EARLY SEASON CROP LOAD ESTIMATION AND YIELD PREDICTION IN APPLE ORCHARDS

A Thesis in
Agricultural and Biological Engineering
by
Thomas Jarvinen

© 2019 Thomas Jarvinen

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2019
The thesis of Thomas Jarvinen was reviewed and approved* by the following:

Daeun D. Choi  
Assistant Professor of Agricultural and Biological Engineering  
Thesis Advisor

Paul H. Heinemann  
Professor and Department Head of Agricultural and Biological Engineering

James R. Schupp  
Professor of Pomology

Tara A. Baugher  
Extension Educator, Tree Fruit

*Signatures are on file in the Graduate School.
Abstract

In apple production, early season yield predictions have applications in labor demand projections, packing-house inventory management, and sales planning. To make these predictions, fruit counts are a useful metric, but it is expensive to measure these by hand. To automate this process, a computer vision system using a Faster R-CNN object detector was developed which detected immature fruit with a precision of 0.85 and a recall of 0.92, and detected mature fruit with a precision of 0.92 and a recall of 0.82. However, for the purpose of counting fruit even a highly accurate detector will fail to find many apples due to occlusion. It has been proposed that analyzing video streams rather than single images could minimize accuracy loss due to occlusion. To this end, the CNN based object detector was integrated with a video multiple-object tracking algorithm to produce fruit counts for early-season yield prediction. This system was demonstrated to be an accurate fruit counting system which is robust to variability in fruit maturity, and tree structure. Video based fruit counting was found to be a strong predictor of yield, predicting with \( R^2 = 0.81 \) when compared against harvest weight. The resulting yield prediction system was tested on trees with varying canopy depths to find a relationship between canopy density and prediction accuracy. No significant difference was found in prediction accuracy over the different pruning severities. The resulting system has potential applications as an early season yield predictor in a commercial setting.
# Table of Contents

List of Figures vi

List of Tables viii

Acknowledgments x

Chapter 1

Introduction 1

1.1 Goals, Objectives, and Research Questions ............... 4

1.1.1 Project Goal ........................................ 4

1.1.2 Specific Objectives ................................ 4

1.1.3 Research Questions ................................. 4

Chapter 2

Literature Review 6

2.1 Introduction .......................................... 6

2.2 Precision Agriculture ................................ 6

2.3 Horticulture ......................................... 7

2.4 Sensing Approaches for Fruit Detection .................. 8

2.5 Object Detection Using Computer Vision ............... 9

2.5.1 Evaluation of Object Detection Methods ............. 11

2.6 Object Counting in Video Sequences .................... 11

2.6.1 Tracking ........................................... 12

2.7 Related Work in Fruit Detection ....................... 13

Chapter 3

Methodology 17

3.1 Description of Field Sites ............................ 17

3.1.1 Mature Stage Field Sites .......................... 17

3.1.2 Immature Stage Field Sites ....................... 18
3.2 Equipment and Data Collection Procedure ................................. 20
  3.2.1 Mature Fruit Data Collection ........................................ 20
  3.2.2 Immature Fruit Data Collection ...................................... 21
3.3 Fruit Detection ..................................................................... 22
  3.3.1 Mature Apple Detection .................................................. 23
  3.3.2 Immature Apple Detection .............................................. 25
3.4 Apple Counting .................................................................... 26
  3.4.1 Mature Stage Fruit Counting ............................................ 27
  3.4.2 Immature Stage Fruit Counting ........................................ 29
  3.4.3 Combining Video Streams ............................................... 31
  3.4.4 Trunk Positioning .......................................................... 32
  3.4.5 Trunk Partitioning .......................................................... 34
  3.4.6 Trunk Tracking .............................................................. 34
  3.4.7 Depth Filtering .............................................................. 36
  3.4.8 Single-tree Granularity Fruit Counts .................................. 36
3.5 Fruit Size Estimation ............................................................. 36

Chapter 4
Results and Discussion ............................................................. 38
  4.1 Apple Detection ................................................................. 38
  4.2 Trunk Tracking ................................................................. 43
  4.3 Fruit Counting ................................................................. 46
    4.3.1 Mature Fruit Counting .................................................. 46
    4.3.2 Immature Fruit Counting .............................................. 47
    4.3.3 Pruning Severity Analysis ............................................. 49
    4.3.4 Yield Prediction ........................................................... 50
  4.4 Recommendations for Further Research .................................. 51
    4.4.1 Future Applications ...................................................... 52

Chapter 5
Conclusion .................................................................................. 53

Bibliography ............................................................................... 54
List of Figures

3.1 Image of a data tree at the FREC research station. Pink, blue, and red flagging tape identify location of apple clusters. Yellow flagging tape marks the tree’s pruning treatment. 19

3.2 Setup for image acquisition. Three Intel Realsense cameras are mounted perpendicular to the orchard row. 21

3.3 Examples of the three levels of apple occlusion considered. Left: light occlusion, middle: moderate occlusion, and right: heavy occlusion. 24

3.4 Visualization of initialization of detection pipeline for the first two frames of a video sequence. (a) Apples are detected in the first frame, (b) apple positions are estimated for the second frame, (c) a new set of detections are computed for the second frame (shown in red), and (d) the two sets of detections are matched to create a single set of apple positions. For subsequent frames this process is followed using the matched set of apples for translation. 28

3.5 Sample annotated image. The red bounding boxes indicate apple positions. Yellow bounding boxes indicate the positions of trunks. 33

3.6 Sample tree partitioning. Red bounding boxes are trunk detection results, green bounding boxes are apple detection results, and blue lines are tree partitions. 34

4.1 Detection results from four commercial orchard blocks. Each image corresponds to a different cultivar: (a) Braeburn, (b) Goldrush, (c) Gala, (d) Honeycrisp. 40
4.2 Detection results for braeburn dataset when no data augmentation is used during training. Apples in the nearest row are not detected, because no apples of this size were represented in the training data. 41

4.3 Detection result demonstrating the arbitrary bounding box problem. The red boxes are the ground truth label, and the blue box is a detection result which was classified as a false positive during evaluation. During labeling, the bounding box was split into two due to minor occlusion from leaves. 45

4.4 Counting output after 2 frames (left), and 20 frames (right) of the video sequence. In the right image, 24 apples were detected, but use of the counting algorithm led to a total count of 34 apples. 31 of 34 detections are true positives. 47

4.5 Comparison of manual fruit count taken on 07/13/208 on FREC immature dataset against fruit count at harvest. More fruit were counted at harvest for 21 of 24 trees. The red line indicates a perfect fit line between harvest counts and manual counts. 48

4.6 Comparison of percent error rates of video counting over the four pruning treatments. 1: LTR = 1.75. 2: LTR = 1.25. 3: LTR = 0.75. 4: Apogee + LTR = 1.25. 50
List of Tables

3.1 Summary of characteristics of all data collection locations. Locations of data taken from commercial farms have been anonymized. 20

3.2 Distribution of apple occlusion levels for the 180 images in the mature stage dataset. Note that over 50% of all apples are heavily occluded. 25

3.3 Distribution of apple labels in the immature fruit detection dataset. 26

4.1 Apple detection accuracy on immature detection dataset for four commonly used convolutional architectures. 38

4.2 Faster R-CNN model generalization results using images from four commercial orchard blocks. The model is trained using data from FREC immature dataset. 39

4.3 A comparison of Faster R-CNN model generalization across datasets. “Imm” denotes immature validation set, and “Ma” denotes mature validation set. 42

4.4 Recall of Faster R-CNN over varying levels of fruit occlusion. 43

4.5 Detection Results of mature fruit video counting algorithm on 30 frame video sequences. 47
4.6 Summary of fruit counting accuracy for the FREC dataset and each pruning severity level. RMSE is calculated in terms of fruit counted using the video counting algorithm as compared to fruit count at harvest. ......................................................... 49

4.7 Model summary for linear regressions comparing fruit counting methods to harvest fruit counts. ................................................................. 51

4.8 Model summary for linear regressions comparing fruit counting methods to harvest fruit weight on an individual tree basis. ............ 51
Acknowledgments

First I would like to express my deep gratitude to Dr. Daeun Choi for her mentorship, constant support, and substantial contribution to all aspects of this study. In addition, thanks go to Dr. James Schupp for his generous contribution of time, resources, and expertise. Without his help, this project would not have been possible. Likewise, Dr. Tara Baugher’s guidance, and assistance in coordinating with commercial growers is greatly appreciated. I would also like to thank her for inspiring and cultivating my interest in agriculture and nurturing my growth as a scholar. I would also like to thank Dr. Paul Heinemann for his keen insight and advice. Thank you to the Penn State Department of Agricultural and Biological Engineering, and the State Horticultural Association of Pennsylvania for their financial support. Finally, I would like to thank my family and friends for supporting me throughout my graduate studies.
Chapter 1  
Introduction

In tree fruit production, decreasing availability of skilled workers and rising production costs [Lampinen et al., 2012] are driving an urgent need for labor-saving technologies. Tree training, thinning, pruning, orchard scouting, and harvest are still predominantly done manually in apple production, making the endeavor highly vulnerable to disruptions in the labor supply. A survey of specialty crop growers found that the areas of highest perceived need for advances in automation and sensing technologies were harvest and thinning [Ellis et al., 2010] — two of the most labor-intensive tasks in tree-fruit production.

From an engineering standpoint, there are two major approaches to alleviating the labor shortage. Mechanical systems, such as orchard platforms and robotic harvest systems, address the shortage directly by reducing labor inputs [Baugher et al., 2009]. The second approach is to develop technologies to support advances in on-farm decision making. In particular, there is an ongoing push in the agricultural community to improve the practice of precision agriculture (PA), which can be defined as “that kind of agriculture that increases the number of (correct) decisions per unit area of land per unit time with associated net benefits”[McBratney et al., 2005]. In practice, this often involves taking a site-specific approach to agricultural management where practices vary to account for local variations in conditions. The advancement of this approach to agricultural production requires bringing more information to growers. Manual investigation of such information — various farming variables such as nutrition, soil moisture, crop health, disease infection, etc. — is often a time-consuming and repetitive task, so automating data acquisition using remote sensing technologies is desirable. Computer vision (CV) is one such class of remote sensing technology which is used
to automatically derive useful information from image inputs. Systems based on CV could address the challenges associated with variability of orchards by extracting usable abstractions about environmental and botanical irregularities directly from the environment. Such CV orchard surveying techniques bear further significance as a necessary precursor to effective robotic orchard systems for automating labor intensive tasks.

Yield prediction is a promising research area which will help growers to plan farming operations, prepare inventory, estimate and schedule labor needs and efficiently allocate labor. Furthermore, forecasts are used by packing houses, wholesalers, and retailers for planning sales, setting prices, and crop storage management. It is desirable to predict yield as early in the season as possible. In particular, both industry groups and the United States Department of Agriculture (USDA) publish their first yield forecasts in early July — making this the de facto target date for producing accurate yield predictions. Predictions can be taken one step further by creating yield prediction maps which track fruit on a tree-by-tree basis. Repeated application of yield prediction mapping throughout a growing season could be used to assess fruit growth rates, and inform “spatially targeted agronomic measures” [Payne et al., 2013] such as precision nutrient applications.

Most approaches to yield forecasting proceed from the premise that the number of fruit on trees is a reliable early indicator of yield. The general approach is to detect and count fruit on trees using CV and use the resulting crop load estimation as an input to a secondary model (e.g., linear regression) for yield prediction. Occlusion of fruit, inconsistent lighting, and color similarity between fruit and canopy, among other factors, are challenging problems which lead to inaccuracies in existing fruit detection algorithms. Cumbersome methods such as controlling lighting conditions, manual block-by-block calibration, or augmenting image data with additional block information are often used to alleviate these issues [Zhou et al., 2012, Zhang et al., 2015]. An ideal system would have minimal overhead in terms of labor and capital inputs, and be suitable to large-scale adoption and automation.

In other lines of research, advances in deep learning have improved the state-of-the-art in object detection and localization [Krizhevsky et al., 2012, Girshick, 2015, Shelhamer et al., 2016]. Deep learning refers to a class of high-dimensional nonlinear curve fitting schemes based on machine learning. These models use optimization
to fit a high number of parameters to ground truth, referred to as training data, for some complex tasks such as object detection. They are called “deep” because the models have a multi-layered hierarchical structure which enables high-level abstractions from data.

In specialty crop production, data are often complex and noisy, so it is useful to extract these high level abstractions rather than use hand-engineered features directly on input data. Until recently, the large volume of training data necessary for deep learning was prohibitive for its adoption to CV applications in specialty crops. However, advances in transfer learning (using a model trained on one data set as a starting point for a model in a new domain) have shown deep learning to be effective with much less data [Marmanis et al., 2015]. This methodology has been shown to be effective in a wide range of agricultural applications [Sa et al., 2016, Bargoti and Underwood, 2017b, Liu et al., 2018].

A properly constructed deep learning approach to fruit detection could provide an accurate model which inherently accounts for shape, color, and texture of fruit under variable light conditions. However, occlusion of fruit by the tree canopy is still a limiting factor. To address this, Tabb et al. [Tabb et al., 2006] proposed using video sequences of fruit canopy from a moving platform to detect fruit from a range of perspectives. In a related work, Roy and Isler [Roy and Isler, 2016] developed a structure-from-motion pipeline for fruit registration in video sequences, but adequate validation of their method has yet to be published. The integration of deep learning for fruit detection with video sequence tracking to ameliorate the occlusion problem is a promising, underexplored approach to yield prediction mapping. Some specific research gaps in the existing literature include inadequate collection of ground truth data of on-farm conditions, analysis of the effects of horticultural practices (such as pruning) on system performance, and using sensing techniques for the detection of immature fruit.

The key goal of this study was to develop an early-season fruit detection algorithm and integrate it with a video-based crop load estimation pipeline. Video taken from two experimental apple orchards (Russell E. Larson Agricultural Research Center, Pennsylvania Furnace, PA and Fruit Research and Extension Center, Biglerville, PA) was collected to train and validate a deep learning system for detection of both mature and immature apple fruit. Fruit tracking, and counting was performed on video sequences of tree canopies in order to produce a fruit count.
A method of trunk tracking and detection was developed to partition counting results into a single tree granularity. Detection results were compared against actual yield data in order to test suitability of the developed algorithm for application in yield prediction. In addition, the fruit counting methodology was tested for several commercial orchards. Finally, the effect of various levels of tree pruning severity on fruit counting accuracy was evaluated.

1.1 Goals, Objectives, and Research Questions

1.1.1 Project Goal

The goal of this project is to develop a system to detect, count, and estimate the size of immature apple fruit on trees to produce early-season yield predictions for apple orchards.

1.1.2 Specific Objectives

The specific objectives of this research are to:

1. Develop a method for detection of apple fruit on trees at various levels of maturity and varying light conditions.

2. Develop a fruit counting system built on a CNN fruit detector, and a multiple object tracking algorithm in order to count unique fruit detections on video sequences.

3. Evaluate the resulting counting system in the field by measuring the performance on apple trees of various canopy depths and varying levels of fruit maturity by comparing estimated yields by the developed algorithm to actual yield measured at harvest.

1.1.3 Research Questions

In addition to the objectives in 3.2, this study aims to investigate several specific research questions.
1. Can a CNN using a limited domain specific dataset alongside transfer learning effectively detect apples on trees throughout the growing season?

2. Does CNN-based apple detection generalize across cultivars and tree architectures?

3. Does final fruit count constitute a good predictor for yield?

4. Does analyzing video sequences for fruit detection effectively minimize accuracy loss due to occlusion?

5. What is the relationship between canopy depth and fruit detection accuracy?
Chapter 2  
Literature Review

2.1 Introduction

This review will introduce the concepts underlying the design decisions addressed in this work. Relevant background in precision agriculture, horticulture, sensing tools for data collection, and approaches to object detection, and video tracking methodologies will be discussed. Finally, recent related works in fruit detection, crop load estimation, and yield prediction will be surveyed.

2.2 Precision Agriculture

The goal of precision agriculture (PA) is the development of effective decision-making systems which can account for both spatial and temporal variation of on-farm conditions in order to increase output, drive down costs, and reduce the environmental footprint of agricultural production. There are two key elements to achieving this — improvements in information gathering capabilities, and development of techniques for processing and utilizing such information. These developments often go hand-in-hand. For instance, precision irrigation techniques require both tools for monitoring water stress, and the ability to control how much water is being applied through a given irrigation system. The result of such a system reflects what precision agriculture aims to accomplish — utilizing fewer resources to get the same or better result.

The field of PA originated in the mid 1970s from the insight that many row crops had high levels of variability within fields [Robert, 2004]. Since then, a wide range
of agricultural products have benefited from numerous PA advances. For instance, yield mapping, crop disease detection, and weed detection technologies are readily available for a number of crops [Robert, 2004]. However, development and adoption of such applications has been slow for the tree fruit industry. One driving reason behind the pursuit of this project is the exploration of the capabilities emerging CV technologies in the context of overcoming challenges which have prevented adoption of PA practices in the past.

2.3 Horticulture

Generally, approaches to yield prediction in tree fruit begin with the hypothesis that the number of fruit present on trees is a good indicator for yield. In apples, fruit count fluctuates early in the season. Final fruit count is generally reached in late June, following a period known as June drop, when fruit often fall from trees. Furthermore, for yield prediction applications which use fruit size as a feature, the period following June drop is desirable because early season factors have a disproportionate effect on final fruit size. Given sufficient resource availability, apple fruit follow an expolinear growth pattern [Lasko et al., 1995] in which fruit grow exponentially during a cell-division phase for approximately 3-5 weeks after bloom. Following this, fruit growth (in terms of weight) is mostly linear over time, with a rate proportional to the number of cells produced during the exponential phase. In principle, a few robust measurements of fruit growth during the linear growth phase could thus be used as strong predictors of final fruit size. Finally, it is worth noting that the push for automated fruit counting is motivated by the fact that manual fruit counts are both highly error-prone, and prohibitively labor intensive to collect.

An additional consideration of interest is the interplay between orchard design and efficacy of CV systems. In an orchard setting, automated detection of fruit is challenging due to the inherent complexity and variability of biological systems. Fruit size and color vary widely between cultivars and throughout the season. The architecture of the tree canopies varies with tree age and training approach, leading to various levels of occlusion of fruit. Finally, orchard environments are often subject to changing inconsistent lighting conditions, which make the consistent acquisition of high quality data challenging. Though these factors certainly have
an effect on the accuracy of computer vision systems, they are rarely addressed directly in the literature.

Variation in canopy density or depth is of particular interest, because it is hypothesized to be a large factor in how likely fruit are to be occluded by the canopy. This variation between canopies is difficult to quantify. A recently proposed method [Schupp et al., 2017] uses the limb to trunk ratio (LTR) to quantify pruning severity in terms of limb cross-sectional area (LCSA) and trunk cross-sectional area (TCSA) in mature tall spindle trees (Eq 2.1). Alternately, it is possible to try to infer canopy characteristics directly from image inputs. For instance, one approach used color thresholding to classify image pixels as either fruit ($F_A$) or foliage ($L_A$). Then, the ratio $L_A/I_A$ was used as a feature for estimating yield [Cheng et al., 2017]. Taking this approach further, systems with access to reliable depth data can estimate canopy volume directly. For instance, Tumbo et al. [Tumbo et al., 2002] demonstrated two methods for accurately measuring canopy volume using ultrasonic and laser measurements. Understanding the effect of orchard design on applicability of computer vision systems is a crucial research area for preparing automated crop monitoring techniques for commercialization.

$$LTR = \sum_{i=0}^{n} \frac{LCSA_i}{TCSA}$$

(2.1)

### 2.4 Sensing Approaches for Fruit Detection

A wide range of sensors have potential for applications in fruit detection. The earliest studies used simple black and white cameras, which mainly open up the use of shape and texture information to identify fruit [Gongal et al., 2015]. Today, color cameras often fill this role, being cheap, universally available, portable, configurable, and available at high resolutions. Furthermore, comparing fruit color to that of the canopy can be an effective way to identify fruit in of itself [Zhou et al., 2012].

To create more robust detection systems, it is often desirable to gather additional geometric or spectral data. Stereo systems use two cameras mounted with a fixed distance between them. Using multiple frames of reference, strong geometric inferences can be made to reconstruct 3D information (3D point clouds or depth
data) from 2D images. Time-of-flight sensing, which measures the distance between objects and the camera directly, is another approach to gather this geometric information. Most available depth sensors are very sensitive to lighting conditions. This renders them difficult to use outdoors unless lighting is controlled directly (for example by taking night images). As an alternative, light detection and ranging (LIDAR) measures the reflected pulses of a laser to generate high-fidelity depth information [Medeiros et al., 2016]. Finally, Multi-spectral and hyper-spectral sensors gather additional information by measuring a wider range of the light spectrum. Analyzing a wider reflectivity profile of fruit can give vital information for image segmentation [Yang et al., 2014]. However, these sensors are often extremely sensitive to illumination, which makes them very difficult to use in the field.

Choosing a sensor constitutes a fundamental trade-off between robustness and practicality. In an orchard setting, consideration must be given to both cost and availability. In particular, it is highly desirable that a system for use in orchards use off-the-shelf components so equipment failure can be quickly and easily addressed; the ability to do their own repairs is a major concern for specialty crop growers considering adoption of a new technology [Lampinen et al., 2012]. In this light, the ubiquity of color cameras make them highly appealing to use for orchard sensing. Conversely, LIDAR and hyper-spectral cameras are less frequently used in the consumer sphere, making them less desirable.

2.5 Object Detection Using Computer Vision

Object detection is a fundamental problem in estimating crop load with CV. It can be defined as the task of detecting and locating each instance of a given class within an image. In general, the idea behind object detection is to define some feature or set of features which distinguishes between objects of a given class and all other objects present in the image. In CV, there are two basic classes of approaches for defining these features: engineered features and learned features (note these approaches are not mutually exclusive). An engineered feature is generally a mathematical description of some property of the object of interest defined by a programmer (e.g. color, shape, position, or texture). A learned feature on the other hand is derived directly from a set of sample solutions to the problem called training data. A full solution to the object detection problem generally uses some
set of these features as input to a classifier in order to detect objects in an image [Amit and Felzenszwalb, 2018].

In this work, deep learning methods are used for object detection. Deep learning refers to a branch of machine learning concerned with the use of many-layered hierarchical models for solving complex tasks. In CV, this generally means convolutional neural networks (CNN). In simple terms, a CNN model is a class of machine learning algorithm which is developed by fitting a set of parameters to a ground truth of labeled data using optimization techniques. This fitting can effectively create hierarchical representations of image features; i.e. “pixels are assembled into edgelets, edgelets into motifs, motifs into parts, parts into objects, and objects into scenes” [LeCun et al., 2010]. In this way, CNNs can be used to develop a set of learned features for the object detection problem. To develop these abstractions, CNNs are generally vastly complex, fitting many millions of parameters using datasets with millions of images.

However, it has been shown that CNNs trained on one object classification task can be effectively adapted to new domains with a small set of training data in a process known as transfer learning [Sharif et al., 2014, Bengio, 2011, Li et al., 2016]. Transfer learning is based on the idea that the fundamental operations in object detection bear similarity regardless of problem domain. Thus, a model which is pre-trained for a general model only needs to be fine-tuned to a specific problem domain, greatly simplifying the learning task. This is desirable because CNNs are the highest performing models on a wide variety CV tasks. This paradigm began when the huge potential for deep learning in object recognition tasks was demonstrated through the development of a stable 8-layered, CNN (referred to as AlexNet in honor of the author) [Krizhevsky et al., 2012]. Prior to this paper, deep CNNs had problems with overfitting, training time, and fundamental instability. The authors put forth a set of training procedures, and data augmentation techniques to stabilize training and minimize overfitting. They also implemented their training on GPUs — a highly parallelized type of processor — to dramatically speed up training.

One of the principal distinguishing factors between different CNN models is architecture. CNNs are built by stacking layers of various building blocks. In theory, there are innumerable ways to assemble these building blocks, but in practice several notable architectures have proven to perform well on a variety of tasks. Popular architectures include ‘AlexNet’ [Krizhevsky et al., 2012], ‘ResNet’ [He et al., 2016],


2.5.1 Evaluation of Object Detection Methods

When evaluating the efficacy of object detection methods, it is common practice to measure accuracy using a set of manual annotations as ground truth. Object positions are generally labeled using simple polygons, such as rectangular bounding boxes to indicate their position. In this study, an object detection is considered a true positive if it overlaps enough with a ground truth bounding box, using the intersection of the bounding box areas divided by the union between the bounding box areas as a metric. Accuracy is reported using precision, recall, and F1 score as metrics. Precision is the number of true positive detections divided by the total number of detections. Recall is the number of true positive detections divided the total number of object labels. Finally, F1 score is the harmonic mean of precision and recall (Eq. 2.2).

\[
F1 = \frac{2 \cdot (\text{precision} \cdot \text{recall})}{(\text{precision} + \text{recall})}
\]  

(2.2)

2.6 Object Counting in Video Sequences

In this work, crop load will be estimated by counting fruit detected across video sequences. To do this, a method to ensure each detected fruit is counted exactly once is necessary. One way to think of this is as a tracking problem. If each apple is tracked from the time it enters the camera frame to the time it leaves the frame,
the apples can be assigned unique identifications. Thus, fruit counting can be considered a special case of the multiple object tracking (MOT) problem. A general definition of the MOT problem is given in [Luo et al., 2014]. This definition is repeated here as the notation will be used throughout this paper. Let $s^i_t$ denote the state of the $i$-th object in the $t$-th frame of a video sequence. Then, the state of all known objects in a given frame can be defined by the vector $S_t = (s^1_t, s^2_t, \ldots, s^M_t)$. To denote an object's state for a sequence of frames, we will use $s_{i_s:i_e} = \{s^i_{i_s}, \ldots, s^i_{i_e}\}$. In this case, $s$ is an object's position, $i$ is the object's ID, and $i_s$ and $i_e$ are respectively the first and last frames in which the object $i$ is present in the video sequences. In addition, the set of observations $O$ associated with a frame $t$ is given by $O_t$. An observation is simply one object detection. Finally, the state of all objects over all frames of the video sequences is denoted by $S_{1:t} = \{S_1, S_2, \ldots, S_t\}$. Given this notation, a general definition of the MOT tracking problem is to optimize the set of states $S_{1:t}$ over the set of observations. A tracking-by-detection approach is used, which means that the observation set is simply the set of all object detections over all frames of the video sequence.

Among MOT approaches, one important distinction is between the online and offline tracking problems. In online tracking, frames are processed strictly sequentially, one pair of frames at a time. In this case, the MOT problem is concerned with optimizing $S_{1:n} = \arg \max P(S_{1:n} | S_{1:n-1}, O_{n-1:n})$. That is, in the current frame $n$, object tracklets are formed strictly based on known tracklets formed in the proceeding frames and any observations from the most recent pair of frames. This approach is generally necessary for real-time applications and is often used in non real-time applications when processing time is a bottleneck, because it is generally cheaper computationally to limit processing to two frames at a time.

### 2.6.1 Tracking

To perform tracking successfully, it is necessary to first form correspondences between successive frames in a video sequence. There are two classes of approaches to this problem — global and local. The global approach uses a set of matching features between images to fit a single transformation function which defines the translation between images at all pixels in both images [Hartley and Zisserman, 2003]. This approach can be appealing because it has strong theoretical geometric foundations.
Unfortunately, these transformations are difficult to compute accurately under complex conditions such as those found in outdoor orchard imagery. The local approach, called optical flow (OF) calculates the translation between images at the pixel level. That is, for a given pixel in frame $t$, OF computes an estimate of that pixel’s position in frame $t + 1$. OF is a widely studied field, and several robust methods were found suitable for tracking in orchard conditions.

### 2.7 Related Work in Fruit Detection

In the existing literature, color factors are a common feature used to distinguish between fruit and background foliage. A straightforward approach is to create empirical thresholds in each of the R, G, and B portions of the color spectrum to detect fruit [Zhou et al., 2012, Roy et al., 2015, Cheng et al., 2017]. Color can also be modeled as a probability distribution, as in the global mixture of Gaussians method, which segments images by developing a set of Gaussian distributions for the color ranges of fruit in ground truth images, and uses a threshold of standard deviations on those distributions to segment fruit [Tabb et al., 2006]. In the case of early season detection, it is impractical to depend entirely on color information to distinguish between fruit and canopy because fruit are still quite green. Empirical color thresholding requires extensive manual calibration that varies by cultivar and time of year. It can also be highly sensitive to variations in lighting conditions.

Shape and texture information are often used instead of or in addition to color. For instance, Circular Hough transforms (CHT) are a commonly used method to detect circles in images [Hung et al., 2015, Choi et al., 2016, Bargoti and Underwood, 2017b]. Since fruit tend to be circular this can be an effective way to isolate fruit, at least when there is little occlusion. The efficacy of this approach was demonstrated in oranges under a range of light conditions by normalizing light intensity [Choi et al., 2016]. Another method [Linker et al., 2012], detected apple pixels with high-confidence apple pixels color thresholds and smoothness measures. Then segmentation (pixel-wise classification) was performed by modeling apple area as a circle. This latter method boasts a 95% detection accuracy for mature apples under ideal lighting conditions, but has many false positives in less ideal conditions.

Hyperspectral imaging has been shown to be very effective for fruit detection in blueberries achieving 97.9% classification accuracy for young blueberries. However,
this classification was done on fruit in a laboratory environment, which is much less challenging than an outdoor environment [Yang et al., 2012]. They used principal component analysis (PCA) to extract key features from the hyperspectral data to be used as inputs for classification trees and multinomial logistic regression. In another work, hyperspectral imaging achieved an 88% detection accuracy in apples using PCA, a hand-tuned normalization method, and a spectral and spatial analysis technique [Safren et al., 2007]. This result is notable because it was achieved without controlling lighting conditions.

To address this latter issue, an approach that used video sequences rather than still images to count the number of fruit on pepper plants was developed [Song et al., 2014]. First, they detected the number of fruit in each image using a bag-of-word model. To do this, they ran a sliding window over points of interest (POI) in each image, where points of interest were identified using a Naive Bayes classifier. Then for a region around each POI (10000 per image), color and texture features were extracted. The resulting features were discretized into 1000 clusters with K-means clustering. These cluster labels were then input into a support vector machine classifier to identify a region as fruit or non-fruit. Finally, an ad hoc statistical approach was developed to match detected fruit across images in the video sequence. They finally achieved a correlation of 74.2% between manual and automatic fruit counts. The application’s proposed use was plant phenotyping.

One alternative approach to video segmentation is to use semi-supervised learning approach to pixel classification [Hung et al., 2015]. In this approach, instead of manually defining features of interest in each image, feature learning is used to automate feature extraction. Here, they built a set of sparse auto-encoders at several scales in order to represent image regions in low dimensionality. Then, the outputs of these auto-encoders were used as input features to a logistic regression classifier to label the output of these multi-scale responses. Finally, the image was modeled as a conditional random field, in order to define local relationships between pixel labels. Fruit counting was performed using images of apple trees taken from an unmanned ground vehicle. They were able to classify apple pixels with a 93.3% accuracy.

Since the publication of the seminal ‘AlexNet’ paper [Krizhevsky et al., 2012] much of the attention of the computer vision community has been focused on deep learning methods for a range of tasks. However, it was not until 2016 that the
first application of deep convolutional neural networks (CNN) was brought to fruit monitoring [Sa et al., 2016]. Sa et al. demonstrated the applicability of CNNs to fruit detection by training a Faster R-CNN object detector using transfer learning to detect sweet peppers. They combined data from both RGB and near infrared (NIR) spectra to detect sweet peppers with an F1 score (accuracy) of 0.8338. Finally, they tested the generalization of this approach by training models for 7 other types of fruit using color images acquired from Google Images. They demonstrated F1 scores around 0.9 for most of these fruits under good lighting conditions. A similar work demonstrated the use of Faster R-CNN for detection of apples, mangoes, and almonds in orchard conditions [Bargoti and Underwood, 2017b]. They were able to achieve F1 scores of 0.904, 0.908, and 0.775 respectively. Furthermore, they tested a number of data augmentation schemes and found that rescaling and flipping images were the most effective data augmentation techniques for training. Finally they also demonstrated model accuracy to asymptote around 200 images for all 3 use cases.

CNNs can also be used for image segmentation instead of object detection. Image segmentation refers to the set of methods which give class labels to each pixel in an image. Bargoti and Underwood [Bargoti and Underwood, 2017a] used a fully-convolutional neural network [Shelhamer et al., 2016] to classify apple pixels in a tree canopy using color images. Furthermore, they were able to increase naive accuracy by formulating an approach to utilizing metadata during training and inference. In a typical CNN, the only inputs to the fully connected output layers were feature maps which were created by the convolutional layers. However, it is possible to simply add more inputs to the fully connected layer to represent additional features of interest. In this case, the authors encoded meta-parameters such as sun position and tree type as integer inputs to the fully connected layers. This augmentation increased pixel F1 score from 0.791 to 0.797. Finally, the authors tested both watershed segmentation and circular Hough transforms to translate pixel segmentations to object detections which could be used to count fruit. The final model predicted fruit count with a $r^2 = 0.85$.

Liu et al. [Liu et al., 2018] combined deep learning object detection with an optical flow based tracking method to count fruit along an entire orchard row for both apples and oranges. First, they detected apples in much the same way as the previous paragraph (without metadata). Then, optical flow was used to estimate
pixel translation between each pair of images in the video. This estimate was input to a Kalman Filters to create a noise tolerant fruit tracker. Then, for each pair of images, fruit are compared and matched. Estimated position and color histograms are used as features to compute matching cost, then apples are mapped from one frame to the next by generating a complete bipartite graph between the sets of apple detections. An optimal matching scheme was computed using the Hungarian algorithm. A structure-from-motion pipeline was used to generate a 3D of the apple positions and to estimate the size of each apple. Finally, outlying apples in terms of size were discarded from the fruit count. This method achieved a 97% accuracy when compared to manual count of fruit in the video sequence.

Throughout the surveyed literature, there are several key research gaps this work aims to address. First, most existing works exclusively rely on image annotations to provide ground truth for evaluating CV methodologies. This work will use manual measurements of field conditions in addition to image annotations to provide ground truth data. Secondly, the impact of horticultural practice on CV system performance has rarely been addressed. This work will provide an analysis of how pruning practices affect accuracy. Thirdly, to the author’s knowledge, there exists no analysis on how well deep learning object detectors can be made to generalize to variations in tree cultivar and fruit maturity.
Chapter 3  Methodology

The study has been separated into two distinct stages — the mature fruit stage, and the immature fruit stage. In the mature fruit stage, a variety of models were tested using video sequences taken of apple canopies during October 2017. These models were used to develop a set of procedures for detection and counting of mature fruit on trees. During the immature fruit stage, this approach was adapted and refined for application to a juvenile fruit dataset collected during June, July, and August 2018. In the following chapter, data collection, algorithm development, and the final fruit counting methodology will be introduced for both the mature and immature fruit processing stages.

3.1 Description of Field Sites

3.1.1 Mature Stage Field Sites

For the mature stage, a set of videos of tree fruit canopies were collected from the Russell E. Larson Agricultural Research Center (ARC), in Pennsylvania Furnace, PA and the Fruit Research and Extension Center (FREC), in Biglerville, PA. These data were collected on mature and nearly mature apple fruit. At Rock Springs, images were recorded on September 26, September 27, October 6, 2017 under both overcast and sunny lighting conditions. At FREC, images were recorded on October 18, 2017 under bright sunny light conditions. The dataset was largely non-discriminatory in regard to choice of trees to be included; a wide range of cultivars were included (including red, green, and yellow varieties), at varying tree sizes, with several different tree architectures. As such, these data will be used for
algorithm prototyping, and training, but not for final validation of yield prediction.

### 3.1.2 Immature Stage Field Sites

For the immature stage, data from several separate sources was collected throughout the 2018 growing season. The most detailed dataset was collected at FREC, for a set of 24 trees trained to varying levels of pruning severity. In addition, measurements were taken at four commercial orchard blocks.

At the FREC site, tall spindle ‘Brak Fuji’/M.9 trees were selected. They are planted with a 91 cm (3 ft.) in-row spacing, with 356 cm (12 ft.) between row spacing. Three levels of pruning severity based upon the LTR pruning severity index for apple trees were adopted for the tests, according to recently published guidelines (Schupp et al., 2017). The LTR is calculated from the sum of the cross-sectional area (LCSA) of all branches on a tree at 2.5 cm from their union to the central leader divided by the cross-sectional area of its trunk (TCSA) at 30 cm height from the graft union (Equation 2.1). The LTR metric provides a measurable way to define the maximum allowable branch diameter (MD) to create different levels of pruning severity. Removing the largest branches in succession down to MD achieves consistent outcomes and allows a greater degree of accuracy and precision for dormant pruning of tall spindle apple trees.

Diameter of all limbs of four representative trees were measured and the LCSA calculated. The trunk was measured at 30 cm height from the graft union, and TCSA calculated. The sum of the LCSA was divided by the TCSA and the resulting LTR regression line was used to develop a maximum allowable limb diameter (MD) for each level of LTR.

Pruning severity treatments were applied by pruning each tree using renewal cuts to remove all the branches larger than MD for each treatment. Trees were pruned to LTR 1.75 (lightly pruned; 1.25 LTR (moderately pruned); or 0.75 (heavily pruned). For each level of pruning severity, three-tree plots were set up with the center tree as the data tree, but with the adjacent tree to each side receiving the same canopy treatment. Also, an additional tree plot of moderately-pruned trees treated with the plant growth regulator prohexadione calcium, (PCa) (kudos™, Fine Amercas, Inc., Walnut Creek, CA) was included. PCa was applied 3 times in the season by handgun sprayer at 187 mg · L⁻¹. Buffer trees separated PCa plots
from plots not receiving this treatment. PCa treatment timings were petal fall, 4, and 6 weeks after petal fall.

The plots were arranged as a randomized complete block design with blocking by tree size (TCSA) and six replications. There was quite a bit of variability in tree size, (TCSA class 6 =23 cm$^2$; TCSA class 1 =39.2 cm$^2$), with the biggest class about 70% larger than the smallest. The randomized complete block design was used to help manage natural variance due to variation in tree size.

Each data tree was marked with colored flagging tape indicating the treatment it belonged to. In addition, three fruit clusters were identified for each data tree so fruit growth could be tracked. Clusters were manually selected with preference for those having more than one fruit and those that were reachable from the ground. Colored flagging tape was tied to the trunk side of the branch belonging to each cluster (Figure 3.1). For some clusters, all fruit abscised between measurements. In this case, new clusters were selected later in the season.

![Figure 3.1.](image)

**Figure 3.1.** Image of a data tree at the FREC research station. Pink, blue, and red flagging tape identify location of apple clusters. Yellow flagging tape marks the tree’s pruning treatment.

An additional set of field sites were selected for evaluating the effectiveness of the method within four commercial orchards in southern Pennsylvania. For these
sites, host growers selected representative high-density tree blocks. For each block visited, a set of 25 trees were selected for video recording. Table 3.1 summarizes all data collection locations.

Table 3.1. Summary of characteristics of all data collection locations. Locations of data taken from commercial farms have been anonymized.

<table>
<thead>
<tr>
<th>Location</th>
<th>Dates</th>
<th>Cultivar</th>
<th>Rootstock</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRECa</td>
<td>10/18/2017</td>
<td>Various</td>
<td>Various</td>
</tr>
<tr>
<td>ARC</td>
<td>09/2017-10/2017</td>
<td>Various</td>
<td>Various</td>
</tr>
<tr>
<td>FRECb</td>
<td>06/2018-08/2018</td>
<td>Brak Fuji</td>
<td>M.9</td>
</tr>
<tr>
<td>Grower 1</td>
<td>08/07/2018</td>
<td>Braeburn</td>
<td>B.9</td>
</tr>
<tr>
<td>Grower 1</td>
<td>08/07/2018</td>
<td>Goldrush</td>
<td>B.9</td>
</tr>
<tr>
<td>Grower 2</td>
<td>08/07/2018</td>
<td>Premier Honeycrisp</td>
<td>M.9 337</td>
</tr>
<tr>
<td>Grower 3</td>
<td>08/07/2018</td>
<td>Gala</td>
<td>B.9</td>
</tr>
</tbody>
</table>

3.2 Equipment and Data Collection Procedure

Two distinct data collection procedures were developed for the two stages of this study. For immature data collection, the data collection process was improved — specifically in order to gain cover of the full fruit canopy during collection.

3.2.1 Mature Fruit Data Collection

A Kinect 2.0 sensor (Microsoft Corporation, Redmond, WA) was used for capturing depth, infrared, and RGB data of apple trees. The Kinect camera records at 30 frames per second with an image resolution of 1920x1080 for color images, and 512x424 for the depth channel. This camera was chosen specifically because it is a readily available sensor with depth sensing capabilities. The sensor was mounted via tripod on the back of a gasoline-powered utility ground-vehicle. Video of tree canopies was recorded with the vehicle driving at approximately 3 km/h parallel to the fruit canopy. A Windows laptop was used to drive the sensor, and it was powered using a 12 volt portable battery.
3.2.2 Immature Fruit Data Collection

During preliminary data collection, a camera was used at just one height, and thus only part of the tree canopy was captured. To accurately count fruit, it is necessary to capture video covering the full tree canopy. To achieve this, three cameras were used to record simultaneously. A vertical camera mount was fabricated to mount on the back of a utility vehicle (Figure 3.2). The mounts were designed to have adjustable camera heights, so that the intended field of view could be easily adjusted based on tree size and distance from the tree canopy. In their default configuration, the cameras were mounted 76 cm (30") apart at heights of 114 cm (45"), 178 cm (70"), and 254 cm (100") respectively. Under this configuration, the total camera field of view covered an approximate range of heights from 45 cm (18") to 381 cm (150") at a distance of one meter from the canopy, which was sufficient to cover nearly all fruit at the FREC location. The cameras’ respective fields of view overlapped by approximately 50 cm when the camera was 1 m from the canopy, which was sufficient to allow for inter-camera frame alignment.

Figure 3.2. Setup for image acquisition. Three Intel Realsense cameras are mounted perpendicular to the orchard row.
For this application, the Kinect 2.0 sensor was deficient for two reasons. Firstly, the Kinect sensor was designed for indoor applications, so its depth sensor exhibited high levels of salt-and-pepper noise in images. Secondly, the Kinect has poor support for multi-camera sensing. To deal with these issues, three Intel Realsense D435 cameras were used. These cameras support depth, color, and NIR imaging like the Kinect, but have fewer issues with noise in the depth channel, and provide better user support for managing camera parameters. One downside with the Realsense cameras is lower image resolution. The Kinect uses 1920x1080 image formats, whereas the Realsense camera only supports a resolution of 1280x720 pixels. Furthermore, depth and NIR images were recorded at a resolution of 848x480 pixels due to data throughput issues associated with recorded data from three cameras simultaneously.

For the field site at FREC, images were collected on May 7, May 23, June 4, June 20, June 29, July 5, July 13, and August 6, 2018. In addition, fruit were counted by hand and fruit measurements were taken on May 23, June 4, June 22, June 29, and July 13 2018. Finally, fruit were harvested, counted, and weighed on October 18, 2018.

The four commercial blocks were visited on August 7, 2018. For each block, 25 trees were selected. Video was recorded for both sides of the canopy. In addition, manual fruit counts were collected for each tree.

### 3.3 Fruit Detection

There are two key challenges in object detection. Each object in an image must be correctly identified, and its boundaries accurately delineated. To achieve these tasks, a convolutional neural network, Faster R-CNN, was trained. Faster R-CNN uses two jointly trained output layers that have shared convolutional layers. The convolutional layers can be thought of as feature extractors. The first output layer, called the region proposal network, takes convolutional features as inputs and outputs regions of interest that may contain objects. The second output layer is an object classifier, which takes regions of interest as input and classifies them [Ren et al., 2017]. An open source implementation of this model programmed in Tensorflow was used [Chen et al., 2017]. For the convolutional layers of the network, several commonly used architectures were tested including ResNet50, ResNet101 [Xie et al., 2016],
Network parameters were initialized using models pre-trained on ImageNet.
Training was performed using stochastic gradient descent via back-propagation with dropout [Srivastava et al., 2014] for regularization, as demonstrated by Krizhevsky et al.[Krizhevsky et al., 2012]. In a given training iteration, loss is either calculated with respect to the classifier or the region proposal network in an alternating fashion. When the classifier is used, ground truth bounding boxes are given as its input, rather than the region proposal network's output, so that it can be trained on true positive data.

Development of a robust dataset is a key step when developing a CNN model. For this application, a dataset consists of a set of images, and corresponding annotations which define the position of each apple in the image via a rectangular bounding box. This dataset is then divided into training set, validation, and testing sets. The training set is used for learning the parameters of the deep learning model. The testing sets used for model evaluation during the development process.
Finally, the validation set (40 images), is reserved for evaluation of the final model and reporting of results. This latter set is necessary because the model has a set of hyperparameters which must be defined prior to the training process. During model development, training is performed repeatedly as hyperparameters are tuned and evaluated on the testing set. This practice can lead to overfitting to the testing set, so the validation set is held separate for evaluating final results.

### 3.3.1 Mature Apple Detection

To build the mature fruit detection dataset, 181 images were extracted from the video taken at FREC and Rock Springs in the 2017 growing season. Images were selected to produce a curated dataset which included variability in lighting conditions, number of apples visible, cultivars represented, and image quality. Images were annotated using Sloth [Slo, 2013], an open source image annotation tool. Each apple visible in the image was annotated using a rectangular bounding box and a label defining severity of occlusion (Figure 3.3). It is worth noting that 55% of the apples labeled were deemed heavily occluded (Table 3.2). This is one of the key reasons that a video based apple counting pipeline was deemed necessary. When such a high proportion of apples are heavily occluded, it is necessary to
develop a method for minimizing the occlusion problem in order to accurately estimate crop load. The labeling procedure was defined as follows:

- Drawing bounding boxes, such that they are the minimum sized box which includes all apple pixels.
- Use the “light occlusion” label for apples which have occludants blocking less than 25% of their surface.
- Use the “moderate occlusion” label for apples which have occludants blocking less than 50% of their surface.
- Label all other apples using the “heavy occlusion” label.

Figure 3.3. Examples of the three levels of apple occlusion considered. Left: light occlusion, middle: moderate occlusion, and right: heavy occlusion.
Table 3.2. Distribution of apple occlusion levels for the 180 images in the mature stage dataset. Note that over 50% of all apples are heavily occluded.

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Images</td>
<td>101</td>
<td>39</td>
<td>41</td>
</tr>
<tr>
<td>Lightly occluded apples</td>
<td>568</td>
<td>202</td>
<td>274</td>
</tr>
<tr>
<td>Moderately occluded apples</td>
<td>436</td>
<td>150</td>
<td>194</td>
</tr>
<tr>
<td>Heavily occluded apples</td>
<td>1337</td>
<td>457</td>
<td>511</td>
</tr>
<tr>
<td>Total Apples</td>
<td>2341</td>
<td>809</td>
<td>979</td>
</tr>
</tbody>
</table>

The images were divided with a train-test-validation split of 60%-20%-20%. Training was performed repeatedly with slightly adjusted hyperparameters. Key parameters identified were image size, non-maximal suppression (NMS) coefficient, anchor scales, and anchor ratios. Image size defines the resolution of the image as it is input into the neural network. Unsurprisingly, the best results were found using the native image resolution of 1920x1080 pixels. With that being said, it was found that reducing image resolution traded modestly reduced accuracy for reduced processing time. NMS is a process for eliminated redundant detections by removing bounding boxes which overlap more than a defined ratio [Rothe et al., 2014]. Tuning this ratio was a crucial trade-off between avoiding false positives, and successfully picking out apple clusters. Anchor scales and ratios defined the sizes and shapes of bounding boxes proposed by the region proposal network. Finally, training time was a crucial parameter. It was found that 10,000 training iterations led to the best results.

3.3.2 Immature Apple Detection

An additional dataset of 177 images was created for immature apple detection (Table 3.3). In this case, level of fruit maturity was a crucial consideration. It is desirable to detect fruit as early in the growing season as possible. However, it is not necessarily clear at what point fruit detection becomes feasible. Ultimately, tractability of data annotation was the determining factor. Accurate training data are necessary for training an effective fruit detector. During the annotation process, creation of accurate training labels was deemed impractical for data taken prior to July 5, so only apples from the July 5 and July 13 image acquisition dates were
used in this dataset. To ensure that video analysis results were not contaminated, no trees from the pruning severity trial were included in this dataset.

The training process followed the same procedure that was used in mature apple detection except that several data augmentation methodologies were tested. Data augmentation is a procedure used in the machine learning community to improve the training data through automated alterations. It is a preprocessing step performed during training to alter training data. The methods used were image translation, flipping, and resizing. None of the methods were found to improve model performance on the immature fruit detection dataset. However, image resizing was found to be necessary for the model to generalize to the commercial orchard dataset. This was because the apples in the commercial dataset were larger, and the camera was closer to the trees during imaging. As a result, apples in the images were nearly twice as big as those found in the training data. Resizing was performed by randomly selecting 50% of a training images, and casting that subimage into the original image size using bilinear interpolation.

| Table 3.3. Distribution of apple labels in the immature fruit detection dataset. |
|----------------------------------|-----------|-----------|-----------|
| Number of images                | Training  | Validation| Test      |
| Lightly occluded apples         | 912       | 288       | 249       |
| Moderately occluded apples      | 443       | 149       | 146       |
| Heavily occluded apples         | 547       | 171       | 193       |
| **Total apples**                | **1902**  | **608**   | **979**   |

### 3.4 Apple Counting

An apple tracking approach was first developed using the mature apple dataset, and later improved and refined for use in immature fruit counting. Both approaches use an iterative detection, matching, and translating pipeline for counting fruit over the course of a video. The final fruit counting algorithm augments this approach by also estimating the tree correspondences for each fruit detected.
3.4.1 Mature Stage Fruit Counting

The object tracking problem in this study is significantly simplified compared to the general multiple object tracking problem because apples on trees are relatively stationary. However, a constraint on the problem is that an explicit statistical model for distinguishing individual apples is impractical because they are similar in appearance. Thus, a pixel-wise rather than object-wise motion estimation approach was used so that motion estimation would not depend on a model of visual apple characteristics. The general framework for in-image motion estimation is optical flow [Farneback, 2003], a method for two-frame motion estimation. Given two consecutive frames in an image sequence, optical flow is an estimate of the translation from a given position in one image to the same position in the other.

The Farneback algorithm [Farneback, 2003] was used for the optical flow computation. The Farneback algorithm is computationally inexpensive — a single frame is processed in 0.1 seconds on a single CPU core — and effective for small frame-to-frame translations. The algorithm is based on two assumptions: (1) The pixel constancy assumption states that for small displacement in time and space between images, the same point in space should have the same pixel value on the image. (2) The continuity of motion assumption states that displacement only varies slowly in space. With these as constraints, the algorithm computes a displacement field, densely mapping each pixel location in one image to its estimated location in the next image. The algorithm is effective in small-displacement settings. Matlab's built-in implementation of the algorithm was used [Mathworks, 2017]. Images were subsampled to 1/4 of their native resolution, which resulted in reduced processing time and improved performance.

The goal of object counting is to identify each unique fruit detection in a video sequence without double counting fruit. Furthermore, counting must also be tolerant of error from both the detector and tracker. An additional consideration in the counting algorithm was to account for tracker drift. Tracker drift occurs when slight errors in translation from one pair of frames compound in future frames. Since the viewpoint of an apple is not constant from one frame to another, the pixel constancy assumption of optical flow does not hold exactly, and thus bounding box positions change relative to the object they refer to over time. If this is allowed to occur, further use of optical flow calculates translation on a bounding box that no
longer corresponds to the object being tracked.

To address these issues the object counting algorithm takes a pipelined approach (Figure 3.4). Below is the general procedure, as well as detailed descriptions for steps two, four, five, and six.

1. Initialize pipeline by detecting apples in first image using Faster R-CNN
2. Translate detections to next image using the displacement between two images computed from optical flow.
3. Perform detection on new image using Faster R-CNN.
5. Remove out of bounds detections.

![Figure 3.4](image)

**Figure 3.4.** Visualization of initialization of detection pipeline for the first two frames of a video sequence. (a) Apples are detected in the first frame, (b) apple positions are estimated for the second frame, (c) a new set of detections are computed for the second frame (shown in red), and (d) the two sets of detections are matched to create a single set of apple positions. For subsequent frames this process is followed using the matched set of apples for translation.

The first crucial step in the tracking pipeline is translation estimation. This can be defined as the task of estimating a set of object positions $\hat{\mathbf{S}}_{t+1}$ given frames $t$ and $t+1$ as well as $\mathbf{S}_t$. To do this, optical flow is computed for the pair of frames $t$ and $t+1$ to get a dense set of translation vectors (one per pixel in $t$). These translation vectors are noisy, so for each apple bounding box, the median vector value was taken. Then, these vectors are simply added to the pixel position values $\mathbf{S}_t$ to estimate apple positions for the next image.
In the matching step, two sets of apple positions are taken as input: the translated observations from the previous frame, \( \hat{S}_{t+1} \), and the new object detections from the current frame, \( S_{t+1} \). Since object appearance is ineffective at distinguishing between apples, only apple position is used for matching. Specifically, for each pair of observations, percent overlap is computed as follows. Let \( p_t^i \) denote the pixel position of the \( i \)-th observation in the \( t \)-th frame. Then, the overlap ratio between two objects \( (\hat{s}_{t+1}^i, s_{t+1}^j) = \frac{\text{area}(\hat{p}_{t+1}^i \cap p_{t+1}^i)}{\text{area}(\hat{p}_{t+1}^i \cup p_{t+1}^i)} \). This overlap ratio is computed for all pairs of observations. Individual apples are then matched in order of percent overlap down to a threshold of 85%. Empirical results found that an 85% threshold allowed for some tracking error and changes in size of detection bounding boxes between images while avoiding false matches. The matching follows the constraint that each apple can only be matched once per image (i.e., no single detection can be matched to more than one other detection). This has the effect of keeping all unique detections from previous images. Once all matches are computed, old bounding boxes corresponding to apples which were successfully matched are discarded in favor of the new detections. This stabilizes tracker drift, by updating position estimates with new information from the detector where possible. Finally, all bounding boxes that were not matched are maintained. Thus, the final matching result \( S_{t+1,\text{final}} = S_{t+1,\text{matched}} \cup \hat{S}_{t+1,\text{unmatched}} \). This allows the algorithm to maintain memory of apples that were visible in previous frames, but are occluded in the current frame. Finally, count is maintained by simply incrementing it each time a new detection is not matched.

Continued use of the algorithm on a sequence led to apples being translated outside the image. Optical flow does not perform well near the border of the image. To avoid this issue, apples within the outermost 100 pixels of the image are considered to be outside the image. Their contribution to total count was maintained, but the bounding boxes were ignored for the purposes of the fruit tracking pipeline.

### 3.4.2 Immature Stage Fruit Counting

Though the fruit tracking approach in the previous sections serves as the basis of the final fruit tracking algorithm, it has several crucial deficiencies that needed to be addressed. One crucial issue was that the Farneback optical flow algorithm
exhibited high levels of error whenever there was an unusually large displacement between frames, indicating a need for a better optical flow approach. A second issue was the lack of an approach for handling false positives. Thirdly, the previous matching approach was not optimal for the offline processing setting in that better matching performance can be calculated using global fruit position information.

To improve optical flow estimation, a range of well-performing algorithms from the literature were tested [Weinzaepfel et al., 2013, Sun et al., 2017, Bouguet, 1999]. One challenge with evaluating optical flow methodologies is a lack of ground truth data. It was not possible to manually annotate ground truth within the dataset, and furthermore, all publically available datasets are substantially different from an orchard setting. Thus, performance was evaluated based simply on inspection of optical flow results. In addition to the Farneback algorithm, three additional optical flow methods were tested. Two of the methods, Deepflow [Weinzaepfel et al., 2013] and PWCNet [Sun et al., 2017], used CNNs for learning optical flow on publicly available datasets. PWC-Net in particular was highly promising because it demonstrated state-of-the-art results with minimal processing time. However, both of these models performed exceedingly poorly in tests. This is likely because the training data for these models — MPI Sintel, an open-source animated film [Butler et al., 2012]— bears little resemblance to orchard settings. Given the excellent performance of these models in other domains, application to fruit tracking is a potentially promising research area if relevant training data were to become available. The final method tested was Lucas-Kanade (LK) optical flow [Bouguet, 1999]. The LK method is similar to the Farneback algorithm, except that rather than finding a dense flow field, it calculates optical flow for a defined set of image features. For this purpose, the centroid of each apple detection bounding box was used as the image feature to track. Ultimately, the LK method was deemed to have the lowest error rate among all methods tested.

The propagation of false positives was ameliorated using a tracklet age constraint. All fruit observations (i.e. each unique detection, $s_{i,e}$) are given an age parameter which is defined as the number of fruit detections assigned to that observation. A new detection has an age of one, and each time it is matched to another observation, the age is incremented. Finally, when tracking is completed, a threshold can be set for tracklet age in order to filter out false positives. Tuning this threshold allows for defining a trade-off between tolerance for false positives and tolerance for false
negatives. Testing by manual inspection showed that an age threshold of three performed best.

Another modification was made to improve the matching optimization algorithm. In the previous work, matching was performed using only information from the current pair of frames (i.e. at time $t$, matching was performed between observation sets $S_t$ and $S_{t-1}$). However, in the offline processing setting, it is possible to optimize the matching problem within a broader context. To achieve this, matching was formulated as a cost-flow network as proposed by Zhang et al. [Zhang et al., 2008]. A cost-flow network is a directed graph where each edge has a capacity which is the maximum amount of flow which can pass through it. In this setting, a flow from one node to another is equivalent to establishing a match, and the maximum flow along an edge is 1. Using this framework, matching can be optimized globally over an entire video sequence using well established network optimization algorithms. However, since objects are typically only present in a short sequence of frames (10-12 frames is typical), this optimization was performed on a sliding window using the memory and computationally bounded successive shortest paths algorithm implemented by Lenz et al [Lenz et al., 2015]. This method improved matching performance, especially in the case where many apples were close together.

3.4.3 Combining Video Streams

For the fruit counting approach described above to be applicable to the crop load estimation problem, it was necessary to develop a method for combining multiple overlapping video sequences, so that fruit counts could be computed for a whole tree. To do this, first a set of matching key-frames were extracted using image timestamps. Then a frame alignment formula was computed using manually annotated feature matches. This alignment was then used to resolve double counting of fruit across sequences.

During video recording, each image was associated with a timestamp using the Unix Epoch (the number of milliseconds that have elapsed since January 1, 1970). Using these values, image frames could be aligned in time. Since videos recorded independently, these times did not align exactly, so this alignment has some error. Video recorded at 30 frames per second, so this time-error is no greater than 33.3 ms which is equivalent to 2.8 cm of camera translation if the camera is moving
at 3km/hour. Due to this inherent error in frame alignment, frame overlap was handled by defining an approximate frame overlap region calculated from their respective fields of view.

### 3.4.4 Trunk Positioning

Given a set of fruit counts over a video sequence, it is desirable to generate a mapping of the positions and densities of these apples. Mapping methods which use only spatial information for this process cannot develop a one-to-one correspondence between each apple detected and the tree to which that apple belongs, because tree positioning is not perfectly uniform in real orchard settings. This study proposes that mapping each detected apple to its most spatially proximal central leader provides a strong biological basis for yield mapping with single-tree granularity.

To do this, it was first necessary to develop a trunk detection model. Trunk positions were added as additional class labels in the fruit detection dataset. Since trunks were not always vertical in the image, a single bounding box was found to be an inappropriate tool for specifying trunk positions. Instead, each trunk position was given a bounding box in continuously non-occluded trunk region (Figure 3.5). This allowed the precise position of the trunk to be known in different parts of the image. Trunks were added as an additional class to the Faster R-CNN detector, and training was completed using the same procedure as before.
Trunk detection can be used to find the position of trees in individual frames, but making a mapping between apples and trees requires correspondences between trunk positions across frames. Furthermore, the previously proposed fruit tracking framework is not applicable here, because the trunk regions which are detected can vary widely from frame to frame. There are also frequent sets of frames in which a trunk is not visible at all. Finally, though trunk spacing is nominally uniform, in practice the gaps between trees vary significantly. The proposed solution used the following framework:

1. Compute trunk detections for all images in the video sequence using Faster R-CNN.
2. For each image perform tree partitioning where possible.
3. Match partitions in adjacent frames.
4. Interpolate trunk positions across occluded regions.
5. Estimate remaining unknown trunk positions.

This algorithm depends on the assumption that each trunk is detected at least one time, and takes the minimum distance between trunks as a parameter; this parameter is important for incorporating known information about tree spacing.
3.4.5 Trunk Partitioning

In a given image, each trunk can be detected zero, one, or multiple times. Trunk partitioning is concerned with classification of each individual trunk detection into a set of all detections in that frame which belong to a unique tree. First, the minimum gap parameter is used to find mutually exclusive detections. Second, one detection is used to find the first band. Then, each remaining detection is assigned to that band if it has an x-position which is less the minimum distance parameter away from any known band, where a band’s position is defined by the x-position of the bounding box in that band with the lowest y-value. This bounding box is chosen to represent the position of the trunk, because it is assumed that a trunk position is closer to where the tree is planted closer to the ground. Finally trunk partitions are defined as the midpoint between the x-positions of any two adjacent bands (Figure 3.6).

![Figure 3.6. Sample tree partitioning. Red bounding boxes are trunk detection results, green bounding boxes are apple detection results, and blue lines are tree partitions.](image)

3.4.6 Trunk Tracking

Matching trunk partitions in adjacent frames is trivial given that both frames contain a valid detection of the given trunk. Optical flow is computed between the frames using the LK algorithm, and then matching is performed using difference
between x-values as a metric. Let $P_i^t$ be partition $i$ at time $t$ have an x-position $x_i^t$. Then $P_i^t$ is matched to $P_{i+1}^t$ if $x_i^t + \Delta x_{flow} - x_{i+1}^t < \text{minimum tree gap}$. All valid matches are computed across the video sequence to form a set of known tree positions. However, in the case of occlusion, there may be subsets of frames for which some tree positions are unknown. Furthermore, in this case, it is not valid to use optical flow to estimate translation, because the flow algorithm would track the occludants rather than the objects of interest.

Instead, x-displacement between successful trunk displacement was used to form a crude estimate of local camera velocity. In this case, the formula, velocity $(\text{pixels/s}) = \text{x-displacement}/\delta \text{time}$ was used. Local estimated velocities were then used to translate trunk positions across frames with missing trunk matches. This translation was repeated until the estimated trunk position either left the camera frame, or overlapped with an existing trunk partition. In the latter case, the two partitions were matched.

One problematic error case with this method is the existence of false positive detections. In particular, it was found that orchard posts are sometimes misclassified as trunks by the object detector. These false positives can lead to violations of the assumption of minimum trunk gap. When this occurs, there are multiple ways to form tree partitions for a given image, and partitions can only be formed arbitrarily. This can lead to misassignment of trunk position, or generating more trunk partitions for an image than there are trees visible. To mitigate this issue, orchard posts were added as a third class in the object detector following the same methodology used for trunks. Since the post detector also has some chance of finding false positives, results from the detector were only used in the case that an image had partition ambiguity. Then, any post detection was used to form a vertical filter in the image that removed any trunk detections sharing x-values with the post detection.

Trunk partitioning was only performed on videos taken from the bottom camera. This camera had the best view of the trunks and trunk positions were closer to vertical in these sequences, which lead to more precise tree positioning. Trunk partitions in this frame were projected vertically into aligned frames from the other cameras.
3.4.7 Depth Filtering

A method of depth filtering was needed in order to avoid double counting fruit detected from video on both sides of the tree. To do this, depth values were taken from known trunk positions. Then during apple counting, apples which had depth values > trunk depth + 10 cm were ignored. This enabled double counts to be largely avoided, while allowing some margin of error in trunk depth to make sure nearly all detected apples are counted at least once.

3.4.8 Single-tree Granularity Fruit Counts

To produce fruit counts with single tree granularity, the following process is followed:

1. Compute trunk detections for all images in the 6 video sequences using Faster R-CNN.
2. For each video sequence, independently execute the fruit tracking algorithm.
3. For the two video sequences from the lowest camera, compute trunk tracking and tree partitioning.
4. Align frames via timestamps, and project trunk partitions vertically to the higher video sequences.
5. For each apple tracklet, select the frame in which it is closest to the center of the image. Assign the apple to the tree partition it is in during this frame. Check if the apple is in an overlap region.
6. For each tree, resolve fruit counts in the overlap regions by taking the count from the video sequence with the most fruit in that region.

3.5 Fruit Size Estimation

No effective method was found to consistently estimate size of fruit on trees. Given a set of apple positions, and corresponding dept data, size estimation should in theory be trivial. After all, size should simply vary proportionally with respect to the number of pixels it takes to span the apple’s diameter, and inversely with the apple’s depth value. In practice, several complicating factors made it extremely
difficult to extract this information properly. Firstly, for a given detection, apple width may or may not be accurate. In many cases, the apple is partially occluded, so the detection result does not reflect the apple’s size. Secondly, only a subset of the pixels within a detection bounding box actually belongs to the apple. In any bounding box, some of the pixels will belong to occludants or background which introduces erroneous depth values into the image. Finally, even in the absence of occlusions, the detector naturally has some inaccuracy in the dimensions of its bounding box which introduces further error in size estimate. Given these constraints, all tested approaches to apple size estimation were found to be highly inaccurate.
Chapter 4  
Results and Discussion

4.1 Apple Detection

The Faster R-CNN object detection models were evaluated in terms of detection accuracy with respect to hand annotations on the validation dataset. Recall, precision, and F1 score were used as metrics for evaluating this accuracy. A detection was defined as a true positive if its bounding box had greater than 40% overlap with a hand annotation where overlap is defined as bounding box intersection over union. This value was chosen because it was often the case that several different sized bounding boxes could be appropriate for heavily occluded apples. A higher threshold caused many detections to be erroneously classified as false positives.

In order to maximize model performance, a series of training runs were performed on a range of commonly used CNN architectures from the object detection literature. Table 4.1 summarizes the results of each architecture on the immature apple dataset. VGG16 was the best performing architecture. All other reported results use the VGG16 architecture.

Table 4.1. Apple detection accuracy on immature detection dataset for four commonly used convolutional architectures.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Recall</th>
<th>Precision</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>0.81</td>
<td>0.80</td>
<td>0.81</td>
</tr>
<tr>
<td>VGG16</td>
<td>0.85</td>
<td>0.92</td>
<td>0.89</td>
</tr>
<tr>
<td>ResNet50</td>
<td>0.83</td>
<td>0.82</td>
<td>0.83</td>
</tr>
<tr>
<td>ResNet101</td>
<td>0.83</td>
<td>0.89</td>
<td>0.86</td>
</tr>
</tbody>
</table>
The ability of the object detector to generalize across various orchard conditions was of great interest. One approach to evaluating generalization was building diverse orchard conditions into the datasets themselves. Specifically, the mature dataset contained both sunny and cloudy light conditions, as well as tall spindle and fruiting wall pruning systems. The immature dataset was more limited in its internal variation, so additional tests were performed to evaluate performance using images from commercial orchards. For each location, five images were annotated. Table 4.2 summarizes these results. The model was found to generalize well to three of the four orchard blocks, though the Honeycrisp block had a low recall rate. Premier Honeycrisp is an early-maturing red apple, so the apples in those images were highly colored. It seems likely that the poorness in model performance on this block can be attributed to the high difference in coloration between the immature training data and the highly red fruit in the images. It is worth noting that precision is a slightly more important metric for an object detector as which is part of a video counting pipeline, because the detector has multiple chances to detect each apple, but it is more challenging to resolve high rates of false positives. Figure 4.1 shows representative detection results from each orchard.

Table 4.2. Faster R-CNN model generalization results using images from four commercial orchard blocks. The model is trained using data from FREC immature dataset.

<table>
<thead>
<tr>
<th>Location</th>
<th>Cultivar</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grower 1</td>
<td>Braeburn</td>
<td>0.90</td>
<td>0.77</td>
<td>0.83</td>
</tr>
<tr>
<td>Grower 1</td>
<td>Goldrush</td>
<td>0.97</td>
<td>0.82</td>
<td>0.89</td>
</tr>
<tr>
<td>Grower 2</td>
<td>Premier Honeycrisp</td>
<td>0.84</td>
<td>0.67</td>
<td>0.78</td>
</tr>
<tr>
<td>Grower 3</td>
<td>Gala</td>
<td>0.97</td>
<td>0.75</td>
<td>0.85</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>0.95</td>
<td>0.75</td>
<td>0.84</td>
</tr>
</tbody>
</table>
Figure 4.1. Detection results from four commercial orchard blocks. Each image corresponds to a different cultivar: (a) Braeburn, (b) Goldrush, (c) Gala, (d) Honeycrisp.
Figure 4.2. Detection results for braeburn dataset when no data augmentation is used during training. Apples in the nearest row are not detected, because no apples of this size were represented in the training data.

In addition, a set of tests were performed on the generalization from one dataset to another. In this case, models were trained using data from the immature dataset, the mature dataset, and both datasets combined. Then, the three resulting models were evaluated on validation sets from both datasets separately (Table 4.3). The motivation behind this test was to garner insight into the interplay between choice of training data, and the set of use cases in which the resulting model is effective. One notable result was that training using the mature dataset resulted in exceptionally poor performance on the immature testing set, but the converse is not true. One interpretation of this result is that a model trained to detect red apples is highly dependent on color features for discriminating between fruit and background. This makes sense because there is a high level of contrast between red apple fruit and green leaves in the fruit canopy. On the other hand, when training for detection of immature fruit, color is a poor discriminator, because there is little difference in color between leaves and fruit.

The performance of the combined training set was also of interest. Though the model performed worse than the individual training sets on their respective
validation sets, it did show effective generalization. This indicates that using a broad dataset is appropriate in settings where orchard conditions are not known a priori, or orchard specific training data are not available. Furthermore, it is important to note that the two datasets were taken using different cameras and different image resolutions. This is notable because it indicates the potential for a high level of flexibility when deploying Faster R-CNN (or similar) object detectors in commercial applications. In the broader context of PA, there is great interest in deploying applications using smart phone technology. In that setting, it is extremely important to understand how changing camera inputs influences detection performance. The results shown here can be viewed as a proof-of-concept which indicate that it is possible to generalize well across multiple cameras, but the degradation in performance found indicates that further study is needed to determine a set of best practices for developing models for use with multiple types of cameras.

Table 4.3. A comparison of Faster R-CNN model generalization across datasets. “Imm” denotes immature validation set, and “Ma” denotes mature validation set.

<table>
<thead>
<tr>
<th>Training set</th>
<th>Precision(Imm)</th>
<th>Recall(Imm)</th>
<th>Precision(Ma)</th>
<th>Recall(Ma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immature</td>
<td>0.85</td>
<td>0.92</td>
<td>0.82</td>
<td>0.85</td>
</tr>
<tr>
<td>Mature</td>
<td>0.82</td>
<td>0.26</td>
<td>0.92</td>
<td>0.87</td>
</tr>
<tr>
<td>Combined</td>
<td>0.83</td>
<td>0.88</td>
<td>0.88</td>
<td>0.82</td>
</tr>
</tbody>
</table>

Detection accuracy was also analyzed with respect to level of occlusion. Only recall is reported, because false positives cannot be classified by level of occlusion (Table 4.4). These results demonstrate the need of an occlusion management mechanism for accurate fruit counting. The detection accuracy is quite high for light and moderate levels of occlusion, but only 50% and 73% of heavily occluded apples are successfully detected respectively for immature and mature fruit. Given that a high proportion of all apples in a given dataset are heavily occluded (more than half in the case of the mature dataset), this accounts for a disproportionate number of the missed fruit.
Table 4.4. Recall of Faster R-CNN over varying levels of fruit occlusion.

<table>
<thead>
<tr>
<th></th>
<th>Overall</th>
<th>Light Occlusion</th>
<th>Moderate Occlusion</th>
<th>Heavy Occlusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immature fruit</td>
<td>0.85</td>
<td>0.95</td>
<td>0.75</td>
<td>0.50</td>
</tr>
<tr>
<td>Mature fruit</td>
<td>0.82</td>
<td>0.90</td>
<td>0.89</td>
<td>0.73</td>
</tr>
</tbody>
</table>

The preceding detection results indicate that the Faster R-CNN model is robust to variations in fruit maturity, color, size, and cultivar given the right training data. Furthermore, the detector also works under variations in lighting conditions and even changes to the source camera. With that being said, the development process of this study also demonstrates a need for caution and extensive testing when developing such a detector. It is difficult to predict whether a particular variation in conditions will cause a decline in performance. As an illustrating example, figure 4.2 shows an example commercial detection result when no data augmentation techniques were applied during training. Almost no apples are detected correctly even though there are many clearly visible fruit in the image, because no representative images (with respect to apple size) were present in the training data. Though every effort was taken to ensure the robustness of the object detector, it is plausible that there exists other possible variations in conditions which this study fails to account for. The construction of an appropriate dataset which accounts for the range of variations present in the target application is an essential process for any CV application.

4.2 Trunk Tracking

Trunk tracking accuracy was analyzed for two video sequences covering each side of the tree row at FREC, with a total of 117 trees. For each of 24 data trees in this row (i.e. the set of data trees), two images were selected (one for video from each side of the tree), and the centroid of each tree in that image was manually labeled. The first metric used was correctness of tree label. That is, are trees are uniquely numbered with ordered values ranging from 1 to 117? For this sequence all labels were accurately assigned. The second metric was distance from estimated trunk position to actual trunk position. This was measured in terms of pixels along the x-axis of the image. Results are reported in terms of root mean squared error (RMSE). For the 48 manually labeled trunk positions, the RMSE was 15.6 pixels.
Note that accurate labels were only possible when the trunk was visible, so the analysis was limited to cases where trunks were at least partially visible. This level of inaccuracy is tolerable given the inherent uncertainty of its application. After all, there is no guarantee that matching an apple to the nearest tree trunk will match it to the correct tree, and incorrectly matched fruit can still be useful for understanding the high-level fruit distribution throughout a block.

The reader may note that no metrics for trunk detection accuracy (e.g. precision, recall, and F1 score) were mentioned. This is because the measures were not found to be representative of any particularly useful interpretation due to the trunk position annotation methodology. Recall, that continuously non-occluded trunk sections were labeled during the annotation stage (Figure 3.5). Unfortunately, it is often an arbitrary determination where to cut off an individual trunk label. Due to this, nominally correct trunk bounding boxes were often labeled as incorrect due to discrepancies between trunk label distinctions (Figure 4.3). For the sake of completeness, the trunk detection model had a precision of 0.71, recall of 0.74, and F1 score of 0.72.
Figure 4.3. Detection result demonstrating the arbitrary bounding box problem. The red boxes are the ground truth label, and the blue box is a detection result which was classified as a false positive during evaluation. During labeling, the bounding box was split into two due to minor occlusion from leaves.

Though the trunk tracking algorithm is highly effective for this dataset, the methodology’s minimal tolerance for error indicates it needs further refinement in future work. In the current method, if any tree is miscounted, it will lead to an off-by-one error for all other trees in the video. This will cause errors in counting wherein counts from one side of a tree will not line up with the other side. To alleviate this issue, it is recommended that future work augment the use of video sequences with additional positional measurements. For example, a GPS or inertial sensor could be used to assign positional references to each image.
Another alternative could be to distribute RFID tags through the orchard, and cross-reference trunk tracklets with known RFID positions through time.

4.3 Fruit Counting

4.3.1 Mature Fruit Counting

For the mature fruit counting algorithm, analysis was limited to comparison against hand annotations. These results are useful primarily in terms of how they informed further developments in the algorithm for immature fruit counting. Three sets of 30 sequential frames each were labeled with apple annotations. The sequences were chosen to represent different lighting conditions (cloudy and sunny) and different levels of fruit density. To get a ground truth apple count, the frames of each sequence were monitored sequentially, and any newly visible apples in each frame were carefully recorded; each new detection was compared to the detection results in all previous frames in the sequence to avoid double counting.

Object counting was evaluated in terms of precision, recall, and F1 score for each video sequence (Table 4.5). It was found that using video sequences made it possible to successfully count nearly every apple in the video sequence. Given many input images, the challenges associated with occlusion were minimized because most apples were clearly visible in at least one image. The average counting error across the three video sequences was 11%. Detection accuracy generalized well across the three sequences despite variations in density of canopy and lighting conditions. Given these results, it is reasonable to conclude that video analysis is a necessary and effective method for resolving the occlusion problem in apple fruit detection.

However, it was also found that the algorithm tends to over count apples due to false positives. This occurs because there are no mechanisms for removing false positives once they enter the system. Thus, each false positive propagates through the whole system. In addition, prolonged periods of occlusion can sometimes cause an apple to be double counted because the tracking process will track whatever object blocks the apple view, rather than the hidden apple (Figure 4.4). Finally, it must be noted that the algorithm is only stable for video sequences with small displacement between images (less than 40 pixels per image). Large displacement leads to failure of the Farneback optical flow algorithm. This motivated the adoption
of LK optical flow for the final fruit counting algorithm.

**Table 4.5.** Detection Results of mature fruit video counting algorithm on 30 frame video sequences.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>Count (true value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence 1</td>
<td>0.89</td>
<td>0.92</td>
<td>0.91</td>
<td>38(37)</td>
</tr>
<tr>
<td>Sequence 2</td>
<td>0.81</td>
<td>0.97</td>
<td>0.88</td>
<td>37(31)</td>
</tr>
<tr>
<td>Sequence 3</td>
<td>0.92</td>
<td>0.99</td>
<td>0.95</td>
<td>88(82)</td>
</tr>
<tr>
<td>Overall</td>
<td>0.89</td>
<td>0.97</td>
<td>0.93</td>
<td>166(150)</td>
</tr>
</tbody>
</table>

**Figure 4.4.** Counting output after 2 frames (left), and 20 frames (right) of the video sequence. In the right image, 24 apples were detected, but use of the counting algorithm led to a total count of 34 apples. 31 of 34 detections are true positives.

### 4.3.2 Immature Fruit Counting

For immature fruit counting, access to ground truth data enabled a more complete analysis of the efficacy of the fruit counting algorithm. It was originally planned that fruit counts would be evaluated against manual counts taken on the same day as the video recordings. However, when these manual counts were compared against fruit counts at harvest, it was found that the manual counts were woefully inaccurate. For 21 of the 24 trees, more fruit were counted at harvest than on July 13 (Figure 4.5). Clearly, this cannot be correct, because after final fruit set, the number of fruit on the tree can only decrease. Furthermore, there is reason to believe that harvest counts are the more accurate of these two measures. At harvest, all fruit were removed from the trees and subsequently counted using a fruit sorter. As a result, it was simple to verify that all fruit had been counted,
because no fruit were left on the tree. Due to the discrepancies between the two counts, CV fruit counting results were compared against harvest fruit counts, but not mid-season fruit counts. Though the error in fruit counts is unfortunate from a research perspective, it does serve to further illustrate one aspect of the need for an automated crop-load estimation system given how error-prone manual fruit counting can be.

![Graph showing comparison of manual fruit count against harvest fruit count.](image)

**Figure 4.5.** Comparison of manual fruit count taken on 07/13/208 on FREC immature dataset against fruit count at harvest. More fruit were counted at harvest for 21 of 24 trees. The red line indicates a perfect fit line between harvest counts and manual counts.

CV fruit counts were compared against harvest fruit count in terms of RMSE. In addition, a set of 5 images per tree were annotated with fruit labels and fruit matching data so that precision, recall, and F1 score of the tracker could be reported for this dataset (Table 4.6). In this case, a fruit was considered a true positive if it overlapped with ground truth bounding boxes in all the frames in which an apple was visible. The CV counting algorithm consistently under counted the total number of fruit in trees by an average of 9.7 fruit, but the precision and recall measures indicate that nearly all fruit visible in the manual annotations were detected successfully. This indicates that even using the multiple points of view allowed by MOT, some apples still were never visible to the cameras. An alternative explanation is that some subset of visible fruit were simply not detected by either the Faster R-CNN model or the ground truth annotations. This is certainly possible, because it is very challenging to consistently detect moderately and heavily occluded
apples. Hand annotations had high rates of false negatives during labeling, which was only ameliorated by multiple labeling passes over each image. Even so, there were sometimes heavily shadowed or occluded portions of the images in which it was ambiguous whether or not a fruit was present.

Table 4.6. Summary of fruit counting accuracy for the FREC dataset and each pruning severity level. RMSE is calculated in terms of fruit counted using the video counting algorithm as compared to fruit count at harvest.

<table>
<thead>
<tr>
<th>Treatment</th>
<th>RMSE</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>All trees</td>
<td>10.4</td>
<td>0.97</td>
<td>0.91</td>
<td>0.94</td>
</tr>
<tr>
<td>LTR = 1.75</td>
<td>14.1</td>
<td>0.98</td>
<td>0.92</td>
<td>0.95</td>
</tr>
<tr>
<td>LTR = 1.25</td>
<td>12.83</td>
<td>0.96</td>
<td>0.93</td>
<td>0.94</td>
</tr>
<tr>
<td>LTR = 0.75</td>
<td>6.00</td>
<td>0.96</td>
<td>0.91</td>
<td>0.93</td>
</tr>
<tr>
<td>Apogee + LTR = 1.25</td>
<td>8.50</td>
<td>0.97</td>
<td>0.91</td>
<td>0.94</td>
</tr>
</tbody>
</table>

4.3.3 Pruning Severity Analysis

In the previous subsection, RMSE counting error for each LTR pruning treatment is reported. On average, more severe pruning treatments did have lower counting error than the less severe treatments. However, using absolute magnitude of the errors fails to account for variation in number of fruit present on the trees. To better understand the influence of pruning severity, it is more appropriate to use percent error as the metric. An analysis of variance was performed with pruning severity as the explanatory variable, and percent counting error as the response variable. No significant difference was found between the four treatments, though the most severe pruning treatment again had the lowest error rate overall. Figure 4.6 shows 95% confidence intervals for the four treatment error rates. Given the sample size used here, it would be inappropriate to definitively conclude that pruning severity has no bearing on fruit detection accuracy. It can be said that the video counting algorithm is a viable method for fruit counting across all levels of pruning severity.
Figure 4.6. Comparison of percent error rates of video counting over the four pruning treatments. 1: LTR = 1.75. 2: LTR = 1.25. 3: LTR = 0.75. 4: Apogee + LTR = 1.25.

4.3.4 Yield Prediction

Fruit count was tested as a predictor of yield using a linear regression model. Three separate counts were used as predictors of yield — manual counts, still image CV counts, and video CV counts. Still image counts were computed by selecting the set of 6 images (one from each video sequence) such that the tree of interest was centered in the frame. Then, manually annotated trunk partitions were used to select the set of apples which were assigned to the tree. Then, total count was simply the sum apple detections in each of the images. This simple method is included as a baseline to give some idea whether the level of complexity introduced by a video-based method is worthwhile. Tables 4.7 and 4.8 summarize the model results for predicting harvest fruit count, and harvest fruit weight respectively.

Among the models, video fruit counts were the strongest predictor of yield with $r^2 = 0.81$ for prediction of harvest weight, and $r^2 = 0.78$ for prediction of harvest fruit count. The strength of these predictions, despite the high variance in error rates across trees, is partly due to the fact that much of the counting error was systematic — across nearly all trees, the counting algorithm under-counted
rather than over-counted fruit. It is also worth noting that on-farm conditions were both atypical and non-ideal for the fruit counting application. Trees at the FREC site had low crop load, and high vigor due to an abnormally high rate of fruit abscission during June drop. This can be attributed to an extremely rainy Spring and an early-season hail event at the orchard site. Though the given yield results do indicate that video fruit counts are effective at predicting most of the variance of harvest fruit counts in this dataset, stronger results are reasonable to expect under more ideal conditions. Thus, early season fruit counts should be considered to be strong feature for building a more robust yield prediction model. More research is needed to develop a methodology for predicting yield given fruit counts as an input. Likely, such a method would require a robust approach to fruit size estimation, a way of tracking the growth of fruit over time, as well as a way to generalize yield prediction across different orchard blocks.

Table 4.7. Model summary for linear regressions comparing fruit counting methods to harvest fruit counts.

<table>
<thead>
<tr>
<th>Model</th>
<th>$r^2$</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harvest count hand count</td>
<td>0.48</td>
<td>12.1</td>
</tr>
<tr>
<td>Harvest count still image count</td>
<td>0.28</td>
<td>14.3</td>
</tr>
<tr>
<td>Harvest count video count</td>
<td>0.78</td>
<td>7.9</td>
</tr>
</tbody>
</table>

Table 4.8. Model summary for linear regressions comparing fruit counting methods to harvest fruit weight on an individual tree basis.

<table>
<thead>
<tr>
<th>Model</th>
<th>$r^2$</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harvest weight hand count</td>
<td>0.53</td>
<td>2.55</td>
</tr>
<tr>
<td>Harvest weight still image count</td>
<td>0.31</td>
<td>3.1</td>
</tr>
<tr>
<td>Harvest weight video count</td>
<td>0.81</td>
<td>1.61</td>
</tr>
</tbody>
</table>

4.4 Recommendations for Further Research

The results of this study demonstrate an effective method for counting fruit on trees across a variety of orchard conditions. Furthermore, fruit counting is demonstrated to be a promising tool for yield prediction and yield mapping applications. However,
several steps are needed to extend these methods to the point of usefulness in a real orchard setting.

Firstly, the author recommends the exploration of image segmentation techniques for fruit size estimation. Though an object detection approach is effective for fruit counting applications, more precision is needed to accurately predict fruit size. Accurate image segmentation could be used to better estimate fruit contours in order to predict diameter, which would resolve many of the problems this study faced.

Image localization was another crucial challenge faced in this study. Though trunk tracking was found to be a viable approach to producing a mapping between apples and the trees to which they belong, error robust approaches to this problem are necessary for wider applications. Positional sensing techniques such as GPS and inertial sensors could be used as error-correcting mechanisms within the trunk tracking framework.

Finally, further work is needed to extrapolate fruit counts to true yield predictions. For instance, data from repeated passes through a single block throughout a season could be used to iteratively improve predictions over time. Also, a generalizable approach to calibrating a yield prediction model to a specific orchard block would be highly useful.

4.4.1 Future Applications

Given further development, the methodology put forth in this study has potential application in cropload estimation as a service. The system requires minimal capital investment in terms of hardware (in total, the set of cameras costs <$1000). From a processing time perspective, the current algorithm could be used to process a useful sample size of trees using commodity grade hardware. Typically, yield estimation is performed on a per-block basis, so simply measuring one or two orchard rows per acre could be sufficient to provide early season yield estimates for a whole orchard. Based on a benchmarked processing time of approximately 0.5 seconds per frame, a single PC could process approximately 60 orchard rows worth of measurements within a single day without any processing time optimizations.
Chapter 5 | Conclusion

This study proposed an effective methodology for apple crop load estimation by counting fruit on trees using computer vision. Several datasets were created for training a Faster R-CNN object detector to detect apples under a variety of orchard conditions. The resulting model was found to generalize well to variations in lighting conditions, fruit maturity, fruit size, and cultivar. This result is significant in that it hints at the potential for development of a general purpose apple detector given appropriate model tuning and dataset development, using existing technology. Such a detector could serve as a platform for future development of precision agriculture applications in apple production.

The fruit detection model was incorporated in a video based pipeline for counting fruit across video sequences. Fruit counting was formulated as a multiple object tracking problem. Fruit tracking was performed using optical flow, and a fruit matching scheme. The resulting fruit counter was found to count fruit with a precision of 0.97 and a recall of 0.91. A multi-camera orchard imaging frame was fabricated to capture video sequences covering a full tree canopy in high-density orchards. Counting results were combined across all video sequences, and mapped to individual trees using a trunk-tracking approach. The resulting method was able to count fruit on trees with single-tree granularity, with a RMSE of 10.4 fruit/tree for early season fruit. This application is highly promising as an early-season yield predictor; harvest fruit weight was predicted with $r^2 = 0.81$. 
Bibliography


58
