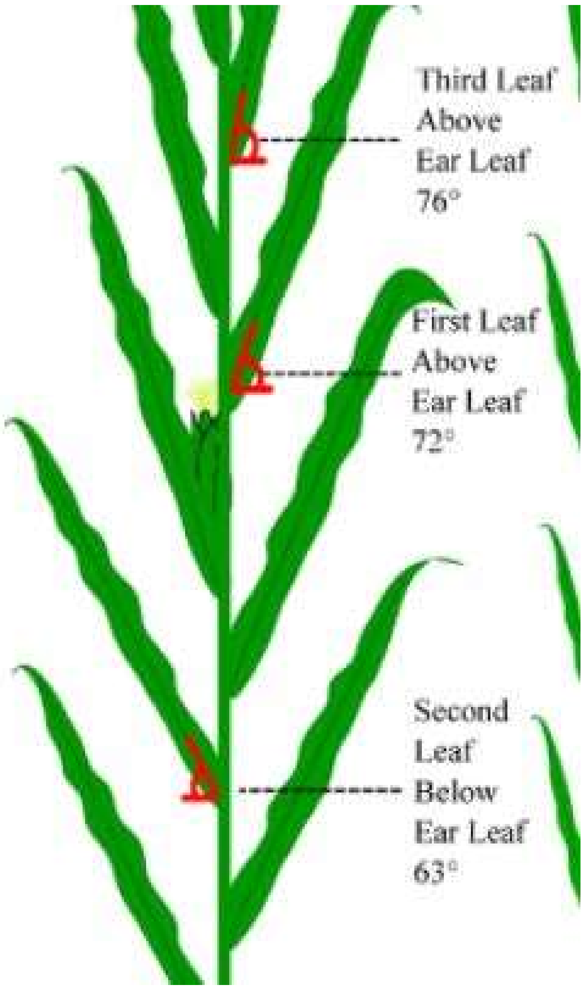

Third Leaf Above Ear Leaf 76°
First Leaf Above Ear Leaf 72°
Second Leaf Below Ear Leaf 63°

Figure 6.8: Above Ear and Below Ear Leaf Angles [36]

The next step upon the extraction of region of interest from the image is to detect the point of contact of leaf and stem. Such a detection is complex considering the transition from the stem to leaf is rather smooth than abrupt. Upon empirical observation of the dataset, it is determined that the majority of the images in the dataset consist of straight lines that run along the length of the leaf. The lines are visible when the image is zoomed in. Figure 39 (zoomed-in Figure 37) shows the lines on the leaf of the plant.

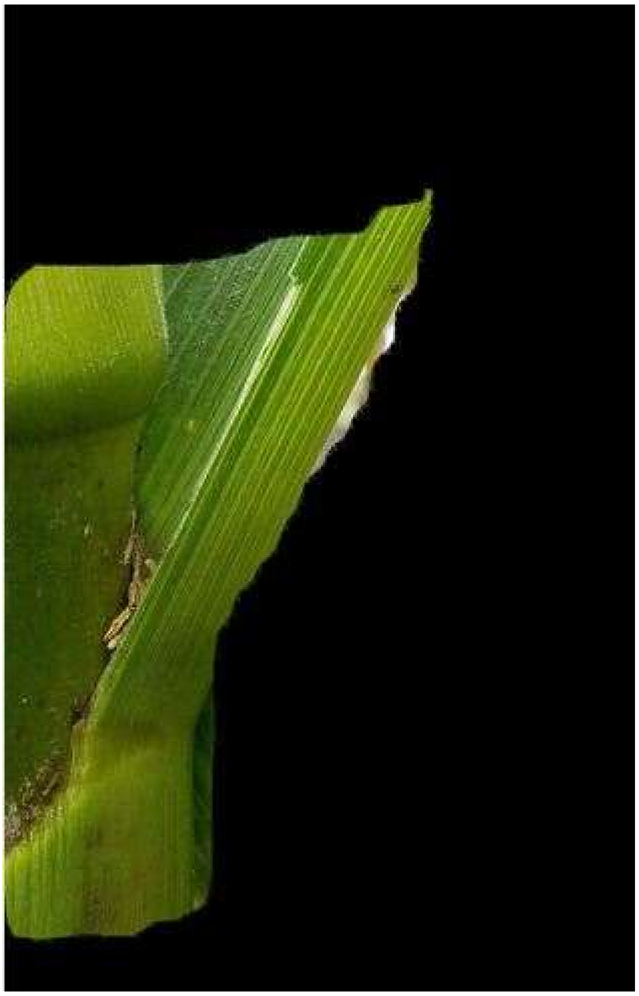


Figure 6.9: Straight Lines on Leaf

Another empirical observation made is that the straight lines along the leaf are parallel to each other. Hence, the straight lines are used as indicators to determine the angle between leaf and stem under the assumption that the base of the stem is perpendicular to the ground. In other words, the angle between leaf and stem is the angle is determined as the angle made by the straight lines that run along the leaf with the horizontal.

Hough Transform

The detection of straight lines in images is done by applying the Hough Lines transform. Typically, straight lines are represented using the slope-intercept method as $y = mx + c$

where m is the slope and c is the y-intercept. However, Hough Transform relies on representing straight lines using the pair of polar coordinates, (ρ, θ). The first parameter, ρ, is the shortest distance from the origin to the line. The second parameter, θ, is the angle between the x-axis and distance line. Figure 6.10 shows the representation of a line in polar coordinates.

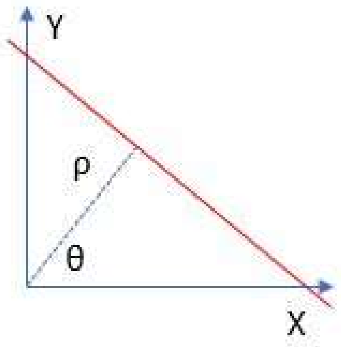


Figure 6.11: Straight Line in Polar Coordinates

The equation of the straight line in polar coordinate system is given by, ρ = x * cos(θ) + y * sin(θ) where (x, y) is a point on the line. A straight line in image space is represented by a point in Hough space, as shown in Figure 6.11.

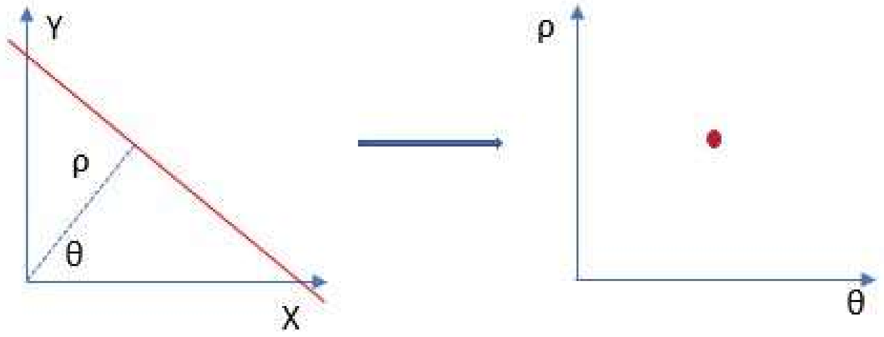


Figure 6.10: Representation of Straight Line in Image Space (left) and Hough Space (right)

Likewise, a group of intersecting lines in the image space form a group of points in the Hough space that appear in the shape of a sinusoid, as shown in Figure 6.12. An infinite number of straight lines in the image space forms a continuous sinusoid in the Hough space. A group of points that represent a straight line are represented by multiple sinusoids that intersect at a common point. In order to detect points that form straight lines in the Hough space, points of sinusoid intersection are to be detected.

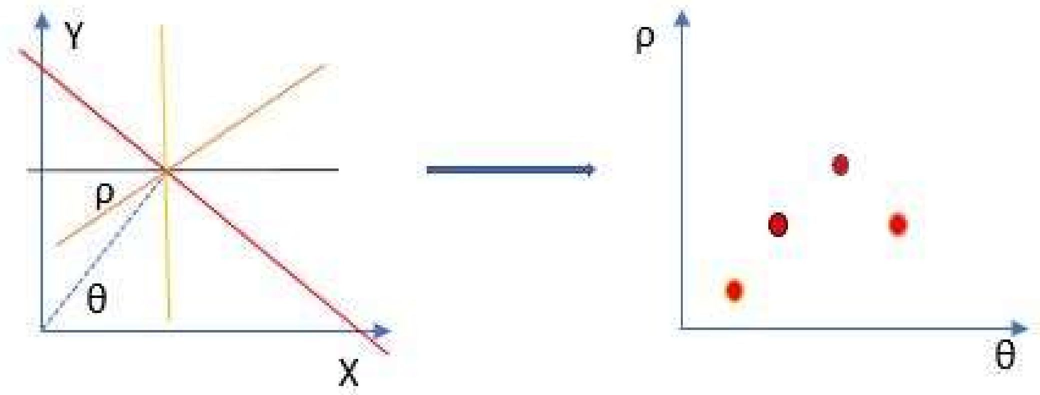


Figure 6.12: Intersecting Lines in Image Space (left) as Sinusoid in Hough Space (right)

Standard and Probabilistic Hough Transforms

The Hough Transform takes the edges from a binary image as input. The Hough Transform maps each of the pixels to multiple points in Hough (or parameter) space. An edge pixel is mapped to a sinusoid in 2D parameter space, (ρ, θ) representing all possible lines that could pass through the point. It is referred to as the voting stage. The sinusoids of collinear points in the Hough space cross each other indicating collinearity. There are two variants of Hough Transform, Standard and Probabilistic [67]. The primary difference between the two variants is the computational complexity. Consider an image that has M pixels as edge points and a Hough space divided in N_ρ x N_θ accumulators. In case of the Standard Hough Transform, each of the M edge pixels is used for computation which means that the computational complexity is $O(M. N_\theta)$ for the voting stage and $O(N_\rho. N_\theta)$

for the search stage. However, in the Probabilistic Hough Transform, only a subset m of M edge pixels is used for computation which means that the computational complexity is reduced to O(m. N_θ) for the voting stage resulting in a faster execution of the algorithm.

The algorithmic steps to estimate leaf angle from the extracted region of interest are as shown in Figure 6.13 and described as follows.

1. Convert the input image (extracted region of interest) from RGB color space to grayscale.
2. Detect contours on the image to identify the location of the plant in the image. It is possible that multiple contours are detected. In such a case, filter the contours to identify the largest contour in the image since the largest contour in the image most likely belongs to that of the plant.
3. Detect the direction in which the leaf shoots from the stem using the moments of the contour. The moments of the contour help to determine the center of the contour. In case the center is present in the right-half of the image, the leaf shoots to the left whereas the leaf shoots to the right if the center of mass is present in the left-half of the image. The knowledge of direction is important since the slope of the straight lines on which the algorithm is based depends

upon the direction of leaf from the stem. The slope of the lines on the leaf is negative if the leaf shoots from left to right in the image, vice-versa otherwise.

4. Detect edges on the grayscale image from step 1. The edge image is required as input for the probabilistic Hough lines algorithm.

5. Apply probabilistic Hough lines transform on the edge image and detect all the lines on the image. The application of probabilistic Hough transform requires the specification of a vote range i.e., the level of confidence that the probabilistic Hough lines algorithm has that a group of points forms a straight line. The vote range for the current image analysis is set to 180 and lowered by 10 on the subsequent iteration in case no lines are detected by the probabilistic Hough lines function in the current iteration. Other parameters that are input to the algorithm are minimum line length i.e., minimum length that any detected line is required to be and maximum line gap i.e., the maximum gap between two lines in order for them to be considered separate lines. If the gap is lower, the lines are merged together. A value of 10 for minimum line length and 50 for maximum line gap are used for the majority of the images in the dataset. Since the parameters are tunable, a universal value that works for any image is hard to define. It is required to be evaluated on a case-by-case basis.

6. In case no lines are detected when vote range reaches zero, it means that the algorithm is not able to detect any straight lines. At that point, the algorithm is terminated. However, if lines are detected, a filter to retrieve the lines specifically on the leaf is applied. The probabilistic Hough lines algorithm determines all the straight lines that meet the criteria, and it is the case that

several of the detected lines indicate the stem rather than the leaf. The lines on the stem are irrelevant and need to be eliminated from the processing. Two constraints, one on slope and another on proximity to image boundary are placed on the detected straight lines to filter out the ones that don't belong to the leaf.

a) **Slope Constraint:** The leaf is generally oriented in comparison to the stem that is almost perpendicular to the ground in the majority of cases. As a result, lines whose orientations are between 80 and 90 degrees are ignored since they are likely to belong to the stem but not the leaf.

b) **Boundary Proximity Constraint:** It is determined empirically that the lines that belong to the leaf are at least 100 px apart from the image boundary. The lines that are closest to the image boundary are those of the stem. Hence, the constraint to ignore lines that are under 100 px in distance to the image boundary works to ignore lines that belong to the stem.

Both constraints are required to be satisfied for a line to be ignored. In certain images that orientation of the leaf is between 80 and 90 degrees. In such cases, without the boundary constraint, the lines pertinent to the leaf are ignored. The lines that pass the filter are assumed to be present on the leaf.

7. Compute the slope of each of the lines and find the mode of the slopes. Mode refers to the most frequently occurring item among a list of items.

8. Retrieve the lines with the most frequently occurring slope for further processing. Only the lines with the orientation of the mode are considered to eliminate any stray lines that may be detected and present on the leaf.

9. Compute the orientation of the lines made with the horizontal as the tan inverse of slope i.e., angle = $\tan^{-1}$(slope). Since all the lines have the same orientation,

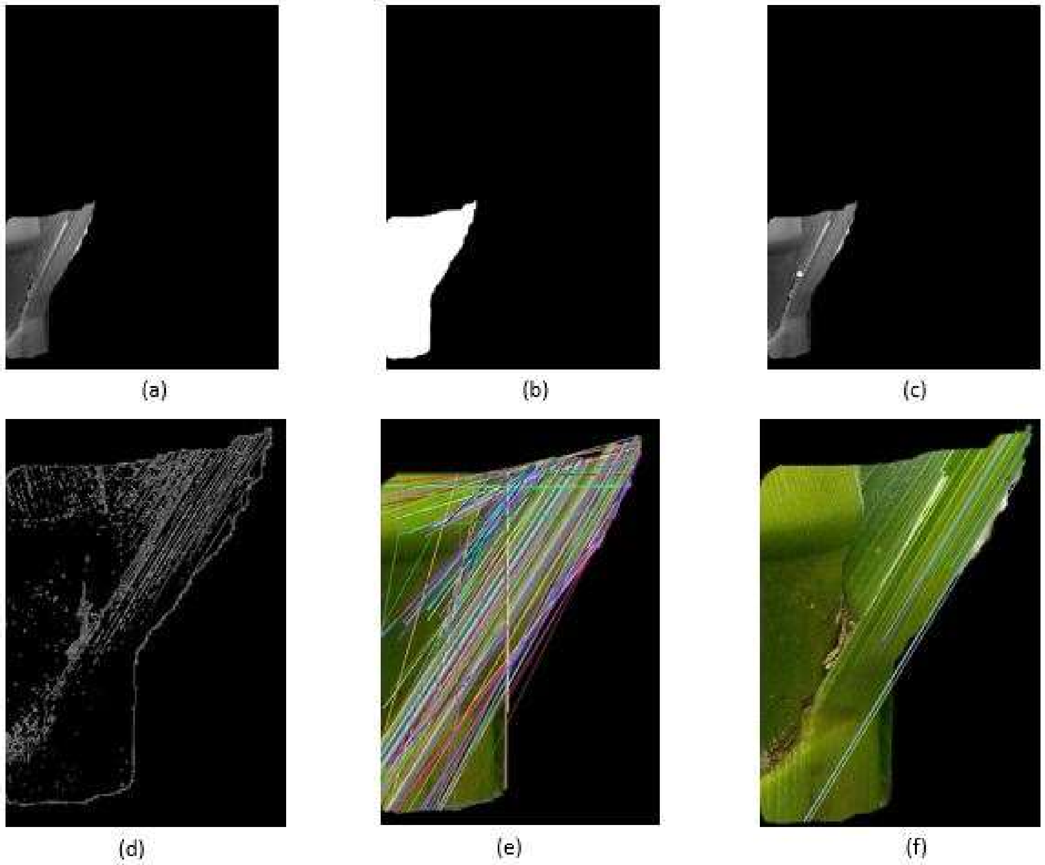


Figure 6.13: (a) Grayscale Region of Interest Image (b) Contours of Region of Interest (c) Center of Contours Plotted on Grayscale Image (d) Edges of Contours (e) All Straight Lines Detected by Probabilistic Hough Lines Algorithm (f) Filtered Straight Lines on Leaf

it is sufficient to pick one line and compute its orientation. The orientation made by the line is deemed the orientation of the leaf with the stem.

Alternate Approach: Another approach that is an adaptation of the approach previously described is as follows.

1. The algorithmic steps until step 6 are the same as described above. The proposed approach deviates from step 7.
2. Upon the determination of lines that belong to the leaf, compute the median of the orientations of lines pertinent to the leaf. The orientations are given by the tan inverse of the slope of the lines.
3. Apply an orientation threshold (under five degrees is recommended) and calculate upper bound and lower bound of orientation where upper bound = median orientation + orientation threshold and lower bound = median orientation – orientation threshold.
4. Apply the upper bound and lower bound as filtering criterion and identify the lines whose orientations lie within the bounds of orientation. The retrieved lines have an orientation that is within x degrees of each other where x is the orientation threshold.
5. Merge the lines that are close to each other by a certain orientation threshold. In other words, apply an orientation threshold (smaller than the orientation

threshold applied in step 3) and create a single line segment whose orientation is the average of each of the line segments within the threshold.

6. Repeat step 5 by increasing the orientation threshold each time until only a single line segment is left.
7. The orientation of the line segment is the orientation between the leaf and stem.

Note: While both approaches provide quality outcomes, the alternate approach is more complex to understand and implement.

6.1.4 Validation Mechanism

A mechanism to ensure that the estimation made by the aforementioned algorithm is devoid of flaws is developed. The idea is to split the region of interest image into two and consider the half that is farthest from the image boundary. The algorithm is applied on the half of the image farthest from the image boundary. In theory, the estimation made by the algorithm on the original region of interest image and the cropped image is the same (or highly similar). Figure 6.14 (a) and 6.14 (b) show an instance where the region of interest and cropped region of interest yield the same result upon the application of the algorithm. A tolerance threshold which indicates the allowable discrepancy in estimations is defined. Any estimations that violate the tolerance threshold may be manually evaluated and corrected. Figure 6.14 (c) and 6.14 (d) show an instance where the estimations on region of interest and cropped region of interest images yield results that exhibit a high degree of variance.

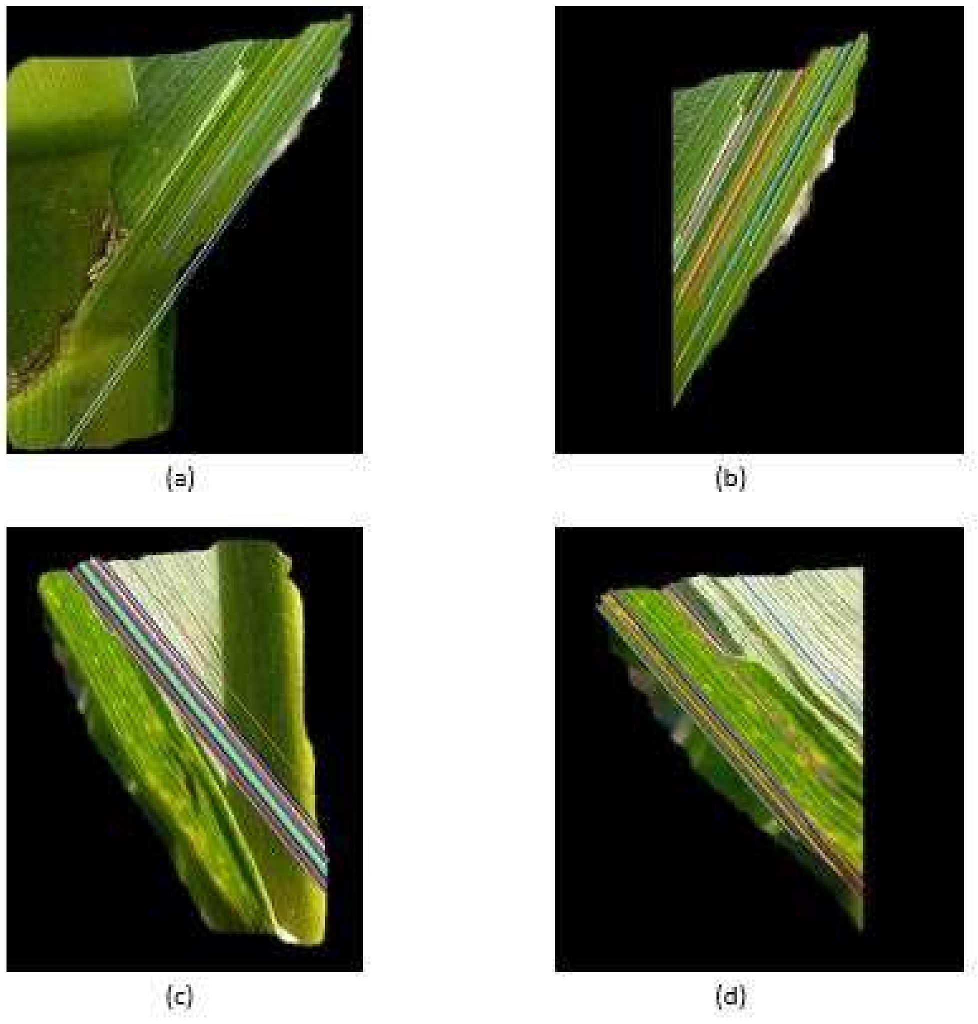


Figure 6.14: (a) Uncropped Image and (b) Cropped Image Showing Identical Orientation Estimates (c) Uncropped Image and (d) Cropped Image Showing Varying Orientation Estimates

6.1.5 Impact of assuming a Vertical Stem

The angle estimated between the leaf and stem assumes that the stem is a vertical line. While stems are close to being vertical, they are not always at a 90°. The estimated angle is closest to the actual when the stem is close to or exactly 90°. However, the more deviant stem is from being a vertical, the more deviant the estimated angle is from the actual.

6.1.6 Impact of Image Sharpness

A key factor that impacts the outcome of the algorithm is image sharpness. Sharpness describes the clarity of detail in a photo. The aspects of resolution and acutance primarily impact sharpness of an image. Since the proposed algorithm relies on detecting lines that are present along the leaf, it is imperative that the lines on the leaf be sharp so the lines may be identified easily. Figure 6.15 shows the algorithm applied on the original and sharpened version of the original image. The lack of sharpness leads to the algorithm's faulty behavior on the original image whereas the algorithm correctly detects lines along the leaf on the sharpened version of the image.

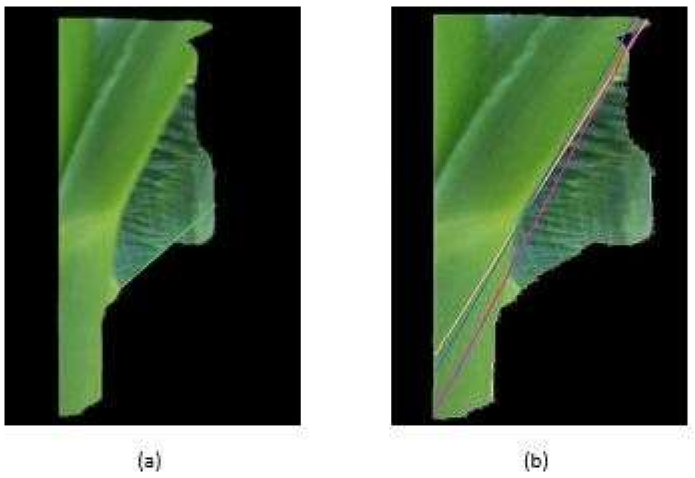


Figure 6.15: (a) Original Image (b) Sharpened Image

The OpenCV-Python library is used to increase the sharpness of the images. The key parameter that[A24] is required is the kernel. A kernel $\begin{matrix}0 & -1 & 0\\ -1 & 5 & -1\\ 0 & -1 & 0\end{matrix}$ of shape 3 x 3 is used to sharpen an image. It is empirically determined that a sharpness of 0.7 or higher is desired for efficient functioning of the algorithm.

6.1.7 Multiple Leaf and Stem Detections

In certain cases, the trained Mask R-CNN model detects multiple leaf and stem instances on a given image. Figure 6.16 shows two instances where multiple leaf and stem instances are detected. The default behavior of the algorithm is to consider the largest contour in the image as the valid instance. However, it may be flawed in cases where the misidentified instance is larger than the actual leaf and stem instance. The validation mechanism defined prior is able to detect the failures due to multiple instances.

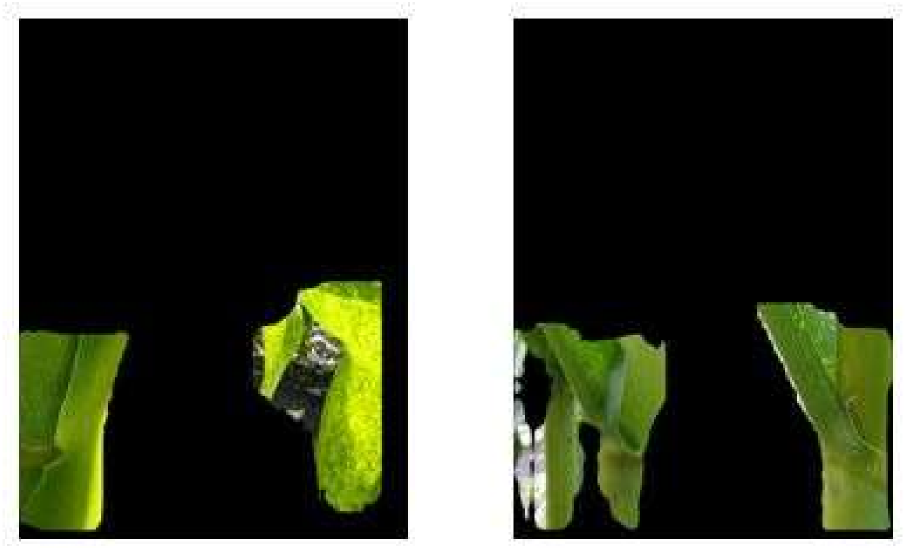
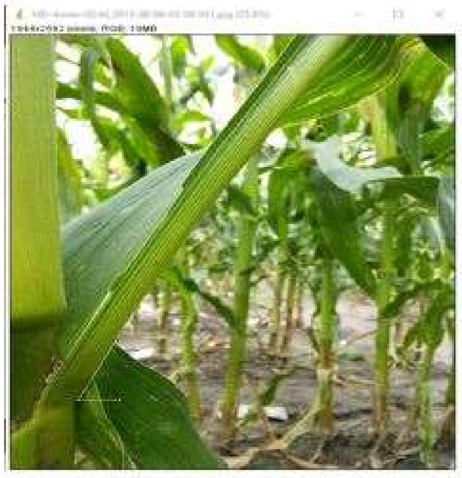
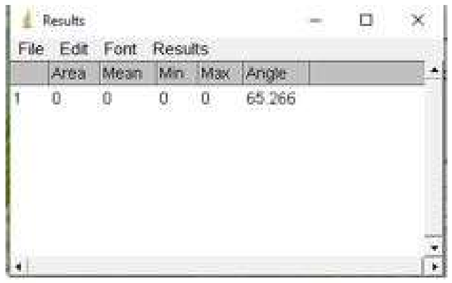


Figure 6.16: Multiple Leaf-Stem Instances in a Single Image

6.1.8 Results

The algorithm is applied on each of the image from the Summer_2015-Ames_ULA and Summer_2015-Ames_MLA datasets and the orientation between leaf and stem is estimated. The results are verified against the orientation estimations made by two graduate students at Iowa State University. The students worked independent of each other as they arrived at their estimations. The students used the ImageJ tool to estimate the orientations. ImageJ is a Java-based image processing tool that was developed at the National Institutes of Health and Laboratory for Optical and Computational Instrumentation. It runs on Linux, Mac OS X and Windows in both 32-bit and 64-bit modes. The features that ImageJ provides for analysis of images is the measurement of area, mean, standard deviation, min and max of selection or entire image, and angles. The angle measurement feature of ImageJ was used by the students to measure the orientation of the leaf. ImageJ allows the user to plot three points on the image that it uses to draw two lines. The angle measurement tool computes the angle between the two lines. Figure 6.17 shows the angle estimation made by plotting two lines (shown in yellow) and the result window output by ImageJ.





	Area	Mean	Min	Max	Angle
1	0	0	0	0	65.266

Figure 6.17: (left) Lines drawn for Angle Measurement (right) Measurement Output by ImageJ

The estimations made the students and algorithm are compared using a metric known as Cosine Similarity. Cosine Similarity is the cosine of the angle between two vectors that are typically non-zero and within an inner product space. It is useful to compare the similarity between two vectors represented in a higher-dimensional vector space. Mathematically, it is defined as the division between the dot product of vectors and product of the magnitude of each vector and is expressed as, $similarity = A.B/||A||\ ||B||$ where A and B are the two vectors compared for similarity. The measure is expressed as a value between 0 and 1. The Scikit-Python library is used to programmatically estimate the cosine similarity and matplotlib is used to plot a line chart of the datapoints. Figures 6.18 and 6.19 show the leaf angle estimates made by the proposed algorithm and student1, and proposed algorithm and student 2 respectively on the Summer_2015-Ames_MLA dataset which consists of 955 images. The cosine similarity between the proposed algorithm and student 1 is 0.98 which indicates a ten-degree difference in orientation between vectors, and proposed algorithm

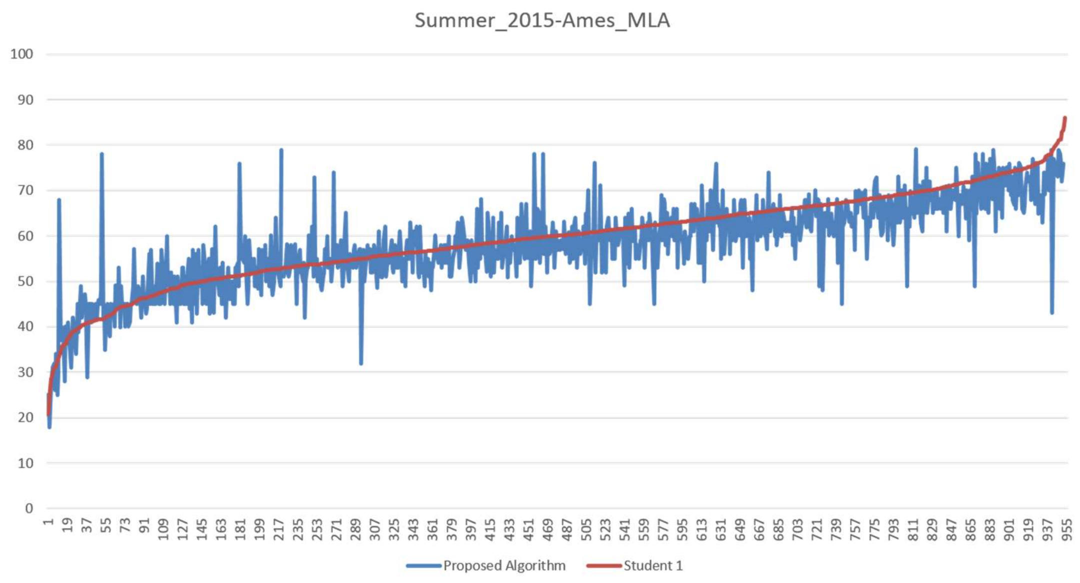


Figure 6.18: Datapoints from Leaf Angle Estimates of Proposed Algorithm and Student 1 for Summer_2015-Ames_MLA Dataset

and student 2 is 0.99 which indicates an eight-degree difference in orientation between the vectors[A25].

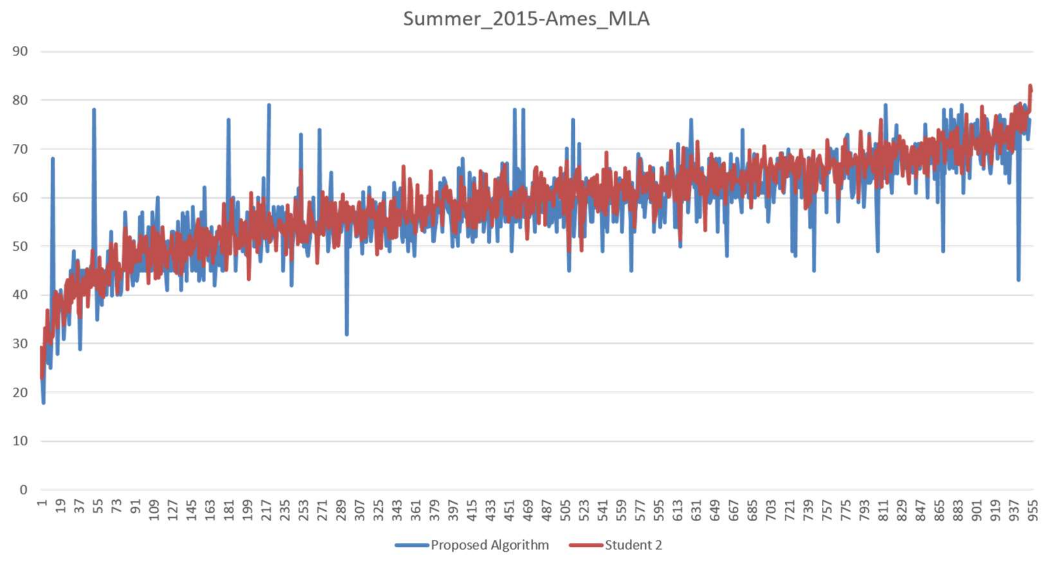


Figure 6.19: Datapoints from Leaf Angle Estimates of Proposed Algorithm and Student 2 for Summer_2015-Ames_MLA Dataset

Figures 6.20 and 6.21 show the leaf angle estimates made by the proposed algorithm and student1, and proposed algorithm and student 2 respectively on the Summer_2015-

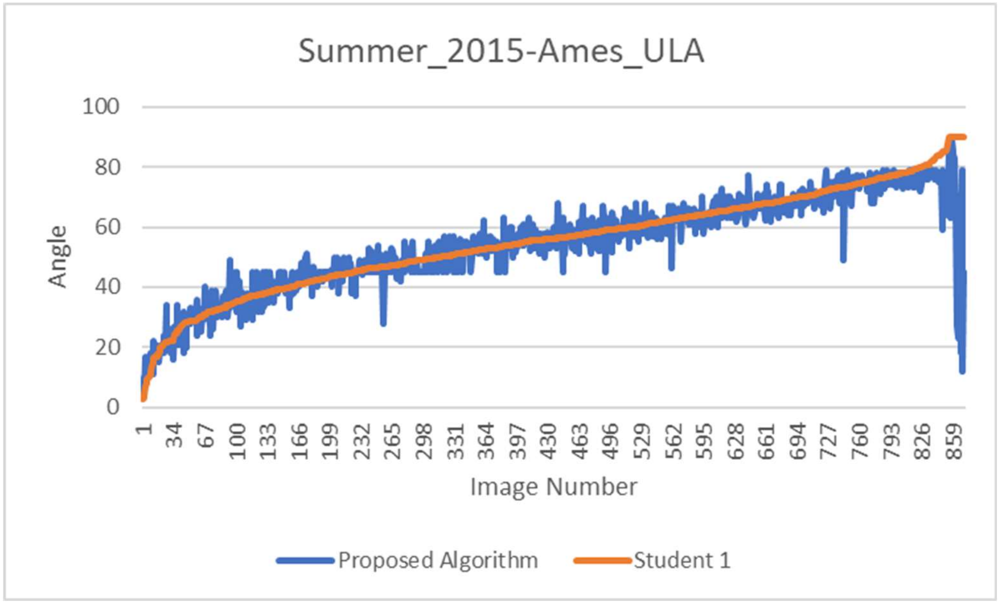


Figure 6.20: Datapoints from Leaf Angle Estimates of Proposed Algorithm and Student 1 for Summer_2015-Ames_ULA Dataset

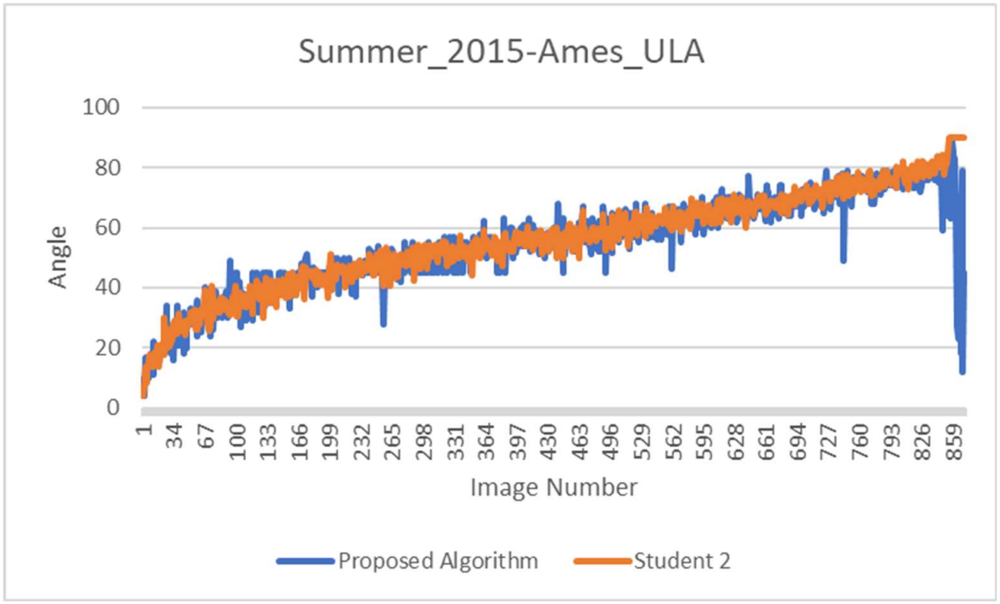


Figure 6.21: Datapoints from Leaf Angle Estimates of Proposed Algorithm and Student 2 for Summer_2015-Ames_ULA Dataset

Ames_ULA dataset which consists of 872 images. The cosine similarity between the proposed algorithm and student 1 is 0.998 which indicates a four-degree difference in orientation between vectors, and proposed algorithm and student 2 is 0.992 which indicates a seven-degree difference in orientation between the vectors. Furthermore, an intuitive outlier analysis is performed on the results to gain insight into the similarity of estimation between the different techniques. The rule used to identify an outlier is, "the difference in estimates between proposed algorithm and a student is greater than 7 degrees". The analysis assumes that the estimations made by the students are accurate. However, the estimations by the students also have a variance between them. The variance in student estimates, given by $\Sigma(\mathrm{e}1 - \mathrm{e}2)/\mathrm{n}$ where e1 and e2 are estimates made by students 1 and 2 respectively, and n is the total size of the data sample, are 2.57 and 2.07 on Summer_2015-Ames_MLA dataset and Summer_2015-Ames_ULA dataset respectively. Hence, the seven degree range to identify outliers is reasonable. Applying the outlier rule, a total of 31 outliers on the Summer_2015-Ames_MLA dataset and 18 outliers on the

Summer_2015-Ames_ULA dataset are detected. While a higher number of outliers is detected upon the initial application of the algorithm, part of them are due to issues with image sharpness and incorrect mask detections made by Mask R-CNN. Such outliers are fixed manually by applying appropriate fixes to the images. Among the estimates that are inliers, a variance of 3.2 degrees and 2.9 degrees is identified between the proposed algorithm and student 1, and proposed algorithm and student 2 respectively on the Summer_2015-Ames_MLA dataset whereas a variance of 2.6 degrees and 2.8 degrees is identified between the proposed algorithm and student 1, and proposed algorithm and student 2 respectively on the Summer_2015_Ames_ULA dataset. The results show that the proposed algorithm outputs results that highly co-relate with the manual measurements made by the students demonstrating the feasibility of the proposed automated technique for leaf angle estimation.

6.1.9 Code Availability

The code for the algorithm is available for download at https://github.com/marven22/Leaf-Angle-Estimation.git . The codebase is developed in an Anaconda environment and uses Python 3.8.3 and OpenCV 4.4.0.44 as the primary set of libraries for image processing.

6.1.10 Future Work and Conclusions

The proposed algorithm is implemented in Python using the OpenCV library. Moving forward, the algorithm will be implemented as part of a real-time android application so plant scientists and individual enthusiasts may leverage the application on a day-to-day basis. The bottleneck for the mobile application is the use of neural network models. The current Mask R-CNN network will have to be adapted so it works on mobile devices using networks such as Mobilenet SSD v1/v2 as the backbone in place of the current ResNet-101 model. Running models with large architectures on resource constrained devices is a challenge that needs to be overcome. The annotation of images is another bottleneck that is laborious and time consuming. The use of eye tracking hardware to annotate the images has the potential to greatly speed up the process of annotating images. The use of a screen to act as the common background over the plants as images are captured is an efficient way to help identify the location of the plant in the image. In case a routine is established, the region of interest in the image may be identified using simple image processing.

Overall, the proposed technique demonstrates the idea of estimating angle between leaf and stem using instance segmentation neural networks and image processing, a task that is laborious and cumbersome when performed manually. The estimation of leaf angles using automation is novel and has the potential to turn into one-of-a-kind utility for the agricultural community upon its implementation as a mobile app.

6.2 Plant Cover Analysis for Companion Planting

Companion planting is the idea of growing multiple species of plants in close proximity so that they reap mutual benefits, such as improved crop yield, soil quality and pest control. One of the foremost instances of companion planting in North America is that of ‘three sisters’ pioneered by the Native Americans. The plant varieties of beans, corn and squash constitute the ‘three sisters’ where their cultivation in close proximity led to the benefits of shelter and growth support for plants in addition to improved soil quality and decreased soil erosion [29]. Other combinations of plant varieties that complement each other's growth include artichoke and cucumber, beetroot and onion, and tomatoes and carrot whereas the combinations of parsnip and carrot, potatoes and pumpkin, and peas and garlic are instances of poor companions that deter one another's growth and development [82]. Hence, the discovery of companion plant species that are able to co-exist is of significant value to the agricultural sector. A key metric that helps estimate the growth of plant species is Fractional Vegetation Cover (FVC). FVC is defined as the percentage of the ground surface covered by vegetation elements from the overhead perspective [2]. Different techniques such as Point-and-Line cover Estimation, Plot-based cover estimation, Ocular-based estimation and semi-quantitative ocular-based estimation are available to estimate FVC at a certain location. However, the techniques currently available are labor-intensive, stochastic and subjective in nature. In addition, the techniques are prone to overestimating the vegetation cover due to field survey design [63]. The need for a technique that precisely analyzes the amount of FVC persists. The article proposes a technique to precisely estimate the amount of FVC in an area from images by drawing inferences from and extending upon the Daubenmire method, a semi-quantitative ocular-based FVC estimation technique. While current techniques estimate the percentage of FVC within images, the proposed technique provides an estimate on the area occupied by FVC in metric units in addition to percentage. The precise estimation of the FVC helps the plant scientists make discoveries of plant varieties that make good companions and offer well-informed suggestions to the agrarian community.

6.2.1 Related Work

The development of vegetation cover rating scales started in the early 1900s. The most widely accepted scale at the time was the rating scale proposed by Braun-Blanchet

[16]. The scale involved estimating and classifying the vegetation cover within a certain area into one of five cover classes from one to five where one indicated cover less than five percent, two indicated cover between five and twenty-five percent, three meant cover between 25% and 50%, four meant cover between 50% and 75%, and five meant cover between 75% and 100%.

RF Daubenmire [16] proposed the canopy coverage method in 1959. It was a semi-quantitative ocular-based estimation technique and regarded as one of the most accurate methods for vegetation cover analyses over the years. It involved meticulously placing a 20- x 50- cm quadrat along a tape on permanently located transects and classifying the amount of vegetation cover into one of six cover classes. The cover classes are similar to Braun-Blanchet's classes with the difference being that cover class five was divided into two classes where cover class five indicated vegetation cover between 75% and 95% and class six indicated vegetation cover between 95% and 100%.

Booth, Cox and Berryman [17] proposed SamplePoint, a free vegetation cover estimation tool from digital images using manual point sampling. The application shows users multiple single-pixel sample points (defaults to 100) on the image and allows them to classify the pixel as belonging to one of nine categories. The application uses the classifications made by the users to identify the percentage of pixels belonging to each of the nine categories. The results are output to an excel spreadsheet and have been comparable to the results from the most accurate field experiments for vegetation cover estimation.

Systat Software Inc. [163] provides specialized scientific software products for research in fields such as environmental sciences, life sciences and engineering. SigmaScan

Pro 5.0, an image processing software tool developed by Systant Software Inc. may be used to estimate the percentage of pixels that belong to vegetation from digital imagery. While the tool was not developed for vegetation cover estimation, it is able to be adapted for the use case.

Patrignani and Ochsner [124] developed a software tool named Canopeo to estimate the fractional green canopy cover from digital images. The tool was developed using Matlab and red-to-green (R/G), blue-to-green (B/G) and excess green index (2G-R-B). Desktop and mobile versions of the tool for android and iOS are available as free downloads. The tool provides a percentage estimate of the amount of green cover within an image and provides a grayscale image highlighting the green cover. However, the tool doesn't provide the vegetation cover estimate in metric units making it hard to reproduce results among experiments across individuals due to difference of height of image capture.

Louhaichi, Hassan, Clifton, and Johnson [85] proposed VegMeasure, a software tool that processes digital imagery collected in a specialized manner known as Digital Vegetative Charting Technique (DVCT). VegMeasure provides classification of imagery and measures change over time. However, DVCT requires a digital camera with built-in GPS that can be mounted to a stand, so images are captured from a fixed height with lens pointed orthogonal to the surface. VegMeasure requires that the camera, its height and orientation be kept constant throughout the image capture process for its estimation and analysis.

Zhang, Wang, and Elfaki [197] conducted research in the area of weed identification using imaging technology and spectroscopy. It was determined that location, color, texture, and shape feature of weeds and crops were the major criteria for weed detection sensor structure and algorithm design. Imaging-based weed sensors were compared with optical weed sensors as part of the experiment. Statistical and neural-network classification models were also used as part of the experiment.

Laliberte, et. al [76] demonstrated that object-based image analysis upon the conversion of RGB scale images to IHS (intensity-hue-saturation) scale images is a viable approach for estimating total cover of vegetation, bare soil, and fractional components of green and senescent vegetation. The image analysis tool used for the experiments was eCognition where the image was segmented into homogenous areas based on three parameters: scale, color and shape.

6.2.3 Materials and Methods

The breeders at the Land Institute in Salina, KS aim to estimate FVC across multiple plant varieties with the goal to identify companion plant species and make quality recommendations may be made to the farming community. The design of the experiment is inspired by the Daubenmire canopy coverage method and visually depicted by Figure 6.22.

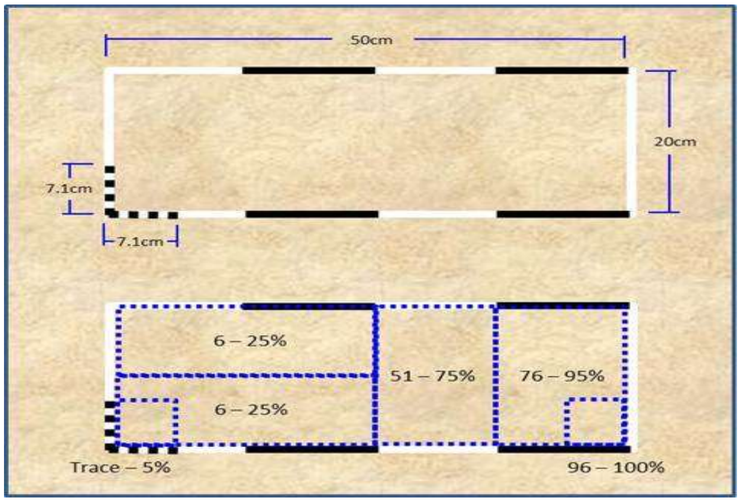


Figure 6.22: Daubenmire Quadrat with Ground Cover Classes [16]

Briefly, the Daubenmire technique involves using a 20 cm x 50 cm quadrat around vegetation cover along a tape on permanently located transects. The vegetation within the quadrat is measured by trained plant scientists based on ocular estimation. Since estimating the precise amount of plant cover from ocular observation is farfetched, plant scientists classify (estimate) the plant cover into one of six cover classes labelled one through six, as defined by Daubenmire. Membership in class one indicates crop cover less than five percent, membership in class two indicates crop cover between five and twenty-five percent, membership in class three indicates crop cover between 25% and 50%, membership in class four indicates crop cover between 50% and 75%, membership in class five indicates crop cover between 75% and 95%, and membership in class six indicates crop cover between 95% and 100%. As part of the experiment, a PVC frame of known dimensionality is dropped across multiple rows of crops as shown in Figure 6.23(a) and the amount of vegetation cover within each of the segments of the PVC frame is estimated independently at regular intervals over a stipulated time frame. Since the experiment progresses over time, accurate readings at each time step are essential to estimate the growth of vegetation. An algorithm that leverages image processing and analysis is proposed to eliminate the need for ocular estimation of FVC and measure FVC from digital images with precision. The development of the algorithm is performed on images from two datasets where one of the datasets comprises 33 images that capture alfalfa at a resolution of 3024 x 4032 pixels and the other comprises 177 images that capture a one-segment PVC frame laid around plants of Kura at a resolution of 5184 x 3456 pixels, as shown in Figure 6.23. Broadly, the algorithm is a three-step process where in the first and second steps are the detection and extraction of the PVC frame and vegetation cover from the image respectively. The final step is the estimation of the amount of vegetation cover within each of the segments of the PVC frame.

The algorithm is developed in Python using the OpenCV image processing library. The code for the algorithm is available at: https://github.com/marven22/Fractal-Vegetation-Cover-Estimation.git .

6.2.3.1. Noise Removal and PVC Frame Extraction from Image

In addition to vegetation cover within the PVC frame, the image consists of area outside the PVC frame. The area outside the PVC frame is irrelevant in terms of vegetation cover estimation within the grid. Hence, the first step of the process is the meticulous determination of the region of interest, i.e., the PVC frame and pixels within the PVC frame. The image processing concepts that are relevant to the process are HSV color space, Perspective Transform and 'bitwise and' operation.

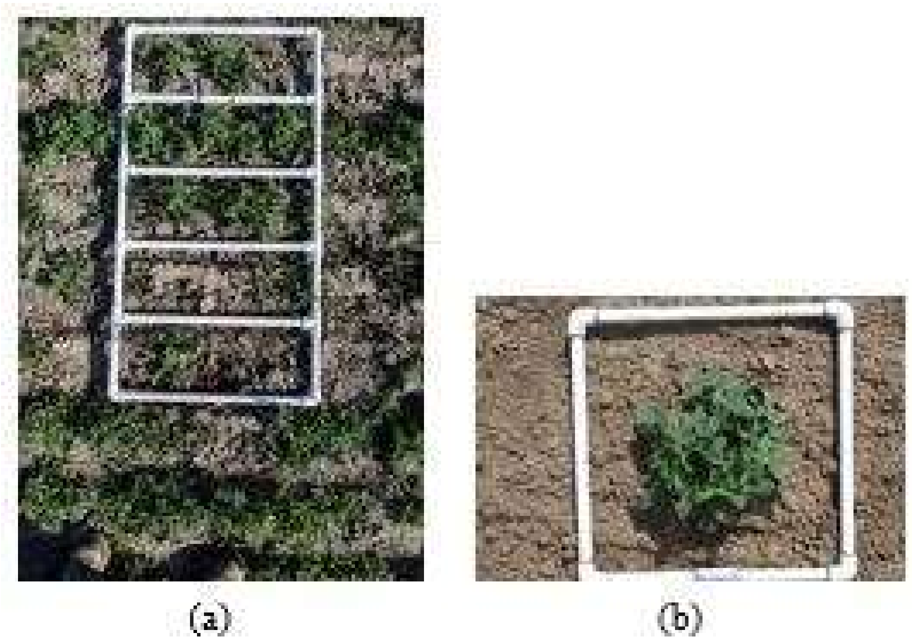


Figure 6.23: (a) Vegetation Cover within Five-Segment PVC Frame
(b) Kura Plant within One-Segment PVC Frame

HSV Color Space: HSV refers to Hue, Saturation and Value which make up the co-ordinates for the color space, as shown in Figure 6.24.

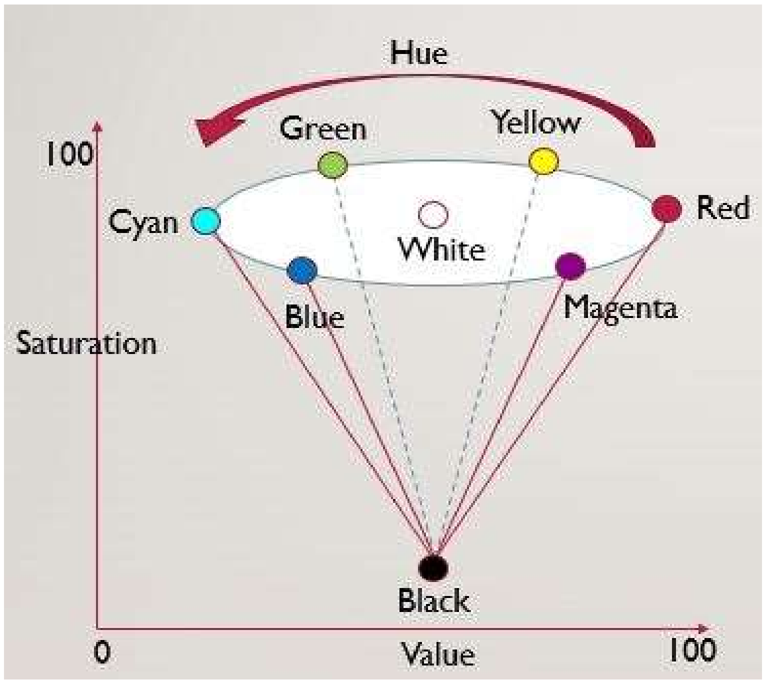


Figure 6.24: HSV Color Scheme

It is a cylindrical color space where the radius represents Saturation, the vertical axis represents Value, and the angle represents Hue. Intuitively, Hue is the dominant color visible to an observer, Saturation is the amount of white light mixed with a hue and Value is the chromic notion of intensity. As Value decreases, the color gets closer to black whereas the intensity of the color increases as Value increases.

Perspective Transform: Perspective Transform corrects the perspective of an image to bird's eye or top view. In other words, it helps view the image as if it were captured with the camera held orthogonal to the surface. The application of the transform immensely helps reduce skew in images captured using freehand. The downside to images being skewed is that objects in skewed images appear to be of a different size and orientation in comparison to their true dimension. In the context of the current algorithm, the application of the perspective transform to the images helps reduce skew, thereby making the process of identifying the PVC frame easier.

Bitwise AND Operation: The 'bitwise and' operation is used to extract a specific region from an image using a mask. The mask is a grayscale image provided to indicate the region of the image required to be extracted. Typically, the pixels of the mask are made to be either white (255) or black (0). The original image is also converted to grayscale and all the regions of the image that are not a black (0) pixel are set to white (255). The 'bitwise and' operation operates on a pixel-by-pixel basis where it identifies the common regions between the two and sets them to white (255).

The algorithmic steps to remove noise and extract the PVC frame are described as follows and the results produced after each step are shown in Figure 6.25.

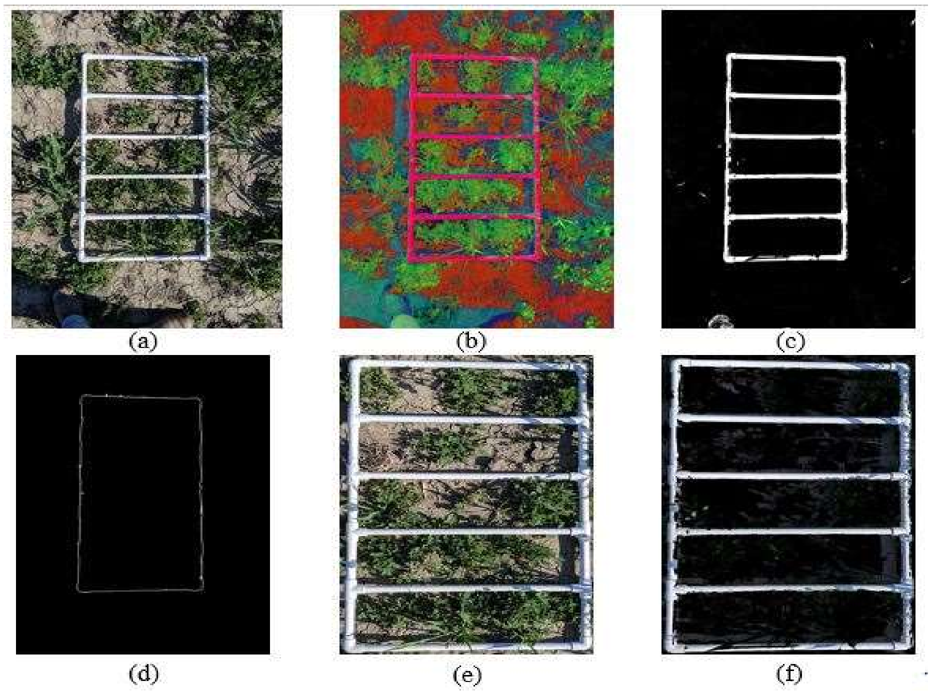


Figure 6.25: (a) Original Image (b) Original Image in HSV Color Scheme (c) Grayscale Image of PVC Frame

(d) Contour of the PVC Frame (e) Original Image without Noise (f) Extracted PVC Frame from Original Image

2. Convert the detected pixels of the PVC frame to grayscale and apply a median blur on the image to smoothen the image, if necessary.

3. Perform canny edge detection on the image and detect contours on the edges obtained.

4. Identify the largest contour of the contours detected. The idea is that the largest contour runs along the circumference of the PVC frame since the contours of each of the individual segments are significantly smaller in comparison to the outer contour which encapsulates the PVC frame.

5. Plot the minimum area rotated rectangle around the contour of the PVC frame.

6. Perform a 'bitwise and' operation on the input image using the minimum area rotated rectangle as the mask. Doing so precisely identifies the location of the PVC frame on the image. The region of the image within the PVC frame is the only portion of the image to be processed for vegetation cover estimation.

7. Apply perspective transform on the image to view the image from the top view. It corrects skew and makes the task of detecting segments on the PVC grid easier.

8. Convert the output of step 7 to HSV format and apply the lower and upper ranges mentioned in step 1 to precisely extract the PVC frame.

6.2.3.2. Vegetation Cover Detection using Simple Linear Iterative Clustering

On the region of interest identified in the previous step, Simple Linear Iterative Clustering (SLIC) is applied to segment the image into multiple super-pixels. To elaborate, the process of partitioning an image into multiple segments by assigning a label to every pixel in the image is called image segmentation. The idea is that segments with the same label are perceptually similar and share common characteristics. Image segmentation helps to identify similar regions in images and extract semantic information from them. While segmenting images by pixel is an intuitive approach, segmenting by super-pixel is more optimal. A super-pixel is defined as a group of pixels that perceptually belong together by sharing common characteristics such as pixel color, intensity and other low-level properties. Modern cameras result in images of high resolution. While the level of detail within the images is excellent, it overburdens the image processing algorithms used to process the images resulting in longer runtimes. The key benefit to using super-pixels over pixels is that they shrink the pixel space required to be processed by the algorithm and in turn, lead to shorter runtimes without compromising accuracy. In addition, a single pixel by itself does not provide any semantic information whereas a super-pixel provides semantic information that helps make key discoveries within images.

Simple Linear Iterative Clustering (SLIC): SLIC is an image processing technique that clusters pixels to generate super-pixels based on their color similarity and proximity in the

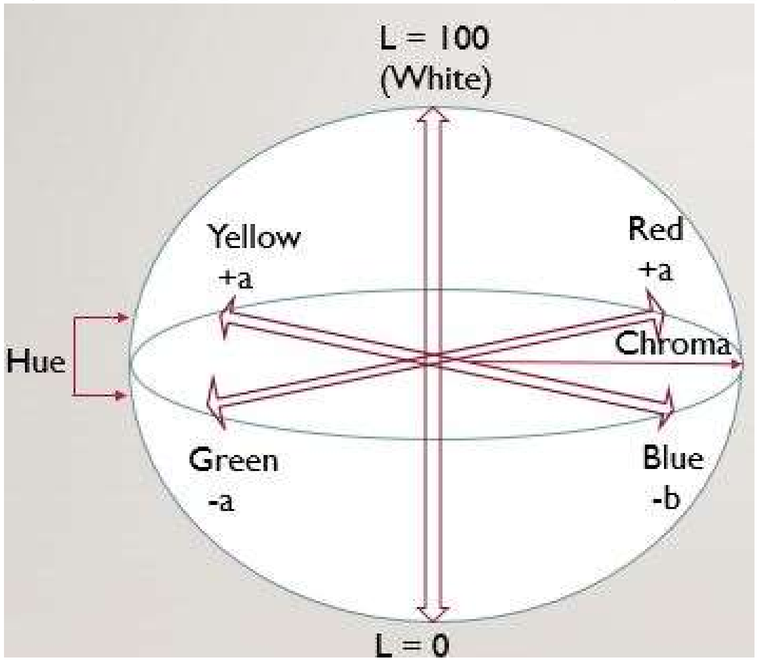


Figure 6.26: CIELAB Color Space

image plane. SLIC uses the five dimensional [LABXY] space for the clustering where [LAB] is the pixel color vector in CIELAB color space and [XY] is the pixel position on the XY plane [166]. The CIELAB color space, as shown in Figure 6.26, is defined by the International Commission on Illumination (CIE) to improve upon the fact that the conventional RGB color space only allows for the representation of 40% of the colors that the human eye can perceive. The L channel represents lightness which ranges from black to white, the A channel represents the value on the axis that ranges from green to red and the B channel represents the value on the axis that ranges from blue to yellow. A key characteristic of the CIELAB color space is that its dimensions are non-linearly scaled in the sense that spatial distance between two colors corresponds to perceptual distance and is uniform across the color space. For instance, consider two pairs of different colors that appear to be equally similar such as blue and light blue, red and light red. When Euclidean distance is computed between the two pairs of colors in the CIELAB color space, it is observed the Euclidean distance between both pairs of colors is the same.

In order to use the Euclidean distance measure in the five dimensional [LABXY] space, the spatial distance between two points is required to be normalized since it is subject to image size. Consider an image containing N pixels and let K be the number of super-pixels in the segmented image. The size of each super-pixel is N/K, and a super-pixel center is present at every grid interval $S = \sqrt{N/K}$. The center of each of the super-pixels $C_k = [L_k, A_k, B_k, X_k, Y_k]$ where k = [1, K] is chosen at regular grid intervals S. The spatial extent of each of the super-pixels is approximately S^2 the area of a super-pixel and the pixels that are associated with any super-pixel center lie within a 2S x 2S area around the super-pixel center on the XY plane [5]. Euclidean distances in CIELAB color space are perceptually meaningful for small distances but not large distances. The distance measure D_s used in SLIC is the aggregate of the LAB distance and XY plane distance normalized by the grid interval S. The mathematical notations are as follows:

LAB distance, $D_{lab} = \sqrt{\ } (l_k - li)^2 + (a_k – a_i)^2 + (b_k – b_i)^2$

XY plane distance, $D_{xy} = \sqrt{\ } (x_k - xi)^2 + (y_k – y_i)^2$

SLIC distance measure, $D_s = D_{lab} + (m/S) * D_{xy}$ where m is the factor that controls compactness of the super-pixel. The greater the value of m, the more compact the cluster.

The SLIC algorithm is similar to any other clustering algorithm in the sense it begins by sampling K regularly spaced cluster centers and moving them to seed location corresponding to the lowest gradient position in a 3 x 3 neighborhood. Image gradients are computed as $G (x, y) = ||I (x + 1, y) – I (x – 1, y) ||^2 + ||I (x, y + 1) – I (x, y – 1) ||^2$ where I (x, y) is the LAB vector corresponding to the pixel at position (x, y) and $||.||$ is the L_2 norm [163]. Each pixel in the image is associated with the nearest cluster center whose search area overlaps the pixel. After all the pixels are associated with the nearest cluster center, a new center is computed as the average LABXY vector of all the pixels belonging to the cluster. The process is iteratively repeated until convergence.

The regions that indicate plant cover are identified by inferring from the semantics of the super-pixels put out by SLIC. The steps in the process are described as follows.

1. Segment the image into a pre-defined number of super-pixels using the SLIC function from the OpenCV image processing library. It is observed that segmenting the image into 300 super-pixels identifies the plant cover best although the algorithm may execute faster

at the expense of accuracy if the image is segmented into fewer super-pixels. The image segmented into super-pixels is shown in Figure 6.27(a).

2. Find the average color value within each of the super-pixels and assign it as the label that represents the super-pixel. It ensures that a given super-pixel has only one color.

3. Find unique labels within the image and apply a filter that identifies the plant cover. An observation that is made based on experimentation is that the super-pixels that consist of plants are green. Hence, all the green labels are identified, and a grayscale image is created where the locations of green labels are plotted in white (255). The image represents the vegetation cover within the image.

4. **(Optional)** Perform a ‘bitwise and’ operation on the image using the grayscale image from step 3 as the mask. The resulting image shows the vegetation cover in the image

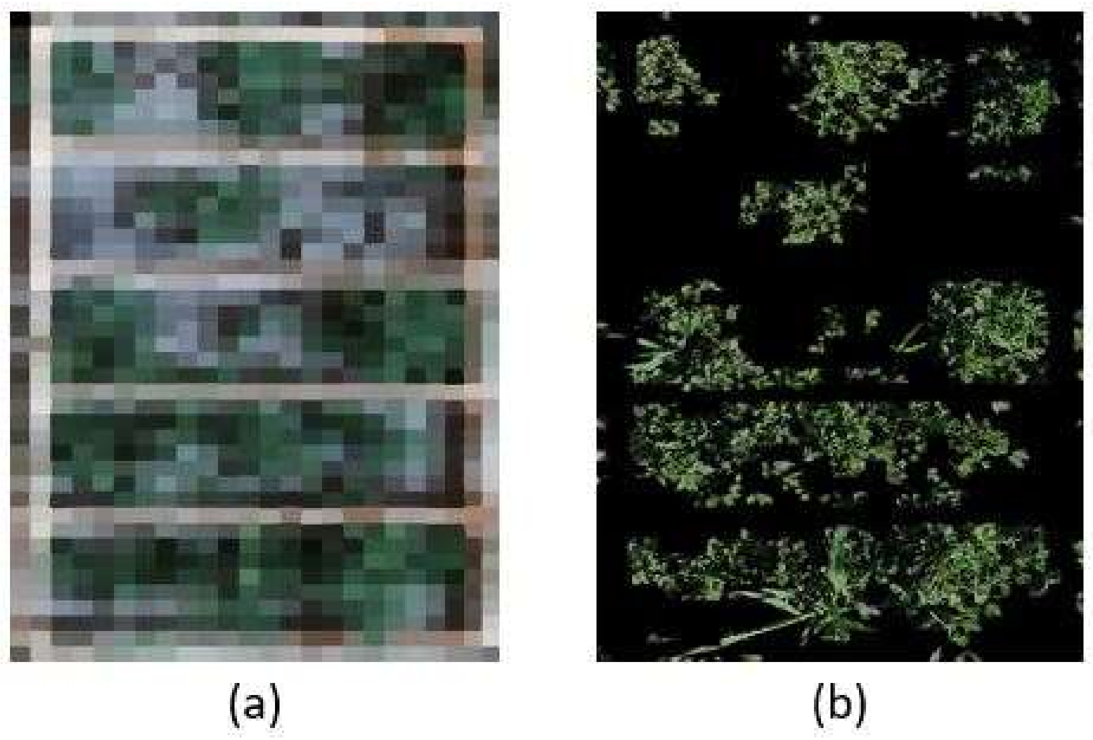


Figure 6.27: (a) Super-pixel Segmentation of Original Image (b) Detected Vegetation Cover in RGB Color Space

within each of the segments of the PVC frame in the 3-channel RGB color space. The step, whose output is as shown in Figure 6.27(b), is merely to observe the vegetation detected within the image for user validation but is not required for analysis.

6.2.3.3. Vegetation Cover to PVC Frame Segment Association

Once the PVC frame and vegetation cover are identified, the next step in the process is to associate the vegetation cover with each segment of the PVC frame. The Hough Lines Transform is used to detect each of the segments of the PVC frame. The Hough Lines Transform is typically used to identify straight lines in images [7][35]. Typically, straight lines are represented using the slope-intercept method as $y = mx + c$ where m is the slope and c is the y-intercept. However, Hough Transform relies on representing straight lines using the pair of polar co-ordinates, (ρ, θ). The first parameter, ρ, is the shortest distance from the origin to the line. The second parameter, θ, is the angle between the x-axis and distance line. The equation of the straight line in polar co-ordinate system is given by, $\rho = x * \cos(\theta) + y * \sin(\theta)$ where (x, y) is a point on the line. The Hough Transform takes the edges from a binary image as input. The Hough Transform maps each of the pixels to multiple points in Hough (or parameter) space. An edge pixel is mapped to a sinusoid in 2D parameter space, (ρ, θ) representing all possible lines that could pass through the point. It is referred to as the voting stage. The sinusoids of collinear points in the Hough space cross each other indicating collinearity. There are two variants of Hough Transform, Standard and Probabilistic [35][69]. The primary difference between the two variants is the computational complexity. Consider an image that has M pixels as edge points and a Hough space divided in $N_\rho \times N_\theta$ accumulators. In case of the Standard Hough Transform, each of the M edge pixels is used for computation which means that the computational complexity is $O(M.\ N_\theta)$ for the voting stage and $O(N_\rho.\ N_\theta)$ for the search stage. However, in the Probabilistic Hough Transform, only a subset m of M edge pixels is used for computation which means that the computational complexity is reduced to $O(m.\ N_\theta)$ for the voting stage resulting in a faster execution of the algorithm.

The algorithmic steps to associate vegetation cover with PVC frame is described as follows.

1. Consider the image of the PVC frame obtained as the output in section *B*.

2. Detect straight lines on the image using the Probabilistic Hough Lines transform. The detected line segments indicate the segments of the PVC frame. A vote measure of 180 is used to start the search for straight lines. In case no lines are identified at the vote range, it is lowered by 10 until lines are detected.

3. Filter the line segments to identify the ones that run along the segments of the PVC frame. Any given segment in the PVC frame should have four lines with slope ~0.0 i.e., horizontal or nearly horizontal lines and two lines with infinite slope i.e., vertical or nearly vertical lines associated with it. In order to detect the lines, parallel lines with slopes less than 15 and slopes higher than 100 are retrieved. Lines with slopes under 15 are oriented at less than 10 degrees with the X-axis whereas lines with slopes above 100 approaching infinity are oriented at over 45 degrees with the X-axis. It is possible that more lines than expected are detected along the PVC frame segment. In order to contain only the expected six lines, any overlapping line segments or ones that are closer in position and orientation than a certain threshold are merged together into a single line segment. All line segments that are separated by less than 10 pixels in distance and 0.2 degree in orientation

are grouped together and only a single line segment from amongst the group is considered for analysis.

4. Extend the line segments along the PVC frame segments by a factor of five (or higher) to ensure that the horizontal line segments intersect with the vertical line segments in case they do not.

5. Determine the segment of the PVC frame to which each of the line segments belong based on the position of the line segments in the image. In case of PVC frames that contain multiple segments, the association of line segments to the PVC frame segments helps estimate the amount of vegetation cover within the PVC frame segments.

6. Consider the line segments associated with each segment of the PVC frame and determine the innermost lines i.e., two inner vertical and two inner horizontal lines. Find the four points of intersection of the innermost lines. The idea is to identify the inner rectangles (polygons) within the PVC frame.

7. Create a grayscale image and plot a polygon using the points of intersection as the four vertices of the polygon. Fill the polygon in white (255), as shown in Figure 6.28(b). In a PVC frame with n segments, n polygons are obtained where each of the polygons indicates a segment within the PVC frame.

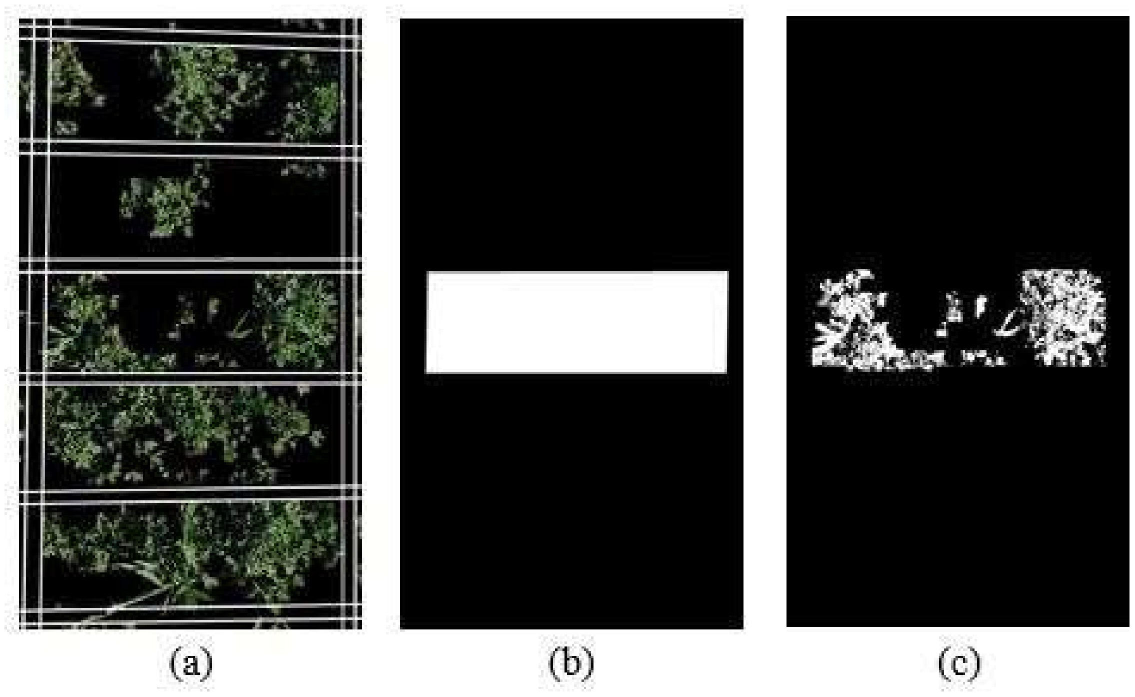


Figure 6.28: (a) Detected Hough Lines for the PVC Frame with Vegetation Cover (b) Polygon Mask for Segment 3 (c) Vegetation Cover within Segment 3

8. Perform a 'bitwise and' operation on the output of section 5.2.3(B), as shown in Figure 6.27(b), using each of the grayscale images as the mask. The pixels indicating vegetation cover are plotted in white (255), as shown in Figure 6.28(c).

6.2.3.4. Amount of Vegetation Cover Estimation

The percentage of vegetation within each segment and the amount of vegetation cover in metric units are estimated by the algorithm. The ratio of the number of pixels that indicate vegetation cover and the number of pixels present in each segment of the PVC frame gives the percentage of vegetation cover within each PVC frame segment. In other words, the ratio of white pixels in images obtained from step 7 and step 8 gives the percentage of vegetation cover. The estimation of vegetation cover in metric units requires the knowledge of dimensions of the PVC frame. The length and width of the inner rectangles of the PVC frame shown in Figure 6.23(a) are 19.75” and 6.75” respectively. The number of pixels present within the rectangle are obtained from polygon masks plotted in step 7 of section 5.2.3(C). Please note that each polygon might have slightly different pixel count since the image may be partly skewed. Since the number of vegetation cover pixels are known from step 8 of section 5.2.3(C), the area of vegetation cover is given by the equation, $Vegetation\ Cover\ (sq.in) = (area\ of\ rectangle\ (sq.in)\ xnumber\ of\ vegetation\ cover\ pixels)/ (number\ of\ polygon\ pixels)$

6.2.4 Results and Discussion

In the absence of ground truth measurements for the vegetation cover within each of the PVC frames, the proposed algorithm is evaluated on two applications SamplePoint and Canopeo. SamplePoint is a desktop application that facilitates foliar cover measurements from nadir imagery by superimposing a systematic or random array of up to 225 crosshairs targeting single image pixels and provides a platform for simple, manual classification of the pixels. SamplePoint is a product that has been developed in collaboration with the USDA and serves as the benchmark for vegetation cover estimation. Canopeo is a tool available as desktop and mobile applications to estimate fractional vegetation canopy cover. Both SamplePoint and Canopeo provide vegetation cover estimates as a percentage of the image size but not in metric units. It is perhaps due to the lack of a reference object in the image. The proposed algorithm is compared to the measurements provided by SamplePoint and Canopeo across 100 Kura plant images, similar to Figure 6.23(b). The Cosine Similarity [190] measure is used to compare similarity among the three measures. Cosine Similarity is the cosine of the angle between two vectors that are typically non-zero and within an inner product space. Mathematically, it is defined as the division between the dot product of vectors and product of the magnitude of each vector and is expressed as, $similarity = A.B/||A||\ ||B||$ where A and B are the two vectors compared for similarity. A plot of the estimates of vegetation cover across the 100 images using each of the three measures is shown in Figure 6.29.

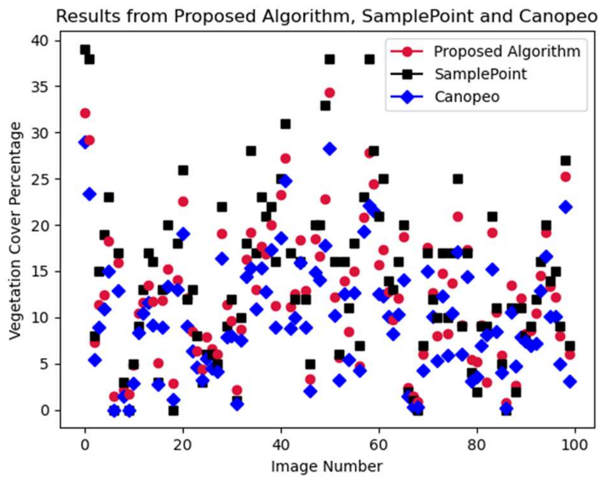


Figure 6.29: Vegetation Cover Estimates by the Proposed Algorithm, SamplePoint and Canopeo

The cosine similarity measure between the proposed algorithm and SamplePoint is 0.995 whereas that between the proposed algorithm and Canopeo is 0.99. It is worth noting that the cosine similarity between SamplePoint and Canopeo is 0.99. The results indicate a high degree of cadence among the techniques and demonstrates the quality of results produced by the proposed algorithm. As observed from the median and mean values, the results produced by the proposed algorithm lie between that of the results produced by SamplePoint and Canopeo. The median of the proposed algorithm is 12.04 whereas that of SamplePoint and Canopeo are 13 and 9.14 respectively. The mean of the proposed algorithm is 12.40 whereas that of SamplePoint and Canopeo are 14.21 and 10.01 respectively. The key benefit to using the proposed technique is that the algorithm provides the measurement of the amount of vegetation cover in metric units along with percentage. It significantly improves reproducibility of the results across multiple individuals conducting the experiments since it eliminates the dependence of the results on the height from which the image is captured.

6.2.5 Future Work and Conclusions

Proposed is an algorithm to estimate the FVC in an area from images to aid in Companion Planting. The proposed algorithm provides an estimate of the vegetation cover as a percentage of the image size and an absolute value in metric units. The algorithm relies on the presence of a PVC frame of known dimensionality in the image to be used as reference to measure the vegetation cover in metric units. The results of the proposed algorithm when compared to SamplePoint and Canopeo exhibit a high degree of similarity. Moving forward, the algorithm will be implemented as a mobile application that the users may download onto their devices. The code for the algorithm is also made open source for enthusiasts to use and develop further.

Chapter 7 - Development Tools

The research conducted as part of the work for each of the chapters relied on the availability of several tools and libraries such as OpenCV, TensorFlow, and PyTorch. The availability of libraries helped in performing tasks such as baselining, diagnostics and troubleshooting along with the development of algorithms themselves. The thesis resulted in the development and usage of three tools for image processing that have been extensively used. While the logic and code for the tools may be available on the internet in bits and pieces, the current chapter documents and describes the tools for readers to comprehend and leverage the tools on their research journeys. Each of the tools is implemented in OpenCV-Python. The three image processing tools are:

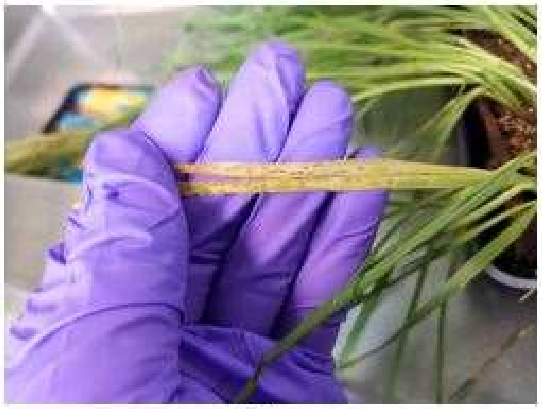
1. Image Thresholding Trackbar
2. Semi-automated Image Cropper
3. Eraser

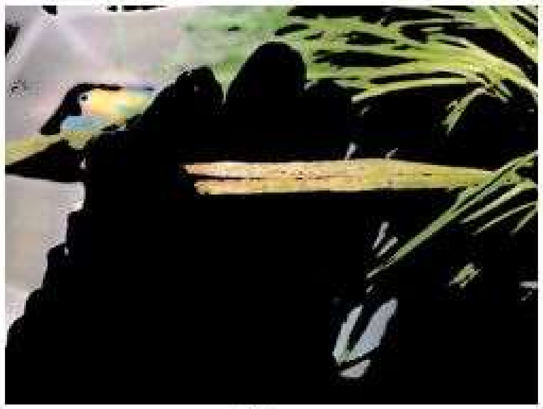
7.1 Image Thresholding Trackbar

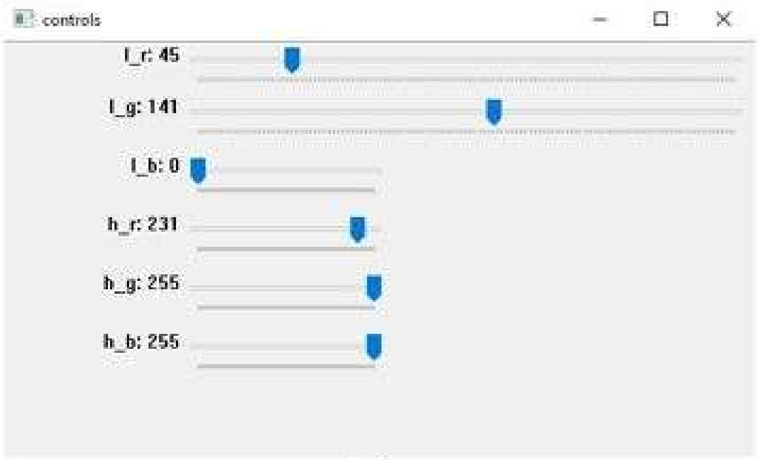
Image thresholding is the technique of segmenting images to extract information. The applications of image thresholding are widespread in the area of digital image processing. Conventional cameras capture information in the RGB color scheme where R is Red, G is Green, and B is Blue. An image is an amalgamation of several pixels of the same or varying color intensities. The intensities of red, green and blue in an RGB image range from 0 to 255 where 0 indicates 0% intensity and 255 indicates 100% intensity. In the majority of image processing applications, determining the appropriate intensities of red, blue, and green is critical to extract key information from images. However, determining the intensities is a task that requires meticulous curation due to each of the channels having a profound impact on the content of the image. The image thresholding tracker helps to curate the values for the red, green, and blue channels so researchers are able to extract information from images easily. Furthermore, the RGB color space is not always the most feasible to extract information from images. Other color spaces such as grayscale, HSV, and LAB provide access to certain detail that the RGB color space doesn't necessarily provide. The proposed image thresholding trackers help threshold images in different color spaces such as grayscale, RGB, HSV and LAB.

7.1.1 RGB Image Thresholding Trackbar

OpenCV provides a 'createTrackbar' function that creates a trackbar. The function takes five arguments – trackbar name, window name to which it is attached, default value that the trackbar is set at the time of creation, maximum value and callback function which is executed every time the value of the trackbar alters. Six trackers, two for each of the red, green, and blue channels are created using the 'createTrackbar' function of OpenCV. Each channel contains two trackbars so a lower bound and upper bound on the channel are able to be specified by the user. As the user alters the values corresponding to each of the channels, all the pixels that do not lie within the upper and lower bounds for each of the channels are subject to thresholding. The process of thresholding involves determining the pixels that lie in within the upper and lower bounds set on the trackbar using OpenCV's 'inRange' function. The function takes three parameters – image, lower bound, and upper bound. The upper and lower bounds are input as numpy arrays where each of the numpy arrays contains three values – red, green, and blue intensities. The function returns all the pixels that lie within the thresholds of upper and lower bounds. The portion of the image that is returned by the 'inRange' function is used as the mask in the 'bitwise_and' operation to extract the desired information from the original image. Figure 7.1 shows the RGB color thresholding trackbar in action on the image of rust-stricken wheat leaf held by a plant science researcher wearing a purple glove.


(a)


(b)



```
l_r: 45
l_g: 141
l_b: 0
h_r: 231
h_g: 255
h_b: 255
```

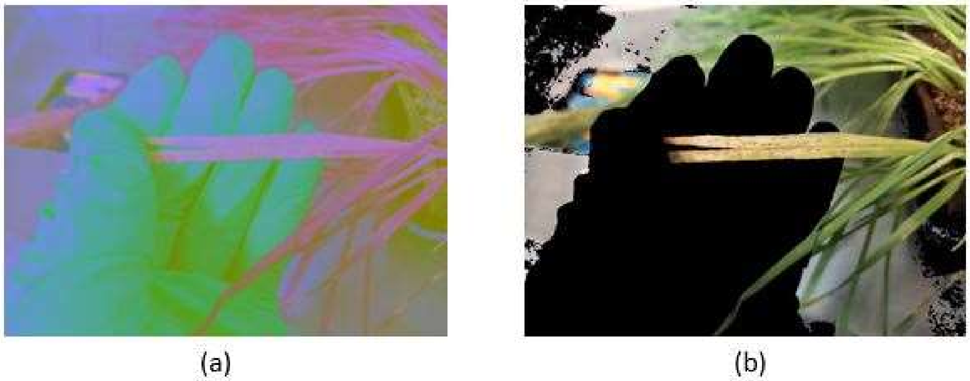
(c)

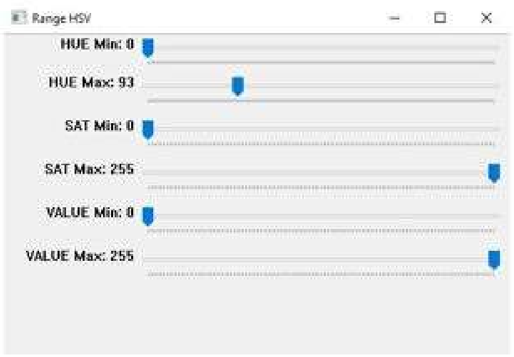
Figure 7.1: (a) Image of Wheat Leaf in Hand (b) Extraction of Wheat Leaf (c) Upper and Lower Thresholds of Red, Green, and Blue Channels

Figure 7.1(a) shows the original image of the wheat leaf in the hand of a researcher. Assuming that the analysis is to be performed on the leaf, the first step in the analytical process is the extraction of region of interest i.e., the wheat leaf. Figure 7.1(b) shows the extraction of the wheat leaf by the application of RGB thresholding. Figure 7.1(c) shows the upper and lower values for the red, green, and blue channels that led to the extraction of the leaf. Altering the values further modifies the thresholding. l_r and h_r corresponds to lower bound of red and upper bound of red respectively. Likewise, for l_g, l_b, h_g, and h_b. The values for each of the channels range from 0 to 255.

7.1.2 HSV and Lab Image Thresholding Trackbars

HSV is the acronym for thresholding based on hue, saturation and value. Hue refers to the color portion of the model expressed as a number from 0 to 360 degrees. Red is represented by values between 0 and 60, yellow by values between 61 and 120, green by numbers between 121 and 180, cyan by numbers between 181 and 240, blue by numbers




Range HSV

HUE Min: 0
HUE Max: 93
SAT Min: 0
SAT Max: 255
VALUE Min: 0
VALUE Max: 255

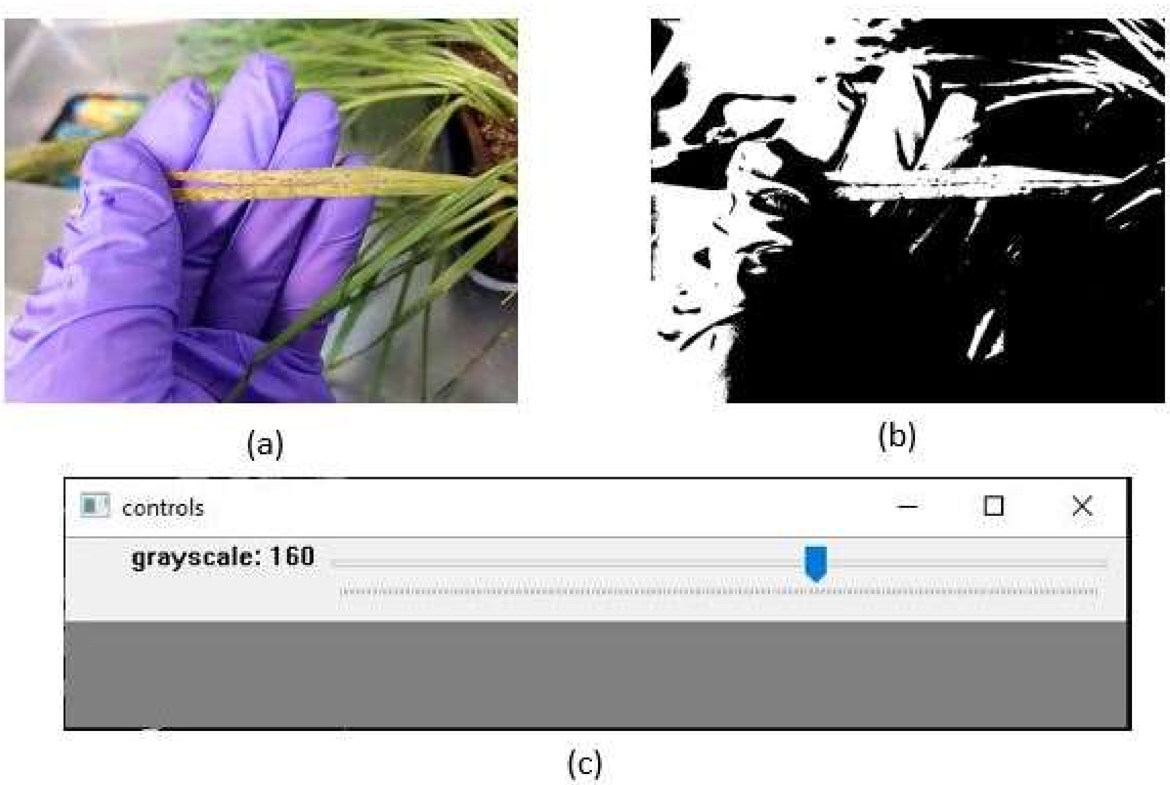
Figure 7.2: (a) HSV Color Model of Figure 7.1(a) (b) Extracted Wheat Leaf using HSV Thresholding (c) HSV Trackbar

between 241 and 300, and Magenta by numbers from 301 to 360 degrees. Saturation describes the amount of gray in a particular color and ranges from 0 to 100 percent. Reducing the amount of saturation towards zero increases the amount of gray within the image and produces a faded effect. Value is synonymous with brightness in the image. It is a value that ranges from 0 to 100 where 0 is completely black and 100 reveals the most color. Figure 7.2 shows HSV thresholding performed on the wheat plant shown in Figure 7.1.

The Lab color space is a color-opponent space with the dimension L for lightness, and A and B to represent color-opponent dimensions based on non-linearly compressed CIEXYZ color space coordinates. The Lab color space includes all perceivable colors which means that its gamut exceeds that of the RGB color model. In addition, the Lab color model is device independent i.e., the colors are defined independent of their nature of creation [128]. The range of L is from 0 to 100 whereas that of a and b is 0 to 127. The HSV and Lab image thresholding trackbars are created in the same manner as the RGB image thresholding trackbar. The only additional step is that the image subject to thresholding is converted to the respective color scheme before the application of the steps outlined in the creation of RGB thresholding trackbar.

7.1.3 Grayscale Image Thresholding Trackbar

The grayscale representation of an image consists of only one channel unlike RGB images that consist of three channels. As a result, the grayscale image thresholding trackbar consists of only one slider whose value ranges from 0 to 255. The value of 0 represents black whereas the value of 255 represents white. Figure 7.3 shows grayscale thresholding in action using the grayscale image thresholding trackbar.


(a) (b)

controls

grayscale: 160

(c)

Figure 7.3: (a) Image of Wheat Leaf in Hand (b) Grayscale Thresholding on Original Image (c) Grayscale Image Thresholding Trackbar at 160

7.2 Semi-Automated Image Cropper

The training of convolutional neural networks involves learning key features from an image. However, the images acquired from the agricultural field contain plenty of background noise in addition to the entity of interest. The training of neural networks on images containing noise yields results that are sub-par. In order to avoid such a scenario, the neural networks are required to be trained only on the region of interest (ROI). Given an image dataset, chances are that the ROI is not present at the same location in each of the images. Hence, cropping a certain region from every image in the dataset is not an option. In addition, the ROI in each image may be of a different size. One of the requirements to train neural networks is that all the training images be of the same size. Different neural networks may take images of different size as input. For instance, the state-of-the-art neural networks of ResNet-50, VGG-16 and VGG-19 require that the input be of size 224 x 224 px. While resizing images may be a simple operation, resizing generally leads to distortion of the objects in the image rendering the image useless for training. Hence, proposed is a semi-automated image cropping tool that extracts the ROI from an image and resizes it to a size of the user's choice without distortion. The proposed tool is a step forward in generating image datasets for neural network/ machine learning model training.

7.2.1 Procedure

The image extractor is a Python-OpenCV that provides headless and user-interface modes. The extraction of ROI from images is defined as a 3-step process:

1. Image thresholding to eliminate background noise.
2. Image cropping to extract ROI
3. Resizing the extracted ROI and centering the image on a canvas of choice.

Image Thresholding

The extractor provides a user interface to apply two different types of thresholding namely, RGB and HSV where the default is HSV. The user interface consists of trackbars and allows the user to apply thresholding to extract the region of interest removing immediate noise in the background. Figure 9 shows the HSV thresholding applied on an image of a wheat plant. The region of interest is present within the glove. As a first step, the glove is removed from the image using HSV thresholding. At this point, the size of the image is unaffected.

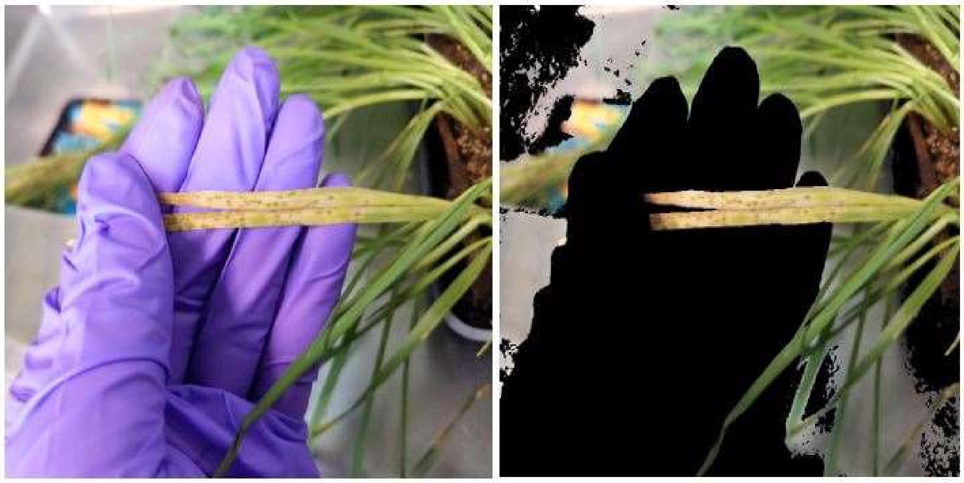


Figure 7.4: (left) Image of Wheat Plant Captured in a Field Setting; (right) HSV Thresholding Applied to Remove Glove from Image

Image Cropping

A tool that allows the user to select the region of interest is provided by the framework. The user is allowed to drag the cursor along the region of interest starting at the top left and ending at the bottom right of the image. The tool crops the image based on the co-ordinates selected by the user. Figure 10 shows the ROI extraction made by the tool.

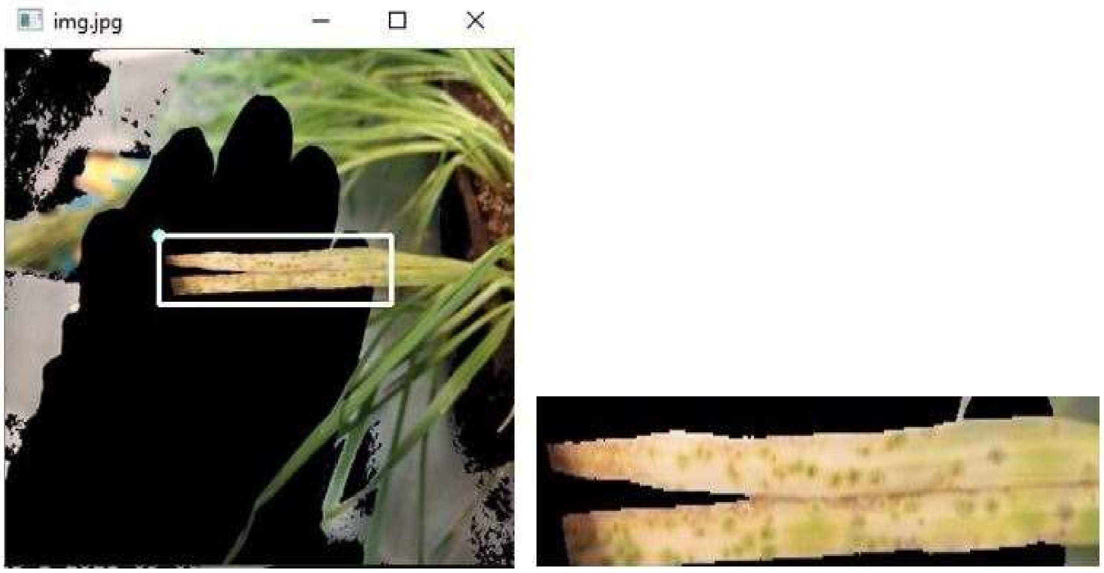


Figure 7.5: (left) ROI Cropping of the HSV Threshold Image; (right) Cropped ROI

Image Resizing

Resizing the ROI extracted in the previous step as-is leads to distortion due to poor aspect ratio that could result. Figure 11 shows the distorted image obtained by resizing the ROI. The ROI obtained from image cropping is of size 361 x 61 px. The ROI is resized to 224 x 224 px. The resulting distortion renders the image unfit for training.

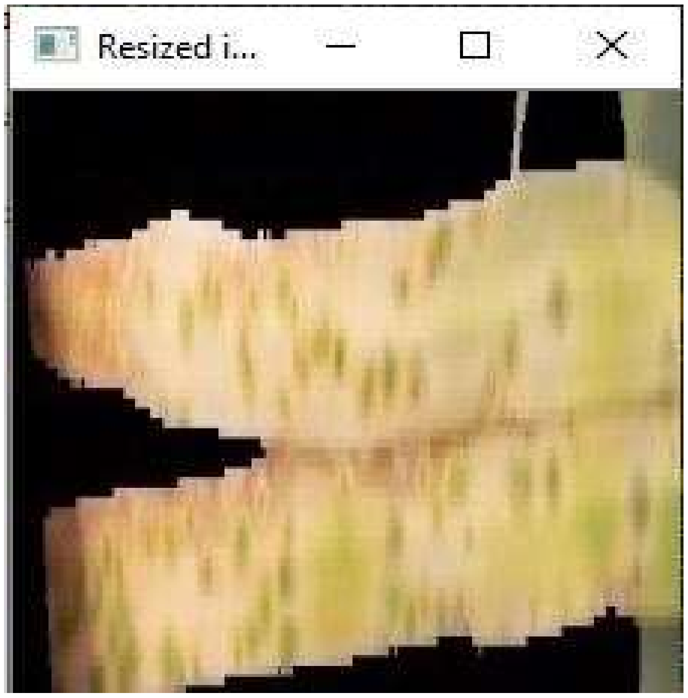


Figure 7.6: Image Distortion from Resizing

Such distortion leads to the images being unsuitable for training on the neural networks since the majority of the image features are lost. The technique to avoid distortion is to copy the ROI onto an n x n canvas that fits the ROI completely and then resize. Such a technique preserves the aspect ratio thereby negating distortion. However, merely preserving the aspect ratio does not provide an image that is fit for training. Figure 12 shows an image generated by copying the ROI onto a larger canvas.

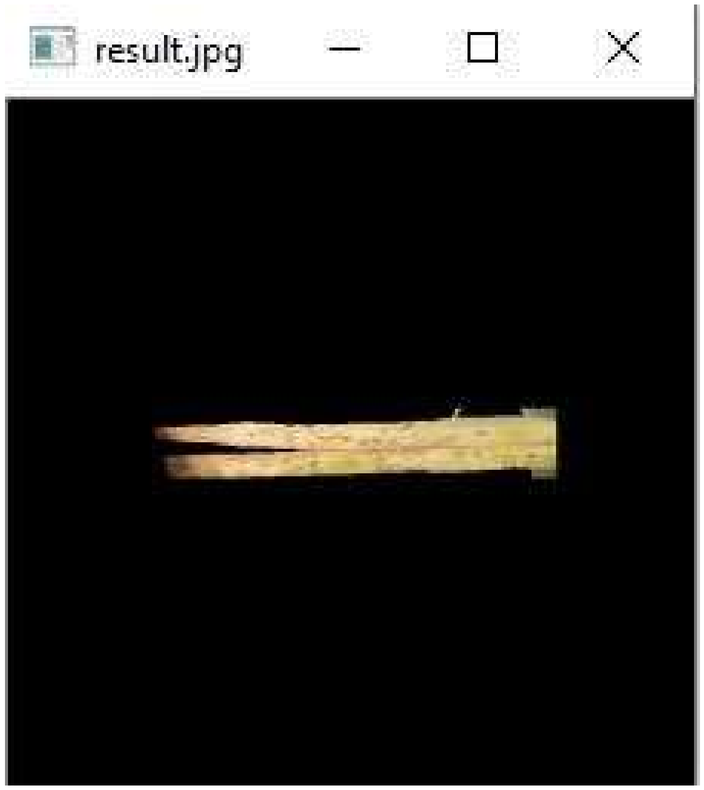


Figure 7.7: ROI Copied onto 600 x 600 px Canvas and Resized to 224 x 224 px

The dimensionality of the canvas is 600 x 600 px whereas the dimensionality of the ROI is 361 x 61 px. The image is then resized to a dimensionality 224 x224 px. While the aspect ratio is preserved, the downside is that the ROI in the final image occupies only a tiny portion of the canvas whereas the majority of the image is occupied by the background. When a feature extractor is trained on such an image, the feature extractor learns the features of the background more than the features of the ROI. Besides, the feature extractor takes significant number of epochs to distinguish the foreground from the background. In such a scenario, there is a high chance that the key features of the ROI are entirely missed by the feature extractor yielding sub-par results. The proposed tool identifies the smallest possible canvas size to fit the ROI and copies the ROI onto the canvas. The canvas is then resized to the dimensionality of the user's choice preserving the aspect ratio. Figure 7.8

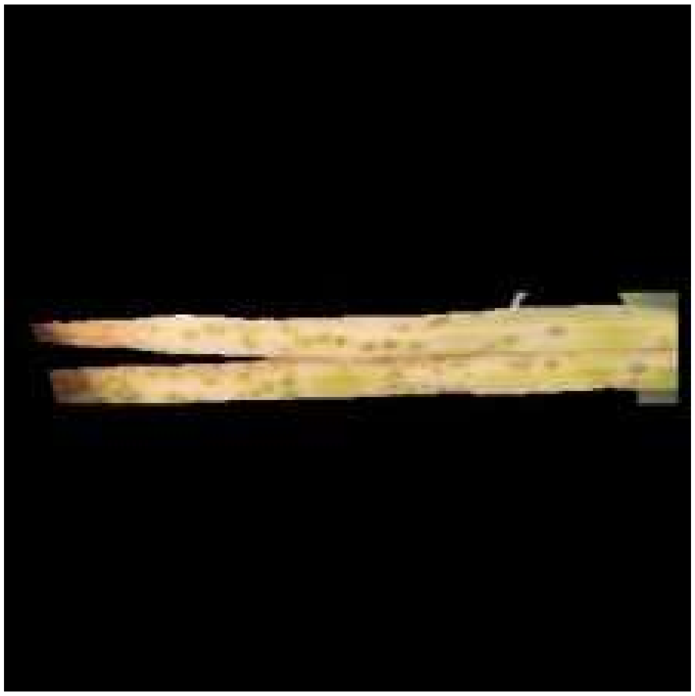


Figure 7.8: ROI Resized to 224 x 224 px from 361 x 61 px

shows the resized image obtained using the proposed tool. The ROI in the resized image occupies a larger area and the clarity of the features is significantly higher.

7.3 Image Eraser

In addition to cropping discussed in section 7.2, another useful application is the ability to remove portions of the image that are undesired. The use case of erasing is different from that of cropping in the sense cropping the image removes pixels from the image and alters the size of the image whereas erasing refers to the act of carefully weeding out portions of the image that hinder the process of information retrieval. The image eraser is a tool that complements foreground extraction tools such as instance segmentation neural networks and OpenCV GrabCut. The goal of the aforementioned extraction tools is the detection of the most informative portion of image. Examples of instance segmentation neural networks include Mask R-CNN and YOLO. Users are able to train the networks to precisely detect a certain part of the image that is most useful for analysis. Likewise, OpenCV GrabCut is used to detect the object in the foreground of the image. While the tools are expected to detect and extracted the desired portions of the image precisely, it is not always the case in practice. Figure 7.9 shows the image of plant whose leaf and stem are focused. There are other plants in the background since the image was captured in a field setting. Figure 7.9 is used to demonstrate the use case for the image eraser.

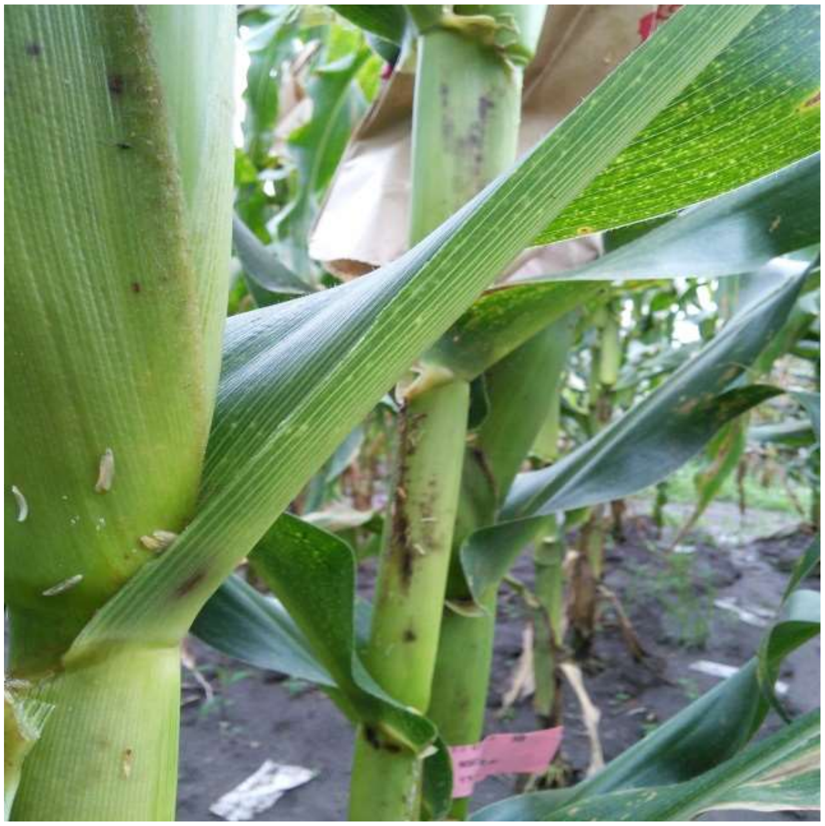


Figure 7.9: Field Image of Leaf and Stem

Assuming that the task at hand is to precisely extract the leaf and stem present in the foreground, OpenCV's GrabCut may be applied. The application of OpenCV's GrabCut algorithm results in the image shown in Figure 7.10.

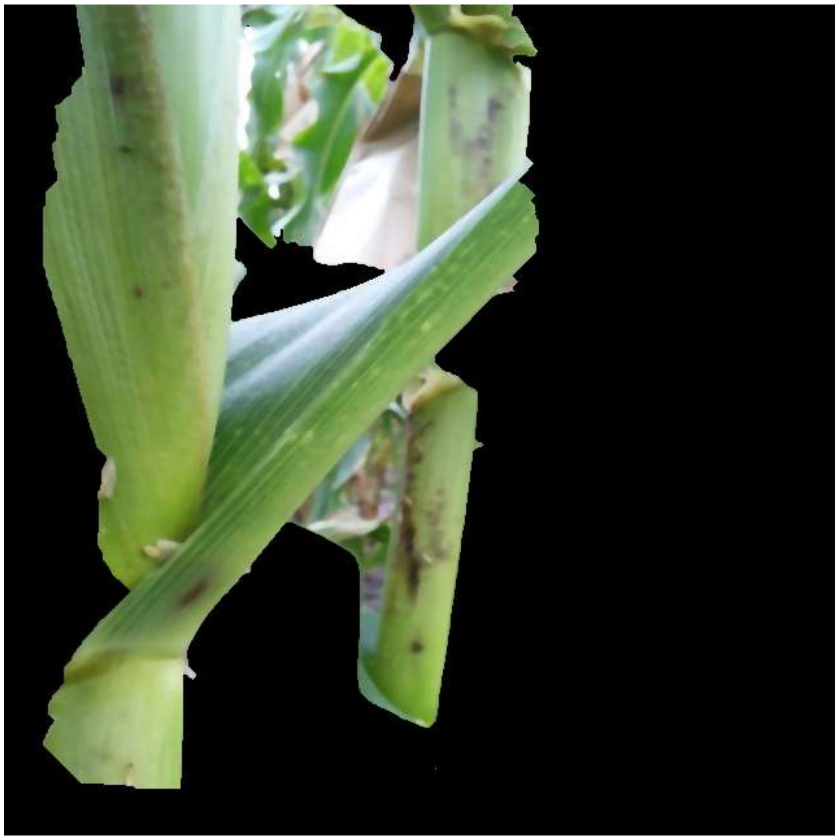


Figure 7.10: Foreground Extraction using OpenCV GrabCut

While the portion of the image showing leaf and stem is present in Figure 7.10, also present are portions of other plants in the background of the image. Foreground extraction is hardly ever perfect owing to the complex nature of the image. All the areas of the image apart from the objects in the foreground area essentially noise in terms of information extraction. The image eraser helps to erase the undesirable portions of the image as a means of noise reduction.

7.3.1 Procedure

The image eraser works by allowing the user to plot a polygon around the object or portion of the image that is undesired. The eraser then eliminates the portion of the image within the polygon. The steps involved in the process are as follows:

1. Plot a polygon around the portion of the image that is not desired. In order to plot the polygon, the tool lets the user plot different points on the image using the left mouse click and joins the points by drawing a line between the points. The right mouse click is used to indicate to the tool that all the points at the desired locations are plotted. Figure 7.11 shows the polygon drawn around the object that is undesired.

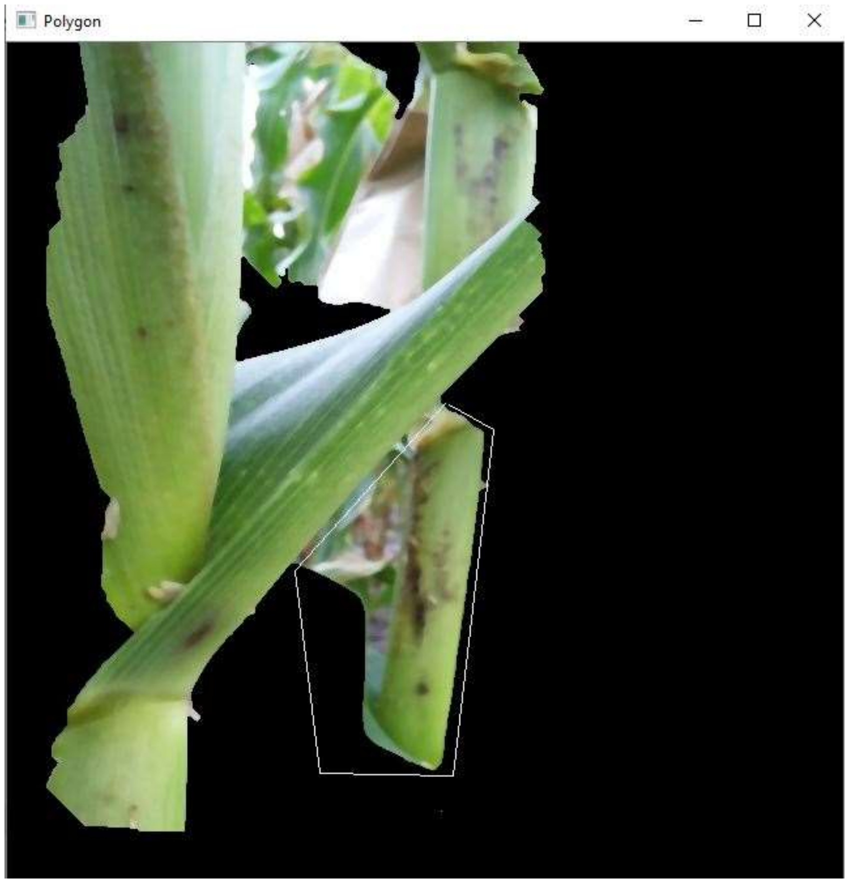


Figure 7.11: Polygon around Undesired Portion of Image

2. The polygon plotted by the user is then filled by the tool in white pixel. Closing the polygon and plotting it in white helps to recognize the region of interest i.e., the region of the image that is required to be erased from the image. Figure 7.12 shows the closed polygon.

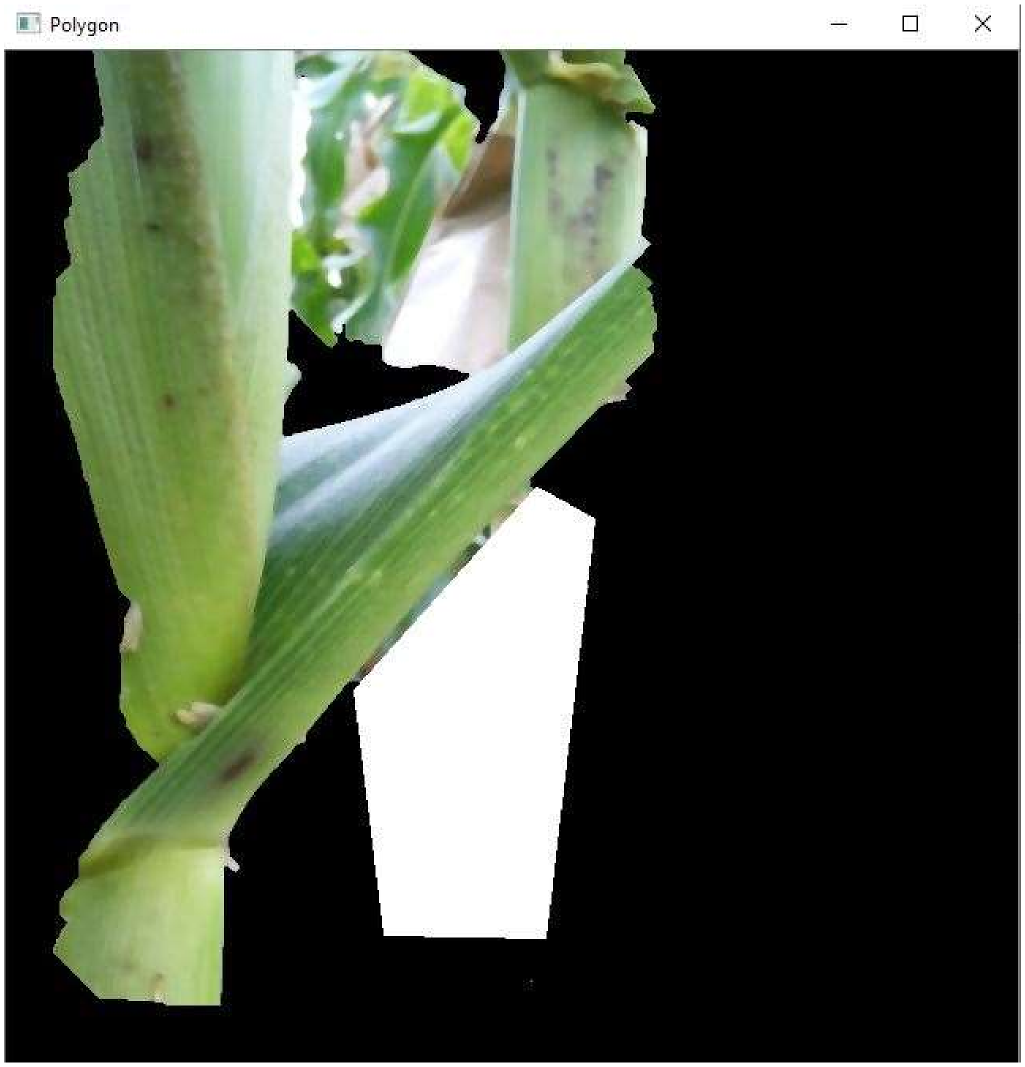


Figure 7.12: Closed Polygon plotted in White Pixel

3. The 'bitwise_xor' operation is performed on the image and the polygon plotted by the user. The 'bitwise_xor' operation returns 0 (false) if both inputs provided to it are true. The idea is that all the pixels where the polygon and image are not 0 (black pixel) are set to 0. Setting the pixel to 0 erases the portion of the image

within the user selected polygon. Figure 7.13 shows the erased portion of the image using the user plotted polygon.

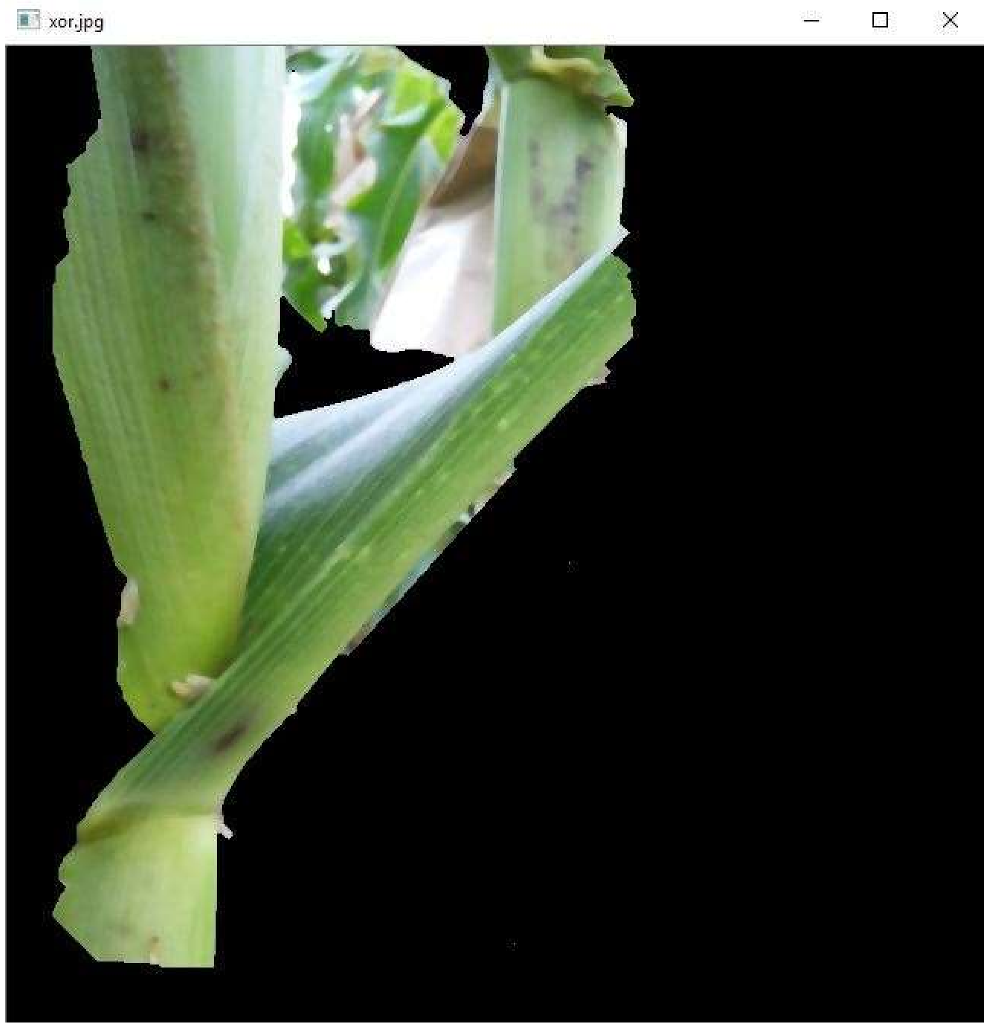


Figure 7.12: Pixels within Polygon erased from Original Image in Figure 7.10

7.4 Code Availability

The code for the tools is available in the GitHub repository at: https://github.com/marven22/Image-Processing-Tools.git . The chapter explains the functionality of three image processing tools namely, the image thresholding trackbar, semi-automated image cropper and image eraser. The tools are handy in the development and testing of image processing applications using OpenCV-Python. While the current implementations are in Python, suitable libraries in C++ are also available for implementation.

Chapter 8 - Conclusions and Future Work

Chapter 2 presents the idea of using low-cost, 3-D printed components to efficiently segment seed kernels for morphometry estimation. In addition, a technique using the conventional watershed method is also proposed for morphometry estimation. The algorithms developed as part of the work are implemented as part of OneKK, an android application developed in collaboration with developers at the Poland Lab in Kansas State University. The application is currently available for download from Google's Play Store and has over 500 downloads. The application also reads from the gyroscope of smart phones to determine the presence of the skew during image capture. The capture of pictures without skew assist in the efficient functioning of the algorithm. In the current state, the algorithms are meant to be used on relatively smaller seeds such as wheat and soy. However, the assimilation of the algorithms for larger seeds such as cassava is to be done in future. The 3-D printed meshes proposed as part of the work may be extendable to the larger seed varieties as well.

Chapter 3 extends the seed kernel morphometry estimation algorithms presented in chapter 2 and presents the technique of applying different kinds of neural networks for seed kernel morphometry estimation. The feasibility of the application of domain randomization to the use case is demonstrated. The combination of synthetic datasets and transfer learning are demonstrated as feasible techniques to achieve high-throughput for seed kernel morphometry estimation. The synthetic image generation framework is proven out on the conventional supervised learning neural network models of VGG and ResNet. In addition, the framework is also proven to provide quality results on self-supervised neural network models and instance segmentation neural network models. Modern day mobile devices are equipped with processors powerful enough to run neural network models. Leveraging the neural network models on mobile devices provides a quality alternative to the image processing solutions in terms of achieving better results since neural networks are able to train and adapt to different problems easily. In future, android applications such as OneKK may be augmented to leverage the power of the neural network models trained on the proposed domain randomization framework to achieve better outcomes.

Chapter 4 proposes an idea to aid the seed packaging industry in packaging seed kernels by count rather than weight owing to its limitations. The idea of stitching multiple slit images together to estimate the number of seeds flowing down a platform delivers quality results, as evident from the experiment. The current idea expands upon the previous idea of tracking seed kernels as they flow down the platform using the technique of centroid tracking. While centroid tracking works well, the algorithm when implemented as an android mobile application, Abacus, takes plenty of processing time and delivers inaccurate results. Although the proposed solution has not been implemented as an android mobile application yet, it is apparent that the algorithm is less resource intensive since it only captures the seed kernels as they flow through the region of interest but doesn’t track the seed kernels all along the video. While the proposed algorithm requires a slow-motion video captured at 60 fps to function efficiently, most modern-day smart phones are equipped with the ability to capture videos at that rate. The next step in the process is to implement the algorithm as part of the Abacus android application developed earlier by Neilsen et al. at Kansas State University. The application when finetuned with the less resource intensive algorithm of slit imaging can be put out for clients and customers to download on Google Play Store. In addition to the use of image processing, other techniques such as the use of microwaves to detect seed kernels will be made part of the experiment to investigate its feasibility.

Chapter 5 carries the aspect of morphometry estimation one step forward and proposes a solution to conduct the volumetric analysis of single seed kernels. The proposed solution is an end-to-end, automated solution wherein the user is provided with a user interface to operate the solution. The proposed setup improves upon the previous iteration developed as part of [24]. The major pitfall of the current iteration is the use of Legos to develop the hardware infrastructure for the project. Moving forward, the hardware is developed using metal to ensure that the hardware is sturdy and more akin to operation in a lab setting such as that of the United States Department of Agriculture (USDA). The setup for the next iteration of the system is shown in Figure 8.1.

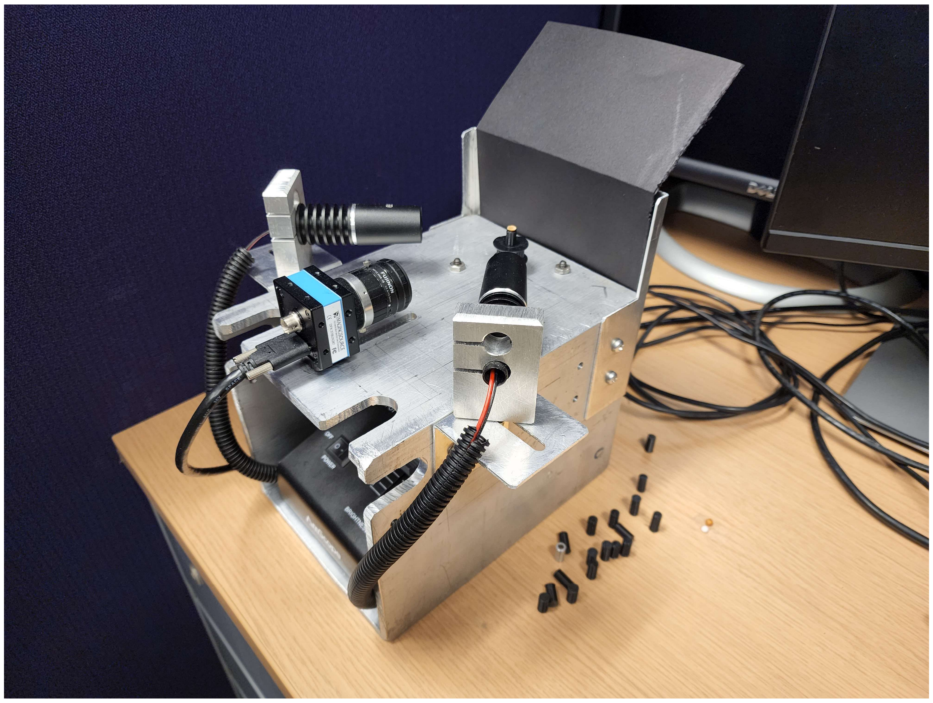


Figure 8.1: Improved Hardware Framework for Volume Estimation of Seed Kernel

The speed of computation is another area that has scope for improvement. Currently, the algorithm captures 36 images, one every 10°, to capture the seed kernel from different angles. The process may be improvised by capturing a video instead of images and meticulously extracting the frames at ten-degree intervals. The material used as the background for seed capture is currently not non-reflective. As a result, a small of the light gets reflected onto the seed kernel which tends to have an impact on the volumetric estimation. The use of a non-reflective material as the background helps to eliminate reflection and improve the accuracy of the results. The system in its current state only estimates the volume of one seed kernel at a time. In the future, a conveyor belt that runs the seed kernels onto the seed station one after the other is of the essence to develop the system into a high-throughput application.

Chapter 6 shifts the focus toward plant phenotyping from seed phenotyping. The chapter presents algorithms for two different use cases, one for leaf angle estimation and another for companion plant determination. Both the use cases presented in the chapter are some of the first and most vetted out solutions. The leaf angle estimation algorithm estimates the angle between leaf and stem to help predict the characteristics of a plant's growth in the field. The algorithm presented uses OpenCV-Python and Tensoflow as the technical stack. The algorithm will be implemented as part of FieldBook, a mobile application developed on the android platform using OpenCV as its backbone for image processing and Mask R-CNN as its backbone for foreground detection. A key aspect that helps improve the functioning of the algorithm is better image capture in the field. Currently, the images captured in the field setting are noisy in the sense they contain plenty of other plants in the background. As a result, the plant of interest is to be detected and identified as the pre-processing step before the leaf angle may be estimated. However, a setup that ensures only the plant of interest is captured helps improve the speed and efficiency of the results of the algorithm. The use of a cardboard as the background is proposed as the simplest means to effect noiseless image capture. Moving forward, the implementation of the algorithm as a mobile application and design of image capture setup will be done to improve the current state of the algorithm and make it more widely adoptable.

The second use of companion plant determination is a project that is being carried out in collaboration with the scientists at the Land Institute in Salina, KS. The algorithm developed as part of the project is perhaps one of the first of its kind since it provides an estimate of the plant area in metric units, unlike other applications that only provide a percentage measure of the plant area. The elimination of skew is one of the key challenges that the algorithm encounters. Moving forward, it is important to leverage the gyroscope of smart phone and other mobile devices as the algorithm is industrialized to address skew. Also, it is important to design an optimal image capture method wherein the images don't contain human shadows because they interfere heavily with the image processing of the algorithm. Moving forward, the algorithm will be tested on other plant images provided by the scientists at the Land Institute in Salina concurrently as it is implemented as a mobile application.

Bibliography

[1] A. Laurentini, "The Visual Hull Concept for Silhouette-Based Image Understanding," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 2, pp. 150–162, 1994.

[2] Abbott, L. (2013). Vegetation Cover Introduction. Retrieved from: https://www.youtube.com/watch?v=TXV5DxEaSR4

[3] Abbott, L. (2013). Vegetation Cover using Ocular Estimation in Plots. Retrieved from: https://youtu.be/wKcKQzNbsds

[4] Abto Software. [Blog] https://www.abtosoftware.com/blog/object-counting-image-processing.

[5] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2010). *Slic superpixels* (No. REP_WORK).

[6] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, *34*(11), 2274-2282.

[7] Aggarwal, N., & Karl, W. C. (2006). Line detection in images through regularized Hough transform. *IEEE transactions on image processing*, *15*(3), 582-591.

[8] Alake, R. (2020). Medium (Blog Post). Retrieved from: https://towardsdatascience.com/understanding-cosine-similarity-and-its-application-fd42f585296a#:~:text=The%20cosine%20similarity%20is%20the,or%20magnitude%20of%20each%20vector.

[9] Aldenhart (2013). Gshield v5 – Synthetos. (GitHub repository) https://github.com/synthetos/grblShield.git

[10] Aleem, M., Raj, R., & Khan, A. (2020). Comparative performance analysis of the resnet backbones of mask rcnn to segment the signs of covid-19 in chest ct scans. *arXiv preprint arXiv:2008.09713*.

[11] Amaravadi, S. (2018). "*Mobile applications for high-throughput seed characterization* (Master's thesis, Kansas State University)".

[12] Arvidsson, S., Pérez-Rodríguez, P., and Mueller-Roeber, B. (2011). A growth phenotyping pipeline for *Arabidopsis thaliana* integrating image analysis and rosette area modeling for robust quantification of genotype effects. *New Phytol*. 191, 895–907. doi: 10.1111/j.1469-8137.2011.03756.x

[13] Atabansi, C. C., Chen, T., Cao, R., & Xu, X. (2021, April). Transfer Learning Technique with VGG-16 for Near-Infrared Facial Expression Recognition. In *Journal of Physics: Conference Series* (Vol. 1873, No. 1, p. 012033). IOP Publishing.

[14] Bah, M. D., Dericquebourg, E., Hafiane, A., & Canals, R. (2018, July). Deep learning-based classification system for identifying weeds using high-resolution UAV imagery. In *Science and Information Conference* (pp. 176-187). Springer, Cham.

[15] Beucher, S., & Meyer, F. (1993). "The morphological approach to segmentation: the watershed transformation". *Mathematical Morphology in Image Processing*, *34*, 433-481.

[16] Bonham, C. D., Mergen, D. E., & Montoya, S. (2004). Plant cover estimation: a contiguous Daubenmire frame. *Rangelands*, *26*(1), 17-22.

[17] Booth, D. T., Cox, S. E., & Berryman, R. D. (2006). Point sampling digital imagery with 'SamplePoint'. *Environmental Monitoring and Assessment*, *123*(1), 97-108.

[18] Bouchard, Louis. What is Self-Supervised Learning? Will machines ever be able to learn like humans? [Blog Post]. Retrieved from: https://medium.com/what-is-artificial-intelligence/what-is-self-supervised-learning-will-machines-be-able-to-learn-like-humans-d9160f40cdd1

[19] Bristeau, P. J., Callou, F., Vissière, D., & Petit, N. (2011). The navigation and control technology inside the ar. drone micro uav. *IFAC Proceedings Volumes*, *44*(1), 1477-1484.

[20] Buckland, M., & Gey, F. (1994). The relationship between recall and precision. *Journal of the American society for information science*, *45*(1), 12-19.

[21] Buters, T., Belton, D., & Cross, A. (2019). Seed and seedling detection using unmanned aerial vehicles and automated image classification in the monitoring of ecological recovery. *Drones*, *3*(3), 53.

[22] C. R. Moore, D. S. Gronwall, N. D. Miller, and E. P. Spalding, "Mapping quantitative trait loci affecting arabidopsis thaliana seed morphology features extracted computationally from images," G3: Genes, Genomes, Genetics, vol. 3, no. 1, pp. 109–118, 2013.

[23] Cao, C. (2020). *Computer vision frameworks for physics-based simulation of liquids and solids*. Kansas State University.

[24] Cao, C., & Neilsen, M. (2020, December). Efficient Seed Volume Measurement Framework. In *2020 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 1328-1334). IEEE.

[25] Chen, W., Lu, S., Liu, B., Li, G., & Qian, T. (2020). Detecting citrus in orchard environment by using improved YOLOv4. *Scientific Programming*, *2020*.

[26] Chen, X., Fan, H., Girshick, R., & He, K. (2020). Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

[27] Chhetri, M., & Fontanier, C. (2021). Use of Canopeo for Estimating Green Coverage of Bermudagrass during Postdormancy Regrowth. *HortTechnology*, *31*(6), 817-819.

[28] Chibane, Julian, Thiemo Alldieck, and Gerard Pons-Moll. “Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion.” (2020)

[29] Colombe, S., Nepal, M. P., Browning, L. B., Miller, M. L., & White, P. T. (2018). Three Sister Crops: Understanding American Indian Agricultural Practices of Corn, Beans and Squash.

[30] Costa, C., Antonucci, F., Pallottino, F., Aguzzi, J., Sun, D. W., Menesatti, P., et al. (2011). Shape analysis of agricultural products: a review of recent research advances and potential application to computer vision. *Food Bioproc. Technol*. 4, 673–692. doi: 10.1007/s11947-011-0556-0

[31] Coulibaly, S., Kamsu-Foguem, B., Kamissoko, D., & Traore, D. (2019). Deep neural networks with transfer learning in millet crop images. *Computers in Industry*, *108*, 115-120.

[32] Culjak, I., Abram, D., Pribanic, T., Dzapo, H., & Cifrek, M. (2012, May). A brief introduction to OpenCV. In *2012 proceedings of the 35th international convention MIPRO* (pp. 1725-1730). IEEE.

[33] da Silva, L. A., Bressan, P. O., Gonçalves, D. N., Freitas, D. M., Machado, B. B., & Gonçalves, W. N. (2019). Estimating soybean leaf defoliation using convolutional neural networks and synthetic images. *Computers and electronics in agriculture*, *156*, 360-368.

[34] Doersch, C., & Zisserman, A. (2019). Sim2real transfer learning for 3D human pose estimation: motion to the rescue. In *Advances in Neural Information Processing Systems* (pp. 12949-12961).

[35] Duda, R. O., & Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, *15*(1), 11-15.

[36] Dzievit, M. J., Li, X., & Yu, J. (2019). Dissection of leaf angle variation in maize through genetic mapping and meta-analysis. *The plant genome*, *12*(1), 180024.

[37] ECE Northwestern retrieved from: http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/images/color11.html

[38] Fenner, M. (Ed.). (2000). *Seeds: the ecology of regeneration in plant communities*. Cabi.

[39] Floyd, D. A., & Anderson, J. E. (1987). A comparison of three methods for estimating plant cover. *The Journal of Ecology*, 221-228.

[40] Ghosal, P., Nandanwar, L., Kanchan, S., Bhadra, A., Chakraborty, J., & Nandi, D. (2019, February). Brain tumor classification using ResNet-101 based squeeze and excitation deep neural network. In *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)* (pp. 1-6). IEEE.

[41] Ghosal, S., Zheng, B., Chapman, S. C., Potgieter, A. B., Jordan, D. R., Wang, X., & Ganapathysubramanian, B. (2019). A weakly supervised deep learning

framework for sorghum head detection and counting. *Plant Phenomics*, *2019*, 1525874.

[42] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Prashant Rai (2020). Ultralytics/yolov5: v3.1 – Bug Fixes and Performance Improvements (Version v3.1). http://doi.org/10.5281/zenodo.4154370

[43] Golbach, F., Kootstra, G., Damjanovic, S., Otten, G., & van de Zedde, R. (2016). Validation of plant part measurements using a 3D reconstruction method suitable for high-throughput seedling phenotyping. *Machine Vision and Applications*, *27*(5), 663-680.

[44] Golzarian, M. R., Frick, R. A., Rajendran, K., Berger, B., Roy, S., Tester, M., & Lun, D. S. (2011). Accurate inference of shoot biomass from high-throughput images of cereal plants. *Plant methods*, *7*(1), 1-11.

[45] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139-144.

[46] Grill, J. B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., ... & Valko, M. (2020). Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*.

[47] Gulzar, Y., Hamid, Y., Soomro, A. B., Alwan, A. A., & Journaux, L. (2020). A Convolution Neural Network-Based Seed Classification System. *Symmetry*, *12*(12), 2018.

[48] Hajar, M. M. A., Lazim, I. M., Rosdi, A. R., & Ramli, L. (2021). Autonomous UAV-Based Cattle Detection And Counting Using Yolov3 And Deep Sort.

[49] Han, Xian-Feng, Hamid Laga, and Mohammed Bennamoun. “Image-Based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era.” IEEE transactions on pattern analysis and machine intelligence 43.5 (2021): 1578–1604.

[50] Hasoi, F., Nakai, Y., Omasa, K. (2009). Estimating the leaf inclination angle distribution of the wheat canopy using a portable scanning lidar. *Journal of Agricultural Meteorology*, *65*(3), 297-302

[51] He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9729-9738).

[52] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

[53] Hedert, T. J. (1995). Stereophotogrammetry of plant leaf angles. *Photogrammetric Engineering & Remote Sensing*, *61*(1), 89-92.

[54] Hendrycks, D., Mazeika, M., Kadavath, S., & Song, D. (2019). Using self-

supervised learning can improve model robustness and uncertainty. *arXiv preprint arXiv:1906.12340*.

[55] Hu, Y., & Zhang, Z. (2021). GridFree: a python package of imageanalysis for interactive grain counting and measuring. *Plant physiology*, *186*(4), 2239-2252.

[56] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).

[57] Huang, R., Pedoeem, J., & Chen, C. (2018, December). YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 2503-2510). IEEE.

[58] Hung, V. (2020). Tensorflow-yolov4-tflite. https://github.com/hunglc007/tensorflow-yolov4-tflite

[59] Islam, S., Khan, S. I. A., Abedin, M. M., Habibullah, K. M., & Das, A. K. (2019, July). Bird species classification from an image using vgg-16 network. In *Proceedings of the 2019 7th International Conference on Computer and Communications Management* (pp. 38-42).

[60] J. Roussel, F. Geiger, A. Fischbach, S. Jahnke, and H. Scharr, "3D surface reconstruction of plant seeds by volume carving: Performance and accuracies," Frontiers in Plant Science, vol. 7, no. June2016, pp. 1–13, 2016.

[61] Jain, D. (2019). Medium [Blog Post]. Retrieved from: https://darshita1405.medium.com/superpixels-and-slic-6b2d8a6e4f08

[62] Jansen, M., Gilmer, F., Biskup, B., Nagel, K. A., Rascher, U., Fischbach, A., et al. (2009). Simultaneous phenotyping of leaf growth and chlorophyll fluorescence via growscreen fluoro allows detection of stress tolerance in *Arabidopsis thaliana* and other rosette plants. *Funct. Plant Biol*. 36, 902–914. doi: 10.1071/FP09095

[63] Jonasson, S. (1988). Evaluation of the point intercept method for the estimation of plant biomass. *Oikos*, 101-106.

[64] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In ECCV, 2016.

[66] Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, *147*, 70-90.

[67] Karmakar, T., Region Proposal Network (RPN) – Backbone of Faster R-CNN [Blog Post]. Retrieved from: https://medium.com/egen/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9

[68] Kehel, Z., Sanchez-Garcia, M., El Baouchi, A., Aberkane, H., Tsivelikas, A., Chen, C., & Amri, A. (2020). "Predictive characterization for seed morphometric traits

for genebank accessions using genomic selection". *Frontiers in Ecology and Evolution*, *8*, 32.

[69] Kiryati, N., Eldar, Y., & Bruckstein, A. M. (1991). A probabilistic Hough transform. *Pattern recognition*, *24*(4), 303-316.

[70] Koc, Ali Bulent. "Determination of Watermelon Volume Using Ellipsoid Approximation and Image Processing." Postharvest biology and technology 45.3 (2007): 366–371.

[71] Komyshev, E., Genaev, M., & Afonnikov, D. (2017). "Evaluation of the SeedCounter, a mobile application for grain phenotyping". *Frontiers in Plant Science*, *7*, 1990.

[72] Kornilov, A. S., & Safonov, I. V. (2018). "An overview of watershed algorithm implementations in open source librarie"s. *Journal of Imaging*, *4*(10), 123.

[73] Kumar, P., Huang, C., Cai, J., & Miklavcic, S. J. (2014). Root phenotyping by root tip detection and classification through statistical learning. *Plant and soil*, *380*(1), 193-209.

[74] Kuznetsova, A., Maleva, T., & Soloviev, V. (2020, December). Detecting Apples in Orchards Using YOLOv3 and YOLOv5 in General and Close-Up Images. In *International Symposium on Neural Networks* (pp. 233-243). Springer, Cham.

[75] Kuznichov, D., Zvirin, A., Honen, Y., & Kimmel, R. (2019). Data augmentation for leaf segmentation and counting tasks in Rosette plants. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 0-0).

[76] Laliberte, A. S., Rango, A., Herrick, J. E., Fredrickson, E. L., & Burkett, L. (2007). An object-based image analysis approach for determining fractional cover of senescent and green vegetation with digital plot photography. *Journal of Arid Environments*, *69*(1), 1-14.

[77] Laurentini, A. "The Visual Hull Concept for Silhouette-Based Image Understanding." IEEE transactions on pattern analysis and machine intelligence 16.2 (1994): 150–162.

[78] Li, L., Zhang, Q., & Huang, D. (2014). "A review of imaging techniques for plant phenotyping". *Sensors*, *14*(11), 20078-20111.

[79] Li, L., Zhang, Q., & Huang, D. (2014). A review of imaging techniques for plant phenotyping. Sensors, 14(11), 20078-20111.

[80] Lin, Y., Tang, C., Chu, F. J., & Vela, P. A. (2020, May). Using Synthetic Data and Deep Networks to Recognize Primitive Shapes for Object Grasping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 10494-10501). IEEE.

[81] Linder, T., Pfeiffer, K. Y., Vaskevicius, N., Schirmer, R., & Arras, K. O. (2020, May). Accurate detection and 3D localization of humans using a novel YOLO-

based RGB-D fusion approach and synthetic training data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1000-1006). IEEE.

[82] Little, B. (2000). *Companion planting in Australia*. New Holland.

[83] Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8759-8768).

[84] Louhaichi, M. (2019). VegMeasure® Software: User Report.

[85] Louhaichi, M., Hassan, S., Clifton, K., & Johnson, D. E. (2018). A reliable and non-destructive method for estimating forage shrub cover and biomass in arid environments using digital vegetation charting technique. *Agroforestry Systems*, *92*(5), 1341-1352.

[86] Lugo, J. J., & Zell, A. (2014). Framework for autonomous on-board navigation with the AR. Drone. *Journal of Intelligent & Robotic Systems*, *73*(1), 401-412.

[87] M. Aharchi and M. Ait Kbir, "A Review on 3D Reconstruction Techniques from 2D Images," pp. 510–522, 2020.

[88] M. MakerGear™, "Makergear m2 3d printer," 2020. [Online]. Available: https://www.makergear.com/products/m2

[89] M. Potmesil, "Generating octree models of 3D objects from their silhouettes in a sequence of images," Computer Vision, Graphics and Image Processing, vol. 40, no. 1, pp. 1–29, 1987.

[90] Mahmood, A., Bennamoun, M., An, S., Sohel, F., Boussaid, F., Hovey, R., & Kendrick, G. (2020). Automatic detection of Western rock lobster using synthetic data. *ICES Journal of Marine Science*, *77*(4), 1308-1317.

[91] Maisonet-Guzman, O. E. (2011). "Food security and population growth in the 21st century". Retr. from http://www.e-ir.info/2011/07/18/food-security-and-population-growth-in-the-21st-century

[92] Mao, J., Tran, D., Ma, W., Naumovski, N., Kellett, J., & Slattery, A. (2020, December). Application of Deep Learning in Automated Meal Recognition. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 58-63). IEEE.

[93] Margapuri, V., & Neilsen, M. (2021). Seed Phenotyping on Neural Networks using Domain Randomization and Transfer Learning. In *2021 ASABE Annual International Virtual Meeting* (p. 1). American Society of Agricultural and Biological Engineers.

[94] Margapuri, V., & Neilsen, M. (2021, December). Classification of Seeds using Domain Randomization on Self-Supervised Learning Frameworks. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 01-08). IEEE.

[95] Margapuri, V., Courtney, C., & Neilsen, M. (2020). Image Processing for High-Throughput Phenotyping of Seeds. In *33rd International Conference on Computer Applications in Industry and Engineering, Las Vegas, NV*.

[96] Margapuri, V., Courtney, C., & Neilsen, M. (2021). Image processing for high-throughput phenotyping of seeds. *EPiC Series in Computing*, *75*, 69-79.

[97] Margapuri, V., Penumajji, N., & Neilsen, M. (2021, December). Seed Classification using Synthetic Image Datasets Generated from Low-Altitude UAV Imagery. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 116-121). IEEE.

[98] Margapuri, V., Rife, T., Courtney, C., Schlautman, B., Zhao, K., Neilsen, M. Fractal Vegetation Cover Estimation using Hough Lines and Linear Iterative Clustering. Pre-print: https://arxiv.org/abs/2205.00366

[99] Margapuri, V., Zhao, K., & Neilsen, M. (2022). Automated Phenotyping of Single Seeds using a Novel Volume Sculpting Framework. In *2022 ASABE Annual International Meeting* (p. 1). American Society of Agricultural and Biological Engineers. (In-Press).

[100] Martin, Worthy N, and J. K Aggarwal. “Volumetric Descriptions of Objects from Multiple Views.” IEEE transactions on pattern analysis and machine intelligence PAMI-5.2 (1983): 150–158. Web.

[101] Mateen, M., Wen, J., Song, S., & Huang, Z. (2019). Fundus image classification using VGG-19 architecture with PCA and SVD. *Symmetry*, *11*(1), 1.

[102] Menzel, M. I., Tittmann, S., Buehler, J., Preis, S., Wolters, N., Jahnke, S., ... & KRAUSE, H. J. (2009). Non-invasive determination of plant biomass with microwave resonators. *Plant, cell & environment*, *32*(4), 368-379.

[103] Microsoft Build retrieved from: https://docs.microsoft.com/en-us/windows/win32/wcs/rgb-color-spaces

[104] Miller, N. D., Haase, N. J., Lee, J., Kaeppler, S. M., de Leon, N., & Spalding, E. P. (2017). A robust, high-throughput method for computing maize ear, cob, and kernel attributes automatically from images. *The Plant Journal*, *89*(1), 169-178.

[105] Mir, R. R., Reynolds, M., Pinto, F., Khan, M. A., & Bhat, M. A. (2019). “High-throughput phenotyping for crop improvement in the genomics era”. *Plant Science*, *282*, 60-72.

[106] Misra, I., & Maaten, L. V. D. (2020). Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 6707-6717).

[107] Mnih, V., & Hinton, G. E. (2010, September). Learning to detect roads in high-resolution aerial images. In *European Conference on Computer Vision* (pp. 210-223). Springer, Berlin, Heidelberg.

[108] Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for

image-based plant disease detection. *Frontiers in plant science*, *7*, 1419.

[109] Monforte, A. J., Diaz, A., Caño-Delgado, A., and van der Knaap, E. (2014). The genetic basis of fruit morphology in horticultural crops: lessons from tomato and melon. *J. Exp. Bot*. 65, 4625–4637. doi: 10.1093/jxb/eru017

[110] Mukti, I. Z., & Biswas, D. (2019, December). Transfer learning-based plant diseases detection using ResNet50. In *2019 4th International Conference on Electrical Information and Communication Technology (EICT)* (pp. 1-6). IEEE.

[111] Muneer, A., & Fati, S. M. (2020). Efficient and Automated Herbs Classification Approach Based on Shape and Texture Features using Deep Learning. *IEEE Access*, *8*, 196747-196764.

[112] Nafi, N. M., & Hsu, W. H. (2020, July). Addressing Class Imbalance in Image-Based Plant Disease Detection: Deep Generative vs. Sampling-Based Approaches. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)* (pp. 243-248). IEEE.

[113] Nai, A. SimCLR: Contrastive Learning of Visual Representations [Blog Post]. Retrieved from: https://medium.com/@nainaakash012/simclr-contrastive-learning-of-visual-representations-52ecf1ac11fa

[114] Neilsen, M. L., Courtney, C., Amaravadi, S., Xiong, Z., Poland, J., & Rife, T. (2017, October). "A dynamic, real-time algorithm for seed counting". In *Proc. Of the 26th International Conference on Software Engineering and Data Engineering*.

[115] Neilsen, M. L., Courtney, C., Amaravadi, S., Xiong, Z., Poland, J., & Rife, T. (2017, October). A dynamic, real-time algorithm for seed counting. In *Proc. Of the 26th International Conference on Software Engineering and Data Engineering*.

[116] Neilsen, M. L., Gangadhara, S. D., & Rife, T. (2016, September). "Extending watershed segmentation algorithms for high throughput phenotyping". In *Proceedings of the 29th International Conf. on Computer Applications in Industry and Engineering, Denver, CO*.

[117] Nepal, U., & Eslamiat, H. (2022). Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs. *Sensors*, *22*(2), 464.

[118] NOBORIO, H, S FUKUDA, and S ARIMOTO. "Construction of the Octree Approximating Three-Dimensional Objects by Using Multiple Views." IEEE transactions on pattern analysis and machine intelligence 10.6 (1988): 769–782. Print.

[119] Odom, F. Easy Self-Supervised Learning with BYOL [Blog Post]. Retrieved from: https://medium.com/the-dl/easy-self-supervised-learning-with-byol-53b8ad8185d

[120] Olsson, K., & Persson, T. (2002). Shape from silhouette scanner.

[121] Oord, A. V. D., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

[122] Parico, A. I. B., & Ahamed, T. (2021). Real time pear fruit detection and counting using YOLOv4 models and deep SORT. *Sensors*, *21*(14), 4803.

[123] Patil, K., Kulkarni, M., Sriraman, A., & Karande, S. (2017, December). Deep learning-based car damage classification. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)* (pp. 50-54). IEEE.

[124] Patrignani, A., & Ochsner, T. E. (2015). Canopeo: A powerful new tool for measuring fractional green canopy cover. *Agronomy Journal*, *107*(6), 2312-2320.

[125] Peng, X. B., Andrychowicz, M., Zaremba, W., & Abbeel, P. (2018, May). Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)* (pp. 1-8). IEEE.

[126] Pound, Michael P et al. "Automated Recovery of Three-Dimensional Models of Plant Shoots from Multiple Color Images." Plant physiology (Bethesda) 166.4 (2014): 1688–1698.

[127] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In ICCV, 2019.

[128] Psychology wiki retrieved from: https://psychology.fandom.com/wiki/Lab_color_space

[129] Purevdorj, T. S., Tateishi, R., Ishiyama, T., & Honda, Y. (1998). Relationships between percent vegetation cover and vegetation indices. *International journal of remote sensing*, *19*(18), 3519-3535.]

[130] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision. Cambridge University Press, 2000. [Online]. Available: https://books.google.com/books?id=MHrYnQEACAAJ

[131] R. P. Herridge, R. C. Day, S. Baldwin, and R. C. Macknight, "Rapid analysis of seed size in Arabidopsis for mutant and QTL discovery," Plant Methods, vol. 7, no. 1, pp. 1–11, 2011.

[132] Rahnemoonfar, M., & Sheppard, C. (2017). Deep count: fruit counting based on deep simulated learning. *Sensors*, *17*(4), 905.

[133] Rastogi, A. Reducing computational constraints in SimCLR using Momentum Contrast V2 (MoCo-V2) in PyTorch [Blog Post]. Retrieved from: https://medium.com/analytics-vidhya/simclr-with-less-computational-constraints-moco-v2-in-pytorch-3d8f3a8f8bf2

[134] Rastogi, A. Understanding SimCLR – A Simple Framework for Contrastive Learning of Visual Representations with Code [Blog Post]. Retrieved from: https://medium.com/analytics-vidhya/understanding-simclr-a-simple-framework-for-contrastive-learning-of-visual-representations-d544a9003f3c

[135] Razmi, T. Mask R-CNN [Blog Post]. Retrieved from: https://medium.com/@tibastar/mask-r-cnn-d69aa596761f

[136] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[137] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*.

[138] Riddle, D.F., 1979. Calculus and Analytic Geometry. Wadsworth Publishing Company, Inc. Belmont, CA, USA. pp. 506 – 507.

[139] Rife, T. W., & Poland, J. A. (2014). Field book: an open-source application for field data collection on android. *Crop Science*, *54*(4), 1624-1627.

[140] Rozantsev, A., Lepetit, V., & Fua, P. (2015). On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding*, *137*, 24-37.

[141] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Berg, A. C. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211-252.

[142] Ryu, Y., Sonnentag, O., Nilson, T., Vargas, R., Kobayashi, H., Wenk, R., & Baldocchi, D. D. (2010). How to quantify tree leaf area index in an open savanna ecosystem: a multi-instrument and multi-model approach. *Agricultural and Forest Meteorology*, *150*(1), 63-76.

[143] Saatkamp, A., Cochrane, A., Commander, L., Guja, L. K., Jimenez-Alfaro, B., Larson, J., ... & Walck, J. L. (2019). A research agenda for seed-trait functional ecology. *New Phytologist*, *221*(4), 1764-1775.

[144] Sagi, O., & Rokach, L. (2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(4), e1249.

[145] Sajjadi, M. S., Bachem, O., Lucic, M., Bousquet, O., & Gelly, S. (2018). Assessing generative models via precision and recall. *arXiv preprint arXiv:1806.00035*.

[146] Sankaran, S., Wang, M., & Vandemark, G. J. (2016). Image-based rapid phenotyping of chickpeas seed size. *Engineering in agriculture, environment and food*, *9*(1), 50-55.

[147] Santos, L., Santos, F. N., Oliveira, P. M., & Shinde, P. (2019, November). Deep learning applications in agriculture: A short review. In *Iberian Robotics conference* (pp. 139-151). Springer, Cham.

[148] Santos, T. T., de Souza, L. L., dos Santos, A. A., & Avila, S. (2020). Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. *Computers and Electronics in Agriculture*, *170*, 105247.

[149] Saranya, K. C., Thangavelu, A., Chidambaram, A., Arumugam, S., & Govindraj, S. (2020). Cyclist detection using tiny yolo v2. In *Soft Computing for Problem Solving* (pp. 969-979). Springer, Singapore.

[150] Schrader, J., Pillar, G., & Kreft, H. (2017). “Leaf-IT: An Android application for measuring leaf area”. *Ecology and Evolution*, *7*(22), 9731-9738.

[151] Schweigert, A., Blesing, C., & Friedrich, C. M. (2020). Deep Learning Based Hazard Label Object Detection for Lithium-ion Batteries Using Synthetic and Real Data. In *Deep Learning Applications* (pp. 137-154). Springer, Singapore.

[152] Seedcount. Next Instruments (2015)., Accessed June 2015.

[153] Shrimali, K. (2018). Convex Hull using OpenCV in Python and C++. (Blog) retrieved from: https://learnopencv.com/convex-hull-using-opencv-in-python-and-c/

[154] Sinoquet, H., Thanisawanyangkura, S., Mabrouk, H., & Kasemsap, P. (1998). Characterization of the light environment in canopies using 3D digitising and image processing. *Annals of Botany*, *82*(2), 203-212.

[155] Sinoquet, H., Thanisawanyangkura, S., Mabrouk, H., & Kasemsap, P. (1998). Characterization of the light environment in canopies using 3D digitising and image processing. *Annals of Botany*, *82*(2), 203-212.

[156] Song, Y. (2016). *Automatic assessment of biological control effectiveness of the egg parasitoid Trichogramma bourarachar against Cadra cautella using machine vision* (Doctoral dissertation, Kansas State University).

[157] Steinmetz solid - Wikipedia, https://en.wikipedia.org/wiki/Steinmetz solid, (Accessed on 10/20/2020).

[158] Steinmetz solid - Wikipedia, https://en.wikipedia.org/wiki/Steinmetz solid, (Accessed on 10/20/2020).

[159] Steven Owen, Timothy Shead, Shawn Martin, "Defeaturing Using Machine Learning", 27th International Meshing Roundtable, Research Note Albuquerque, NM, Oct 2018

[160] Sur, R. SimCLR – A Simple Framework for Constrastive Learning of Visual Representation [Blog Post]. Retrieved from: https://rittikasur16.medium.com/simclr-a-simple-framework-for-contrastive-learning-of-visual-representation-799a903c2779

[161] Sural, S., Qian, G., & Pramanik, S. (2002, September). Segmentation and histogram generation using the HSV color space for image retrieval. In *Proceedings. International Conference on Image Processing* (Vol. 2, pp. II-II). IEEE.

[162] Suzuki, S. (1985). “Topological structural analysis of digitized binary images by border following”. *Computer Vision, Graphics, and Image Processing*, *30*(1), 32-46.

[163] Systat Software, Inc. Retrieved from: https://www.environmental-expert.com/software/sigmascan-version-pro-50-automatically-images-analyze-software-560982

[164] Tanabata, T., Shibaya, T., Hori, K., Ebana, K., & Yano, M. (2012). “SmartGrain: high-throughput phenotyping software for measuring seed shape through image analysis”. *Plant Physiology*, *160*(4), 1871-1880.

[165] Theckedath, D., & Sedamkar, R. R. (2020). Detecting Affect States Using VGG16, ResNet50 and SE-ResNet50 Networks. *SN Computer Science*, *1*(2), 1-7.

[166] Thevenot, A. (2020). Medium. [Blog Post]. Retrieved from: https://towardsdatascience.com/understand-and-visualize-color-spaces-to-improve-your-machine-learning-and-deep-learning-models-4ece80108526

[167] Tian, X., & Chen, C. (2019, September). Modulation pattern recognition based on Resnet50 neural network. In *2019 IEEE 2nd International Conference on Information Communication and Signal Processing (ICICSP)* (pp. 34-38). IEEE.

[168] Tinmay Thaker, VGG 16 Easiest Explanation [Blog Post]. Retrieved from: https://medium.com/nerd-for-tech/vgg-16-easiest-explanation-12453b599526

[169] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017, September). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 23-30). IEEE.

[170] Toda, Y., Okura, F., Ito, J., Okada, S., Kinoshita, T., Tsuji, H., & Saisho, D. (2020). Training instance segmentation neural network with synthetic datasets for crop seed phenotyping. *Communications biology*, *3*(1), 1-12.

[171] Toth, D., Cimen, S., Ceccaldi, P., Kurzendorfer, T., Rhode, K., & Mountney, P. (2019, May). Training deep networks on domain randomized synthetic X-ray data for cardiac interventions. In *International Conference on Medical Imaging with Deep Learning* (pp. 468-482). PMLR.

[172] Trammel, M. (2020). Seed Certification: What Is It and What Does It Mean for You. *Noble News and Views, Volumes, 38(4), 1-2.*

[173] Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., ... & Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 969-977).

[174] USDA. Ag and Food Sectors and the Economy [Blog Post]. Retrieved from: https://www.ers.usda.gov/data-products/ag-and-food-statistics-charting-the-essentials/ag-and-food-sectors-and-the-economy/

[175] Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., & Schmid, C. (2017). Learning from synthetic humans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 109-117).

[176] Vuola, A. O., Akram, S. U., & Kannala, J. (2019, April). Mask-RCNN and U-net ensembled for nuclei segmentation. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)* (pp. 208-212). IEEE.

[177] W. N. Martin, W. N. Martin, and J. K. Aggarwal, “Volumetric Descriptions of Objects from Multiple Views,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-5, no. 2, pp. 150–158, 1983.

[178] Walter, A., Liebisch, F., & Hund, A. (2015). Plant phenotyping: from bean weighing to image analysis. *Plant methods*, *11*(1), 1-11.

[179] Ward, D., Moghadam, P., & Hudson, N. (2018). Deep leaf segmentation using synthetic data. *arXiv preprint arXiv:1807.10931*.

[180] Wei, J., Zhang, N., Wang, N., Lenhert, D., Neilsen, M., & Mizuno, M. (2005). Use of the “smart transducer” concept and IEEE 1451 standards in system integration for precision agriculture. *Computers and Electronics in Agriculture*, *48*(3), 245-255.

[181] Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, *3*(1), 9.

[182] Whan, A. P., Smith, A. B., Cavanagh, C. R., Ral, J. P. F., Shaw, L. M., Howitt, C. A., & Bischof, L. (2014). “GrainScan: a low cost, fast method for grain size and colour measurements”. *Plant Methods*, *10*(1), 23.

[183] Wikipedia. Retrieved from: https://en.wikipedia.org/wiki/Interquartile_range

[184] Wiley, V., & Lucas, T. (2018). “Computer vision and image processing: a paper review”. *International Journal of Artificial Intelligence Research*, *2*(1), 29-36.

[185] WinFOLIA by Regent Instruments Inc. Retrieved from: https://regentinstruments.com/assets/winfolia_software.html

[186] Winseedle. Regent Instruments (2000)., Ville de Quebec: Instruments ` Regent Inc., Accessed June 2015.

[187] Wu, L., Hoi, S. C., & Yu, N. (2010). Semantics-preserving bag-of-words models and applications. *IEEE Transactions on Image Processing*, *19*(7), 1908-1920.

[188] X. Han, H. Laga, and M. Bennamoun, “Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era,” IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–1, 2019.

[189] X. Ren and J. Malik, “Learning a Classification Model for Segmentation,” Proceedings of International Conference on Computer Vision (ICCV), Nice, 13-16 October 2003, pp. 10-17. doi:10.1109/ ICCV.2003.1238308

[190] Xia, P., Zhang, L., & Li, F. (2015). Learning similarity with cosine similarity ensemble. *Information Sciences*, *307*, 39-52.

[191] Yang, Myongkyoon, and Seong-In Cho. “High-Resolution 3D Crop Reconstruction and Automatic Analysis of Phenotyping Index Using Machine Learning.” Agriculture (Basel) 11.10 (2021): 1010–.

[192] Yann LeCun. Predictive learning, 2016. URL https://www.youtube.com/watch?v= Ount2Y4qxQo.

[193] Yehao (2020). COCO-annotations_convert. Github.com/ujsyehao. https://github.com/ujsyehao/COCO-annotations-darknet-format

[194] Yu, X., & Guo, X. (2021). Extracting Fractional Vegetation Cover from Digital Photographs: A Comparison of In Situ, SamplePoint, and Image Classification Methods. *Sensors*, *21*(21), 7310.

[195] Zakharov, S., Kehl, W., & Ilic, S. (2019). Deceptionnet: Network-driven domain randomization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 532-541).

[196] Zhang, H., Zhang, N., Wang, N., Yang, Q., & Hu, J. (2011). A general agricultural information management architecture for distributed wireless sensor network. In *2011 Louisville, Kentucky, August 7-10, 2011* (p. 1). American Society of Agricultural and Biological Engineers.

[197] Zhang, N., Wang, N., Elfaki, M. S., CHAISATTAPAGON, C., & Fan, G. E. (2006). Weed identification using imaging technology and spectroscopy. *Environmental Control in Biology*, *44*(3), 161-171.

[198] Zhao, T., & Nevatia, R. (2003). Car detection in low resolution aerial images. *Image and Vision Computing*, *21*(8), 693-7

[199] Zou, X., Mõttus, M., Tammeorg, P., Torres, C. L., Takala, T., Pisek, J., ... & Pellikka, P. (2014). Photographic measurement of leaf angles in field crops. *Agricultural and Forest Meteorology*, *184*, 137-146.