DEEP LEARNING-BASED PANICLE DETECTION BY USING HYPERSPECTRAL IMAGERY

by

Ruya Xu

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Civil Engineering

Lyles School of Civil Engineering

West Lafayette, Indiana

August 2020

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF COMMITTEE APPROVAL

Dr. Melba Crawford, Chair
Lyles School of Civil Engineering

Dr. Ayman Habib
Lyles School of Civil Engineering

Dr. Mohammad Jahanshahi
Lyles School of Civil Engineering

Approved by:
Dr. Dulcy M. Abraham

ACKNOWLEDGMENTS

I would like to express my very great appreciation to Dr. Crawford for her patient guidance, constructive suggestions and criticism during the development of this research work. I would also like to thank Dr. Habib and Dr. Jahanshahi, for their advice in refining the details in my master thesis document. I would like to thank the TERRA group for their assistance with the collection of the data. I wish to thank my team members from Laboratory for Applications of Remote Sensing (LARS) group for their help in this work. Karoll, for her help in improving the thesis document and oral presentation. Ali and Taojun, for their technical support on Deep Learning and Orthorectification, respectively. I am particularly grateful for the assistance given by Azam for her suggestions on building the model and the reminders for the parameters that are important to pay attention to. I would also like to thank the staff of the Civil Engineering department for giving me instructions on graduation. Finally, I wish to thank my parents for their support and encouragement throughout my study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

10-cv	10-fold cross validation
DBSCAN	Density-based Spatial Clustering of Applications with Noise
CCNN	Counting Convolutional Neural Network
CNN	Convolutional Neural Network
DSM	Digital Surface Model
FCN	Fully Convolutional Network
FCRN	Fully Convolutional Regression Networks
FV	Fisher Vector
GSD	Ground Sampling Distance
GCP	Ground Control Point
HPC	High Performance Computing
ISPRS	International Society for Photogrammetry and Remote Sensing
ITaP	Information Technology at Purdue
IU	region intersection over union metric
LDA	Linear Discriminant Analysis
LSA	Least Square Adjustment
MLR	multinomial logistic regression
MV	Machine Vision
NDSM	Normalized Digital Surface Model
NDVI	Normalized Difference Vegetation Index
PCA	Principal Component Analysis
ReLu’s	Rectified Linear Units
RNN	Recurrent Neural Network
SDAE	Stacked Denoise Autoencoder
SDAE	Spatial updated Deep Autoencoder
SGD	stochastic gradient descent
SIFT	Scale Invariant Feature Transform
SLIC	Simple Linear Iterative Clustering
STD	Standard Deviation
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicles

ABSTRACT

Sorghum, which is grown internationally as a cereal crop that is robust to heat, drought, and disease, has numerous applications for food, forage, and biofuels. When monitoring the growth stages of sorghum, or phenotyping specific traits for plant breeding, it is important to identify and monitor the panicles in the field due to their impact relative to grain production. Several studies have focused on detecting panicles based on data acquired by RGB and multispectral remote sensing technologies. However, few experiments have included hyperspectral data because of its high dimensionality and computational requirements, even though the data provide abundant spectral information. Relative to analysis approaches, machine learning, and specifically deep learning models have the potential of accommodating the complexity of these data. In order to detect panicles in the field with different physical characteristics, such as colors and shapes, very high spectral and spatial resolution hyperspectral data were collected with a wheeled-based platform, processed, and analyzed with multiple extensions of the VGG-16 Fully Convolutional Network (FCN) semantic segmentation model.

In order to have correct positioning, orthorectification experiments were also conducted in the study to obtain the proper positioning of the image data acquired by the pushbroom hyperspectral camera at near range. The scale of the DSM derived from LiDAR that was used for orthorectification of the hyperspectral data was determined to be a critical issue, and the application of the Savitzky-Golay filter to the original DSM data was shown to contribute to the improved quality of the orthorectified imagery.

Three tuned versions of the VGG-16 FCN Deep Learning architecture were modified to accommodate the hyperspectral data: PCA&FCN, 2D-FCN, and 3D-FCN. It was concluded that all the three models can detect the late season panicles included in this study, but the end-to-end models performed better in terms of precision, recall, and the F-score metrics . Future work should focus on improving annotation strategies and the model architecture to detect different panicle varieties and to separate overlapping panicles based on an adequate quantities of training data acquired during the flowering stage.

1. INTRODUCTION

1.1 Research Motivation

Sorghum is one of the most important cereal crops grown worldwide, both for its food value and as a biofuel, particularly in environments experiencing excessive heat and drought. Panicles, which are the flowering part of a sorghum plant, are important because of their relationship to the stage of plant growth and yield (Kumar et al., 2013). Phenotyping, which involves measuring characteristics of plants to obtain a quantitative description of their biophysical properties, includes the acquisition of information about panicles in sorghum, including the number per plot for different varieties (Walter et al., 2015). It is very difficult to count panicles over a large area and to monitor them through different periods of the reproductive cycle because their characteristics can vary significantly over time and across varieties. Recent development of remote sensing technologies is contributing to non-destructive, image-based phenotyping, seeking to reduce time-consuming, costly manual measurements, and extend measurements over large areas using high spatial resolution imagery (Walter et al., 2015). For example, Olsen et al. (2018) and Oh et al., (2019) investigated use of RGB images via a UAV platform which covered the whole field multiple times per season, coupled with local weather (temperature) data to detect panicles.

Studies have been conducted to detect rice panicles based on remote sensing multispectral imagery and LiDAR data (Kumar et al., 2013). However, detection of sorghum panicles based on hyperspectral image data remains mostly unexplored, largely because of the lack of available hyperspectral data at high spatial resolution. Multispectral and hyperspectral imagery provide chemistry-based information in multiple windows of the electromagnetic spectrum (Shimoni et al, 2019). Multispectral data have a few disjoint bands, while hyperspectral data have hundreds of contiguous bands, providing the opportunity to discriminate objects with similar spectral signatures. Instead of acquiring imagery as a 2D array, hyperspectral data acquired by pushbroom sensors are essentially a 3D cube representing spectral and spatial content in a scene. Hyperspectral reflectance data have been shown to provide unique information for multiple applications in agriculture, including disease detection (Liu et al., 2010), weed detection (Okamoto et al., 2007), fruit quality evaluation (Nagata et al., 2004), plant phenotyping (Brugger et al., 2019) and nitrogen content prediction (Bruning et al., 2019) either by a hand-held spectrometer (Liu et al., 2010), in controlled facilities (Nagata et al., 2004; Okamoto et al., 2007), or from mobile platforms (Change et al., 2017; Malambo et al., 2018, Masjedi et al., 2019). The rich spectral information provides potential opportunities for increased applications of hyperspectral data. Relative to panicle detection, both the spatial and the spectral domains provide information to distinguish panicles from the leaves and ground, especially in the early season when the color of panicles is similar to the leaves.

In order to detect panicles, various classification strategies have been applied to RGB imagery and other remote sensing data such as LiDAR. Generally, the approaches are categorized as pixel-based classification and object-based classification. Pixel-based classification assigns a label to each pixel by using the intensity or texture information from the image, while object-based classification detects the class of each object. For example, Tang et al. (2012) converted RGB data to the hue, saturation, intensity (HSI) color space, then performed a region growing based segmentation on the thresholded hue component to detect maize tassels. Malambo et al. (2018) applied Density-based Spatial Clustering of Applications with Noise (DBSCAN) to detect panicles from LiDAR data, in conjunction with object-based segmentation. This approach segments images into small patches first by using a clustering or a segmentation method such as Simple Linear Iterative Clustering (SLIC) (Xiong et al., 2017). A classifier, such as a Support Vector Machine (SVM) or Logistic Regression, is then applied to detect panicles. As an approach developed for classification of complex data, a deep learning model also yielded remarkable performance in detecting objects like rice panicles (Xiong et al., 2017) or leaves of individual maize plants (Jin et al., 2018).

To the best of our knowledge, limited studies have investigated panicle detection using high dimensional hyperspectral image data. Firstly, it is difficult to acquire large quantities of calibrated high quality hyperspectral reference data , which is necessary for supervised classification. Secondly, spectral redundancy of contiguous bands makes it more difficult to analyze than RGB image data with only three channels, or multispectral data with a few, typically uncorrelated bands. Because of the potential contribution of hyperspectral data to improve panicle detection, there is significant justification for additional research to explore this capability. In this thesis, an end-to-end, pixel-to-pixel deep learning model is investigated for the detection of panicles from high resolution hyperspectral image data.

1.2 Challenges to the Research

Sorghum panicles are extremely challenging to detect because they exhibit a wide range of temporal and variety dependent characteristics, including color, texture, size and shape. For example, panicles in Figure 1.1 (a) are red, while panicles in Figure 1.1 (b) are pink on the same date, and their shapes are different. Even for the same variety of panicles on the same date, e.g., Figure 1.1 (e), the color might also vary due to the illumination or the stage of development, which can change rapidly during the growing season. For instance, Figures 1.1 (c) and (f) show two different types of panicles on the same date. In Figure 1.1 (c), senescing leaves have the same color as panicles, but in Figure 1.1 (f), some panicles are still green, making it difficult to discriminate them from leaves. Further, in Figure 1.1 (d), dry leaves might be detected as panicles, while green panicles are likely to be merged with a non-panicle area. These issues all increase the difficulty of discriminating the panicles. As a result, visible color and shape information alone from an RGB image might not be adequate for panicle detection.

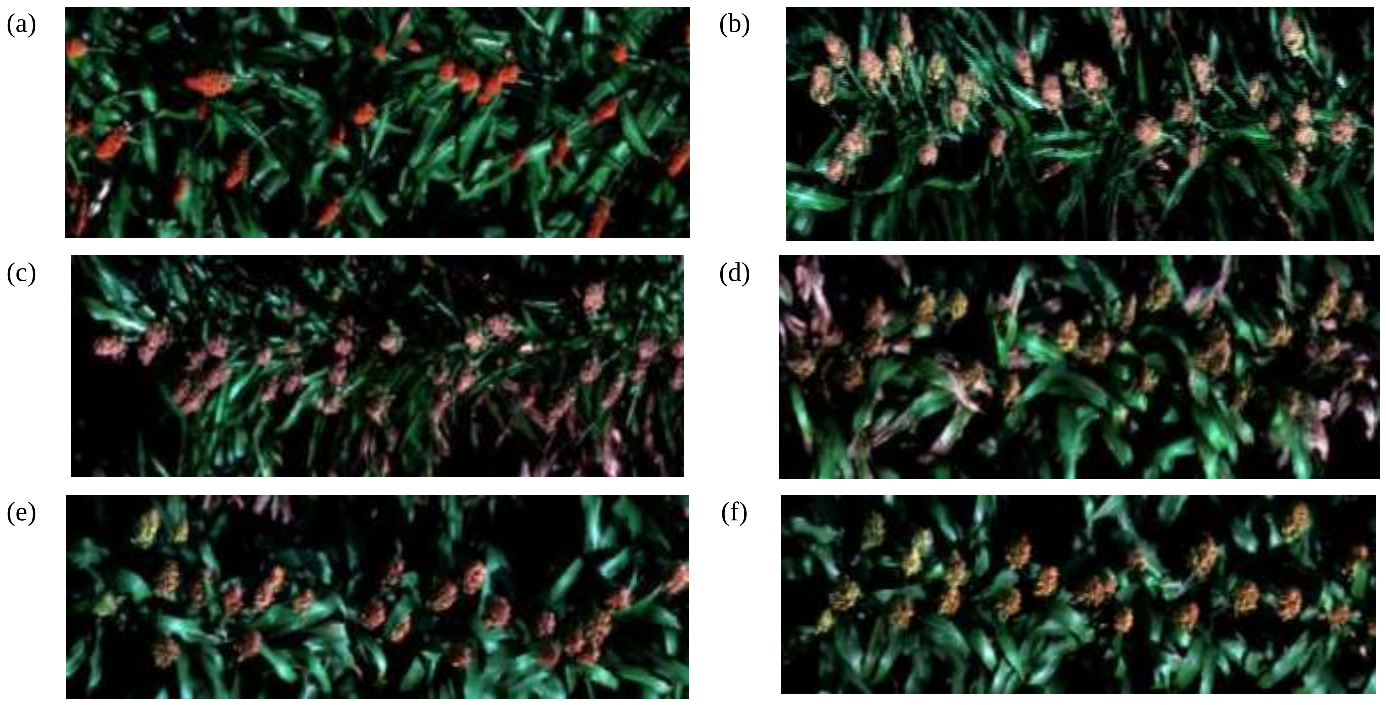


Figure 1.1. These figures depict panicles with different characteristics in shape and color to demonstrate the difficulty of detection

Acquisitions of hyperspectral image data are more limited than for RGB and multispectral data, and typically have a lower spatial resolution because of the reduced energy sensed by detectors within narrow spectral bands. In this thesis, extremely high spatial resolution data (~5 mm) were collected from a ground vehicle platform, which required special pre-processing to obtain orthophotos because of the perspective projection for the pushbroom sensor. Compared to other platforms such as Unmanned Aerial Vehicles (UAV), the lower height of the boom on the platform and resulting higher resolution Digital Surface Model (DSM) data made the orthorectification more difficult as discussed in Chapter 3.

Pre-processing of hyperspectral data prior to analysis is complex, both because the data sets are large, and there is significant spectral correlation between bands in some parts of the spectrum. Therefore, dimensionality reduction via feature selection or extraction is typically proposed. However, although dimension reduction methods reduce redundant features and computation time, useful information may be lost during this procedure. To make full use of the spatial and spectral signatures, dimensionality reduction with a focus on extracting informative features is explored relative to the hyperspectral data acquired for panicle detection.

Deep learning models have been demonstrated to be successful for classification of complex image datasets. However, large quantities of training data are required to train a deep learning model, and current annotation tools for object detection are problematic for hyperspectral data. Even if the data are labeled manually, few software packages can visualize hyperspectral data. Further, to build the architecture, deeper layers could potentially extract more informative features, but require more parameters, which might result in overfitting. For hyperspectral image data, a limited number of layers is not adequate because of its complexity. However, the number of parameters would be dramatically increased compared to RGB data (the most common data source for image-based deep learning architectures). As a result, it is important to focus on the tradeoff between the number of parameters and the depth of the network.

1.3 Thesis Overview

Chapter 2 includes a review of relevant literature, both in the field of object detection applied to RGB images and hyperspectral image classification, along with a discussion of different approaches which were utilized for datasets with different spatial and spectral resolutions. The proposed data pre-possessing and post-processing methods are described in Chapter 3. The pre-processing procedure involving orthorectification of close range hyperspectral imagery was included as an additional contribution. Panicle detection, which is the main objective of the thesis, focused on implementation of the FCN-8s deep learning model within VGG-16. Results of the deep learning model are included in Chapter 4, together with a comparison of different FCN architectures that were investigated for the hyperspectral image data. Finally, conclusions and opportunities for future work are described in Chapter 5.

2. BACKGROUND AND RELATED WORK

In this chapter, approaches for detecting panicles and similar plant objects such as wheat ears from high resolution remotely sensed data are described. Both traditional and deep learning models developed using RGB and LiDAR remote sensing data in recent studies are discussed. Because few panicle/object detection studies have been based on hyperspectral image data, multiple approaches are considered. Object-based classification methods for hyperspectral image data that include proposed strategies noted in Chapter 1 are outlined, followed by a discussion of the weaknesses and strengths compared to their application to RGB image data. Recent research on hyperspectral image classification using deep learning models is summarized, and its potential relevance to detecting panicles is explored. Finally, a brief evaluation of different approaches is included to justify the architecture proposed in this thesis.

2.1 Panicle detection studies based on traditional models using remote sensing data

Multiple studies that have focused on panicle detection and panicle counting are based on RGB image data acquired in controlled facilities (Duan et al., 2015; Guo, Fukatsu, Ninomiya, 2015; Zhu et al., 2015). Two general categories of approaches have been used: feature selection with pixel-based classification and object-based binary classification. In the first, images which may or may not contain panicles, are selected. A feature selection method is then used to identify and choose relevant features such as shape, color, or texture from both types of images, and a classifier is applied to label pixels as panicle or non-panicle. For example, Duan et al. (2015) used an I_2 color plane to extract color features from multi-angle panicle images and applied hysteresis thresholding as the classifier. Guo et al. (2015) employed the Scale Invariant Feature Transform (SIFT) to extract features from panicle and non-panicle images, and then used a k-means classifier to generate visual words from the features. Flowering images were then detected by an SVM supervised classifier. Another more complex approach, which was developed by Zhu et al. (2016) to detect wheat ears, takes advantage of multiple kinds of features, including low level color features, as well as high level features from SIFT or the Fisher Vector (FV) from patches, followed by a linear SVM classifier. The disadvantages of these pixel-based approaches, which have mostly been applied to RGB images, are that they may not be able to extract the shape of each panicle (which can vary widely), and the spectral information in RGB images is limited. It should also be noted that these studies are all based on images acquired in controlled facilities where the number of datasets is limited, and the images are in perspective projection from multiple angles with shape deformation. As the remote sensing technology that is suitable for outdoor environments developed, the strategies gradually transitioned to object based binary classification.

Object based binary classification has been used to detect panicles in remote sensing RGB images which have been acquired over large areas. These approaches typically include two steps. Images are first converted to small objects, and a general classifier is then used to classify the objects. One method relies on SLIC (Xiong et al., 2017; Olsen et al., 2018). Superpixel regions are generated and then used to annotate panicles and segment images acquired by a UAV to create small objects. Olsen et al. (2018) used logistic regression to classify panicles based on the superpixels and constrained the number of panicles to be monotonically increasing over time. A deep learning model can also be used as a classifier in the second step. For example, Xiong et al. (2017) developed a Convolutional Neural Network (CNN) after transforming the image pixels to superpixels. Figure 2.1 shows the structure of the model. It contains 3 convolutional layers and 3 pooling layers, followed by one fully connected layer to predict the class of the central pixel. A drawback of this method is that it requires large quantities of labeled data which are time consuming and tedious to obtain. In order to deal with image data with a limited number of labels, Ghosal et al. (2019) proposed an active learning inspired weakly supervised deep learning method. It trained the model on randomly selected data first and used the trained model on unlabeled data for human verification. Olsen et al. (2018) also adopted a semi-automatic annotation tool based on the color features of each superpixel. Oh et al. (2019) implemented this tool by using a feedback mechanism based on the detection tool, providing a density map as the output.

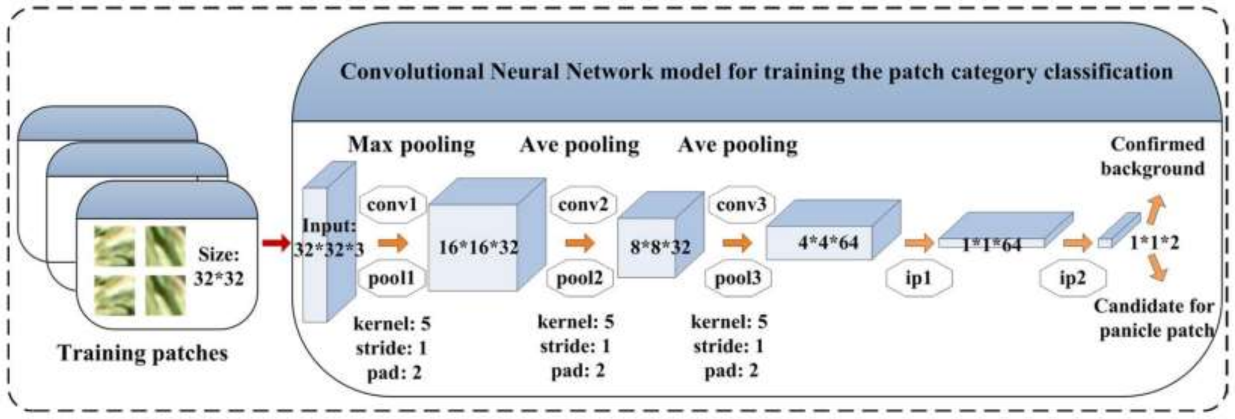

Convolutional Neural Network model for training the patch category classification
Training patches
Size: 32*32
Input: 32*32*3
Max pooling
Ave pooling
Ave pooling
conv1
pool1
16*16*32
conv2
pool2
8*8*32
conv3
pool3
4*4*64
ip1
1*1*64
ip2
1*1*2
kernel: 5 stride: 1 pad: 2
kernel: 5 stride: 1 pad: 2
kernel: 5 stride: 1 pad: 2
Confirmed background
Candidate for panicle patch

Figure 2.1. CNN model used by Xiong et al. (2017). The input of the model is the image data with three channels. The output of the model is the class of the central pixel. Reprinted from 'Panicle-SEG: a robust image segmentation method for rice panicles in the field based on deep learning and superpixel optimization,' Xiong et al., 2017, *Plant Methods*, *13*(1), 104.

LiDAR data have also been used in panicle detection via a segmentation approach applied to 3D point cloud data. As noted earlier, Malambo et al. (2019) proposed DBSCAN based on some general spectral and morphological characteristics of LiDAR for detecting sorghum panicles, including height, point density and color information by using - colorized LiDAR point cloud data to detect panicles. Similarly, Velumani et al. (2019) applied voxel-based segmentation and a mean shift segmentation to the 3D point cloud, followed by an Otsu's threshold selection method to classify the unlabeled segments to wheat ears. The Faster R-CNN deep learning approach was recently used to segment maize from the background at the individual plant level in a controlled facility (Jin et al., 2018).

2.2 Segmentation of Hyperspectral Images with Deep Learning models

To the best of our knowledge, few studies (Miao et al., 2020) have investigated panicle segmentation based on remote sensing multispectral and hyperspectral images. Miao et al. (2020) explored seven supervised classification approaches, including Multinomial Logistic Regression (MLR), SVM and Linear Discriminant Analysis (LDA) to segment the sorghum panicles grown in a greenhouse from hyperspectral images (acquired by a Headwall Photonics, Inc., Fitchburg, MA, hyperspectral camera), and manually labeled pixels from the images. As the development of remote sensing technology has advanced, collecting hyperspectral image data with extremely high spatial resolution from wheeled vehicles and UAVs is viable, so hyperspectral imagery can now be explored to detect panicles. Several relevant studies based on hyperspectral image data provide insight for panicle detection. Two-step object-based binary classification approaches have been used to classify objects in hyperspectral images (Alam et al., 2016; Zhao et al., 2017). Similar to Xiong et al. (2017) and Olsen et al. (2018), this two-step approach first generates superpixels from hyperspectral data using SLIC, then classifies the objects. Alam et al. (2016) developed a feature vector for each pixel containing both spectral and spatial features following the application of SLIC on the feature vector, and then used a convolutional neural network to classify the superpixel to each land cover class. Similarly, Zhao et al. (2017) reduced dimensionality via Principal Component Analysis (PCA). They then performed SLIC on the first three principal components to generate superpixels, followed by classification via SVM, coupled with an active learning strategy to limit the number of training samples and perform the classification.

Deep learning can be applied to both feature extraction and classification. As the convolutional layer can perform feature extraction, while the fully connected layer can serve as the classifier, an end-to-end model that merges the two steps for the hyperspectral data is possible. As a result, a deep learning model could potentially be used to classify hyperspectral data for panicle detection. The most common deep learning model in image analysis is CNN. In RGB images, the input of the CNN is usually $3 \times w \times h$, where 3 is the number of channels, and w and h are the spatial dimensions. In the example noted previously, after performing SLIC on the image data, Xiong et al. (2017) used a 2D CNN to learn the class of superpixels. For an end-to-end deep learning model on image data, Oh et al. (2019) investigated a tuned version of the counting CNN (CCNN) model on 4 input channels (RGB image and thermal time data), as shown in Figure 2.2. The model had 8 convolutional layers and 2 pooling layers in a modified architecture, which yielded higher accuracies. The output was a density detection map for detecting and counting panicles.

However, due to the high dimension of hyperspectral data, it is difficult to force hyperspectral data into a classical deep learning framework. In order to produce a robust deep learning architecture and not waste the unique information from the hyperspectral image, a different convolutional architecture is required.

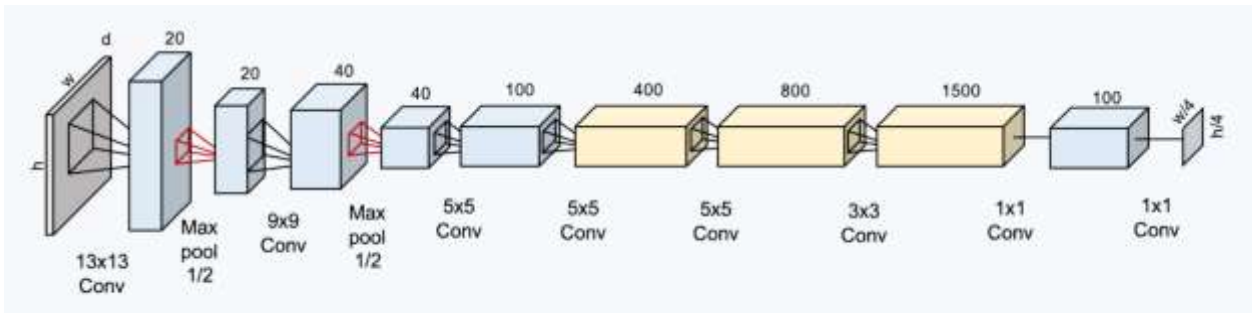


Figure 2.2. Counting CNN (CCNN) used in Oh et al. (2019) The output channel and filter size were modified to refine the model. Reprinted from “Counting and Detecting Sorghums”, by Oh et al., 2019.

2.2.1 Classification of Hyperspectral Data in Deep Learning Models

When solar radiation interacts with objects, energy is absorbed, transmitted, and reflected. Reflectance is related to the chemistry-based characteristics of a target and if unique, may be analyzed using pattern recognition methods for classification. Based on this knowledge, each pixel in a hypercube can be classified by its spectral signature, especially when the spatial resolution is low, as texture or shape related features are difficult to extract from the image. Thus, the hypercube can be considered as a set of 1D spectra, and the associated input to deep learning models is a 1D tensor of each pixel. For example, Hu et al. (2015) used CNN to classify a hyperspectral image directly in the spectral domain, and it outperformed some traditional models such as SVM. However, using only the spectral features ignores spatial features in higher spatial resolution data, which can be important for discrimination. Moreover, Audebert et al. (2019) found that 1D spectra do not fully utilize the data and are easily affected by noise.

2.2.2 Unsupervised Feature Extraction and Spectral-Spatial Classification in Deep Learning Models

Since deep learning frameworks developed for RGB data cannot be used directly on hyperspectral image data because of its high dimension, an alternative approach focuses on reducing the spectral dimension. In particular, the most common idea is to use an unsupervised dimension reduction approach such as PCA or an autoencoder, and then treat the output of the first step as the input image followed by the CNN based classifier.

The most direct way the reduce dimensionality of hyperspectral data and exploit relevant portions of the spectrum is to compute indices, such as the Normalized Difference Vegetation Index (NDVI) as an alternative to the original spectral bands. For example, Audebert et al. (2016) used the NDVI as well as the DSM data and Normalized Digital Surface Model (NDSM) from LiDAR data to create three channel image data for urban classification. Then they proposed a deep learning model (SegNet) to perform the semantic segmentation on the International Society for Photogrammetry and Remote Sensing (ISPRS) Vaihingen 2D Semantic Labelling dataset (Rottensteiner et al., 2012). The selection of appropriate band indices requires prior knowledge of the object feature. Principal component analysis, which projects high-dimensional data into a lower-dimensional embedding space while retaining the variance in the original data, is the most common unsupervised feature extraction approach. For example, Jiao et al. (2017) used PCA on the spectral dimension for the whole hyperspectral image, and the first three principal components were reserved for classification in the FCN-8s model. Similarly, Makantasis et al. (2015) utilized PCA to project high dimensional hyperspectral image data to three channel RGB-like image data, followed by a standard 2D CNN for the classification step. However, PCA is a linear projection and cannot exploit the nonlinear properties in hyperspectral data, and is not related to discrimination per se. Khodr and Younes (2011) tested several linear and nonlinear dimension reduction methods on hyperspectral data and found that nonlinear techniques like Sammon Mapping (Sammon, 1969) and Isomap (Tenenbaum, 2000) performed better than PCA based on eight quality criteria including entropy, variance, first and second spectral derivatives.

Autoencoders, which are deep learning based feature extraction techniques, seek to encode the raw data to achieve high fidelity reconstruction of the original data signature and allow for dimension reduction with minimal information loss and noise removal (Audebert et al., 2019). Xing et al. (2016) utilized the Stacked Denoise Autoencoder (SDAE) method, which is very robust to noise, to reduce the dimension of the hyperspectral data, followed by a logistic regression approach to complete the classification. In another study of autoencoders, Ma et al. (2016) proposed the spatial updated deep autoencoder (SDAE) to extract spectral-spatial information and tested the output features by comparing results achieved with different kinds of classifiers. However, the purpose of dimension reduction in hyperspectral data is to mitigate the impact of noisy bands and highly correlated bands, not simply to force the high-dimensional data to have the same number of inputs to classifiers as RGB data with three input channels. It should also be noted that unsupervised dimensionality reduction can sometimes lead to worse performance than using the original data because it might remove information that is useful discriminant information for classification (Du, 2003).

2.2.3 Spectral-spatial classification in Deep Learning Models

As noted previously, the CNN model can perform as both a feature extraction and a classification method. This is because CNN can be represented in two parts, the convolutional layer and the fully connected layer. The convolutional layer uses the kernel filter to extract features from the original data, while the fully connected layer classifies the feature acquired from the convolutional layer. As a result, using an end-to-end CNN model directly on the hypercube is possible. However, it is problematic to perform kernel calculations on the whole hypercube for hyperspectral data because of the large memory requirements. To deal with this challenge, dimension reduction can be implemented in the first two to three convolutional layers. Alternatively, the raw hypercube can be divided into small patches, which could be spatially adjacent regions of each pixel. One approach leverages a multi-scale filter bank such as the architecture used by Lee and Kwon (2017), which includes a convolutional layer with three different kernels, one to extract spectral features while the other two exploit spatial-spectral correlated signatures to create a joint spatial-spectral feature map as the input of the Fully Convolutional Network (FCN), as can be seen from Figure 2.3. It is important to note that the FCN is able to generate predictions for all pixels from the input, which makes the model more efficient. Another approach, which is computationally advantageous, operates on small spatial patches instead of the whole image. The input size is generally $k \times k \times b$, where k is the number of neighborhood pixels, and b is the number of bands. The output is the label of the central pixel of the input matrix. The architecture depends on the dimension of the kernel. For example, Slavkovikj et al. (2015) flattened the spatial dimension to produce a two-dimensional input matrix of size $(k \times k) \times b$ and then used a 2D CNN for classification. One promising technique directly uses 3D CNN on the input patches, operating simultaneously on the three dimensions. For example, Hamida et al. (2018) produced the new 3D deep learning approach illustrated in Figure 2.4. The model extracts the spectral features with an $1 \times 1 \times b$ kernel and extracts the spatial features with a $k \times k \times 1$ kernel. Instead of producing 2D feature maps, the 3D CNN produces 3D feature cubes that extract features like collocated spectral signatures and differences of absorption between bands from the convolutional layers. This is more straightforward for classification and requires fewer parameters and layers. Chen et al. (2016) demonstrated that 3D convolution improves the quality of the classification result. Yang et al. (2018) also concluded that it was superior to 2D CNN for the classification of data in the same small patches.

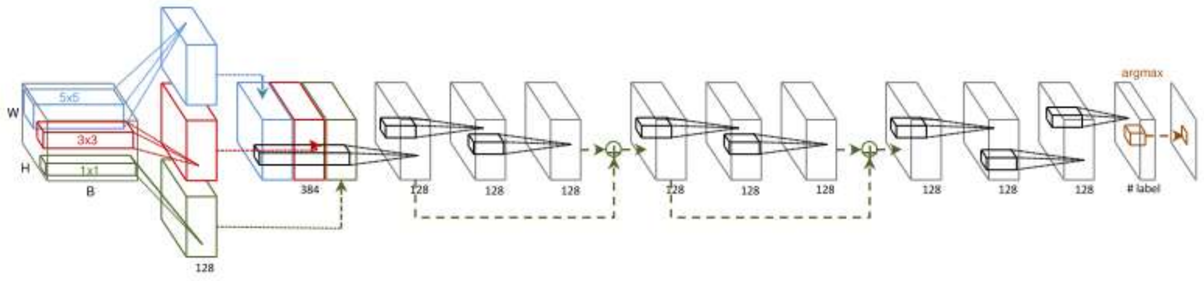


Figure 2.3. Contextual CNN proposed by Lee et al. (2017) The first layer contains three 3D convolutional layers to generate a feature bank. Reprinted from 'Going deeper with contextual CNN for hyperspectral image classification,' Lee et al., 2017, *IEEE Transactions on Image Processing*, *26*(10), 4843-4855.

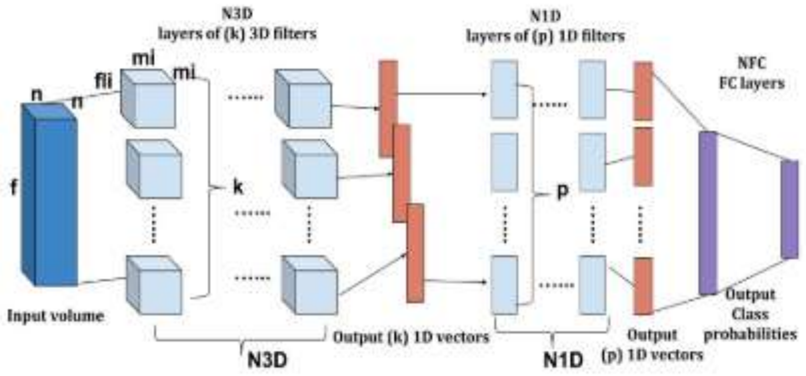


Figure 2.4. 3D CNN used by Hamida et al. (2018). The input is a 3D cube. After several 3D convolutional layers, the output becomes a 1D vector serving as a fully connected output. Reprinted from '3-D deep learning approach for remote sensing image classification,' Hamida et al., 2018, *IEEE Transactions on Geoscience and Remote Sensing*, *56*(8), 4420-4434.

2.2.4 Hyperspectral image classification using other Deep Learning models

Other deep learning models, such as Recurrent Neural Networks (RNN), can also be used for hyperspectral image classification. RNN uses "history" to retrieve past information and is often used to process time series data. In this case, the 3D hypercube can be treated as a set of sequential 1D data. For example, Mou et al. (2017) used RNNs to classify hyperspectral data based on the assumption that both long-range and short-range dependencies could be modeled in the spectral domain. It performs better than 1D CNN using only spectral data because it considers both local and global values in the spectrum.

2.2.5 Selection of the Deep Learning model architecture

Multiple well-validated deep learning architectures have been designed for hyperspectral image classification. To select the optimal model, different factors need to be considered. For example, as the quantity of training data increases, a deeper model can be chosen. Data quality, such as noise and spatial resolution, also affects deep learning model results. As noted earlier, Audebert et al. (2019) tested CNN models with different kinds of convolution on patches from multiple datasets and reported that spatial resolution impacts model results. When the spatial resolution is very coarse, each pixel contains a mixture of ground materials, and 1D convolution performs better. With a higher spatial resolution, 3D convolution on small patches from image data with size $3\times 3\times k$ begins to achieve higher performance. However, if the spatial resolution is very high, the qualities of both 1D and 3D convolutions are worse because spatial features of the center pixel in small patches are difficult to be represented. Stretching the 3D patches to a 2D matrix also encounters the same problem. A larger input size is required in this case, which is computationally intensive for 3D convolution. To reduce computation, either reducing the dimension (Jiao et al., 2017) or modifying the model structure (Lee et al., 2017) on the larger input size (e.g., $100\times 100\times k$) can be used for hyperspectral data with high spatial resolution. Most labels for the deep learning model are pixel level classes. FCN is efficient in this respect as it predicts several pixels simultaneously, and the transferred 3D FCNs extract meaningful features (Audebert et al., 2019). Oh et al. (2019) also imply that Fully Convolutional Regression Networks (FCRN) have the potential to solve the down sampled problem that occurred in their research. As a result, an FCN using a 2D filter in a convolutional layer is proposed in this thesis. Since a traditional FCN network utilized for hyperspectral data requires some form of prior procedure, both dimension reduction and structure modification are used to make it an end-to-end model. Results from this architecture are presented in Chapter 4.

3. METHODOLOGY

In this chapter, details of the methodology used in this study to detect panicles from high resolution data acquired by a wheeled vehicle are introduced. Pre-processing approaches related to creating hyperspectral orthophotos from close-range images are also summarized. The second part of this chapter describes the basics of the FCN and the modification required for the analysis of hyperspectral image data. Specifically, the encoder used for FCN, VGG-16, and the modifications required to transform from the classifier to a prediction map are introduced.

3.1 Orthorectification of Hyperspectral Data acquired by the Purdue PhenoRover

Hyperspectral data for this project were collected by a Headwall Photonics, Fitchburg, MA Machine Vision pushbroom hyperspectral sensor, which operates over the range of ~400-1000 nanometers, on a wheeled vehicle with a custom designed sensor boom. The pushbroom scanner, as shown in Figure 3.1, captures a 1-D array that is perpendicular to its driving direction. The raw data are collected by the sensor over the field as the combination of all the strips, each of which is a perspective projection with a non-uniform scale and lack of coordinate information. Before selecting labels and classifying the images, it was necessary to generate orthophotos from the reflectance data, transforming the perspective projection (Figure 3.2 a) to the orthogonal projection (Figure 3.2 b).

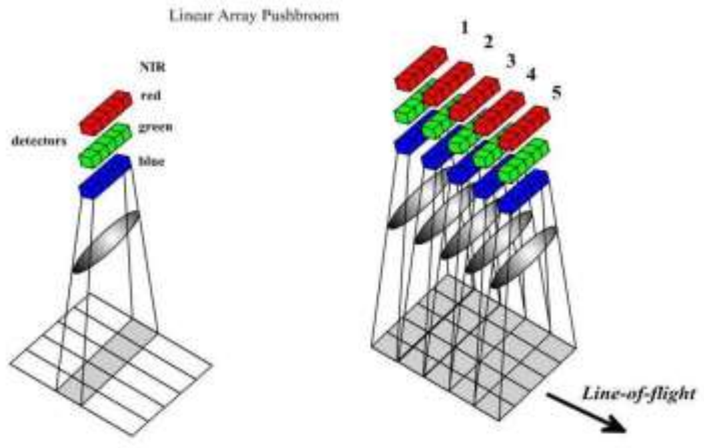


Figure 3.1. Representation of the perspective and image acquisition geometry of Linear Array Pushbroom sensors

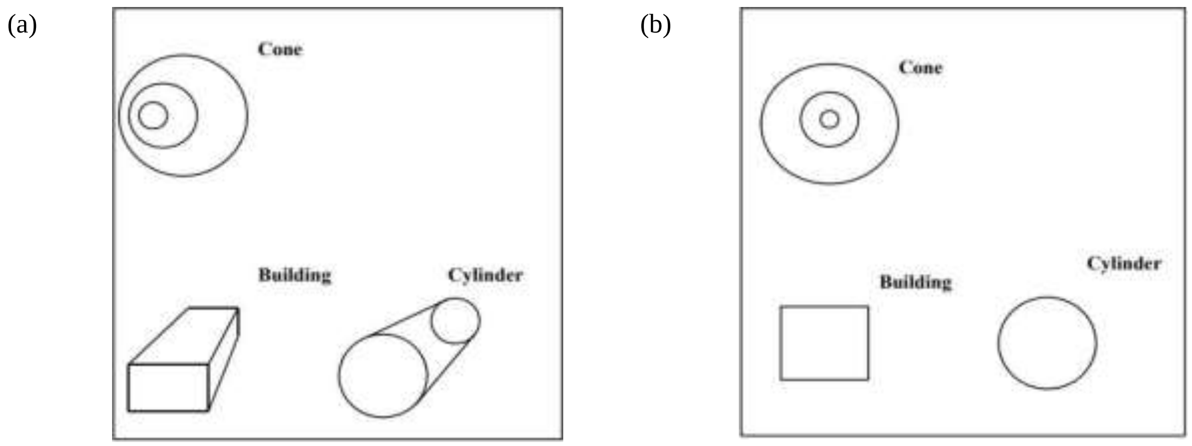


Figure 3.2 Representation of image projections from different angles:
(a) Perspective Projection, (b) Orthogonal Projection

3.1.1 Point positioning function

Each pixel from the raw image has a mathematical collinear relationship with the object point from the ground coordinate system and the perspective center, which can be described by the point positioning function in Eqn 3.1 (Habib et al., 2018). The coordinate system of the pushbroom sensor, which is shown in Figure 3.3, contains three coordinate systems: the mapping frame, the GNSS/INS body frame and the pushbroom scanner coordinate system.

$$r_I^m = r_b^m(t) + R_b^m(t) r_c^b + \lambda_i R_b^m(t) R_c^b r_i^c \tag{3.1}$$

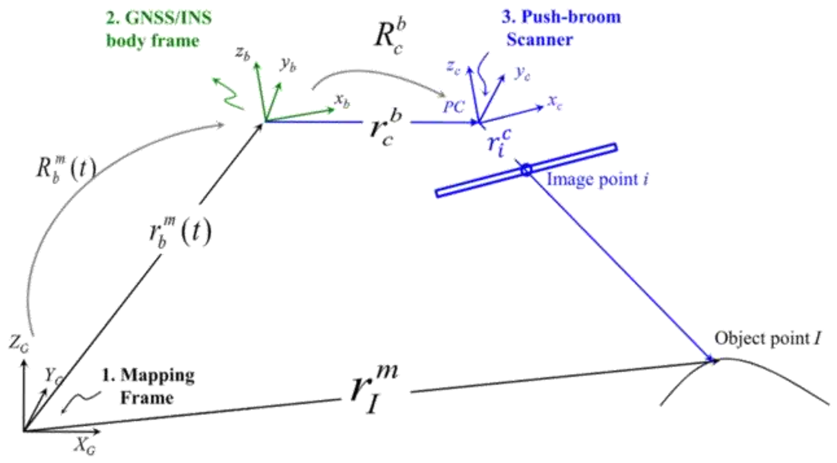


Figure 3.3. Schematic diagram of the collinearity equations and definition of the coordinate systems for a pushbroom scanner. Reprinted from ‘Boresight calibration of GNSS/INS-assisted push-broom hyperspectral scanners on UAV platforms,’ by Habib et al., 2018. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *11*(5), 1734-1749.

It is important to note that in Eqn 3.1, r_p^q represents the spatial offset of point p relative to the coordinate system associated with point q, while R_p^q represents the rotation matrix that transforms coordinate system p to coordinate system q. The first coordinate system is the ground coordinate system, which follows the North American Datum of 1983 (NAD 83) geodetic reference system in this thesis. The variable r_I^m refers to the object ground coordinate (X_I, Y_I, Z_I). The second coordinate system is the IMU body frame (b) (the coordinate of the GNSS/INS system (X_b, Y_b, Z_b) in the IMU body frame is usually the origin (0,0,0)). The variables r_b^m and R_b^m represent the position and orientation information of the IMU body frame relative to the mapping frame. At the exposure time t, $r_b^m(t)$ is the ground coordinate of the origin of the IMU coordinate system. The last coordinate system is the pushbroom scanner frame; the position of the sensor is also at the origin of its coordinate. The ground coordinate of the perspective point is computed as $r_b^m(t) + R_b^m(t)r_c^b$ in Eqn 3.1, where r_c^b is the lever arm relating the pushbroom sensor to the IMU body frame. Also, in this system, the relationship between the image and the perspective center is shown in Figure 3.3, and r_i^c (x_c, y_c, z_c) is the vector that denotes the relationship in Eqn 3.1. In the pushbroom scanning system, the scan line is usually set vertically under the perspective center. In this case, y_c is a constant zero value, while z_c is the focal length, the distance between the perspective center to the scan line. The value of λ_i refers to the non-uniform scale factor. For each point *i* in the raw image, the corresponding object coordinate of *I* in the mapping coordinate system can be calculated by the point positioning function, Eqn 3.1. Conversely, given the object coordinate of a point *I* in the mapping coordinate system, the pixel coordinate *i* of the raw image can also be calculated by the inverse function.

3.1.2 Boresight calibration

In order to generate an accurate orthophoto, the position and orientation information need to be obtained via geometric calibration using targets with known positions. With the help of integrated Global Navigation Satellite Systems and Inertial Navigation Systems (GNSS/INS), direct georeferencing is possible. This means that the position and orientation of each pixel in an image can be directly determined instead of using a traditional bundle adjustment with a large number of ground control points (GCPs) (Habib et al., 2011; Kersting, Habib, & Bang, 2011). The lever arm and boresight matrix relating the camera and IMU mapping frame must be measured by a system calibration procedure. Since the lever arm can be measured manually, the strategy is to estimate the boresight angles, R_b^c from Eqn 3.4, which is derived from Eqn 3.2. The idea is to enforce the perspective center, the image object and the corresponding object point to be collinear. In this thesis, the method proposed by Habib et al. (2018) is used to estimate the boresight angles. It takes advantage of the reformulated point positioning function Eqn 3.2 and decomposes the boresight matrix to the product of two rotation matrices, as shown in Eqn 3.3, in order to avoid gimbal lock. In this case, the $R_b^{c'}$ is a nominal matrix, while $R_{c'}^c$ is a rotation matrix which is almost parallel to $R_b^{c'}$. The initial value of $R_{c'}^c$ is set to zero which is updated by an iterative Least Squares Adjustment (LSA).

$$r_i^c = \frac{1}{\lambda_i} R_b^c R_m^b(t)(r_I^m - r_b^m(t)) - r_c^b) \qquad 3.2$$

$$r_i^c = \frac{1}{\lambda_i} R_{c'}^c R_b^{c'} R_m^b(t)(r_I^m - r_b^m(t)) - r_c^b) \qquad 3.3$$

3.1.3 Pushbroom image orthorectification via an indirect method

As noted previously, each pixel i in the raw image has a corresponding ground coordinate in the mapping frame that is determined by using a point positioning function. Either a direct or an indirect method can be used to generate an orthophoto from the raw image. As a direct method, forward projection directly finds the corresponding object coordinate, while backward projection (an indirect method) finds the point from the generated ortho-map to the original image data. Because the forward projection needs to project regular points to a set of irregular points, back projection is often the better choice for orthorectification (Mikhail, Bethel, and McGlone, 2001).

In this thesis, the back projection method is used for the hyperspectral imagery acquired by the Machine Vision (MV) pushbroom sensor. The back projection function in Eqns 3.8 and 3.9 can be obtained Eqns 3.4 to 3.7 based on the point positioning function.

$$r_I^m - r_b^m(t) = R_b^m(t)(r_c^b + \lambda_i R_c^b r_i^c) \qquad 3.4$$

$$R_m^b(t)(r_I^m - r_b^m(t)) - r_c^b = \lambda_i R_c^b r_i^c \qquad 3.5$$

$$\lambda_i r_i^c = R_b^c R_m^b(t)(r_I^m - r_b^m(t)) - r_c^b) \qquad 3.6$$

$$\lambda_i r_i^c = \begin{matrix} \lambda_i x_i \\ \lambda_i y_i \\ \lambda_i c \end{matrix} = \begin{matrix} N_x \\ N_y \\ D \end{matrix} \tag{3.7}$$

$$x_i = c \times \frac{N_x}{D} \tag{3.8}$$

$$y_i = c \times \frac{N_y}{D} \tag{3.9}$$

The steps required to generate the orthophotos are:

1. Define a coarse orthophoto trajectory by evaluating the point positioning function on the first and last points of each scan line.
2. Find the coordinates of the four corners from the minimum and maximum trajectory points. Generate an orthophoto map using these values.
3. As shown in Figure 3.4, for each pixel in the orthophoto map, the ground coordinate can be calculated by using the corner coordinate and the resolution (5mm in our data), so the height value can also be derived from the DSM data. Then the back projection function assigns the pixel coordinate of the raw image. Because of the characteristics of the pushbroom scanner, the calculated *y* pixel coordinate should be close to zero. An iterative process is therefore used to find the ideal pixel coordinate.

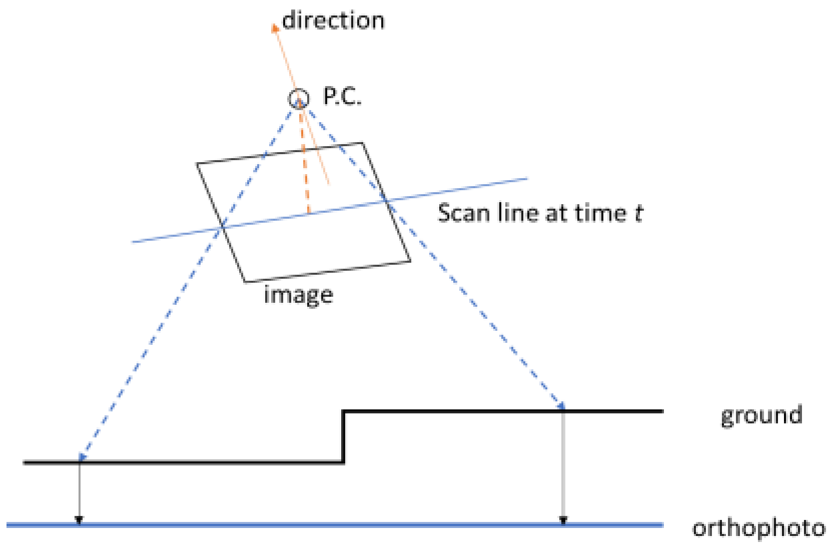


Figure 3.4. Schematic drawing that illustrates back projection using point positioning function

However, since our data are collected using a terrestrial vehicle platform, the distance between the sensor and the ground is much smaller (1-3m) than for a UAV (~ 40m for our studies), and the Ground Sampling Distance (GSD) is much smaller than for the sensor on the UAV platform. This results in a detailed, highly variable DSM, particularly over targets such as tall plants. The resulting higher resolution orthophoto from our platform also has a lower "overall" height than the corresponding DSM obtained from a UAV platform over the same area. In order to obtain a higher quality orthophoto that could be used for panicle detection, a locally filtered DSM is proposed to smooth the orthorectification result. This also improves the physical characteristics of the panicles observed in the image. However, additional future efforts should be focused on the generation of high fidelity orthophotos to obtain more precise positions. Further exploration of filters such as Savitzky-Golay (Savitzky &Golay, 1964) and other adaptive filtering approaches should be considered further for smoothing the DSM.

3.2 Classification of Hyperspectral images

The classification of panicles is treated as a binary detection process, determining an element as 'whether or not' according to a certain quantitative property. The group with that property is a positive class (e.g. panicles), while the other is a negative class (e.g., soil and leaves). The two groups are not usually symmetric.

3.2.1 Model selection: Fully Convolutional Network

A CNN is used in this thesis as the classifier to perform binary classification on the hyperspectral image data. Recently, CNN has been demonstrated to be superior for supervised classification of many complex data sets, compared to other traditional classifiers like SVM. Eqn 3.10 and Eqn 3.11 show the computational steps of CNN. It uses the convolution operations and kernels that can be learned from a backpropagation procedure during training.

$$f(x) = \sum_{i=1}^{m} w_i x_i + b \tag{3.10}$$

$$y = sign[f(x)] \tag{3.11}$$

As noted in Chapter 2, the spatial resolution and spectral resolution of the hyperspectral image data impact the performance of the CNN. For hyperspectral images with narrow spectral bands, a classifier based on 1D spectra can extract the features from correlated spectra, but ignores all spatial information. 3D convolution learns 3D spatial-spectral features from small image patches with fewer parameters. High spatial resolution data provide spatial information, although it is minimal when the image data are subsetted to small patches. To utilize high spatial resolution datasets, 2D convolution from large patches seems to be promising for this application. As an end-to-end, pixel-to-pixel model, the FCN generally performs well in image semantic segmentation and classification.

An FCN can be derived from contemporary classification networks, such as AlexNet, VGGNet, and GoogLeNet, which are all commonly used in image analysis. Among these, AlexNet has five convolutional layers and three fully connected layers and utilizes the max pooling function and ReLU's activation function to enhance the results and reduce the computation time. VGG-16 utilizes very small (3×3) convolutional layers so that the model is deeper than AlexNet. Compared to VGG-16, GoogLeNet uses filters at different scales to extract features. An FCN naturally operates on an input of any size and produces an output with the same resampled spatial dimension, and thus can be used in semantic segmentation. It includes two structures, the encoder and decoder steps, where the image is first down-sampled into feature maps, followed by the up-sampling procedure to a coarse prediction map. To find the optimal encoder, Long et al. (2015) conducted tests on AlexNet, VGG-16, and GoogLeNet and found that among these three networks, FCN-VGG-16 performed best with the 56.0 mean IU (region intersection over union metric) on PASCAL VOC 2011 validation sets. In this thesis, VGG-16 is used as the basic structure to build the fully convolutional architecture. It is a relatively deeper but simplifier architecture for a large-scale image compared to AlexNet and GoogLeNet.

VGG -16 contains thirteen convolutional layers, three fully connected layers, five pooling layers, and other nonlinear hidden layers, such as activation function and dropout layers, as shown in Figure 3.5. Like other CNN architectures, VGG-16 first generates a feature map, followed by a fully connected layer and a classifier layer to learn the label of the input. The feature map is developed by the combination of linear and nonlinear layers. The convolutional layer is the linear layer that extracts features through a kernel, which is the cross product of the input matrix and the kernel as shown in Eqn 3.10. Other parameters in the convolutional layer include stride and padding to control the size of the output matrix. In VGG-16, the stride and padding are both set to 1, so that the output size is the same as the input size. The pooling layer is a nonlinear layer that helps translate the output from a previous layer to a smaller size while maintaining valuable information. The relative location of a feature is assumed to be more important than its exact location.

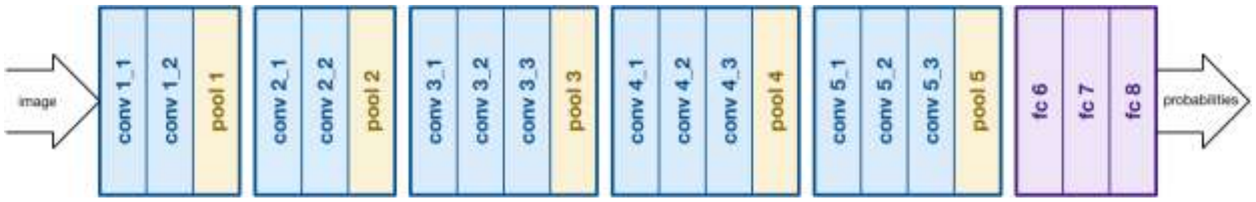


Figure 3.5. Network structure of VGG16 which contains 13 convolutional layers, 5 pooling layers and 3 fully connected layers. Reprinted from *CNN Architecture: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more…*, by Siddharth Das, 2017, https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Max pooling, which selects the maximum number as the representative value within a rectangular neighborhood, is selected in the pooling layers. As the neural network becomes deeper, some problems may occur, including vanishing gradients and overfitting. To avoid the vanishing gradient problem, an activation function is used to represent the data differently, potentially providing useful information. A nonlinear transformation is performed on the input, making it capable of learning and performing more complex tasks. VGG-16 uses the Rectified Linear Units (ReLu’s) function, shown in Eqn 3.12, to activate the linear convolution map result by setting negative values to zero. Dropout is also used to prevent overfitting in deep learning architectures. It randomly drops out some weights of hidden nodes in the network during training of the model, which can be considered as deleting a part of the network structure, but retains their weights for the next training epoch. A SoftMax function, which is the extension of logistic regression from a dichotomous classifier to a multi-class problem, is introduced to construct the loss function for the supervised training in VGG-16.

To convert classification networks into fully convolutional networks, three steps are required:

1. Transform the fully connected layers into convolutional layers. There are three fully connected layers in VGG-16 as noted before, which are fc6, fc7 and fc8, shown in Figure 3.6. The three fully connected layers output a prediction vector with the number of classes, which enables the feature map to output a prediction map, referred to as the heatmap. When using the VGG-16 classifier as the encoder of the FCN, the fully connected layer fc6, fc7 is reconstructed to conv6 and conv7 as shown in Figure 3.6. After that, instead of converting fc8, a score map for all the pixels is produced.

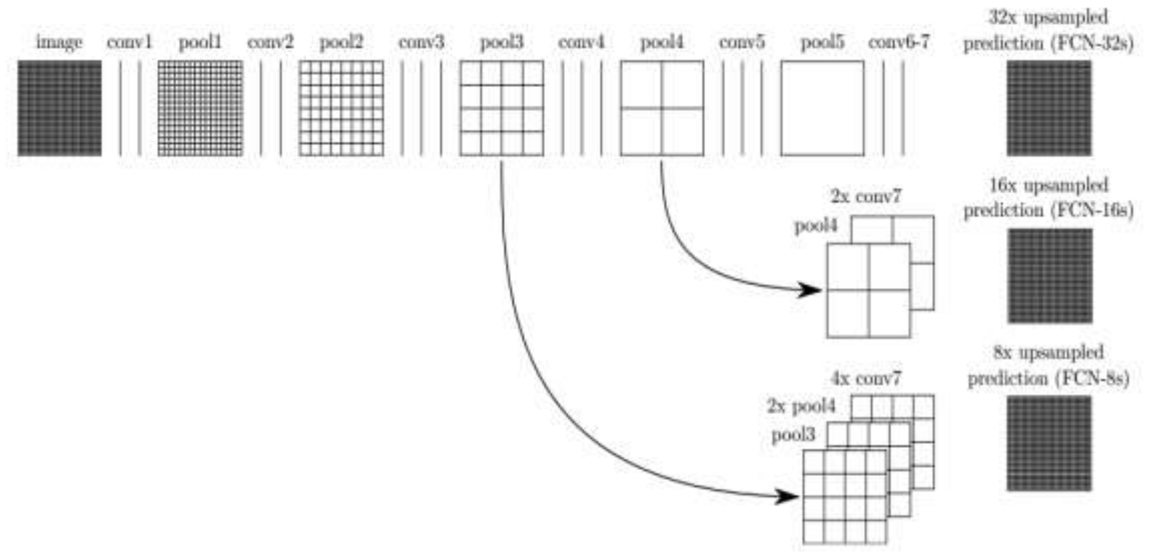


image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32x upsampled prediction (FCN-32s)
2x conv7
pool4
16x upsampled prediction (FCN-16s)
4x conv7
2x pool4
pool3
8x upsampled prediction (FCN-8s)

Figure 3.6. Expanding fully connected layers in VGG16 to FCN by adding prediction layers and deconvolutional layers. Reprinted from ‘Fully convolutional networks for semantic segmentation,’ Long, Shelhamer, Darelll, 2015, In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3431-3440).

2. Add a deconvolutional layer to the prediction layer. The deconvolutional layer is an upsampling process which makes the output larger than the input(an up convolution or transposed convolution procedure). The objective is to generate the input signal from the feature map (Zeiler et al., 2010), which is a top-down decoder process compared to convolution. By using learned filters, the deconvolutional layer can extract mid-level concepts such as edges and lines. In this case, it is used to upsample the prediction map so that the size of the output matrix is the same as the input matrix. After step 2, it generates an upsampled prediction map. The most straightforward way to produce the segmentation map is to directly add the deconvolutional layer with stride 32 after the score map so that

it is the same size as the input matrix. It is referred to as FCN-32s because of the number of strides.

3. Since the output of FCN-32 is quite coarse, it can be refined by combining the upsampled prediction layer with the score layer in a lower layer. For example, after pool-4 in Figure 3.6, a score map can be generated by adding a convolutional layer on pool-4 with the same number of output channels as classes. We refer to it as score_pool4 to distinguish it from the final score map. The upsampled prediction layer is obtained from a transposed convolutional layer with stride two on the score map. The segmentation map can be built by implementing the deconvolutional layer with stride 16 after merging the score_pool4 layer and the upsampled prediction layer. The same step can be performed on pool3 together with pool4 to calculate the combination twice before applying the deconvolutional layer. The two architectures are FCN-16s and FCN-8s respectively as shown in Figure 3.6. The Figure also shows that when the stride number is smaller, the model is deeper, and FCN-8s has the deepest layer among the three models, FCN-8s, FCN-16s and FCN-32s. It is concluded that by fusing prediction layers with lower layers, FCN-8s produces a finer feature map for image segmentation (Long, Shelhamer, & Darell, 2015). As a result, VGG16-FCN-8s with a pretrained VGG-16 is used in this thesis. Stochastic gradient descent (SGD) with momentum is used as the optimization method in the learning procedure. It calculates the gradient of the loss function to determine the direction and size on each training sample and uses the momentum approach to enhance the relevant direction and decrease the oscillations.

$$f(x) = \begin{cases} 0 & for\ x \leq 0 \\ x & for\ x > 0 \end{cases} \qquad 3.12$$

3.2.2 Model Modification

Because of the characteristics of hyperspectral imagery, FCN-8s cannot be applied directly, as it is structured to operate on a three-dimensional hypercube. To make it applicable to hyperspectral data, some modifications of the architecture are required. The weight of the modified layer was set using Glorot initialization (Glorot & Bengio, 2010) while other layers used the weight from pretrained VGG-16. In this thesis, three approaches with different model modifications are proposed to the original deep learning model.

PCA was investigated first as an unsupervised dimension reduction approach, following Jiao et al. (2017) who implemented FCN-8s as noted in Chapter 2. By using 99% of the variance as a threshold, seven principal components were selected as the input data, so the input channel of the first layer in the deep learning model was modified to seven. Reducing the dimensionality of the data also helps reduce the computation, although as noted previously, principal components are not necessarily ideal features for discrimination, as the criterion is unrelated to classification.

To mitigate the effect of unsupervised linear dimension reduction, the second approach focused on changing the architecture of the deep learning model. The convolutional layer of the model can be seen as a feature selection procedure, while the prediction layer is related to classification. In this way, the most straightforward method to accommodate the hypercube is to change the first layer of the FCN. The first layer in a classical FCN is a convolutional layer with the kernel size of 3. The hyperspectral image data can be considered as an image with *b* input channels, where *b* refers to the number of bands. This is a simple but effective method for two reasons: 1) the output channel of the first layer is 64, which in some way reduces the number of dimensions, and 2) it maintains the model as an end-to-end, pixel-to-pixel process without any other pre-processing procedure which could potentially lose information in the original data.

The last approach explored the generation of a joint feature map called the multi-scale filter bank before applying the structure of VGG-16 to the data. It uses two convolutional filters with different sizes: $1 \times 1 \times b$, $3 \times 3 \times b$, where *b* is the number of bands. The first filter extracts the spectral features while the second filter seeks to incorporate spatial-spectral information. Zero padding is used to force the output to be the same size. The two feature maps are then combined, followed by the FCN. Like the previously described method, the model can be applied to the original high dimensional hyperspectral image data. It also reduces the dimension in a supervised way. By learning through backpropagation, the filter band fits filter parameters to learn the features from the spatial-spectral domain.

To distinguish the second and the third approaches, the first method is called 2D-FCN, while the second method is referred to as 3D-FCN. These two end-to-end approaches share the same disadvantage that of potentially being impacted by the data redundancy when the input data is too large as it is a 3D patch-wise classification. For the convolutional layer, the total number of parameters is $(n \times m \times l+1) \times k$, where n and m are the filter size, l denotes the number of input channels, and k represents the number of output channels. In the original FCN, the filter size of the first layer is 3×3, the number of input channels is 3 and the number of output channels is 64; thus, it needs to compute 1792 parameters. When the original raw data channels are used, the number increases to 78,796. At the same time, the number of parameters becomes 2588 with the joint feature map. When training on a small dataset, a more complex architecture would degrade the performance (Oh et al., 2019). Comparing the two approaches, they both require many more parameters than the original network, with the 3D-FCN having both a very large number of parameters and thereby requiring enormous quantities of associated training data, and the training rate is slower. However, the model of 3D-FCN is deeper, and the filter bank considers both spectral and spatial domains, so that a better result could be expected with larger datasets.

Results related both to the pre-processing procedure and to orthorectification are included in Chapter 4. In panicle detection, the three deep learning models discussed in Chapter 3 are all explored for the hyperspectral image data in Chapter 4.

4. EXPERIMENTAL RESULTS

This chapter includes results from both the improved orthorectification approach for close range pushbroom sensors and detection of panicles from the resulting hyperspectral images. Results of detection using three tuned versions of FCN-8s models which include dimension reduction are reviewed and evaluated.

4.1 Remote Sensing data

Data analyzed in this thesis were collected by a high clearance tractor with a boom referred to as the PhenoRover. The platform has three types of sensors: two FLIR GrassHopper3 GigE 9.1 MP color cameras, two Velodyne HDL-32E LiDAR units, and a Headwall MV camera to semi-automatically collect co-registered RGB, LiDAR data, and hyperspectral VNIR data. The Headwall MV VNIR sensor acquires data at wavelengths ranging from 400 - 1000 nm. Dual GNSS antennas as well as an Applanix POS LV 125 are also onboard for direct georeferencing. The RGB and LiDAR sensors are installed on both the left and right side of the center, and the single hyperspectral camera is mounted in the center of the boom as shown in Figure 4.1. The LiDAR sensors generate the point cloud data to characterize the physical structure of the canopy, as well as the DSM data for the RGB and hyperspectral data orthorectification. The hyperspectral data have 5mm spatial resolution at 2m above the canopy and 2.24 nm spectral resolution with 272 bands.

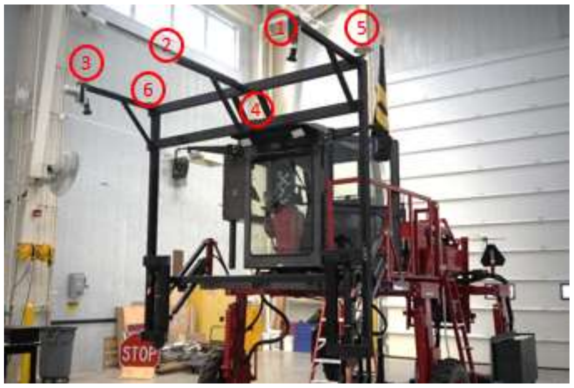


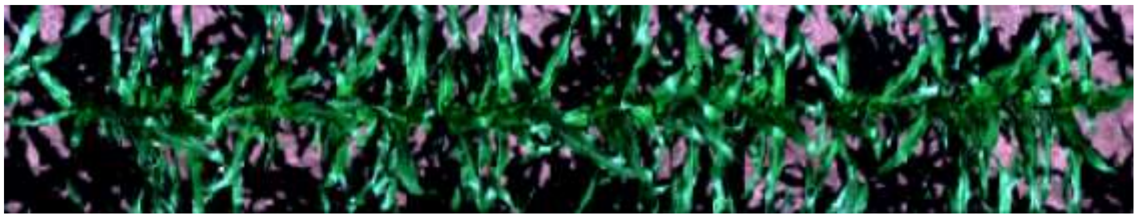
Figure 4.1. PhenoRover and the sensors on the platform including: (1) HDL-32E LiDAR, (2) Headwall MV, (3) HDL-32D LiDAR, (4) POS LV 125, (5) GNSS Antennas, (6) GNSS Antennas

A limited number of acquisitions were available for panicle detection in the 2018 growing season at the Purdue University Agronomy Center for Research and Education (ACRE). Three fields had sorghum experiments: 9D, which contained 12 sorghum geonotypes, hybrid calibration (Field 54 middle) with 18 genotypes, and the test cross diversity panel (Field 54 South) with 630 genotypes. For these fields, the flowering date was observed around July 1st for Field 54, but there is no specific date recorded for Field 9D. It can be assumed that the flowering date of Field 9D should have been about at least 2 weeks later, because the planting date of Field 9D (June 4th) was about 4 weeks later than Field 54 (May 8th). There were only four dates after July 1st: July 13th, July 17th, August 28th, and September 19th, for which PhenoRover data were acquired (see Table 4.1). Figure 4.2 represents four orthorectified images from the four corresponding dates by using the SpectralView (Headwall Photonics, Inc) data processing software provided with the sensor. From Figure 4.2 it can be seen that panicles on July 13th and July 17th are still green, and are difficult to discriminate from leaves. Changes were made to the platform in mid-July, and only the data on last two dates were fully processed. Also, panicles on August 28th and September 19th are easier to discriminate, although it is more desirable to use data in the early season to track flowering. Figure 4.3 shows multiple dates of RGB UAV example data acquired over the 2017 hybrid calibration panel (data used by Oh et al. (2019)). Panicles in these images were distinct from late July and were used to develop a model for flowering time for various varieties. In future years, hyperspectral data should be acquired by the PhenoRover during the flowering period at least twice per week during flowering, if possible.

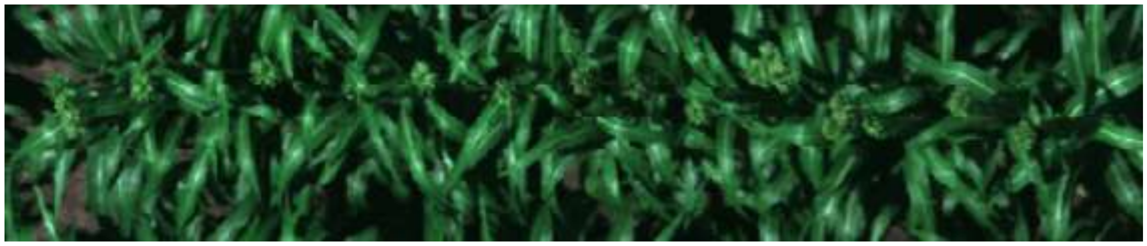
Table 4.1Post flowering Data Acquisitions by the PhenoRover

Data Collection Date	Field Surveyed
20180713	Field 9D
	Field 54 Middle
	Field 54 South
20180717	Field 54 South
20180828	Field 9D
20180919	Field 9D

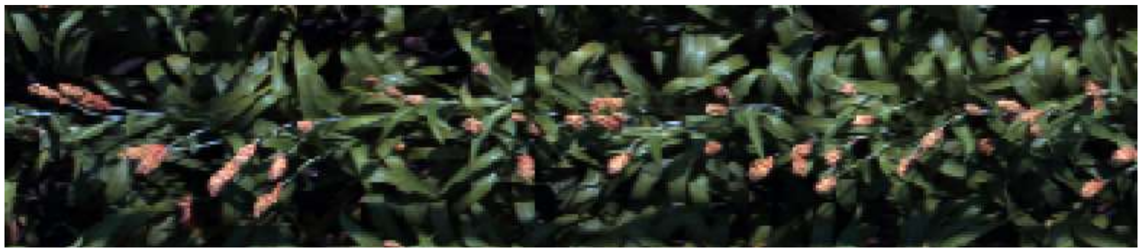
(a)



(b)



(c)



(d)

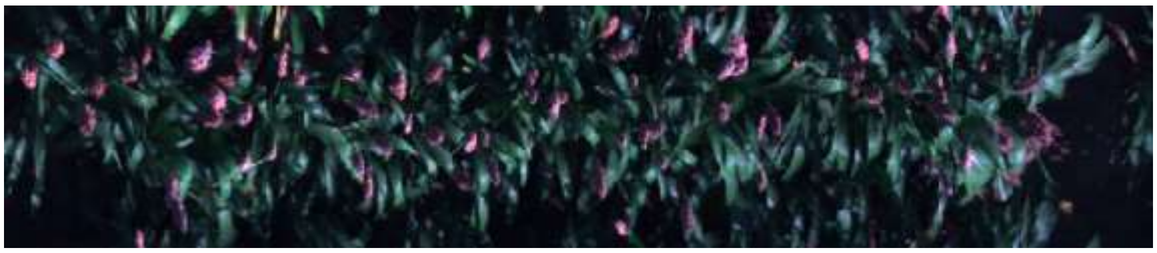


Figure 4.2. RGB bands of hyperspectral data on four dates in 2018: (a) 20180713, Field 9D, (b) 20180717, Field 54 South, no data available from Field 9D, (c) 20180828, Field 9D, (d) 20180919, Field 9D.

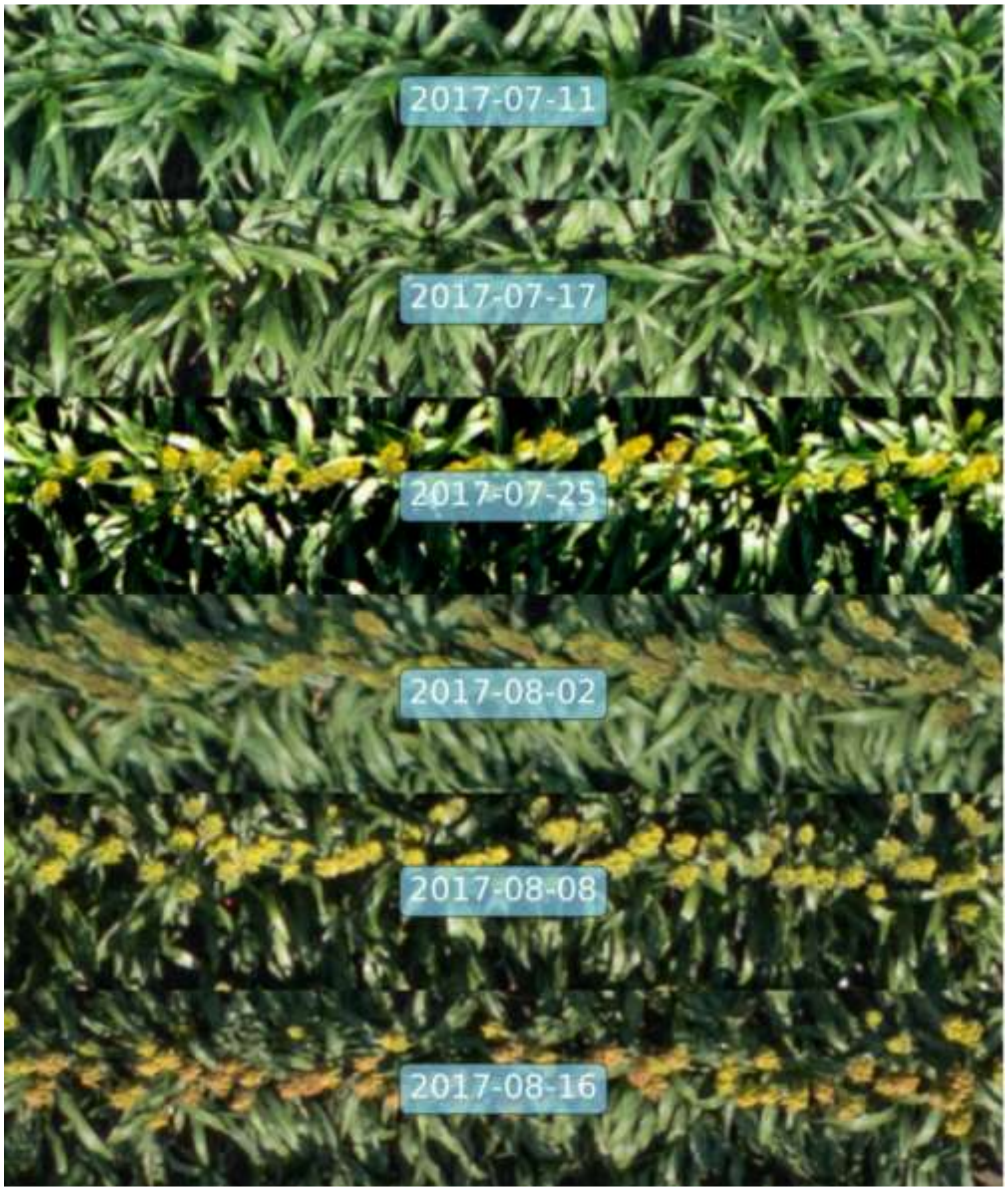


Figure 4.3. Example RGB images in 2017 hybrid calibration panel from 2017/07/11 to 2017/08/16. Reprinted from “Counting and Detecting Sorghums”, by Oh et al., 2019.

The data selected for this thesis were collected from Field 9D on 9/19/2018 because the data for the boresight calibration were also collected on the same date. Half of the field was planted in maize, and the remainder in sorghum, as shown in Figure 4.4. Data were first pre-processed from digital numbers to calibrated radiances by using the SpectralView software. The Empirical Line Method (ELM) was used to obtain reflectance for each pixel, and the spectral bands were binned by two to obtain 137 bands with resolution 4.4nm. In this thesis, the ground reference data of field targets for ELM-based reflectance were acquired by an SVC 1024i field spectrometer (see Figure 4.5).

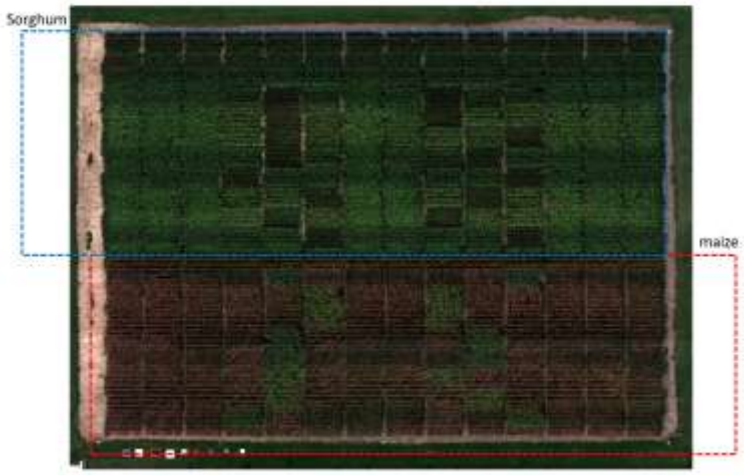


Figure 4.4. Orthophoto of test field, 9D. The blue area is sorghum and the red area is maize. The targets at the bottom of the image were used for acquiring the ground reference spectral data.

Figure 4.5. Spectroradiometer SVC HR-1024i (Spectra Vista Corporation) used for collecting the reflectance of ground targets.

4.2 Evaluation Metrics

Multiple metrics can be used to evaluate the results of classification. Long et al. (2015) used four metrics for semantic segmentation for a Fully Convolutional Network: pixel accuracy, mean accuracy, mean intersection over union (mean IU), and frequency weighted intersection over union (weighted IU). However, since this is a binary classification, the mean accuracy provides less information than the accuracy of panicle and non-panicle classification, respectively. Saito and Rehmsmeier (2015) found that precision and recall are more relevant than other metrics in binary classification. Precision and recall are related, but can be balanced by an F-measure. Accuracy, precision, recall and F-measure are reported in this research.

4.2.1 Confusion Matrix

The confusion matrix, also referred to as the error matrix, provides performance of a learning algorithm in terms of the correctly classified and misclassified pixels by the "users" and "producers". It can be used to better understand misclassified pixels, as well as identify correctly classified pixels. Table 4.2 represents the confusion matrix for the binary panicle classification problem. Rows of the confusion matrix denote the predicted class (users) while the column is associated with the actual class (producers). It contains four variables: True Positives (TP), False Positives (FP), False Negatives (FN) and True Negatives (TN). The positive instance refers to a panicle while a negative instance, also called the background instance is non-panicle as described in Chapter 3. True Positives (TP) represent the number of pixels that are correctly classified as panicles. False positives (FP) denote the background pixels which are classified as panicles, while false negatives (FN) are related to the pixels that were incorrectly classified as background. True Negatives (TN) represent the corresponding background pixels that are correctly classified as such.

Table 4.2. Parameters in the confusion matrix for binary classification

		Actual class (Producers)	
	Total Population	Panicle	Non-panicle
Predicted class (Users)	Panicle	True Positives	False Positives
	Non-panicle	False Negatives	True Negatives

4.2.2 Accuracy

Overall accuracy reports how many pixels in total are classified correctly, including panicle and non-panicle pixels. However, accuracy alone is not reliable for evaluation when the dataset is unbalanced. For example, in our case, the non-panicle area is much larger than the panicle area, so that even if the model classifies all the pixels as non-panicle, the accuracy can still have a high value. Therefore, precision and recall are necessary in this case.

4.2.3 Precision, Recall and F-measure

Precision (Eqn 4.1) is defined as the accuracy of the positive class, the panicle in this thesis. Recall (Eqn 4.2) is the accuracy of the positive class among the relevant classification outcomes and relates to the sensitivity of the classification. In the context of this study, precision is the accuracy of panicles that are correctly classified among all the pixels that are predicted as panicles, while recall is the fraction of properly predicted panicles among all the true panicle pixels (See Table 4.2).

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

While precision and recall explain the rate and sensitivity of the accuracy, they interact with each other, (i.e. a high precision may lead to a low recall.) The F-measure includes both as shown in Eq. 4.3.

$$F = 2 \cdot \frac{Precision \cdot recall}{Precision + recall} \tag{4.3}$$

4.3 Experimental Results

In this section, the results related to orthorectification of the near range data from the PhenoRover are presented, followed by the post-processing analysis for panicle detection. The results of three

FCN-8s tuned versions for classification of panicles were evaluated using accuracy, precision, recall, and the F-metric.

4.3.1 Geometric Correction

Figure 4.6 and Figure 4.7 show the raw image cube in Field 9D on September 19th without any geometric calibration. Figure 4.6 shows the subset from a raw image cube with color targets used for ELM. Figure 4.7 is a subset from a raw image cube which consists of half or each of two four row plots, separated by an alley. The shapes of the targets in Figure 4.6 and shape of panicles and leaves in Figure 4.7 are distorted, especially for objects near the edge of the image cube, and when there are changes in heights over a small area, such as between the two plots (see Figure 4.7).

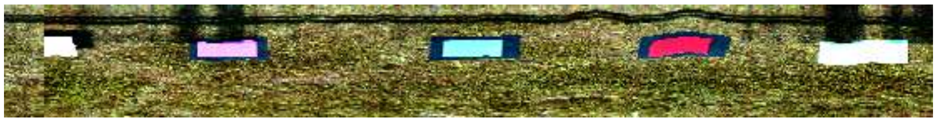


Figure 4.6. A subset of a raw image cube with targets.

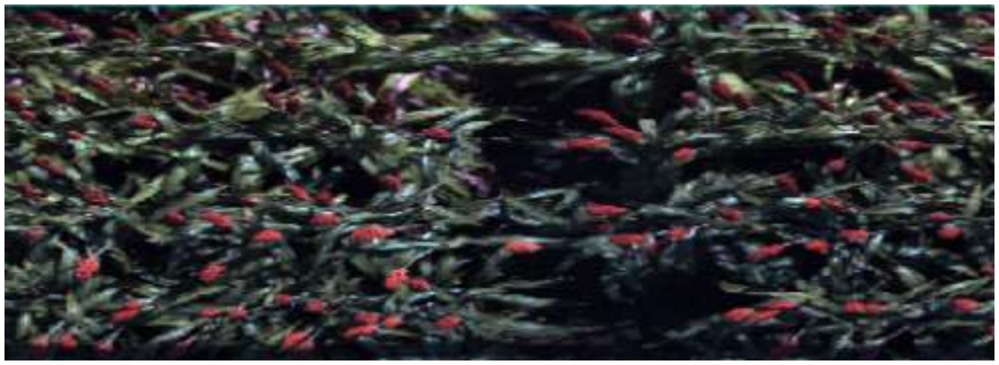


Figure 4.7. A subset from a raw image cube over two plots acquired from east to west, separated by an alley

Figure 4.8 shows two rows of a plot at nadir after orthorectification generated by back projection (Habib et al., 2018), using the LiDAR-based Digital Surface Model (DSM) data with 4cm resolution shown in Figure 4.9. Both the orthorectification result from the DSM data (Figure 4.8 (a)) and the DSM data (Figure 4.9) are unsmoothed, which results in very low quality, noisy images. To obtain higher quality orthophotos, the 95th percentile DSM data associated with each cube was investigated, and the result is shown in Figure 4.8 (b). From this image (Figure 4.8b), the panicles can be more easily recognized, and the approximate positioning of the rows in the whole field can be visualized better.

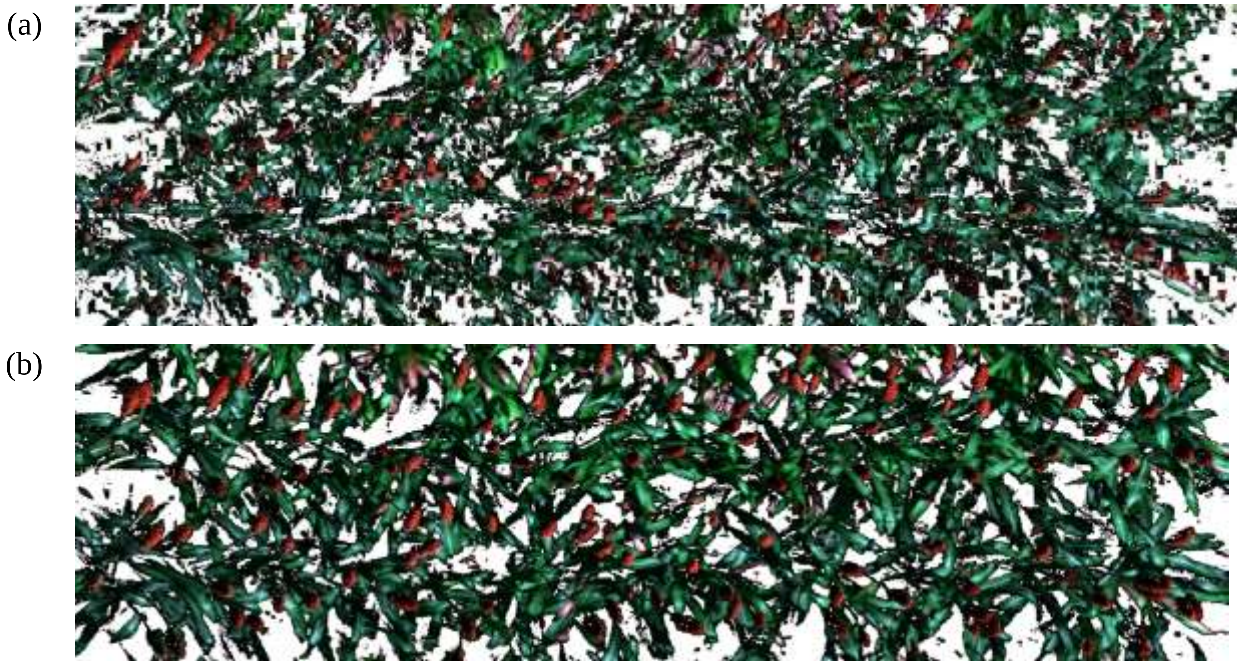


Figure 4.8. Image resulting from orthorectification using different Digital Surface Models: (a) unsmoothed DSM, (b) 95 percentiles DSM over the field as the height value

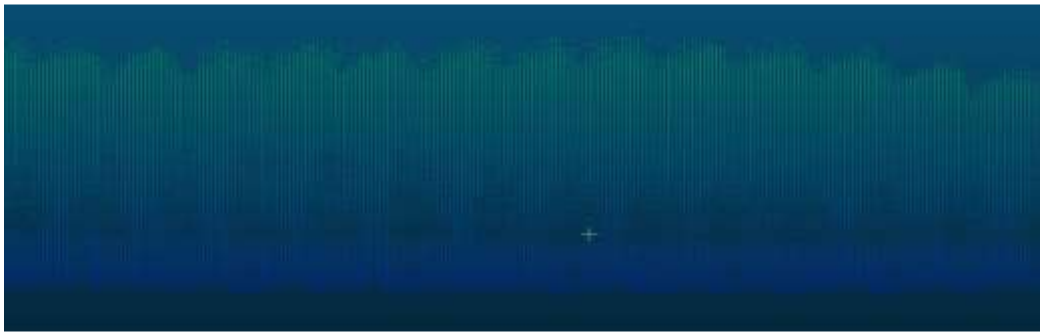


Figure 4.9. Side View of the Digital Surface Model (DSM)

To obtain an orthophoto with accurate positioning and to detect the panicles in each plot, improved geometric correction was required. In this thesis, the image data collected for boresight calibration was used to quantitatively evaluate results of the orthorectification experiments. It has the advantage that it is a flat area with no vegetation, so that it was not affected by the height variability of the surface, as observed with the maize and sorghum. Several raw image data sets were acquired from the calibration field, and Figure 4.10 shows the orthorectification results of one of the raw image cubes collected on September 19th, the same date as the data used for panicle detection. As shown in the three images in Figure 4.10, the white dots are the targets used for evaluation, while the true positions of the targets are shown as red dots. The orthorectification was evaluated by comparing the location of the white targets and the ground reference information. Parameters used in the back projection function (Habib et al., 2018) can be derived from both the Ground Control point (GCP)-based and Tie Point-based methods (Habib et al., 2018). Setting parameters such as the lever arm and focal length as fixed or flexible yielded different results. Figure 4.10 (a) and (c) acquired over the calibration targets show the impact of a single height, but different parameters as listed in Table 4.3 and Table 4.4. When inappropriate parameters are used, the targets are shifted from their true position. Optimal values of the eight parameters relative to minimum root-mean-square error (RMSE) were computed and used to orthorectify the distorted image data. To determine the effect of the height, Figures 4.10 (a) and (b) assume different height values in conjunction with the parameters in Table 4.3. Figure 4.10 (a) uses the average height of the ground reference target (181.554 m), and Figure 4.10 (b) uses the average height from the DSM data (181.283 m). The targets in Figure 4.10 (b) are stretched from their accurate position due to the use of the wrong height value. However, field conditions with crops of variable heights at near range are quite variable. Using the 95 percentile DSM data (commonly assumed to be the top collar of the plant) resulted in the Z value in the point positioning function (Eqn 3.1) being the same for all the pixels in the image data. Therefore, strategy for resolving the orthorectification problem focused on the effect of unsmoothed height value in the DSM.

Table 4.3. Mounting parameters derived from fixed level arm.

Omega	Phi	Kappa	LA_X	LA_Y	LA_Z
1.061	16.199	-0.022	1.32	-1.01	-0.40

Table 4.4. Mounting parameters derived from flexible lever arm.

Omega	Phi	Kappa	LA_X	LA_Y	LA_Z
0.0562	13.730	0.467	0.801	-1.058	-0.40

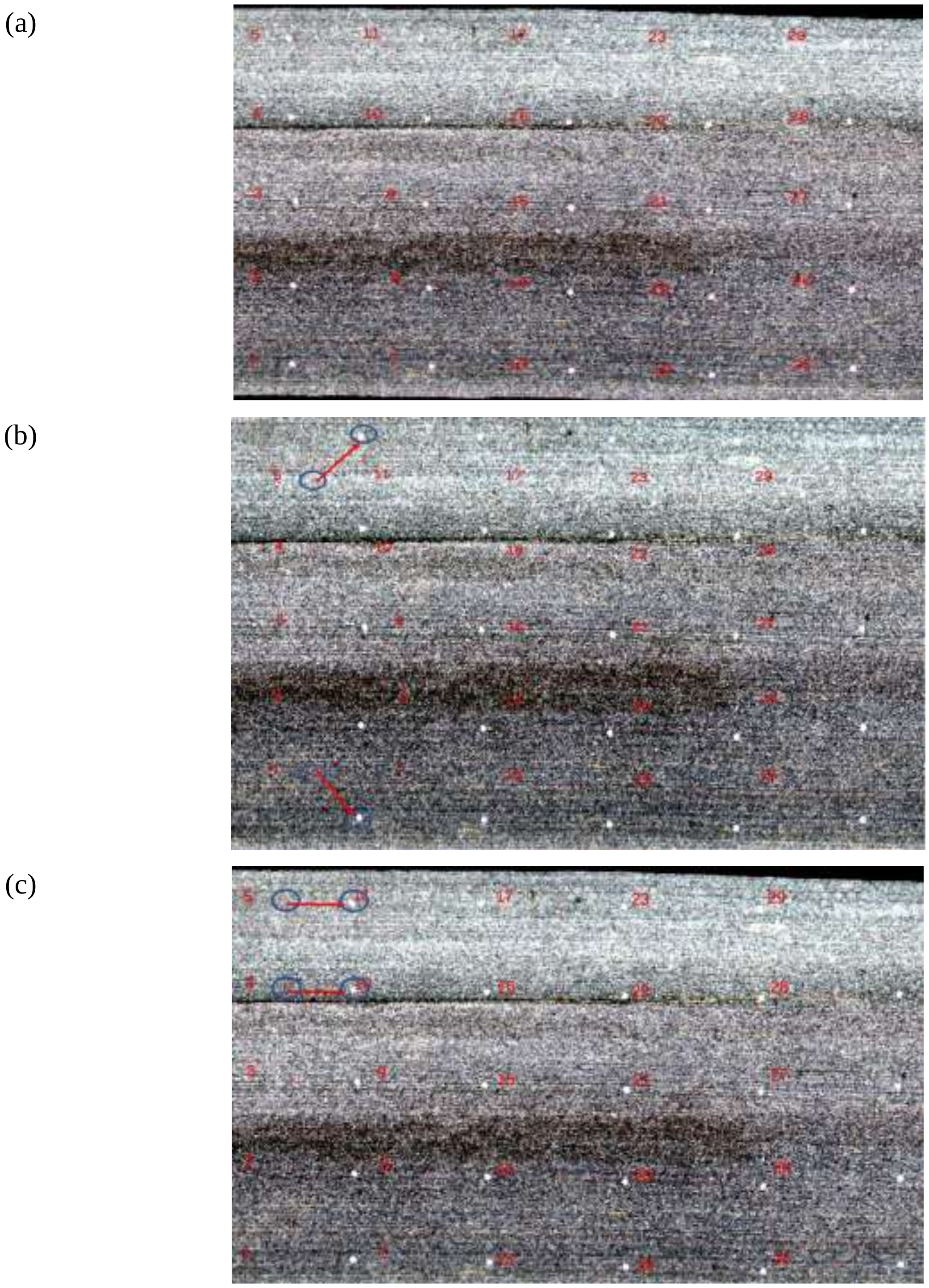


Figure 4.10. Image acquired for calibration after orthorectification by using different height values and mounting parameters: (a) mean of target heights and mounting parameters in Table 4.3, (b) mean DSM value and mounting parameters in Table 4.3, illustrating the stretch problem, (c) mean of target heights and mounting parameters in table 4.4, illustrating shift problem

In this thesis, the follow-up tests on orthorectification were focused on the maize in Field 9D in July 18th because the height of maize was more homogeneous than the sorghum. The DSM used for investigating the improved orthorectification strategy is shown in Figure 4.11. It should be noted that this research was actually conducted after the panicle detection part of the thesis, where the orthophoto generation was achieved using the Headwall SpectralView software. It is acknowledged that those orthorectified data are shifted. Because only the center two rows were used, distortion in the cross track direction was minimal. Figure 4.12 (a) shows results obtained using the original 4 cm DSM, where the image quality is very poor (analogous to the orthorectification results on sorghum as in Figure 4.8 (a)). Figures. 4.12 (b) to Figure 4.12 (d) show orthorectification results using DSMs obtained when applying average filters with *0.1×0.1 m*, *0.5×0.5 m*, and *1×1 m* window size, respectively to the original DSM. The orthorectified image result shown in Figure 4.12 (b) still has a poor image quality and a large distortion, which is improved in Figure 4.12 (c) and (d) with only distortions at the edge of the image and the areas between two plots. Among the four images, Figure 4.12 (d) has the best performance with the *1×1 m* window size, which is similar to the 30"distance between the rows. It was concluded that smoothing the DSM with an appropriate filter improves the quality of orthorectification result.

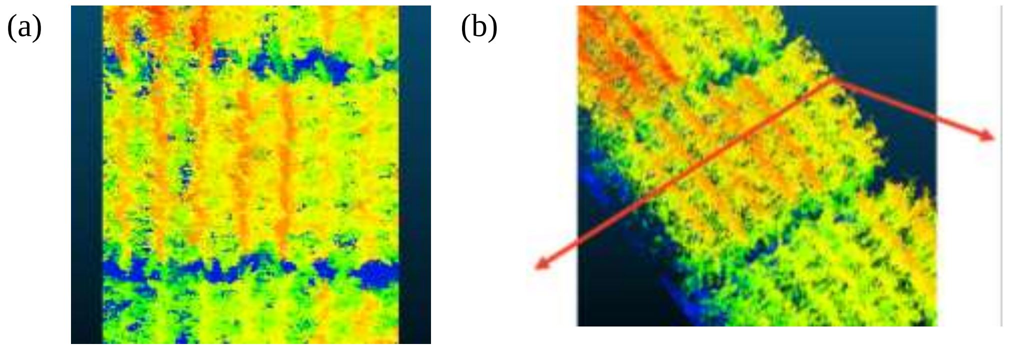


Figure 4.11. DSM used for orthorectification from different side of view: (a) Top view, (b) Oblique

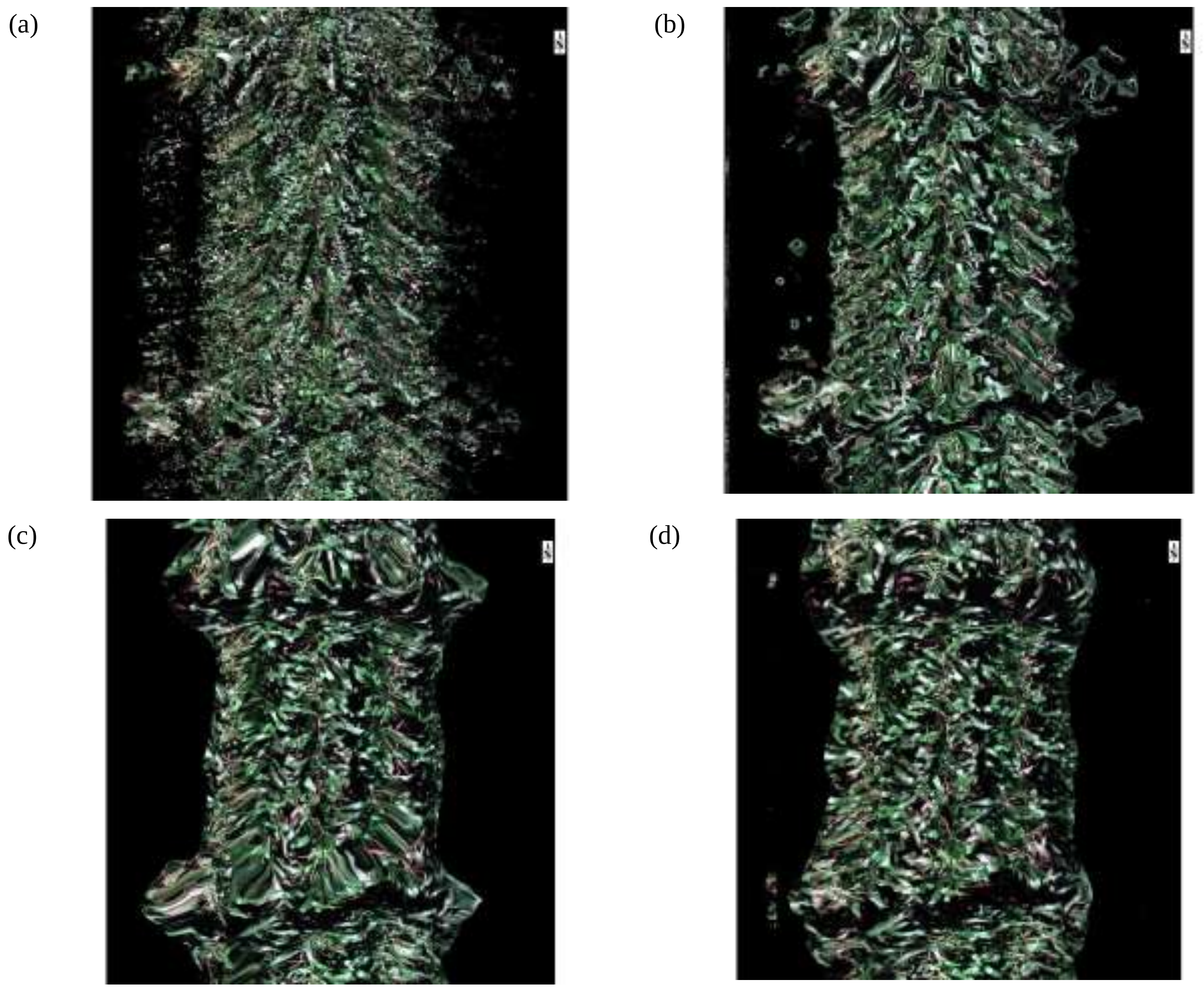


Figure 4.12. Maize image resulting from orthorectification with different DSMs: (a) original DSM, (b) smoothed DSM with 0.1*0.1m window size, (c) smoothed DSM with 0.5*0.5m window size, (d) smoothed DSM with 1*1m window size

The impact of using the Savitzky-Golay (S-Golay) (Savitzky & Golay, 1964) adaptive filter was then investigated for smoothing the DSM. Figure 4.13 contains a plot showing the side view of the orthorectification results on the single transect, as shown in Figure 4.11 (b) for different window sizes/SG parameters. Figures 4.14 a-c shows resulting orthorectified images using the respective parameters. The original DSM in Figure 4.13 indicates 7 peaks are associated with the individual rows, as it is evident in Figure 4.11 (a). The impact of interpolated 3D point cloud from the image after orthorectification using the S-Golay adaptive filter with different filter sizes and the average filter is shown in the plots. The average 100x100 filter (red) also contains 7 peaks, but the positioning is shifted from the original peaks. The original 4cm DSM data (blue) exhibits high localized variability, which impacts the resulting orthophotos, as noted previously. The S-Golay filter results show both the impact of window size and the threshold for the minimum height. For example, Figure 4.14 (b) and (c) have the same window size, but Figure 4.14 (b) uses the mean value as the minimal height while Figure 4.14 (c) uses 75 percentile DSM height. Figure 4.14 (c) shows that the distortion in the edge and the area between the two plots is reduced after modifying the minimal height. For these experiments, the adaptive filter applied to the original DSM data improved the image quality and decreased the cross-track distortion. This approach was used by subsequent research by the team on tassel detection.

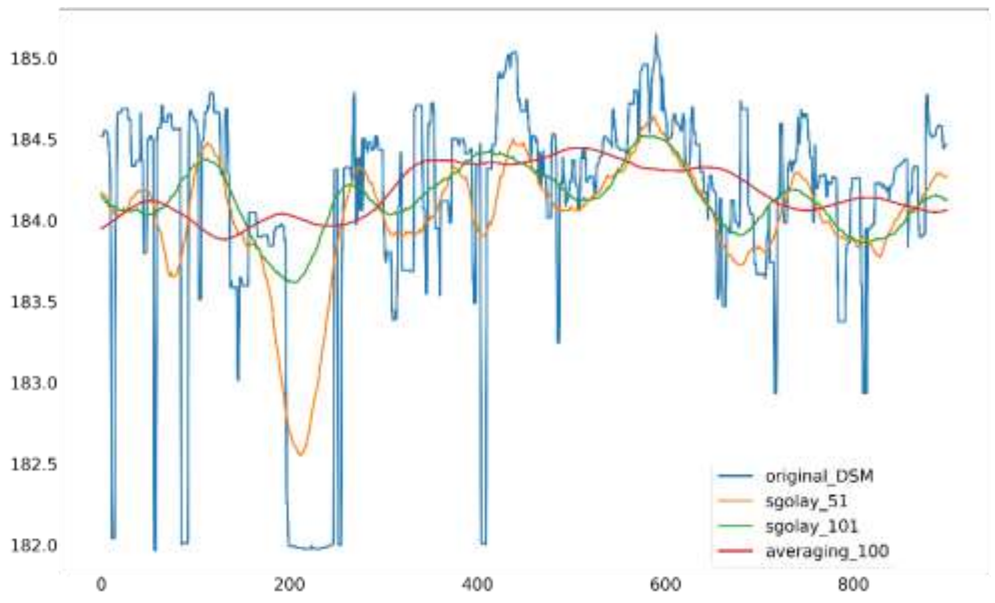


Figure 4.13. Cross-track view across multiple rows of the maize field

(a) (b) (c)

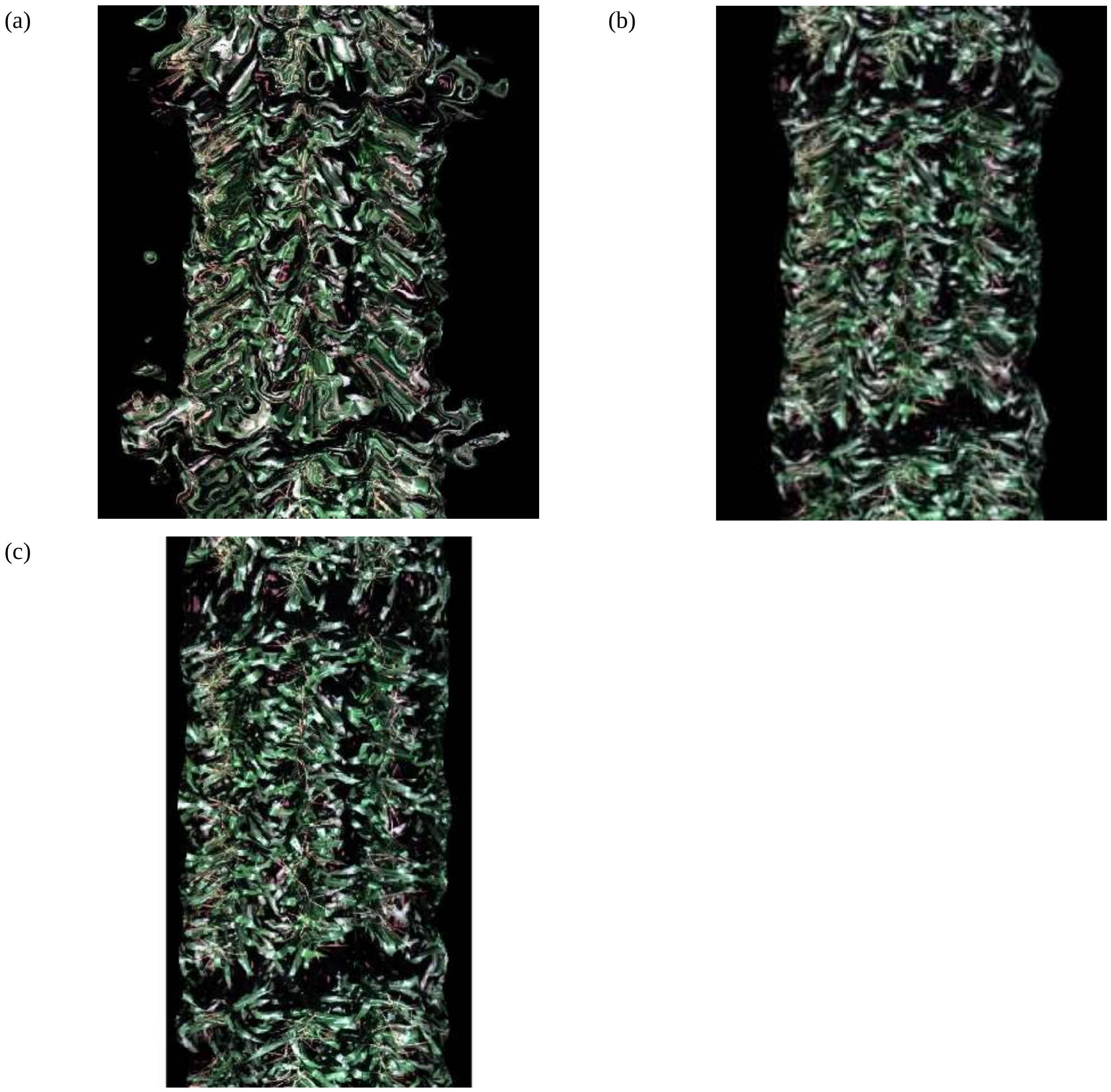



Figure 4.14. Orthorectification results by using Savitzky-Golay adaptive filter on DSM with different parameters: (a)window size= 51, using mean as minimal height (b) window size=101, using mean as minimal height (c) window size= 101, using 75% as minimal height

4.3.2 Panicle Detection

In the following section, the label selection and image annotation are first summarized, followed by discussion of performance of the architectures and the four metrics of the three models including the validation and test.

4.3.2.1 Label Selection

After the geometric correction and conversion to reflectance, labeled data were acquired, and the corresponding labels were used to train the supervised deep learning model. Obtaining annotated reference data for panicle detection with hyperspectral data is challenging. In this thesis, the labels were generated manually by using the ENVI/Harris software (Figure 4.15), which can visualize and process multispectral and hyperspectral image data in RGB bands and select regions of interest (ROI) area from the image. Figure 4.16 shows an example of annotated labels. The black area represents the background including soil, leaf and other objects, while the white areas denotes the panicles. This is a tedious, time consuming process. The annotation tool adapted by Oh et al. (2019) was also investigated for RGB images generated from hyperspectral data, but the results were density maps, which were not accurate enough to be used as binary prediction maps in this thesis. Therefore, future work is needed to simplify the process of selecting labels in either manual or automatic/semi-automatic approaches. For example, a pretrained model could be used to predict the panicles initially, followed by editing.

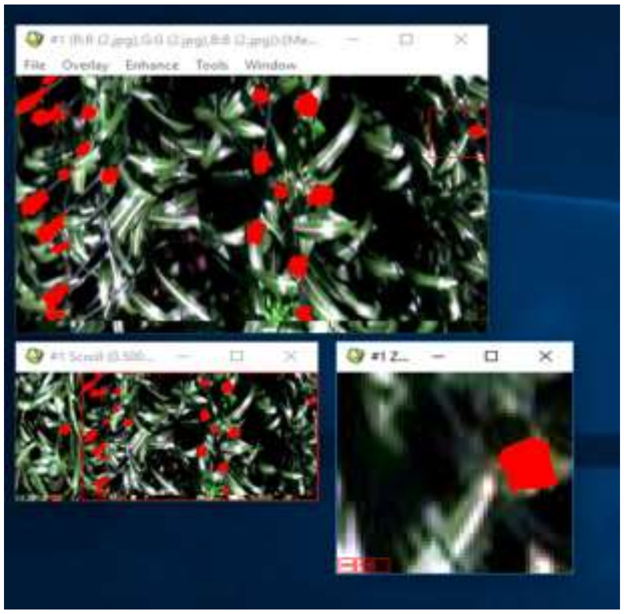


Figure 4.15 Manual label selection by using ENVI displaying RGB bands

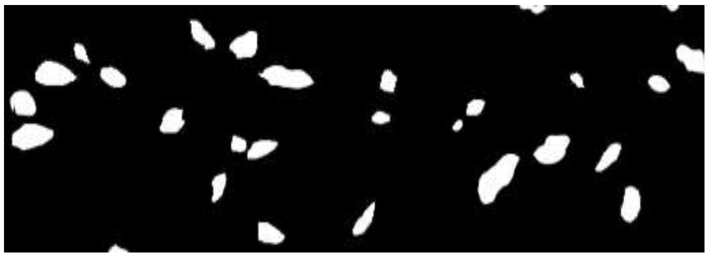


Figure 4.16 Example labels annotated for panicle detection.

4.3.2.2 Model Architecture

As noted in Chapter 3, three different models based on FCN-8s were explored in this research: PCA&FCN, 2D-FCN and 3D-FCN. As noted in Ch 2, PCA&FCN is a two-step pixel-based classification approach. It reduces the dimension of the hyperspectral image, and then is adapted to the FCN-8s with 7 input channels, which represent 99% of the variance in the image. The other two approaches are end-to-end models and use the full spectral bands as input. Figures 4.17 to Figure 4.20 show the structure of FCN-8s, PCA&FCN, 2D-FCN and 3D-FCN. The modified part from the original FCN-8s is outlined in red. From the four figures, we can see that the architecture was modified on the first layer, so the remaining layers could use the pretrained weights of VGG-16. The modified layer used the Glorot initialization (Glorot, Bengio, 2010). The batch size was set as 1 because of the large memory required by the hypercubes. The learning rate was a fixed number (1e-10) and was tested first to make sure the model was trainable. Three hundred thirty-four (334) subsets of hyperspectral cubes, each of size of 256×512×137 were extracted from the pre-processed hyperspectral image and used for panicle detection. Among these cubes, 80% of the images were used as training data, 10% for validation, and 10% as test data. The test data were selected randomly, but contain panicles from different varieties of sorghum and images with different quality. Ten-fold cross-validation (10-cv) was performed to obtain the mean accuracy and to avoid the impact of randomly selected data for validation. The models were first trained on the randomly selected validation data with 500 epochs, and the results were used as the pretrained weight on 10-cv with 500 epochs for each fold. The model was run on multiple GPUs: the Purdue Gilbreth High Performance Computer (HPC), mango HPC (4 GPUs) from the VIPER Lab and Google Co-laboratory. The deep learning model used the Pytorch python package, and the principal components were extracted using the function from the Spectral Python package.

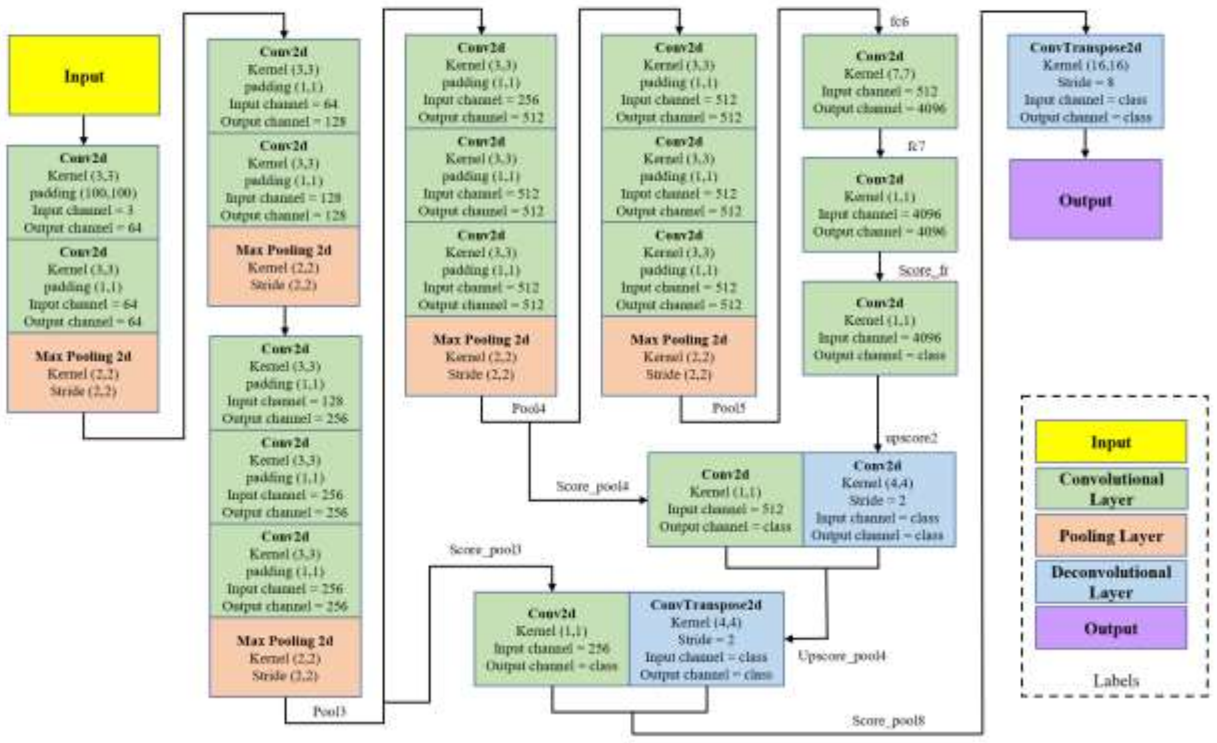

Input
Conv2d Kernel (3,3) padding (100,100) Input channel = 3 Output channel = 64
Conv2d Kernel (3,3) padding (1,1) Input channel = 64 Output channel = 64
Max Pooling 2d Kernel (2,2) Stride (2,2)
Conv2d Kernel (3,3) padding (1,1) Input channel = 64 Output channel = 128
Conv2d Kernel (3,3) padding (1,1) Input channel = 128 Output channel = 128
Max Pooling 2d Kernel (2,2) Stride (2,2)
Conv2d Kernel (3,3) padding (1,1) Input channel = 128 Output channel = 256
Conv2d Kernel (3,3) padding (1,1) Input channel = 256 Output channel = 256
Conv2d Kernel (3,3) padding (1,1) Input channel = 256 Output channel = 256
Max Pooling 2d Kernel (2,2) Stride (2,2)
Pool3
Conv2d Kernel (3,3) padding (1,1) Input channel = 256 Output channel = 512
Conv2d Kernel (3,3) padding (1,1) Input channel = 512 Output channel = 512
Conv2d Kernel (3,3) padding (1,1) Input channel = 512 Output channel = 512
Max Pooling 2d Kernel (2,2) Stride (2,2)
Pool4
Conv2d Kernel (3,3) padding (1,1) Input channel = 512 Output channel = 512
Conv2d Kernel (3,3) padding (1,1) Input channel = 512 Output channel = 512
Conv2d Kernel (3,3) padding (1,1) Input channel = 512 Output channel = 512
Max Pooling 2d Kernel (2,2) Stride (2,2)
Pool5
fc6
Conv2d Kernel (7,7) Input channel = 512 Output channel = 4096
fc7
Conv2d Kernel (1,1) Input channel = 4096 Output channel = 4096
Score_fr
Conv2d Kernel (1,1) Input channel = 4096 Output channel = class
upscore2
Score_pool4
Conv2d Kernel (1,1) Input channel = 512 Output channel = class
Conv2d Kernel (4,4) Stride = 2 Input channel = class Output channel = class
Score_pool3
Conv2d Kernel (1,1) Input channel = 256 Output channel = class
ConvTranspose2d Kernel (4,4) Stride = 2 Input channel = class Output channel = class
Upscore_pool4
Score_pool8
ConvTranspose2d Kernel (16,16) Stride = 8 Input channel = class Output channel = class
Output
Input
Convolutional Layer
Pooling Layer
Deconvolutional Layer
Output
Labels

Figure 4.17 Structure of FCN-8s with VGG-16 as the encoder

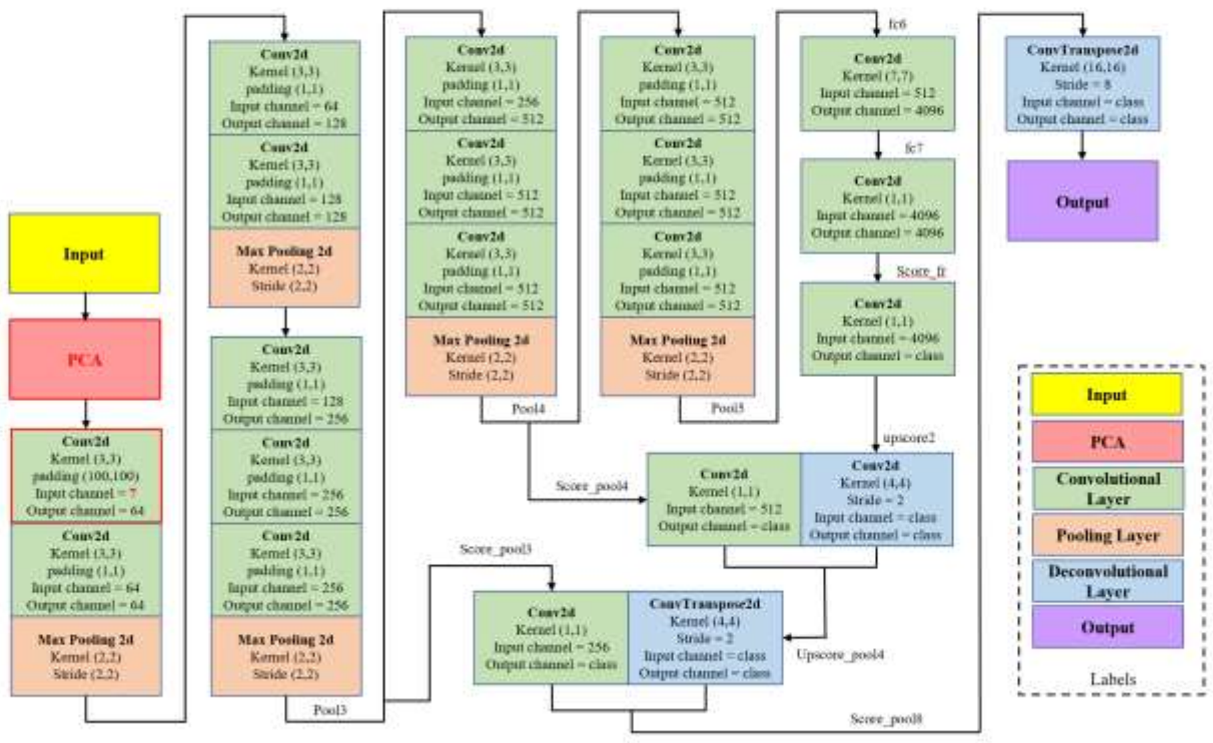

Input
PCA
Conv2d Kernel (3,3) padding (100,100) Input channel = 7 Output channel = 64
Conv2d Kernel (3,3) padding (1,1) Input channel = 64 Output channel = 64
Max Pooling 2d Kernel (2,2) Stride (2,2)
Conv2d Kernel (3,3) padding (1,1) Input channel = 64 Output channel = 128
Conv2d Kernel (3,3) padding (1,1) Input channel = 128 Output channel = 128
Max Pooling 2d Kernel (2,2) Stride (2,2)
Conv2d Kernel (3,3) padding (1,1) Input channel = 128 Output channel = 256
Conv2d Kernel (3,3) padding (1,1) Input channel = 256 Output channel = 256
Conv2d Kernel (3,3) padding (1,1) Input channel = 256 Output channel = 256
Max Pooling 2d Kernel (2,2) Stride (2,2)
Pool3
Conv2d Kernel (3,3) padding (1,1) Input channel = 256 Output channel = 512
Conv2d Kernel (3,3) padding (1,1) Input channel = 512 Output channel = 512
Conv2d Kernel (3,3) padding (1,1) Input channel = 512 Output channel = 512
Max Pooling 2d Kernel (2,2) Stride (2,2)
Pool4
Conv2d Kernel (3,3) padding (1,1) Input channel = 512 Output channel = 512
Conv2d Kernel (3,3) padding (1,1) Input channel = 512 Output channel = 512
Conv2d Kernel (3,3) padding (1,1) Input channel = 512 Output channel = 512
Max Pooling 2d Kernel (2,2) Stride (2,2)
Pool5
fc6
Conv2d Kernel (7,7) Input channel = 512 Output channel = 4096
fc7
Conv2d Kernel (1,1) Input channel = 4096 Output channel = 4096
Score_fr
Conv2d Kernel (1,1) Input channel = 4096 Output channel = class
upscore2
Score_pool4
Conv2d Kernel (1,1) Input channel = 512 Output channel = class
Conv2d Kernel (4,4) Stride = 2 Input channel = class Output channel = class
Score_pool3
Conv2d Kernel (1,1) Input channel = 256 Output channel = class
ConvTranspose2d Kernel (4,4) Stride = 2 Input channel = class Output channel = class
Upscore_pool4
Score_pool8
ConvTranspose2d Kernel (16,16) Stride = 8 Input channel = class Output channel = class
Output
Input
PCA
Convolutional Layer
Pooling Layer
Deconvolutional Layer
Output
Labels

Figure 4.18 Structure of PCA&FCN, the PCA modified for the first seven principal components as the input data.

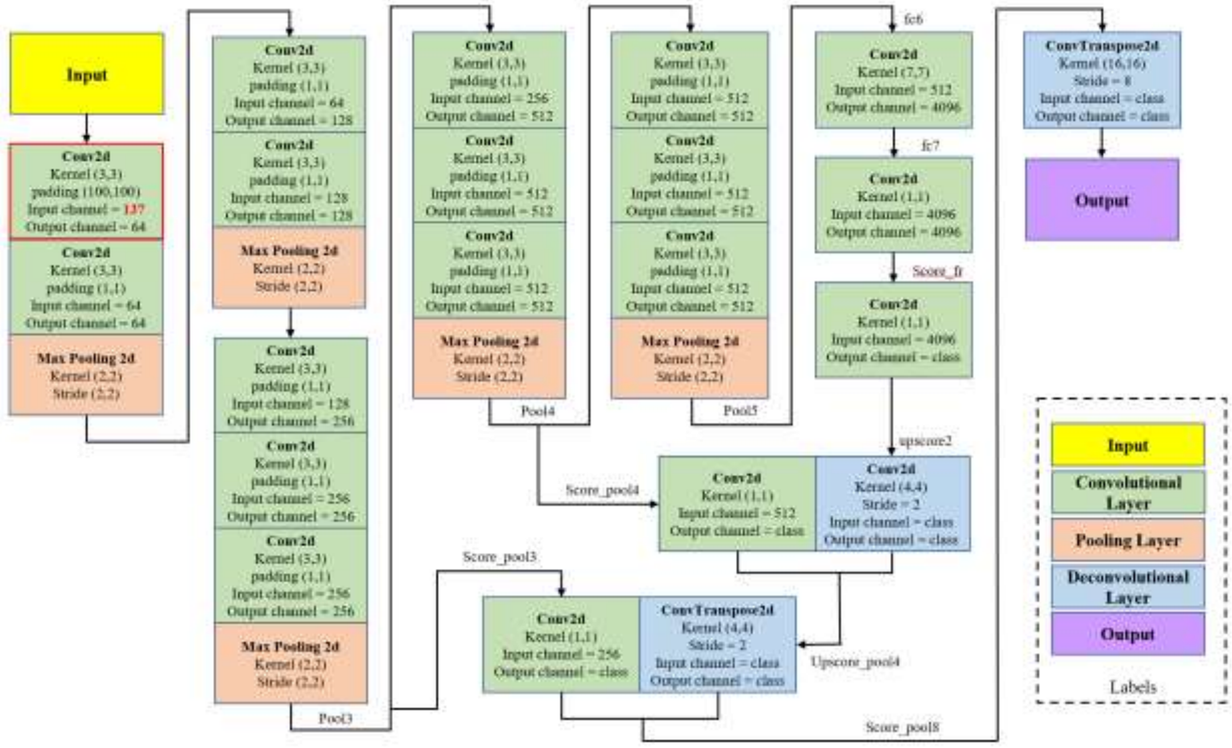


Figure 4.19 Structure of 2D-FCN, the number of the first channel was changed to the number of bands, which is 137 in this thesis.

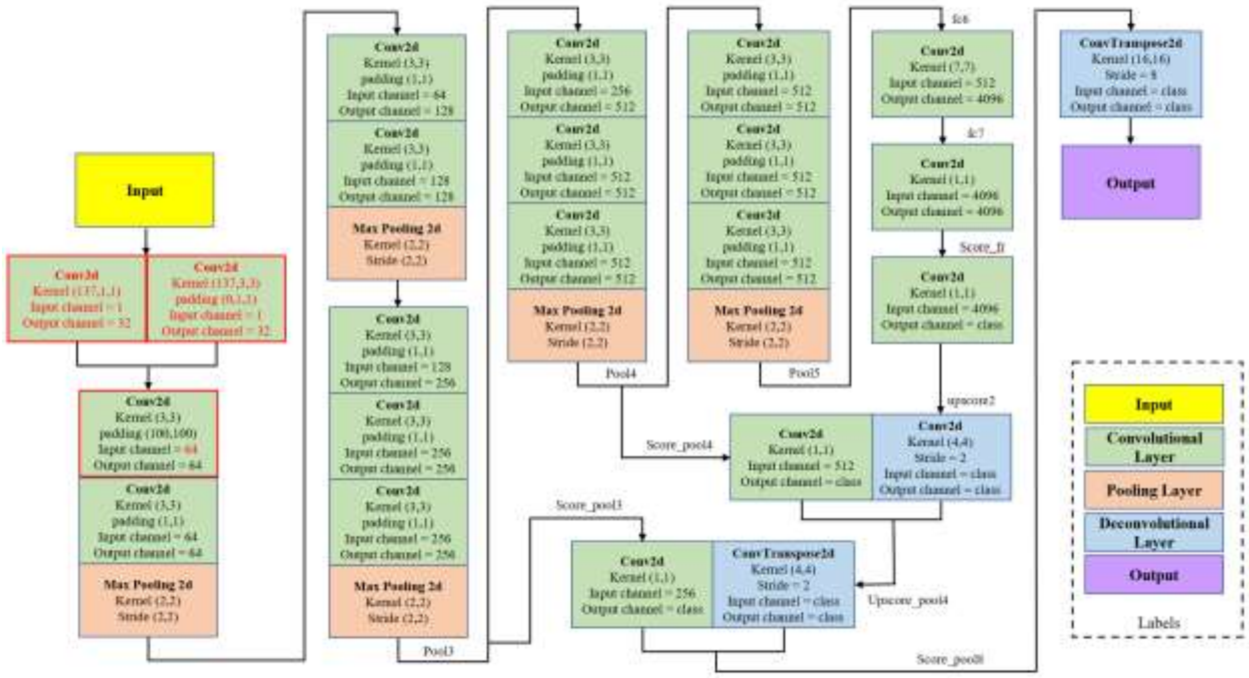


Figure 4.20 Structure of 3D-FCN. A filter bank (Lee, Kwon, 2017) was added before the FCN-8s structure.

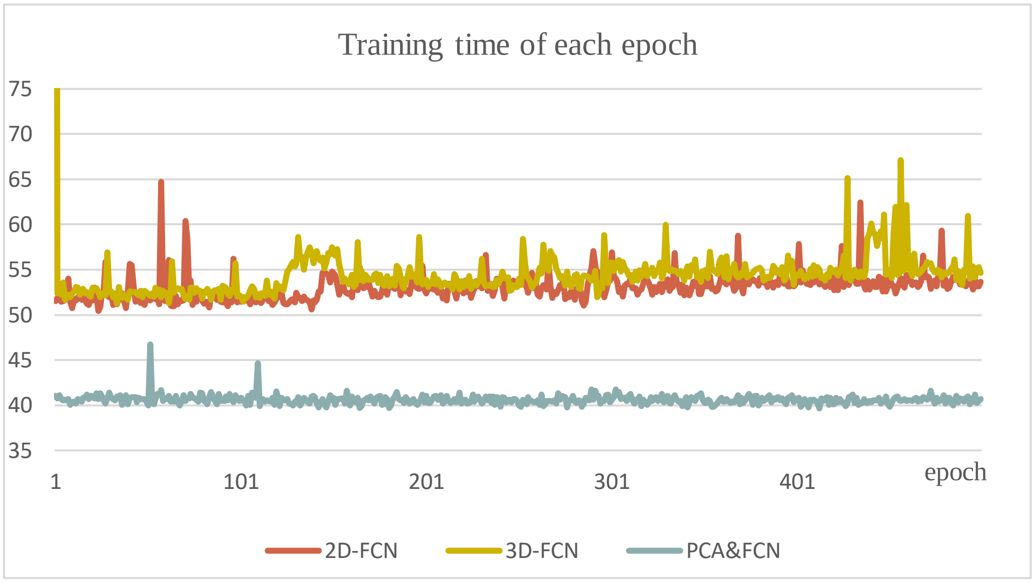
For the convolutional layer, the total number of parameters is $(n \times m \times l+1) \times k$, where n and m are the 2D filter size, l denotes the number of input channels, and k represents the number of output channels. For the 3D convolutional layer, the filter size contains three variables. In the original FCN, the filter size of the first layer is 3 ×3, the number of input channels is 3 and the number of output channels is 64; thus, as noted in Ch 2, it needs to compute 1792 parameters. The input size for the FCN is 256×512×3 with 100 zero padding to ensure the size of the input matrix for the following layers, so the memory requirement of FCN is (256+100) × (512+100) ×3, which is 653,616 bytes. Table 4.5 lists the memory and number of associated parameters for each of the models. Since the modifications of the models are on the first layer as shown in Figure 4.17 to Figure 4.20, PCA&FCN, 2D-FCN are calculated on the first layer, while 3D-FCN considers the first layer as well as the filter bank. 2D-FCN and 3D-FCN both require significantly more parameters than the original network.

Table 4.5 Complexity of the three models

Model	Memory (bytes)	Parameters
PCA&FCN (conv1)	1,525,104	4,096
2D-FCN (conv1)	29,848,464	78,796
3D-FCN (filter bank&conv1)	49,857,536	80,832

Since the models were processed on multiple GPUs for 10-cv, the computation time of the models running on the same GPU was selected for comparison. Figure 4.21 shows the training time of a Gilbreth GPU and a mango GPU for each epoch. For PCA&FCN, the computation time includes the calculation of PCA. It can be seen that the time for the first 100 epochs and the last 100 epochs is unstable. Among the three models, PCA&FCN requires the lowest time. In addition, the training time for 2D-FCN is slightly shorter than 3D-FCN, which is related to the model structure. The total validation loss error for all the pixels is shown in Figure 4.22. In this figure, PCA&FCN has much larger validation loss error than 2D-FCN and 3D-FCN, and using first 5 principal components (pc) is worse than 7 pc. 3D-FCN has the lowest loss error among the four models and 2D-FCN is only slightly larger than 3D-FCN.

(1)



(2)

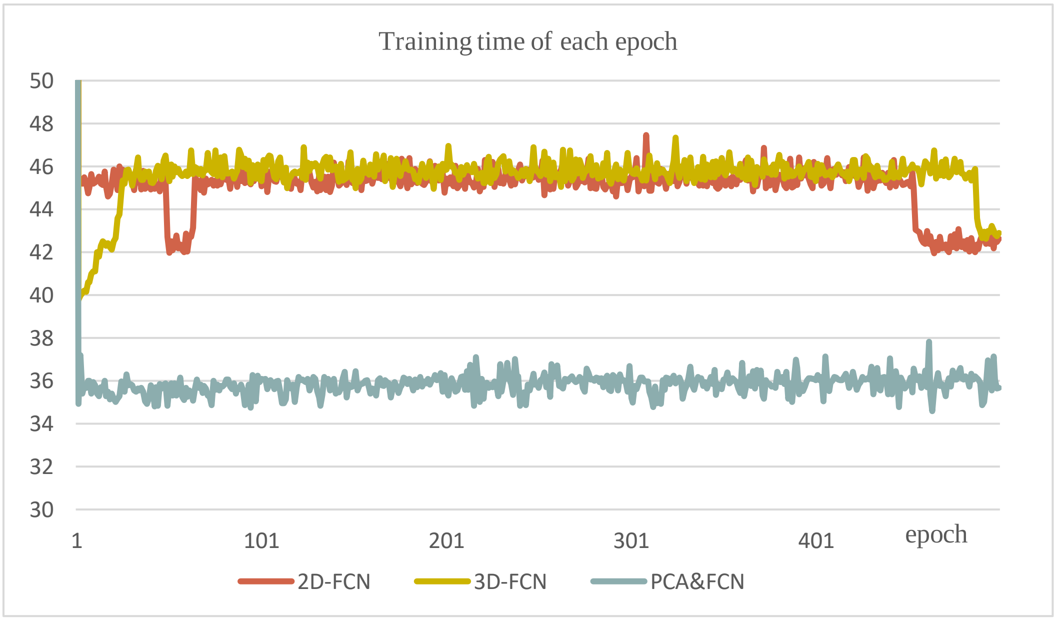


Figure 4.21 Computation time of each epoch for PCA&FCN, 2D-FCN and 3D-FCN on different GPUs: (1) Gilbreth, (2) mango GPU 1

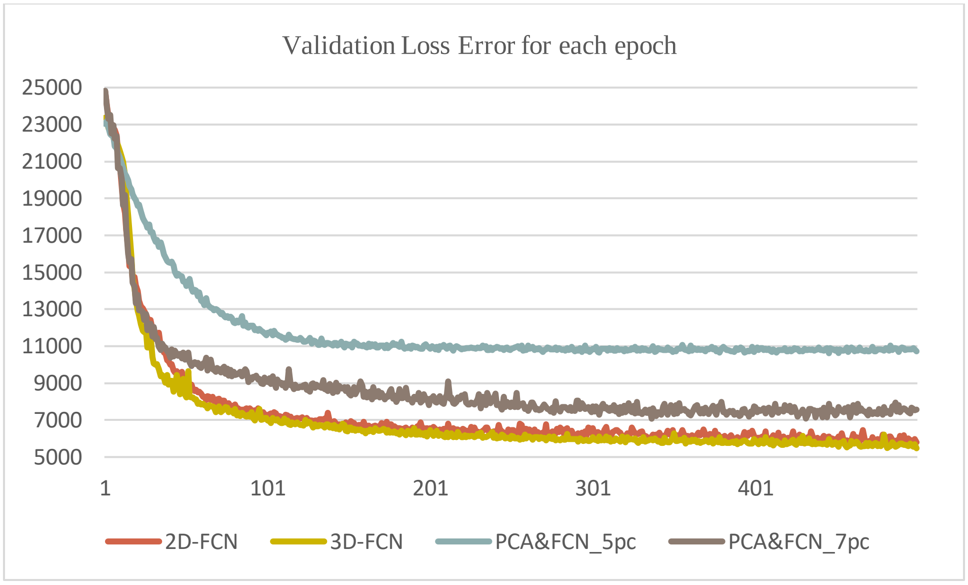


Figure 4.22 Validation loss error of the three models.

4.3.2.3 Panicle Detection performance on test images

Thirty-three images, 10% of the total images, were used to test the performance of the deep learning model. The images were selected randomly, but it is important to note that the test images should contain panicles with different varieties and images with different qualities when the number of training samples are limited. The test data were evaluated to ensure that the required diversity was achieved. Figure 4.23 shows the result of panicle detection from reflectance data using the three approaches. Figure 4.23 (a) to (c) show detection results using PCA&FCN, 2D-FCN and 3D-FCN, respectively. Almost all the panicles were detected by the three models. Using the end-to-end model also seems to yield superior results to the 2-stage process, at least for this dataset. However, panicles that are overlapping or too close to each other cannot be detected separately. This also illustrates the need for additional reference data and more advanced annotation tools that evaluate and possibly split clusters.

(a)

(b)

(c)

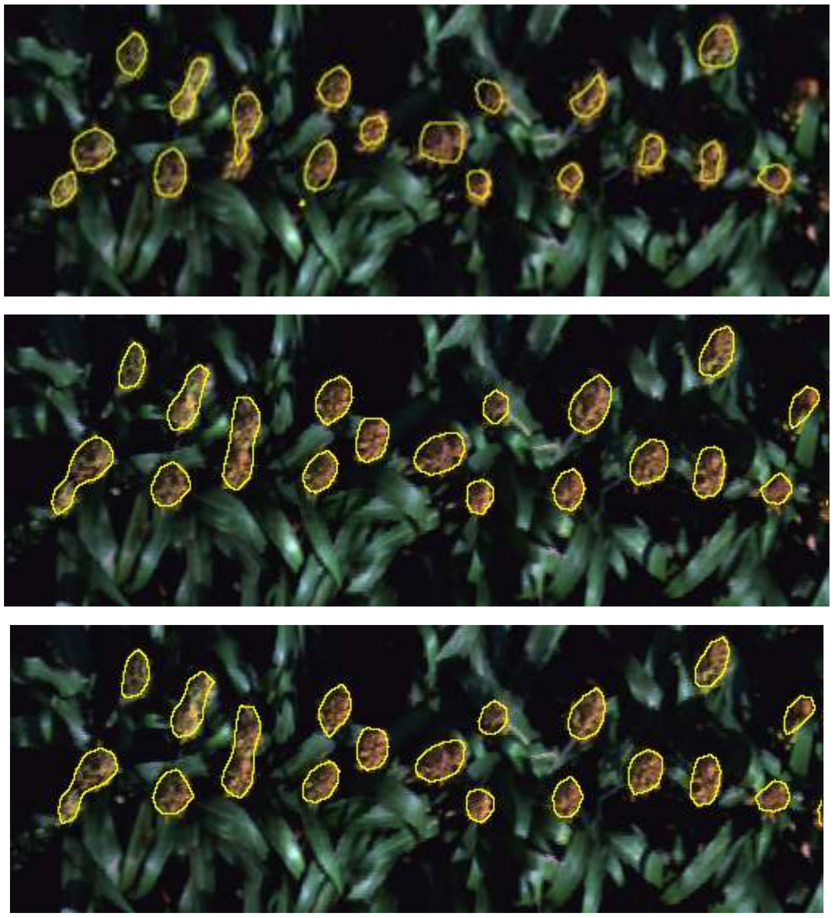
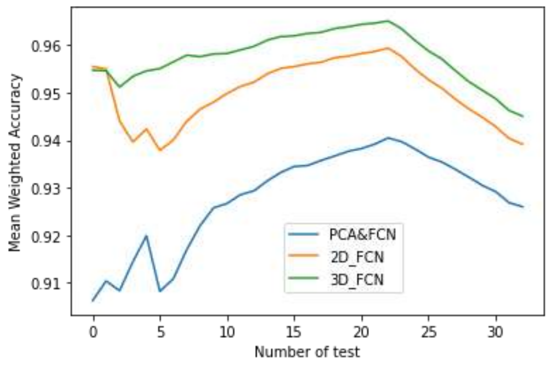


Figure 4.23. Panicle detection results by using different approaches: (a) PCA&FCN-8s (b) 2D-FCN-8s, (c) 3D-FCN-8s

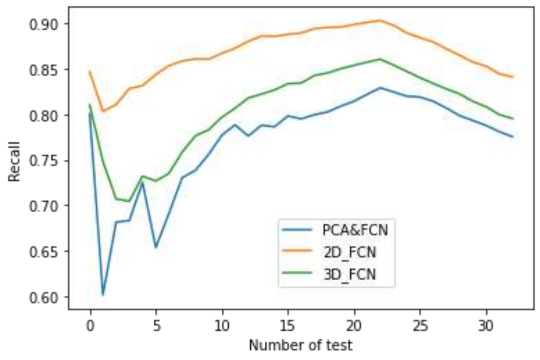
Table 4.6 shows the results of panicle detection based on the four metrics. The standard deviations (std) of the four metrics are all less than 0.02, so the results of 10-cv are stable for all the three models. 3D-FCN performs best with the mean accuracy of 0.945, and F-measure of 0.745 among the three models. Results from 2D-FCN are only slightly worse than 3D-FCN in terms of precision and the F measure. The results obtained from the test images are shown from Figure 4.24 (a) to (d). From these four figures, we can see that: (1) The mean accuracies of all the images are >92% for all three models, among which 3D-FCN has the highest values. (2) Results are image dependent, and specifically impacted by shadow, overlapping panicles, and differences in varieties. Poor quality images, which are usually located in the corner of each plot, also have worse results. For example, Figure 4.25 shows three test images. Figure 4.25 (a) is test image #5 with low accuracy, recall and precision. Figure 4.25 (b) is test image #10 which has high recall but low precision, and

Figure 4.25 (c) image #23, has both high recall and high precision. Among the three images, it is obvious that the first image contains a lot of dead leaves which have a similar spectral signature to panicles within the plot, so that the panicle is difficult to be distinguished. Panicles in the last two images are easy to be recognized. However, the second image results still have a low precision, which implies that the performance of the model depends on the spectral characteristics of the panicles. Future work should consider variety-specific characteristics of panicles. (3) 3D-FCN has a higher precision but a lower recall than 2D-FCN. The F-measure for 3D-FCN is slightly higher than 2D-FCN. (4) The model that uses PCA as the feature extraction method and then fits the data to FCN consistently has the worst performance, which implies that the two step method based on these features is worse than the end-to-end approach.

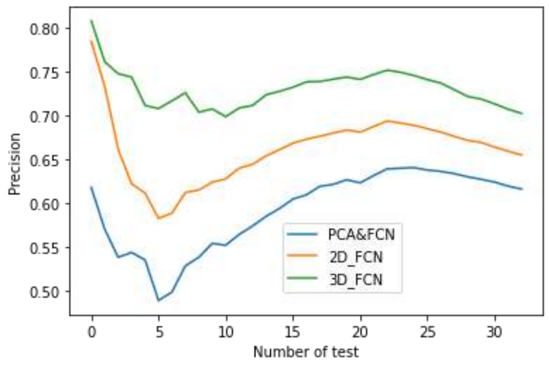
(a)



(b)



(c)



(d)

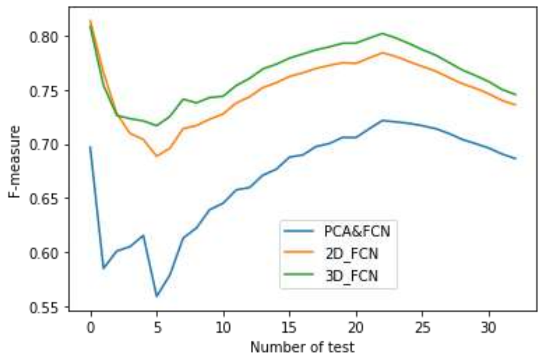


Figure 4.24. Results of 30 test images with different varieties:
(a) Accuracy, (b) Recall, (c) Precision, (d) F-measure

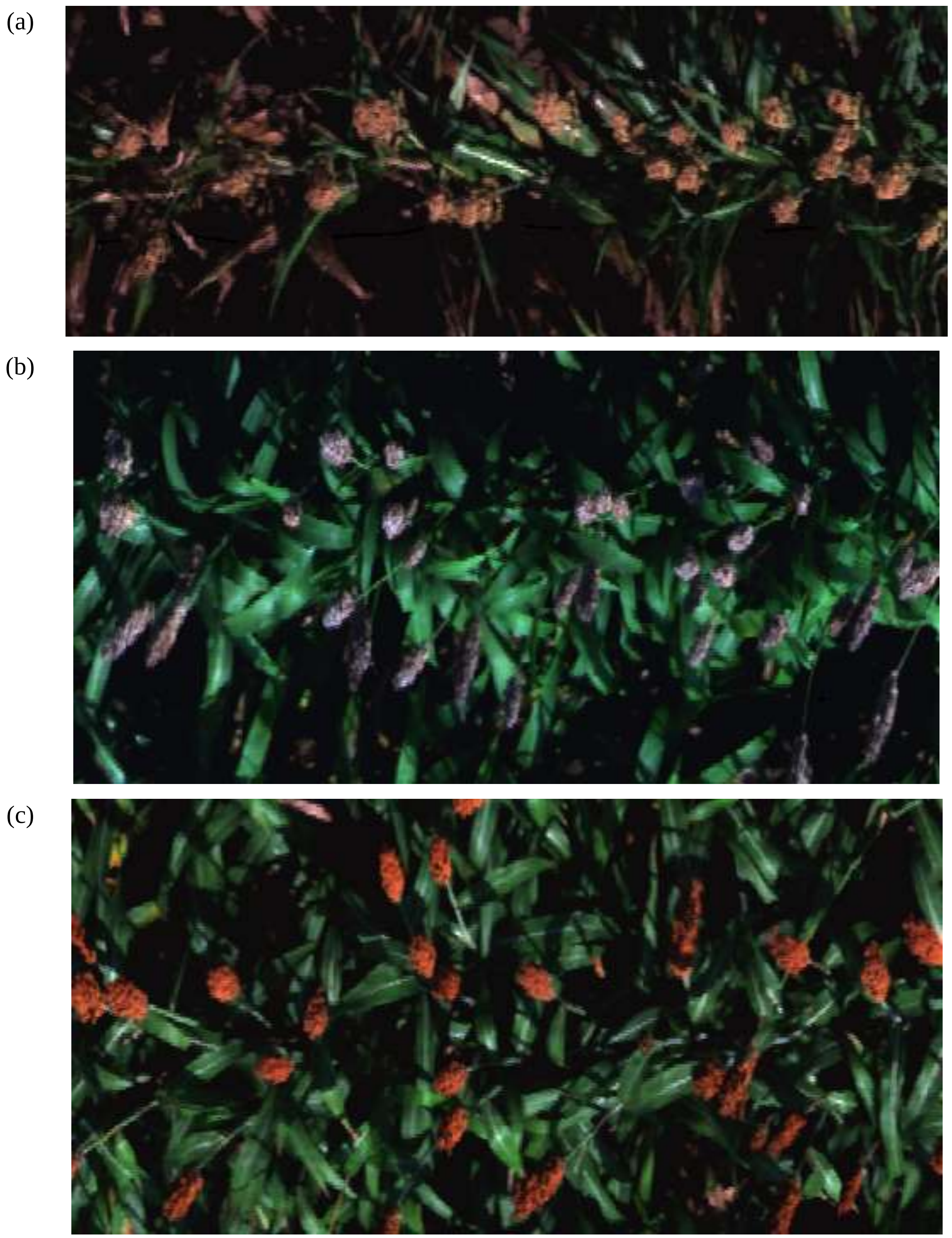


Figure 4.25. 3 images from 30 test images displayed with RGB bands:
(a) image #5, (b) image #10, (c) image #23

Table 4.6 Classification results on test images based on four metrics

test	**Accuracy (std)**	**Precision (std)**	**Recall (std)**	**F-measure (std)**
2D-FCN	0.94(1.5e-3)	0.66(0.01)	0.84(0.01)	0.74(2.8e-3)
3D-FCN	0.94(8.4e-4)	0.70(0.01)	0.80(0.01)	0.75(1.0e-3)
PCA&FCN	0.93(1.9e-3)	0.62(8.5e-3)	0.78(0.01)	0.69(4.6e-3)

To summarize, this chapter first introduced the dataset and some procedures of pre-processing. One of the pre-processing steps, orthorectification was specifically described as minor contribution. After transforming to the regularized 3D hypercube, the results of the three tuned version of deep learning models were evaluated based on four metrics. The overall contributions and future work are discussed in the last chapter.

5. CONCLUSIONS AND FUTURE WORK

The objectives of this research were to 1) to explore the potential contribution of hyperspectral data to the detection of sorghum panicles, and 2) to investigate deep learning architectures for classification of hyperspectral remote sensing data. Hyperspectral imagery with high spatial and spectral resolution over the VNIR portion of the spectrum were collected from a wheeled vehicle platform (the PhenoRover) and provided the testbed for the study. Three tuned versions of the FCN deep learning architecture were adapted to the 3D hypercube: 1) PCA&FCN, where PCA was used as the dimension reduction method followed by classification based on a modified architecture of FCN, 2) 2D-FCN, where the number of input channels was modified to correspond to the number of spectral bands, and 3) 3D-FCN, which used a filter bank to extract spatial-spectral features of the hyperspectral data. A secondary objective of the research was to improve the orthorectification of hyperspectral data acquired by a pushbroom sensor at close range. This was necessary, as the original orthorectified data were geometrically distorted. One component of this research involved investigation of spatial filters for the high resolution LiDAR DSM.

The contributions of this research include:

- Improved orthorectification of close range hyperspectral data. Panicle detection for sorghum using hyperspectral data from the PhenoRover required orthorectification to achieve the correct positioning of the rows in the field. When testing the back-projection orthorectification method used in Habib et al. (2018) for UAV data, the quality of the orthorectification results was unsatisfactory for panicle detection because the sensor is more sensitive to the mounting parameters and the changes of height over a very short spatial interval due to the close range of the objects to the sensor. In addition, the variation in the height of the objects caused a cross track stretching problem with the data, resulting in images with inaccurate positioning. The shifting problem was resolved by using a more accurate boresight calibration, and the experiments showed that applying a smoothing filter to the original DSM improved the quality of the orthorectified image.
- Smoothing of the LiDAR DSM. The filter size impacted the quality of the resulting image. The highest quality orthorectified images were obtained using filters whose spatial extent was approximately the size of the row width (about 1m). Moreover, the Savitzky-Golay

filter not only improved the image quality, but also reduced the distortion at the edges of the image, and in the area between two plots.

- Deep Learning Architectures. Results obtained for different deep learning architectures differed in computational requirements. PCA&FCN is a two-step pixel-based approach, and as such, had the lowest computational overhead. 2D-FCN and 3D-FCN are both end-to-end models, where 2D-FCN uses the first input layer as the dimension reduction procedure. 3D-FCN uses a deeper network than 2D-FCN with a larger computational requirement (49,857,536 bytes for memory and 80832 for parameters), and the 3D kernels in 3D-FCN extract both spatial and spectral features. Comparing the three tuned versions of FCN, it was concluded that all three pixel-based approaches, PCA&FCN, 2D-FCN, and 3D-FCN were able to detect panicles in the example images. 2D-FCN and 3D-FCN performed better than the PCA&FCN model in terms of the classification metrics. Furthermore, 2D-FCN had a higher recall than 3D-FCN, while the latter had higher precision. Relative to the hyperspectral input, the 3D hypercube including the full range of hyperspectral data yielded better results than using principal components as the input. The image quality and the unbalanced numbers of different varieties also have an impact on the results.

Several challenges were encountered in the study.

- The success of deep learning models is largely dependent on having large quantities of high quality training data. This was difficult for the PhenoRover data cubes, which are complex to process.
- The approaches that were considered focused on supervised learning, requiring labels. The reference data in this thesis were selected manually. Visualization of hyperspectral data was problematic for annotation, and image annotation tools explored in this research were time consuming and difficult. In the future, it is advised that an automatic or semi-automatic approach should be identified to increase annotation efficiency. A testbed of data should be developed to train networks. It might be possible to utilize these pretrained networks, with transfer learning for other fields/dates, thereby reducing the required training for each experiment.
- The panicles evaluated in this study did not represent the full range of colors and shapes during the season, and available data was collected at the very end of the season, when they

were the most visible. Future acquisitions should be planned, where possible, to include collections during the flowering period.

- Further studies need to consider the problem of overlapping panicles. For example, a density map, such as proposed by Oh et al. (2019), can be used instead of a prediction map. The model could also potentially be improved by modifying the architecture or using other feature extraction approaches. Object-based segmentation or other nonlinear dimension reduction methods could be also tested to determine whether improved results can be achieved for hyperspectral image data.
- Because LiDAR data were collected by the platform at the same time as the hyperspectral data, it might be useful to take advantage of the height information to detect panicles and determine their volume, as well as to count the number of panicles for each plot Colorized LiDAR might also be effective, if the LiDAR data have adequate density.

REFERENCES

Alam, F. I., Zhou, J., Liew, A. W. C., & Jia, X. (2016, July). CRF learning with CNN features for hyperspectral image segmentation. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (pp. 6890-6893).

Audebert, N., Le Saux, B., & Lefèvre, S. (2016, November). Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. In *Asian conference on computer vision* (pp. 180-196). Springer, Cham.

Audebert, N., Le Saux, B., & Lefèvre, S. (2019). Deep learning for classification of hyperspectral data: A comparative review. *IEEE Geoscience and Remote Sensing Magazine*, *7*(2), (pp. 159-173).

Brugger, A., Behmann, J., Paulus, S., Luigs, H. G., Kuska, M. T., Schramowski, P., ... & Mahlein, A. K. (2019). Extending Hyperspectral Imaging for Plant Phenotyping to the UV-Range. *Remote Sensing*, *11*(12), 1401.

Bruning, B., Liu, H., Brien, C., Berger, B., Lewis, M., & Garnett, T. (2019). The development of hyperspectral distribution maps to predict the content and distribution of nitrogen and water in wheat (Triticum aestivum). *Frontiers in Plant Science*, *10*, 1380.

Chen, Y., Jiang, H., Li, C., Jia, X., & Ghamisi, P. (2016). Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, *54*(10), (pp. 6232-6251).

Du, Q. (2003, October). Band selection and its impact on target detection and classification in hyperspectral image analysis. In *IEEE Workshop on Advances in Techniques for Analysis of Remotely Sensed Data, 2003* (pp. 374-377).

Duan, L., Huang, C., Chen, G., Xiong, L., Liu, Q., & Yang, W. (2015). Determination of rice panicle numbers during heading by multi-angle imaging. *The Crop Journal*, *3*(3), (pp. 211-219).

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.

Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256).

Guo, W., Fukatsu, T., & Ninomiya, S. (2015). Automated characterization of flowering dynamics in rice using field-acquired time-series RGB images. *Plant methods*, *11*(1), 7.

Habib, A., Kersting, A., Bang, K., & Rau, J. (2011). A novel single-step procedure for the calibration of the mounting parameters of a multi-camera terrestrial mobile mapping system. *Archiwum Fotogrametrii, Kartografii i Teledetekcji*, *22*.

Habib, A., Zhou, T., Masjedi, A., Zhang, Z., Flatt, J. E., & Crawford, M. (2018). Boresight calibration of GNSS/INS-assisted push-broom hyperspectral scanners on UAV platforms. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *11*(5), (pp. 1734-1749).

Hamida, A. B., Benoit, A., Lambert, P., & Amar, C. B. (2018). 3-D deep learning approach for remote sensing image classification. *IEEE Transactions on geoscience and remote sensing*, *56*(8), (pp. 4420-4434).

Hu, W., Huang, Y., Wei, L., Zhang, F., & Li, H. (2015). Deep convolutional neural networks for hyperspectral image classification. *Journal of Sensors, 2015*.

Jiao, L., Liang, M., Chen, H., Yang, S., Liu, H., & Cao, X. (2017). Deep fully convolutional network-based spatial distribution prediction for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *55*(10), (pp. 5585-5599).

Jin, S., Su, Y., Gao, S., Wu, F., Hu, T., Liu, J., ... & Pang, S. (2018). Deep learning: individual maize segmentation from terrestrial lidar data using faster R-CNN and regional growth algorithms. *Frontiers in Plant Science, 9*, 866.

Kersting, A. P., Habib, A., & Bang, K. I. (2011, January). Mounting parameters calibration of GPS/INS-assisted photogrammetric systems. In *2011 IEEE International Workshop on Multi-Platform/Multi-Sensor Remote Sensing and Mapping* (pp. 1-6).

Khodr, J., & Younes, R. (2011, October). Dimensionality reduction on hyperspectral images: A comparative review based on artificial data. In *2011 4th International Congress on Image and Signal Processing* (Vol. 4, pp. 1875-1883). IEEE.

Kumar, A. A., Sharma, H. C., Sharma, R., Blummel, M., Reddy, P. S., & Reddy, B. V. (2013). Phenotyping in sorghum [Sorghum bicolor (L.) Moench]. In *Phenotyping for Plant Breeding* (pp. 73-109). Springer, New York, NY.

Lee, H., & Kwon, H. (2017). Going deeper with contextual CNN for hyperspectral image classification. *IEEE Transactions on Image Processing*, *26*(10), (pp. 4843-4855).

Liu, Z. Y., Shi, J. J., Zhang, L. W., & Huang, J. F. (2010). Discrimination of rice panicles by hyperspectral reflectance data based on principal component analysis and support vector classification. *Journal of Zhejiang University Science B, 11*(1), 71-78.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 3431-3440).

Ma, X., Wang, H., & Geng, J. (2016). Spectral–spatial classification of hyperspectral image based on deep auto-encoder. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *9*(9), (pp. 4073-4085).

Makantasis, K., Karantzalos, K., Doulamis, A., & Doulamis, N. (2015, July). Deep supervised learning for hyperspectral data classification through convolutional neural networks. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (pp. 4959-4962).

Malambo, L., Popescu, S. C., Horne, D. W., Pugh, N. A., & Rooney, W. L. (2019). Automated detection and measurement of individual sorghum panicles using density-based clustering of terrestrial lidar data. *ISPRS journal of Photogrammetry and Remote Sensing*, *149,* (pp. 1-13).

Masjedi, A., Carpenter, N. R., Crawford, M. M., & Tuinstra, M. R. (2019). Prediction of sorghum biomass using uav time series data and recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, (pp. 0-0).

Miao, C., Pages, A., Xu, Z., Rodene, E., Yang, J., & Schnable, J. C. (2020). Semantic segmentation of sorghum using hyperspectral data identifies genetic associations. *Plant Phenomics*, *2020*, 4216373.

Mou, L., Ghamisi, P., & Zhu, X. X. (2017). Deep recurrent neural networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *55*(7), (pp. 3639-3655).

Nagata, M., Tallada, J. G., Kobayashi, T., Cui, Y., & Gejima, Y. (2004). Predicting maturity quality parameters of strawberries using hyperspectral imaging. In *2004 ASAE Annual Meeting* (p. 1). American Society of Agricultural and Biological Engineers.

Oh, M. H., Olsen, P., & Ramamurthy, K. N. (2019). Counting and Segmenting Sorghum Heads. *arXiv preprint arXiv:1905.13291*.

Okamoto, H., Murata, T., Kataoka, T., & HATA, S. I. (2007). Plant classification for weed detection using hyperspectral imaging with wavelet analysis. *Weed Biology and Management*, *7*(1), (pp. 31-37).

Olsen, P. A., Ramamurthy, K. N., Ribera, J., Chen, Y., Thompson, A. M., Luss, R., ... & Abe, N. (2018, October). Detecting and counting panicles in sorghum images. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)* (pp. 400-409).

Rottensteiner, F., Sohn, G., Jung, J., Gerke, M., Baillard, C., Benitez, S., & Breitkopf, U. (2012). The ISPRS benchmark on urban object classification and 3D building reconstruction. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences I-3 (2012), Nr. 1*, *1*(1), (pp. 293-298).

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, *10*(3).

Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, *100*(5), (pp. 401-409).

Savitzky, A., & Golay, M. J. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, *36*(8), (pp. 1627-1639).

Slavkovikj, V., Verstockt, S., De Neve, W., Van Hoecke, S., & Van de Walle, R. (2015, October). Hyperspectral image classification with convolutional neural networks. In *Proceedings of the 23rd ACM international conference on Multimedia* (pp. 1159-1162).

Tang, W., Zhang, Y., Zhang, D., Yang, W., & Li, M. (2011, November). Corn tassel detection based on image processing. In *2012 International Workshop on Image Processing and Optical Engineering* (Vol. 8335, p. 83350J). International Society for Optics and Photonics.

Tenenbaum, J. B., De Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, *290*(5500), (pp. 2319-2323).

Velumani, K., Elberink, S. O., Yang, M. Y., & Baret, F. (2017). Wheat ear detection in plots by segmenting mobile laser scanner data. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences, 4*.

Walter, A., Liebisch, F., & Hund, A. (2015). Plant phenotyping: from bean weighing to image analysis. *Plant Methods*, *11*(1), 14.

Xing, C., Ma, L., & Yang, X. (2016). Stacked denoise autoencoder based feature extraction and classification for hyperspectral images. *Journal of Sensors*, *2016*. (pp. 1-10).

Xiong, X., Duan, L., Liu, L., Tu, H., Yang, P., Wu, D., ... & Liu, Q. (2017). Panicle-SEG: a robust image segmentation method for rice panicles in the field based on deep learning and superpixel optimization. *Plant Methods*, *13*(1), 104.

Yang, X., Ye, Y., Li, X., Lau, R. Y., Zhang, X., & Huang, X. (2018). Hyperspectral image classification with deep learning models. *IEEE Transactions on Geoscience and Remote Sensing*, *56*(9), (pp. 5408-5423).

Zeiler, M. D., Krishnan, D., Taylor, G. W., & Fergus, R. (2010, June). Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 2528-2535).

Zhao, P., Zhou, S., Yang, Y., & Hu, Y. (2017). Classification method of hyperspectral remote sensing image based on SLIC and active learning. *Computer Engineering & Applications*, *3*.

Zhu, Y., Cao, Z., Lu, H., Li, Y., & Xiao, Y. (2016). In-field automatic observation of wheat heading stage using computer vision. *Biosystems Engineering*, *143*, (pp. 28-41).