Yield Measurement System for Seed Corn: Improving Dynamic Weight Accuracy and Harvest Area Determination

A Thesis Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Fatima Ellyn Murillo
December 2016

Acknowledgements

I would first like to thank my major professor Dr. John Wilkerson of the Biosystems Engineering and Soil Sciences Department at the University of Tennessee. His bold ideas and unconventional methods for analyzing and solving problems will always be an inspiration to me. He consistently allowed the research I did to be my own, but steered me in the right direction when I needed help.

I would also like to thank the experts who served in my committee: Dr. Paul Ayers of the Biosystems Engineering and Soil Sciences Department at the University of Tennessee and Dr. Mongi Abidi of the Electrical Engineering and Computer Science Department at the University of Tennessee. They challenged me to think outside of the box, and without their expertise and patience, I would not have accomplished this paper. Also, I would like to give special thanks to Dr. Hairong Qi for her advice in image processing and pattern recognition and Dr. Al Womac, a wordsmith of agricultural machinery jargon.

I would like to acknowledge Sun Xiaocun, SAS Certified Advanced Programmer, for her statistical expertise. I spent countless hours with my committee trying to determine which statistical method to use in this paper, and she was able to guide me through proper procedures almost instantly. Also, thanks to Samantha Gardner and Gimgun Loi for reading over my entire thesis when my tired eyes were incapable of catching sentence fragments and misspelled words. I am very grateful for their comments and corrections.

Finally, I must express my profound gratitude to my parents, Lyndon and Elizabeth Murillo, and my fiancé, Chance Frana, for their unwavering support and encouragement throughout my years of study and research. I would like to thank all my friends and family that have cheered for me along the way, this accomplishment would not have been possible without them.

Author

Fatima Murillo

Abstract

A prototype weight-based yield mapping system for seed corn production was developed at the University of Tennessee (UTK) and field tested in Iowa. The first chapter of the following study focuses on assessing the accuracy of this yield mapping system which employs a novel yield prediction and analysis software called Yield Analyzer. Yield Analyzer was designed using a rule-based system for producing yield maps with minimal user input by automatically determining acceptable ranges for known dependent variables that contribute to dynamic weight measurement errors.

The second chapter of this thesis covers the development of a non-intrusive, machine vision technique to measure true width of crop entering a header during harvesting. The development of this technology would further contribute to the overall yield prediction accuracy by providing necessary information for calculating real-time changes in the area component of yield.

Using a rule-based system for yield data processing, Yield Analyzer produces two levels of site-specific yield measurements. At the first level of data acquisition, cart weight measurements compared to certified scale weights at an average absolute difference of 6.07 %. At the second level of data acquisition, weight, length, and yield measurements had a higher degree of variance.

For determination of effective header width, two vision-based classification methods were tested from real-time harvesting video data. The first method used color features for crop detection performed > 90 % accuracy at 0.50 - 0.75 standard deviations from mean color feature descriptors. A linear support vector machine classifier trained with image SURF descriptors performed at > 95 % classification accuracy when images from the entire video dataset were used for training.

Table of Contents

List of Tables

List of Figures

Introduction

Yield Monitoring Systems

Yield monitors have become an integral component to many modern farming operations since it became commercially available for combines in the early 1990s (Griffin, 2010). These systems are designed to collect geo-referenced yield measurements and allow producers to evaluate the performance of their crops and assess variability within their fields. From these evaluations, producers have the ability to make informed decisions for optimizing the management and production of their operations.

Harvesting techniques are not standard across all varieties of crop; therefore, yield monitoring systems must be crop-specific and/or harvester-specific. Though data acquisition systems for various yield monitors may differ, the data from yield monitoring systems for any crop are subject to similar errors. B. Blackmore and Marshall (1996) analyzed data collected from grain yield monitoring systems and discovered six main error sources that contribute to yield data inaccuracies. The following is a list of the attributes that contribute to error in no particular order:

1) Lag and fill times of material through the machine,
2) Error due to GPS,
3) Material loss,
4) Material flow through the harvesting machinery,
5) Sensor accuracy and calibration, and
6) Unknown crop width entering the header

It is important to make corrections for each of these error sources in order to increase the yield measurement accuracy of these systems. Yield monitors producing significant amounts of error can lead to producers making unnecessary changes to their current field operation procedures based on the evaluations of inaccurate yield data. For this reason, there have been many studies

focused on developing solutions for correcting these errors, though no methods have been standardized (Sudduth & Drummond, 2007).

Yield is a measurement of the quantity of crop harvested over a given area. As seen in Equation 1, yield is made up of three components: weight, length, and header width. Each of these components is measured by different hardware within a yield monitoring system, and a yield measurement is then calculated from each of the measured components. The methods for obtaining a measurement for each of these components may differ depending on the type of crop that is being harvested and the equipment used.

$$Yield = \frac{W}{L \times H} \quad (1)$$

Where
W = weight measurement of harvested crop (kgs),
L = distance travelled since last measurement (m), and
H = width of crop entering the header of a harvester (m).

In an ongoing study conducted by the University of Tennessee (UT), a yield monitoring system for seed corn was designed, prototyped, and field-tested on two pickers and four weighing carts(in-tow or side loading) during a commercial-scale harvesting operation . To obtain the weight component of a yield measurement, this system used weight-based scales typically designed for static measurement systems. For the length component, GPS data was collected and the distance between each measurement was calculated. For the *H* component of a yield measurement, the system currently assumes a constant width throughout the harvesting operation.

While all errors must eventually be addressed, the overall objective of this study was to address error sources #5 and #6 discovered by B. Blackmore and Marshall (1996). This proposal is divided into two chapters providing a separate discussion for sensor accuracy and unknown crop harvest

width. In the first chapter, the error attributed to sensor accuracy (#5) is discussed as it pertains to the weight component of a yield measurement. A post-harvest, data processing method was used to increase the accuracy of reported yield measurements. In Chapter 2, the error attributed to unknown crop harvest width (#6) is discussed along with an in-lab, proof-of-concept for a vision-based approach to measuring the actual width of crop entering the header.

Objectives

The overall objective was to develop a system for determining accurate, site-specific yield measurements for seed corn. This study will contribute to the continued development of the weight-based, yield monitoring system for seed corn developed by UT. The first goal was to evaluate the use of the weight-based scales in a dynamic harvesting operation. This evaluation was conducted by using a post-harvest method for determining accurate weight measurements based on the operational conditions of the machine when measurements were taken. The second goal was to develop a visual means of measuring the actual harvest width throughout the harvesting operation. Specific objectives were:

1) To evaluate a rule-based technique for measuring site-specific yield variability within a seed corn field.
2) To validate the yield measurement accuracy under field harvest conditions.
3) To evaluate computer vision techniques for row-crop detection.

Chapter 1

Rule-Based Technique For Improving Yield Accuracy

Background & Review of Literature

In the United States alone, more than 90 million agricultural acres are designated for planting corn (Capehart, 2016). Since the discovery of hybrid seed corn in the early 1900s, the use of hybrid seed corn over conventional open-pollinated varieties of corn became widespread. By the mid 1960s, hybrid seed for corn made up over 95% of farmland dedicated to corn production (Fernades-Cornejo, 2004). Unlike conventional corn production, where combines are used to harvest corn and separate kernels, in seed production corn must be harvested with husks intact in order to protect the seed. Yield monitoring systems have been developed for conventional corn harvesting methods, but there is no commercially available option for seed corn.

In 2013, a weight-based yield monitoring system for seed corn was designed and prototyped at the University of Tennessee. Weight-based yield monitoring systems have been used for the peanut, sugar beet, and potato row crops to name a few industries (Schneider, Von Rawlins, Han, Evans, & Campbell, 1996; Thomas et al., 1999; Walter, Hofman, & Backer, 1996). Walter et al. (1996) design a slide bar weighing system to measure the load of crop on a conveyor system that performed at < 3% error during an in-field study. In a study for measuring yield of citrus, Whitney, Miller, Wheaton, Salyani, and Schueller (1999) designed a system that implemented four shear load cells measuring the weight of pallet bins containing harvested fruit with large correlation between the measure and actual yield ($r = 0.83$, $p = 0.0001$).

The ultimate goal of using yield monitoring systems is to develop maps that allow producers to visualize the yield variability within their fields. To make use of the yield data collected from the field, the data must first be calibrated, analyzed for errors, and corrected. There are several popular yield editing programs available for adjusting and filtering yield data. Sudduth and Drummond (2007) developed Yield Editor, to identify and remove outlying observations from raw yield data. Yield Editor is a widely used program provided through the U.S. Department of Agriculture that

implements filters to edit yield data for commercially available yield monitoring systems like Ag Leader or Greenstar. The latest version of this software, Yield Editor 2.0, gives users the ability to select from 12 different filters. Of the filters used in Yield Editor 2.0, the three that address outliers in weight measurements are the maximum yield (MAXY), the minimum yield (MINY), and the standard deviation of yield (STDY) filters.

The MAXY and MINY filters require user input for threshold values that represent the minimum expected yield and the maximum expected yield for a given field. These filters require prior knowledge of the expected performance of the fields, which may be difficult for new users without sufficient historical data. Additionally, by setting thresholds for expected maximum and minimum yields, yield measurements that may be accurate but fall outside of the range of expected yields would be completely removed. Nevertheless, these methods are commonly used, and in some cases, are the only filters that are applied to yield data (Simbahan, Dobermann, & Ping, 2004).

The third filter, STDY, identifies data points that are greater than a user-determined number of standard deviation coefficients from the mean of the entire field. This approach to removing outliers in the yield data has been studied by several researchers. Thylen and Murphy (1996) suggest that yield measurements greater than two times the standard deviation of the field mean should be removed, and Ping and Dobermann (2005) suggest that the STDY threshold be set to three. Simbahan et al. (2004) suggest that there should not be a set threshold, but that the value should be adjusted based on the range of the true yield variation for each field. Sudduth and Drummond found this parameter difficult to set and discovered that using 3 standard deviations led to the removal of what may have been valid data (2007).

Three additional filters used in Yield Editor 2.0 that reject yield points are the maximum velocity (MAXV), minimum velocity (MINV), and smooth velocity (SMV) filters. The MAXV and MINV

filters remove samples taken when velocity of the harvester falls outside of the expected harvesting velocity range. The SMV filter removes samples taken while the harvester experiences any rapid change in velocity.

Though Yield Editor and similar yield correction methods offer the removal of seemingly erroneous data, these tools heavily rely on user input and some statistical filtering. The focus of the study discussed in this chapter evaluates a novel yield analysis and mapping technique for increasing yield measurement accuracy without filtering yield data. Yield Analyzer is currently programmed for specific use with UT's yield monitoring system for seed corn.

Objectives

Yield monitoring systems consist of three distinct parts: real-time acquisition of yield attributes (weight and area) and other attributes that impact the quality of the yield attributes, yield validation, and yield mapping. This studied focused on the yield data analysis and mapping techniques. The specific objectives of the study are:

1. Evaluate the rule-based, yield mapping technique implemented in Yield Analyzer, a yield analysis software developed by researchers at UT.
2. Validate in-field, dynamic weight measurements collected by the system using certified scale weights.

Prior Study

Researchers at UT worked closely with an international commercial seed production company to develop and test the yield monitoring system. Harvesting operations for this producer required the use of several machines for the harvesting and transporting of seed corn from the field to the production facility. Table 1 lists all the harvesting equipment and other frequently used terms for describing the harvesting operation.

Table 1. Seed corn production jargon and definitions.

Term	Definition
Picker, Harvester	Equipment used for harvesting
Chase cart	Tractor with towed cart used for transferring crop from harvester (picker) to a tractor trailer. Used for side loading and static transfers.
Cart	Cart towed by picker or tractor instrumented with weighting system.
Tractor Trailer	Semi-trailer and road tractor used for hauling harvested material from field to processing facility.
Field	Area of land used for growing seed corn. Has a pre-measured shapefile with harvest boundaries.
Processing Facility	Central terminal location for seed corn processing.
Weigh Station	Location at processing facility where tractor trailer weights are recorded by a certified scale.
Yield	Harvested crop (weight) divided by harvested area.
Yield Monitoring System (YMS)	Data acquisition system used for collecting real-time yield data.

For any given field, a different combination of harvesters, chase carts, and tractor trailers may be used. The configuration of machines is determined by the production manager's preferences. The possible machine configurations and operational states of the machines are illustrated in Figure 1 through Figure 5. In Figure 1, three main machine units used during the harvesting process are illustrated. A semi-tractor trailer, not illustrated, is used to transport the harvested material from the field to the processing facility for drying, sorting, and packaging..

Figure 2 through Figure 5 illustrate possible machine configurations and operational states that occur during the harvesting process. Though these may not be all possible configurations and operational states, they are some of the most common. In this study, the only configuration of machines used for data acquisition was a harvester with towed storage and a single corresponding chase cart. The operational states illustrated in Figures 2, 3, and 4 were all possible operational states that were identified for the machine configuration used.

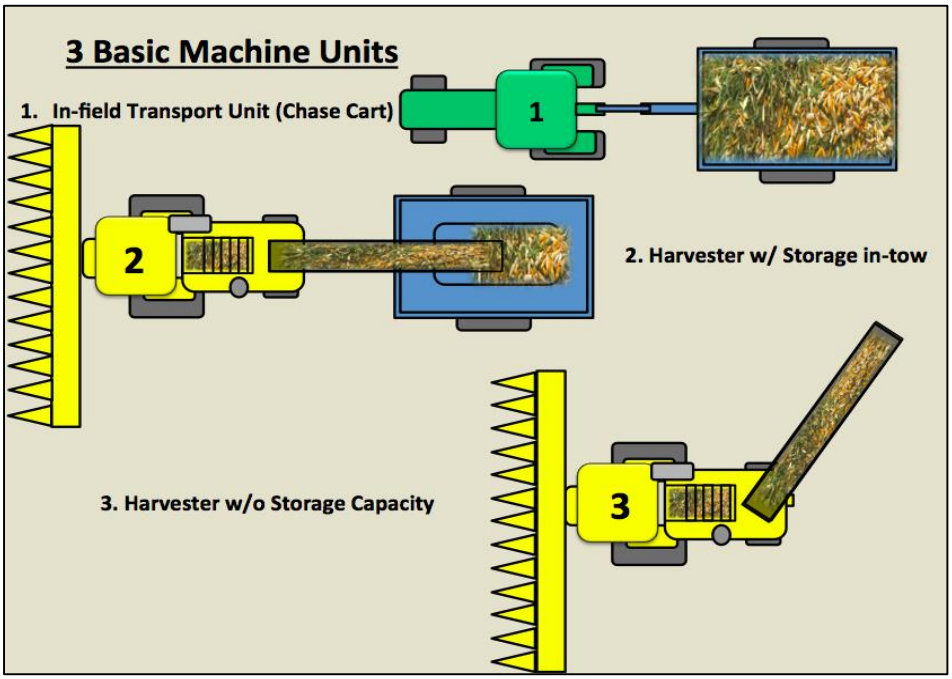


Figure 1. Seed corn harvesting machine machines units used during harvest

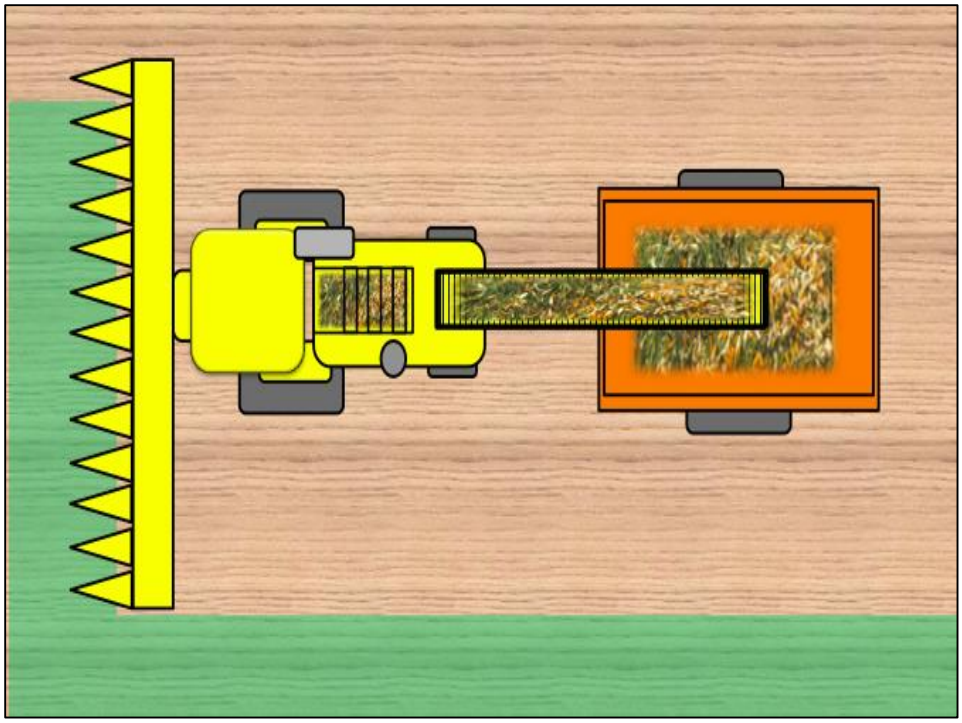


Figure 2. Dynamic loading into a towed cart.

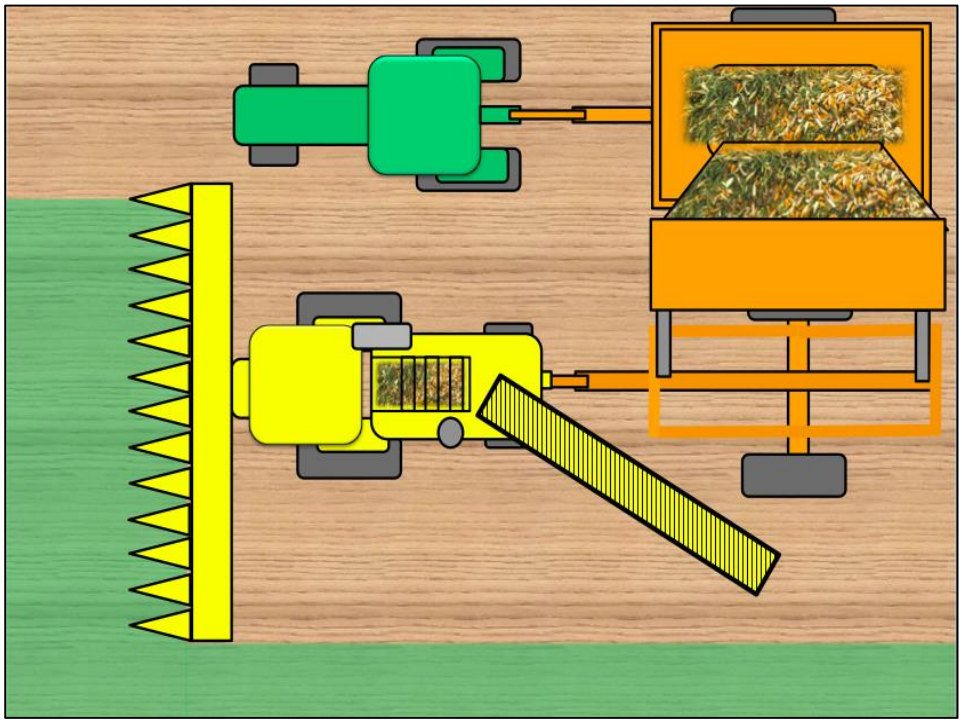


Figure 3. Load transfer from harvester to chase cart.

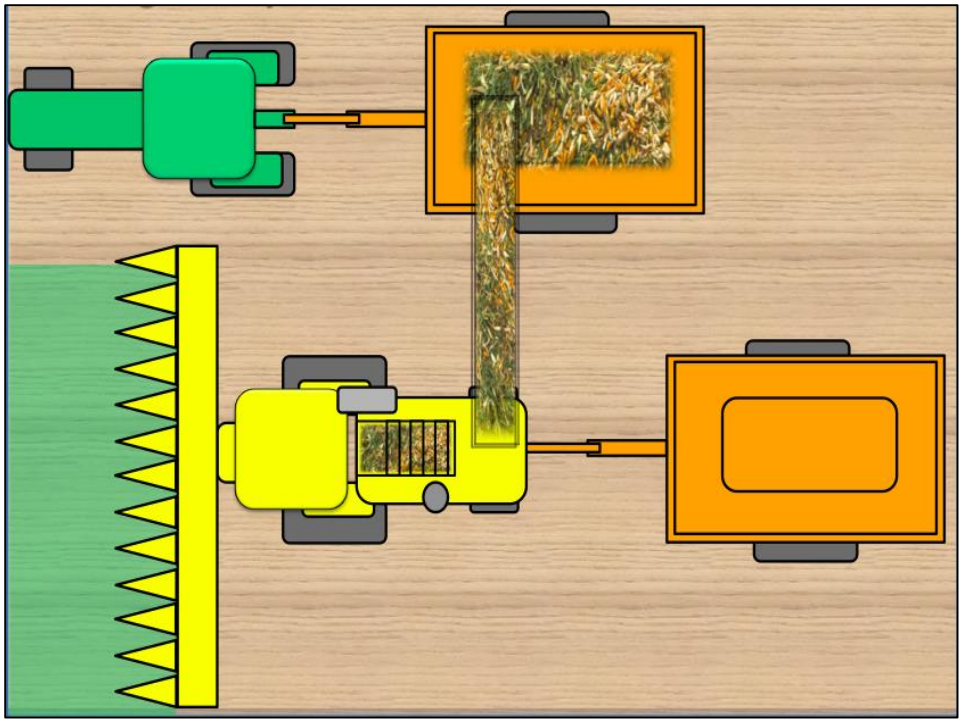


Figure 4. Side loading.

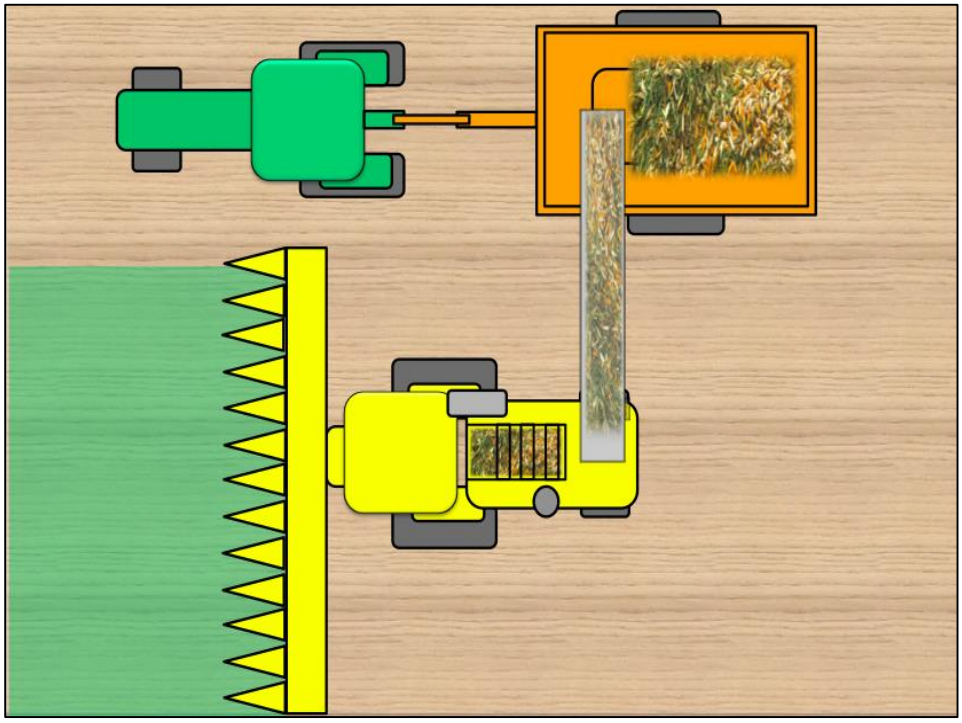


Figure 5. 100% Side loading

Methods and Materials

Yield Monitoring System Description

The embedded system, seen in Figure 6, is programmed to collect data from an on-board GPS unit (Trimble Copernicus II) and an in-cab scale display interfaced with three load cells. In 2013, scale readings were recorded by interfacing the display unit with the data acquisition unit via an RS232 interface. Three load cells (Avery Weigh-Tronix) mounted to the two axles and the hitch of a trailer cart towed by the harvester or tractor. Therefore, yield measurements were represented by the accumulation of harvested crop over a known distance.

To accommodate for the multi-machine harvesting configuration, the system uses wireless communication devices to communicate with peripheral systems via Wi-Fi and RF data modems. Auxiliary sensors may easily be adapted to wirelessly communicate with the central unit. The discreet design requires no user input and has no display monitor. Data is extracted from the system through a USB interface.

Weighing System

Limited by the inability to modify the harvesting equipment used for harvesting seed corn, the yield monitoring system was designed to use existing load cells for measuring the accumulated weight of corn in the trailer carts. Commercially available load cells designed specifically for agricultural applications allowed for the integration of a weighing system with virtually no influence to the operation of any of the machines used in the study. Three weigh bars and a model 640M indicator from Avery Weigh-Tronix, make up the weighing system used in the design.

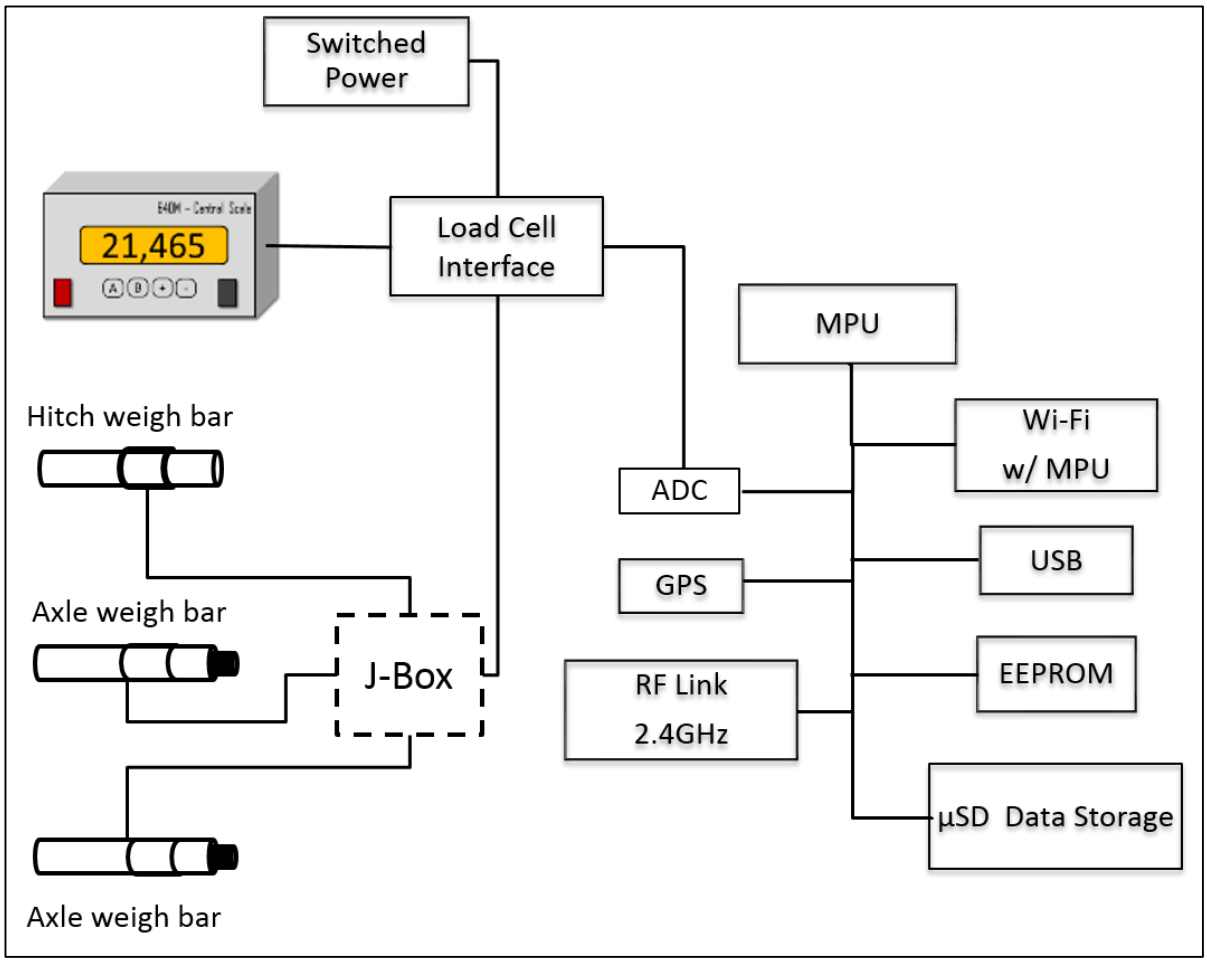


Figure 6. Schematic of yield monitoring hardware configuration.

Data Acquisition

This study analyzes the performance of two YMS over five commercial hybrid corn seed fields located in Iowa. One YMS was installed on a harvester unit, and the second was installed on the corresponding chase cart unit. Each system operated at a sampling rate of 1 Hz, collecting each of the attributes list in Table 2. From the raw data collected by the system, additional attributes were calculated for data analysis purposes. These calculated attributes are seen and described in Table 3.

The five fields used in this study varied based on the row length of a single pass in the field. The range of lengths evaluated were approximately 480 m to 800 m across. Fields 1 and 2 measured >750 m across, Fields 3 and 5 measured between approximately 700 m and 600 m across, and Field 4 measured < 500 m across.

Table 2. System-acquired attributes.

Attribute	Description
System ID	Unique ID
UTC	Universal Time Coordinate (GMT)
Latitude	Degree (WGS84)
Longitude	Degree (WGS84)
Speed Over Ground (SOG)	mph
Course Over Ground (COG)	Degrees
Scale Reading	Reading from load cell interface. Used for weight determination.

Table 3. System-calculated Attributes.

Attribute	Description and Units
Cart Weight	3-point running average (Lbs)
Change in Cart Weight	Change in weight from previous (Lbs)
Distance Travelled	Distance to previous point (m)
Change in SOG	Used for detecting acceleration (mph/s)
Change in COG	Used for detecting change in angular velocity (degrees/s)
Yield	Weight over area travelled (kg/m)
Header Width	Constant value. Either 12 or 14 row. 30”/row.

Yield Data Analysis

As part of UT’s development of a mass-based yield monitoring system for seed corn, a user-driven, post-harvest program, called Yield Analyzer, was written to take in the raw data collected from the fields, process the data via a rule-based technique, and generate a shape file that represents yield measurements at a user-defined spatial resolution as outlined in Figure 7. Yield Analyzer extracts multiple levels of information that are discusses in the following sections. Six distinct Yield Analyzer tasks are:

1. Conversion of raw data into a conventional data format
2. Determination of the operational machine state for each data point
3. Calculation of total time machine spent in each operational state
4. Identification of anchor points used for representing yield variation
5. Detection of all load transfers from harvester to chase cart and chase cart to tractor trailer
6. Generation of geospatial vector data for yield mapping purposes

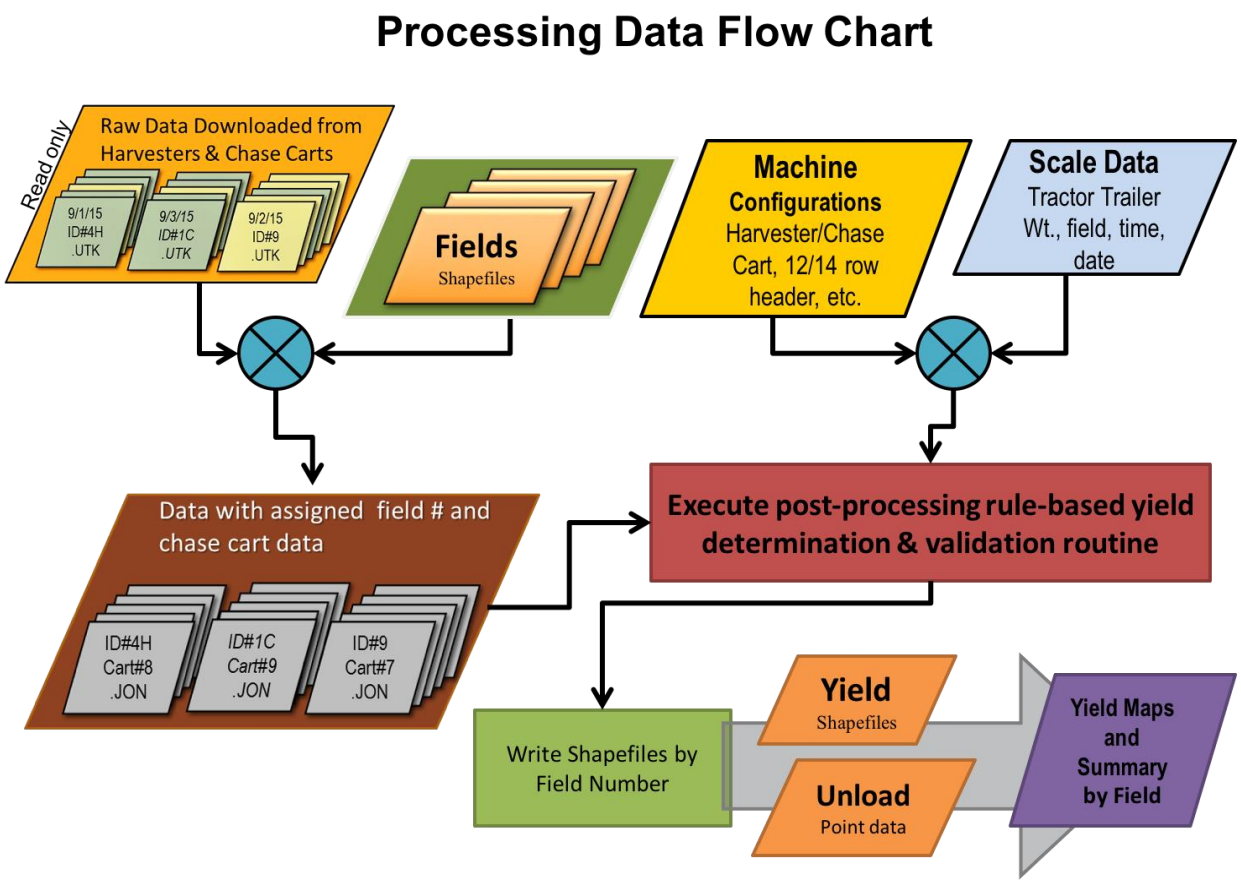


Figure 7. Flow of data from raw and input data to Yield Analyzer output.

Determination of Machine States of Operation

The data collected by the YMS provides information that can be used to classify the operational state of the machine for each data point. In Figure 8, the speed over ground and accumulative weight of a harvester trailer are plotted as a function of time (UTC). With initial analysis of the data, certain operational states of the harvesting machine can be predicted as noted in Figure 8 where there is a peak in weight at 25,055 Lbs followed by a sharp drop in weight to near 0 Lbs, and the speed of the vehicle decreases to 0 mph. This behavior, for example, can be associated with the transfer of load from a harvester to a chase cart. Other patterns have been associated with the operational states of: *starting up*, *harvesting*, *unloading*, *waiting*, *side loading*, and *other*. The *other* state includes irrelevant or indeterminate states.

Classification of the operational states in which samples were taken, provided an additional attribute used to determine the quality of the other attributes measured. Additionally, by determining operational states of the in-field machines, producers would have the ability to assess not only the productivity of their fields, but also the operational efficiency of their harvesting system.

The metrics used to define each operational state are listed in Table 4. However, specific criteria used for identifying operational states are beyond the scope of this project. Each of the metric limitations are determined by pre-harvest user input, prior knowledge, or field statistics.

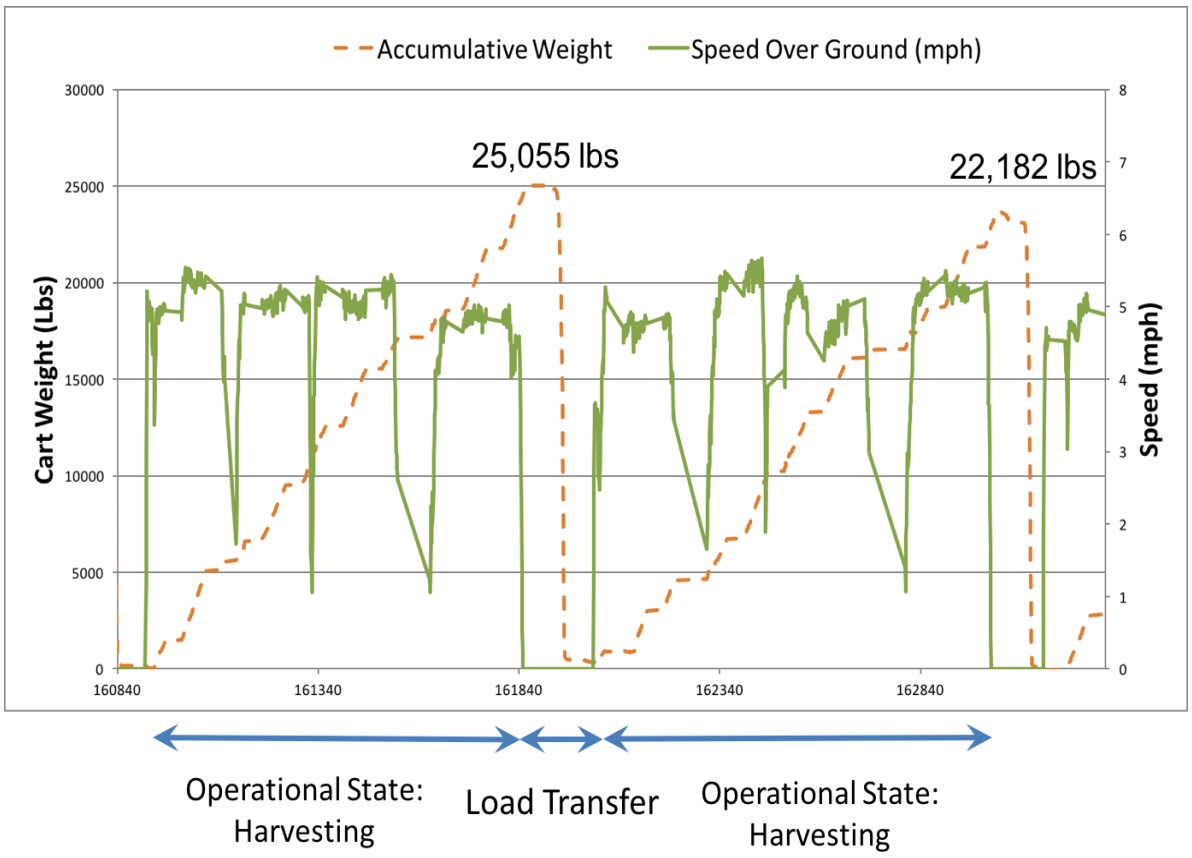


Figure 8. Time domain of harvester velocity and towed weighing cart (Wilkerson, 2015).

Table 4. Rule Configuration Metrics

Rule Metric	Description
maxAcceleration	Maximum acceleration (mph/s)
minIntegrationLength	User-defined. Shortest allowed polygon length for yield representation. Target length is midpoint of min- and maxIntegrationLength. (m)
maxIntegrationLength	User-defined. Longest allowed polygon length. (m)
maxTurn	Greatest turn rate allowed (degrees/s)
minYield	Minimum yield required for polygon (lbs/m)
maxDeltaWeight	Greatest increase in weight allowed (lbs/s)
minOperatingSOG	Below this is not considered harvesting/side loading (mph)
maxOperatingSOG	Above this not considered harvesting/ side loading (mph)
minSogStdDevs	Considers points this many standard deviations below the mean as not harvesting/side loading
maxSogStdDevs	Considers points this many standard deviations above the mean as not harvesting/ side loading
sideloadCOGDif	Checks if harvester COG matches chase cart COG for side loading determination. (degrees)
sideloadSOGif	Checks if harvester SOG matches chase cart SOG for side loading determination. (mph)
sideloadDist	Checks distance between harvester and chase cart for side loading determination (m)

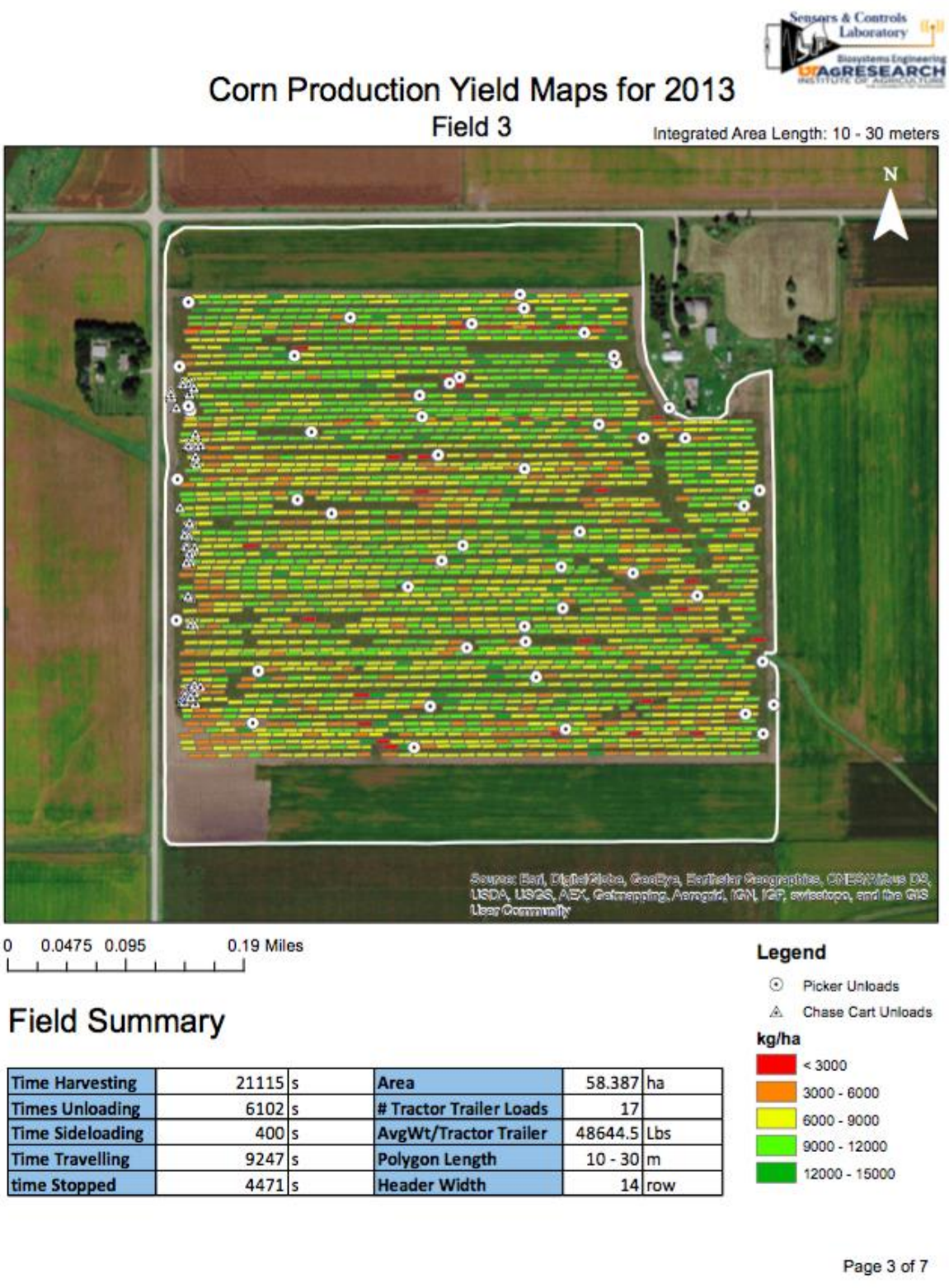
Yield Mapping

Conventionally, pre-processing for yield data includes the removal of samples collected outside of the field boundaries, the removal of samples that represent start- and end-pass delays, and shifting the raw data to correct for the delay of crop flow through the system. Similarly, Yield Analyzer applies these preliminary processes to the raw data using field boundary shapefiles and a constant lag shift of 6 seconds for both the chase cart and the harvester.

In Yield Analyzer, not all points from a field dataset are used to produce yield maps. Unlike most yield analysis software, Yield Analyzer does not use filters to remove yield data. Instead, Yield Analyzer searches for points throughout the dataset that meet a set of criteria that would suggest a high degree of accuracy in the measurement. The criteria, or rules, are determined on the basis of physical limitations, expert knowledge, and field statistics. The points that meet the criteria are called anchor points and are used for yield representation when producing maps.

Yield Analyzer defines yield measurements over an area not a point. This area is referred to throughout this paper as polygons. Users define the range of desired integration length for yield representation and yield maps are generated accordingly. Figures 9, 10, and 11 are examples of yield maps generated at 10 -30 m, 30 - 50 m, and 70 - 90 m spatial resolution settings.

Yield Analyzer takes the average of the user-defined range and searches for anchor points at intervals of that distance. The anchor points determine the starting and ending points for yield representation. Since weight measurements are accumulated weight, yield is calculated using Equation 1 where W is the difference of weight from the starting and ending anchor points, L is the distance between the two points times, and H is the assumed the header width.



Field Summary

Time Harvesting	21115	s	Area	58.387	ha
Times Unloading	6102	s	# Tractor Trailer Loads	17	
Time Sideloading	400	s	AvgWt/Tractor Trailer	48644.5	Lbs
Time Travelling	9247	s	Polygon Length	10 - 30	m
time Stopped	4471	s	Header Width	14	row

Figure 9. Yield map with spatial resolution set to 10 - 30 m yield representations.

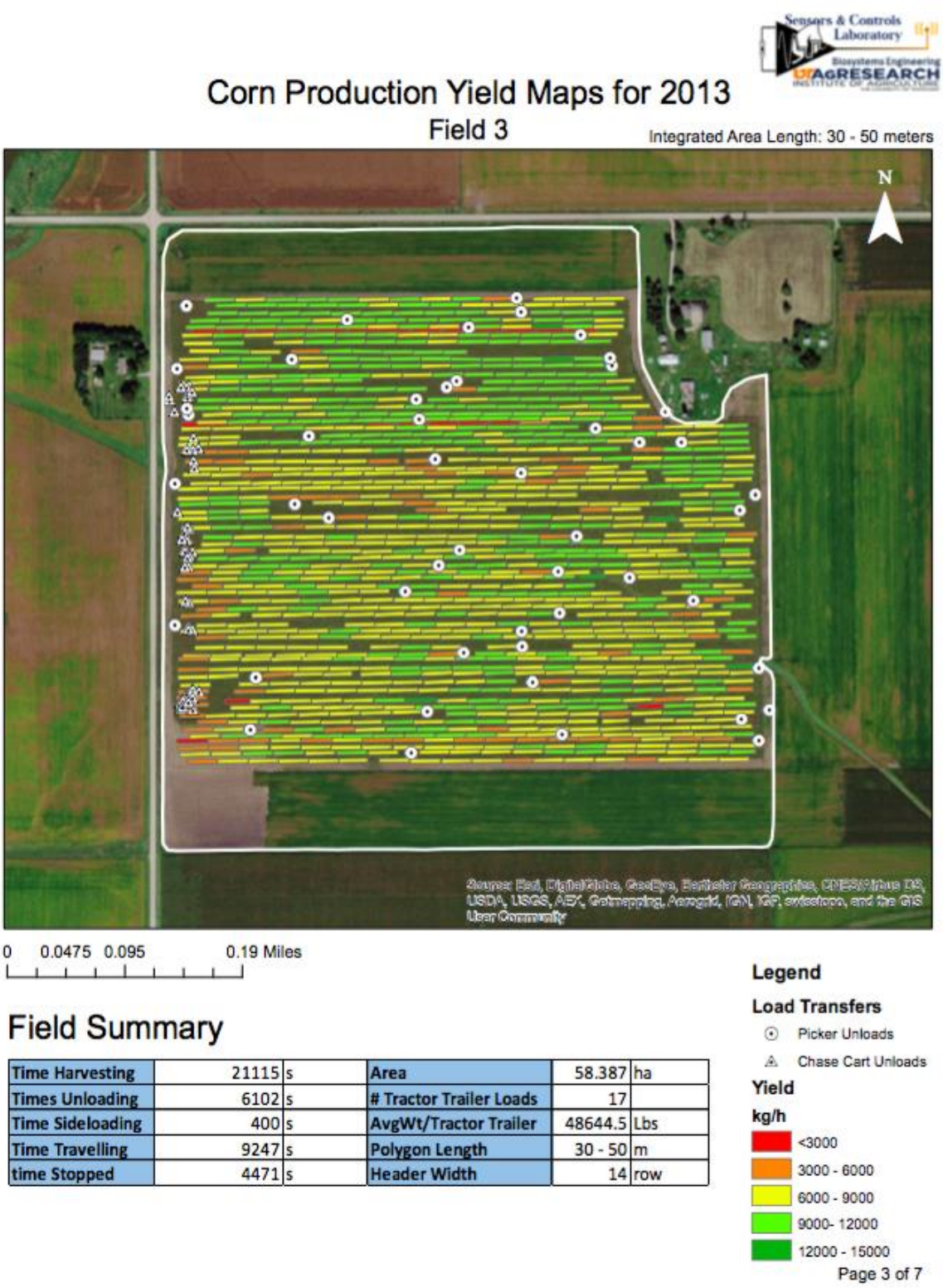
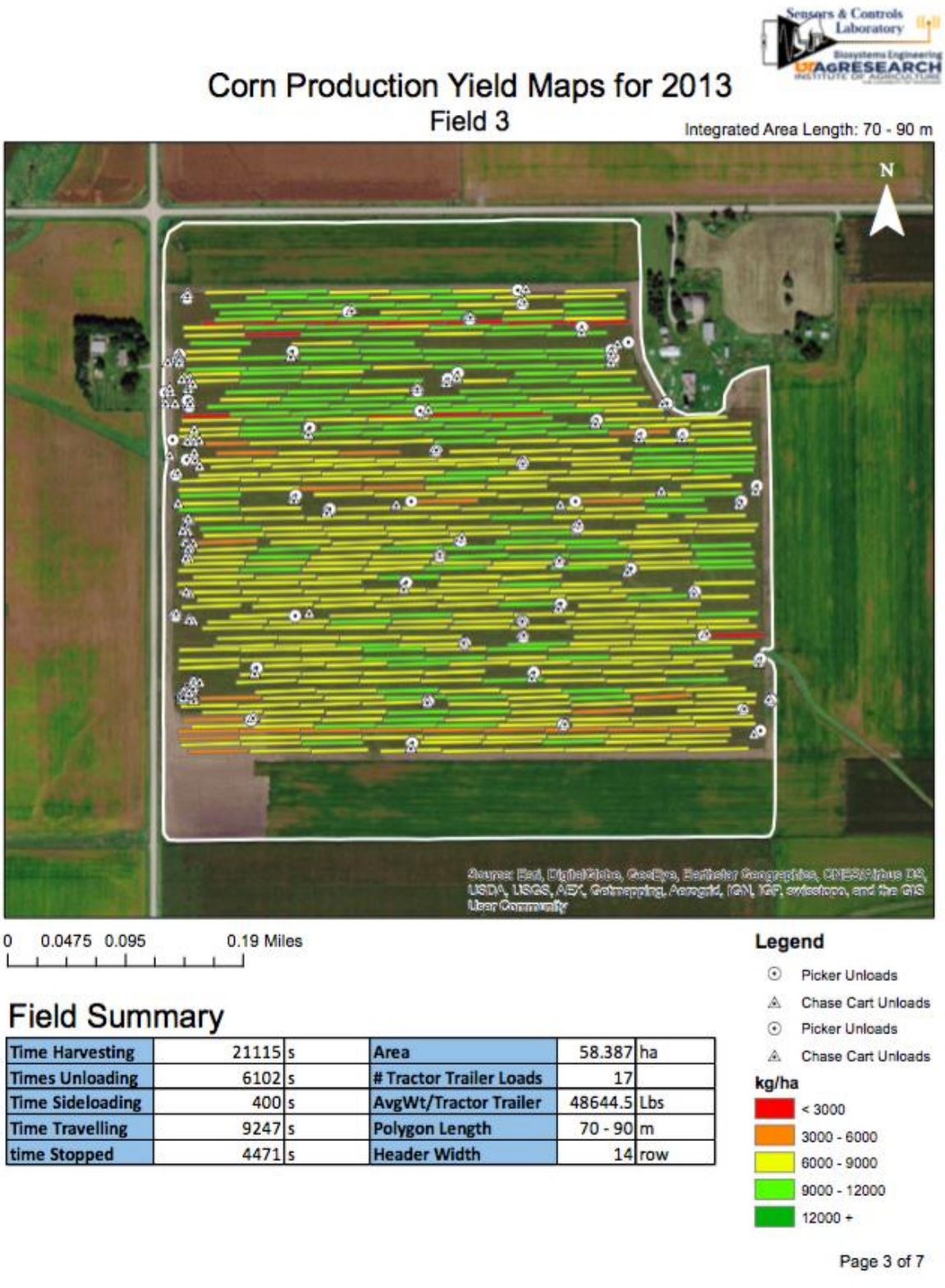


Figure 10. Yield map with spatial resolution set to 10 - 30 m yield representations.



Field Summary

Time Harvesting	21115	s	Area	58.387	ha
Times Unloading	6102	s	# Tractor Trailer Loads	17	
Time Sideloading	400	s	AvgWt/Tractor Trailer	48644.5	Lbs
Time Travelling	9247	s	Polygon Length	70 - 90	m
time Stopped	4471	s	Header Width	14	row

Figure 11. Yield map with spatial resolution set to 70 - 90 m yield representations

Each polygon in the yield maps represented above is determine by locating anchor points in the raw data that have been determined to be in good standing based by the rule-based system. Yield is then determined by taking the yield accumulated from the starting anchor point to ending anchor point and dividing that by the distance between anchor points. Figure 12 illustrates a general example of how polygons are determined.

Yield Analyzer searches for anchor points throughout the raw data and aims for intervals based on polygon length setting. If the program determines that a point does not meet the criteria outlined by Yield Analyzer, the program will continue to look at the surrounding points on either side to find an anchor point until the interval exceeds the minimum or maximum polygon length settings. If the program is unable to find an anchor point with the polygon length settings, the program will look ahead the length of a polygon and search for a new starting anchor point.

All polygons formed are associated to the corresponding chase cart unload weight measurement. This correspondence provides a means to validate the yield determined by the sum of the polygons that correspond to a single chase cart unload. Additionally, the polygon yield measurements can be calibrated based on the truth values from the chase cart every time there is a load transfer from the harvester to the chase cart.

Validation

The validation of the weight measurements obtain in the field is two part. First, chase cart unload weights are compared to the corresponding tractor trailer weights measured on certified scales. This part of the validation shows how the weighing system performs under static conditions in the field as chase cart unload weights are determined in a static state right before load transfer to the tractor trailer. During the harvesting operation in which the data set was collected, weight data was collected from certified scales. This scale data provides true weight data for every tractor trailer

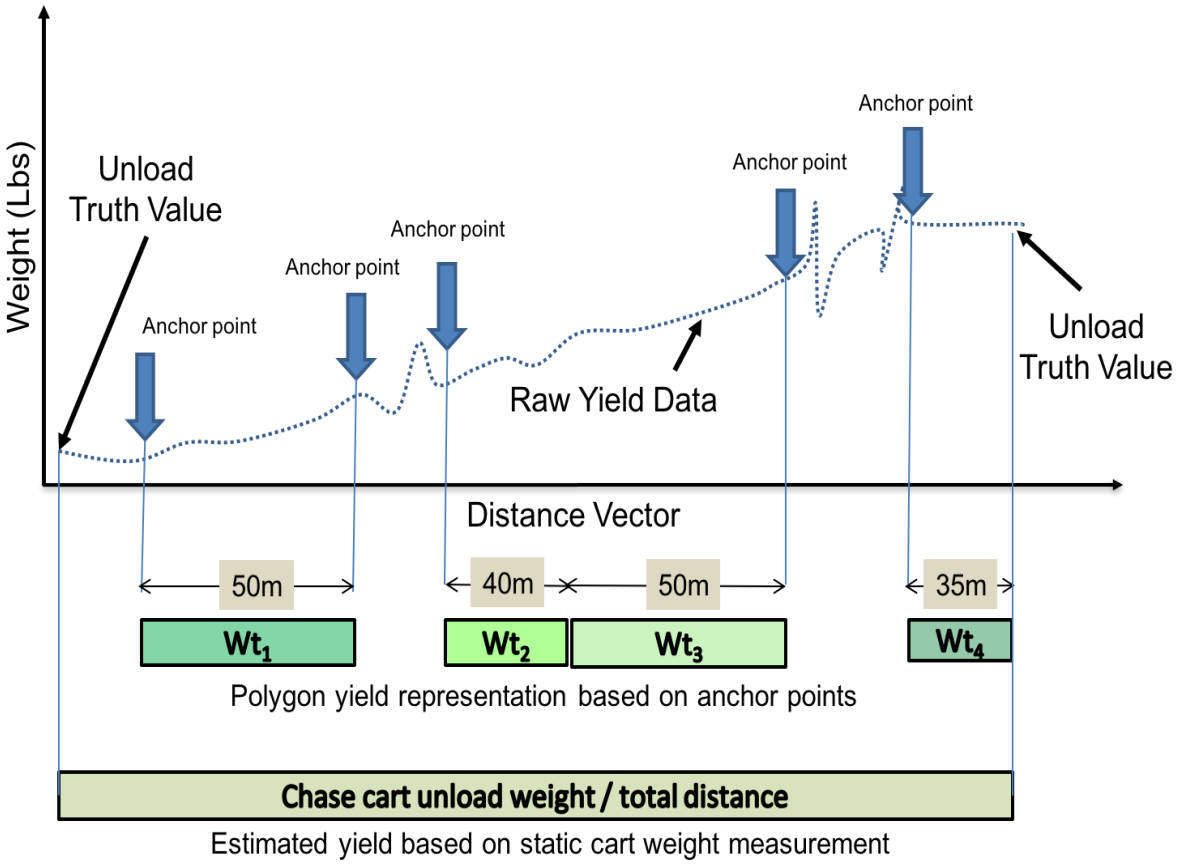


Figure 12. Polygon determination and validation to chase cart yield measurements.

load which is equivalent to 2-3 load transfers from chase carts. Each tractor trailer load is traced backed to the chase cart load transfers, and each chase cart load has an associated harvester unload. This data will be used to validate the in-field yield monitoring system performance and variability in accuracy with respect to spatial resolution.

The second part of the validation is to compare the yield measurements determined by the polygons. The polygon yield measurements are calculated from data obtained under dynamic conditions within the field. A comparison of the polygon yield measurements and the chase cart yield measurements are made in the Results section. Lastly, polygon length settings can vary based on user preference. In order to test the repeatability of the yield measurements at varying polygon length settings, a sensitivity analysis between five different polygon lengths settings were evaluated.

Results and Discussion

Chase Cart to Tractor Trailer

The first comparison is made between each tractor trailer unload and the corresponding weight measurements from chase cart unloads. Every tractor trailer load weighed at the processing facility can be traced back to the chase cart unloads the crop originated from. This association is made based on the recorded dates and times of load transfers by the tractor trailer operators and the date and time attributes for each chase cart unload detected by Yield Analyzer. Figures 13 - 17 illustrate how the sum of the chase cart unloads compare with each tractor trailer load. Only weight measurements are compared at this level because tractor trailer weights do not correspond to any measured area.

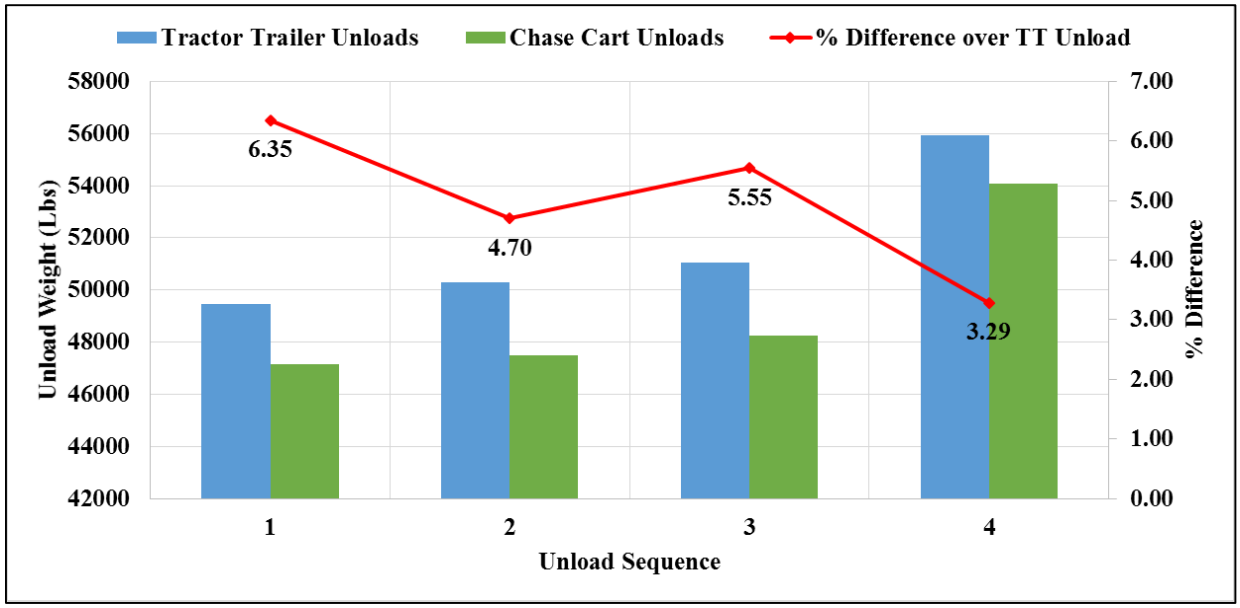


Figure 13. Chase cart to tractor trailer load comparison for Field 1.

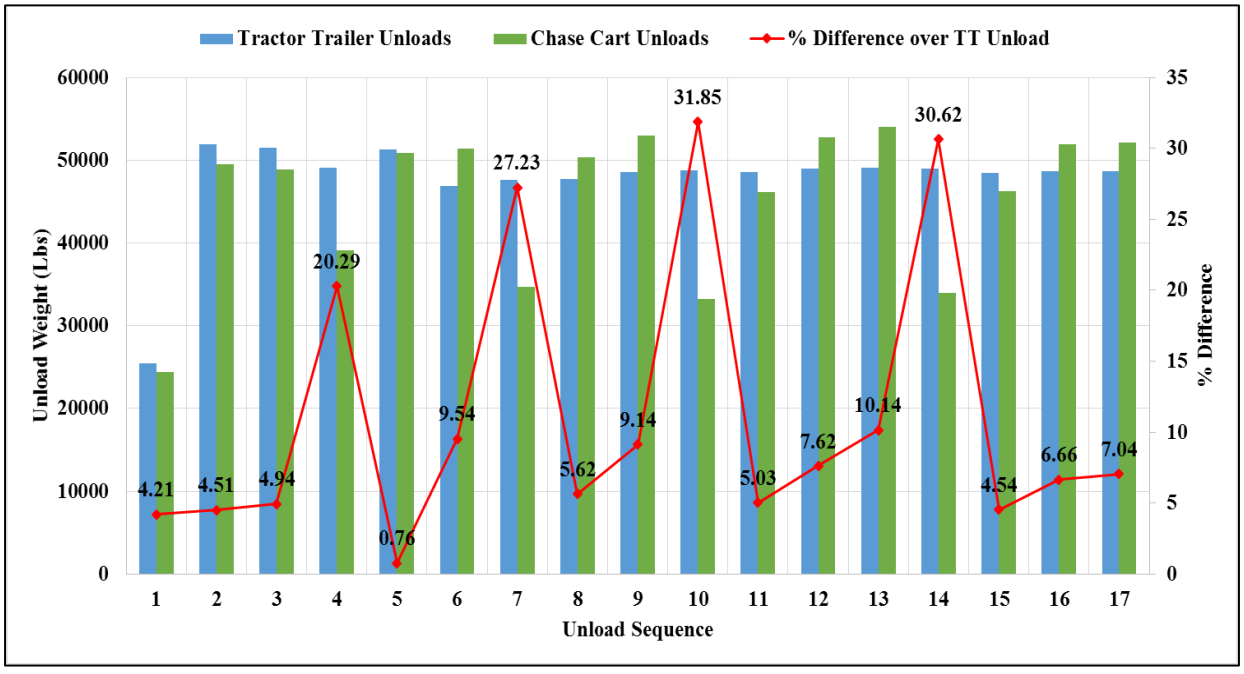


Figure 14. Chase cart to tractor trailer load comparison for Field 2.

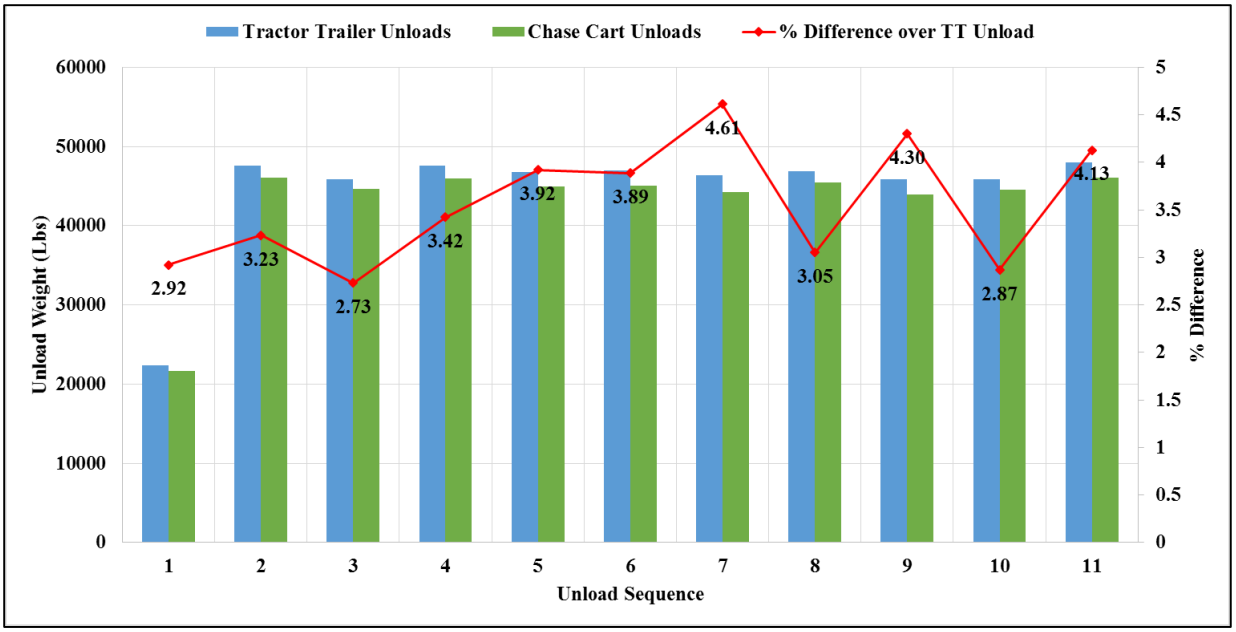


Figure 15. Chase cart to tractor trailer load comparison for Field 3.

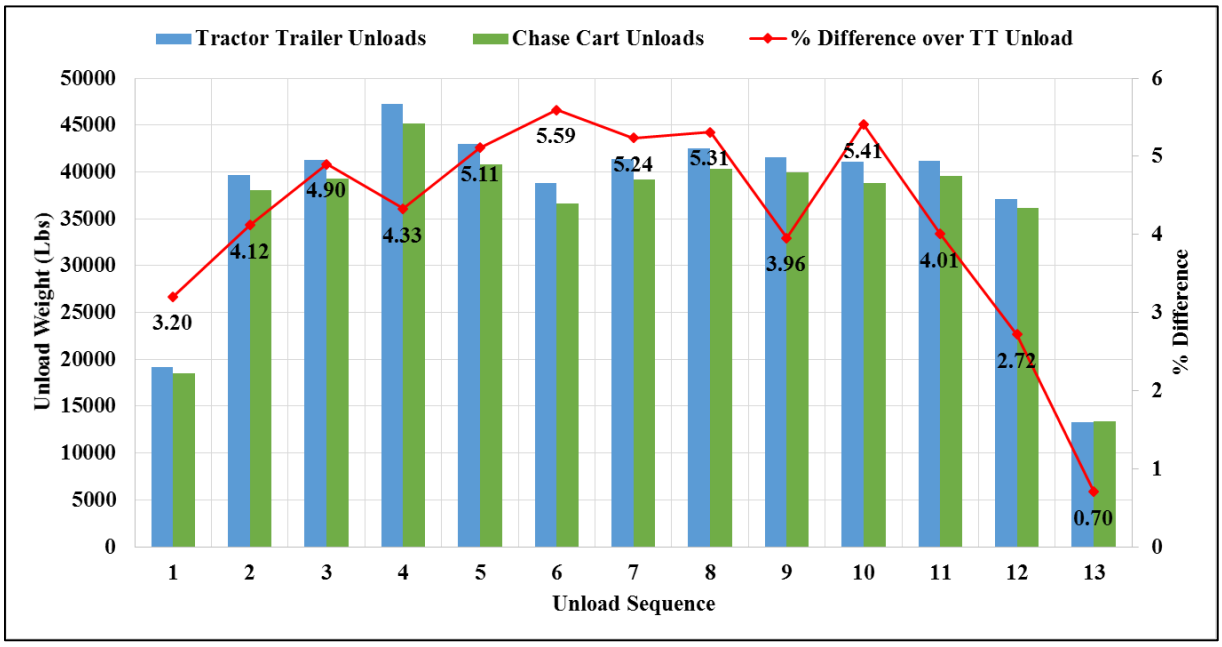


Figure 16. Chase cart to tractor trailer load comparison for Field 4.

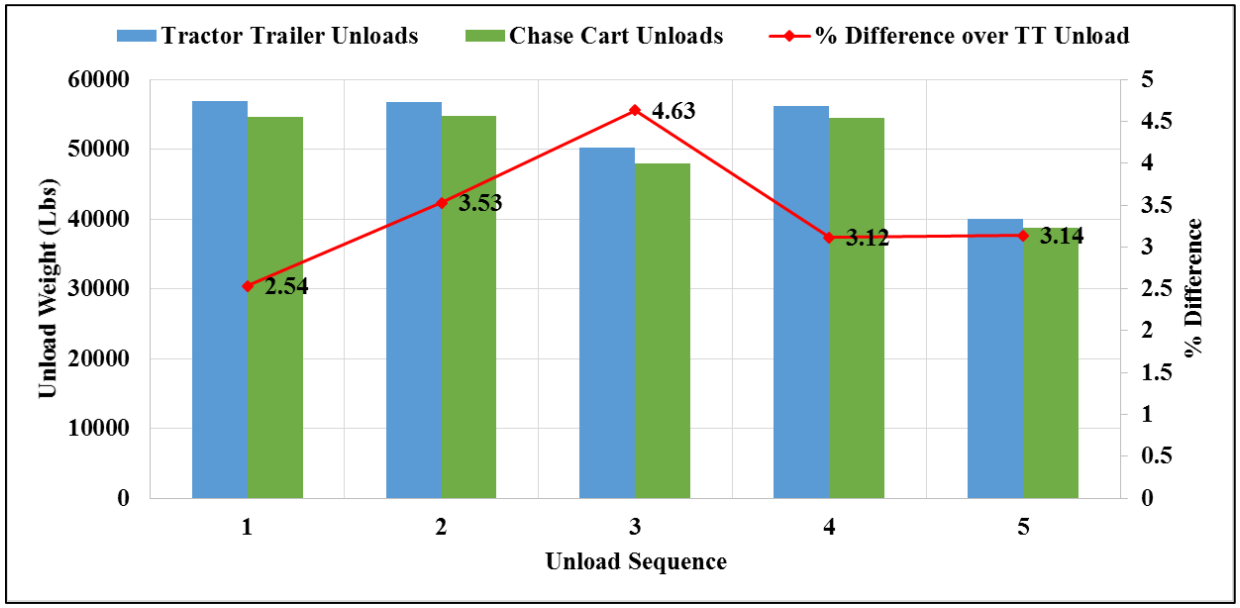


Figure 17. Chase cart to tractor trailer load comparison for Field 5.

Figure 13 illustrates that Field 1 required four tractor trailer loads to transfer the crop from the field to the processing facility. The sum of the detected chase cart unloads that correspond to each tractor trailer load was different in weight by no more than 6.35 % . For most of the fields, evaluated in this study, the sum of the chase cart unloads measure < 6% difference in weight.

In Figure 14, Field 2 had four extreme differences between the chase cart weights and the corresponding tractor trailer weight. These patterns are not comparable to the rest of the field comparisons where the tractor trailer weight always exceeds the sum of the corresponding chase cart weights with a 6% difference. It is believed that the reason for the significant differences between the chase cart weights and the tractor trailer weights for Field 2 was due to the inability to accurately match the chase carts with the corresponding tractor trailer vehicles.

This inability to associate the detected chase carts weights with the tractor trailer weights should not penalize the performance of the system. Instead, it is recommended that future versions of the yield monitoring system should implement a means for automatically detecting the identity of the

tractor trailer in which chase cart unloads are transferred and the time associated with the transfer of that load into the tractor trailer. Implementing this feature into the system would remove the responsibility of machine operators manually recording these events.

The comparison results between the tractor trailer loads and the chase cart unloads would suggest that a weight-based system is a viable means of measuring site-specific data. At this level of measurement acquisition, the overall mean absolute percent difference between the chase cart unloads and the tractor trailer loads was 6.40%.

Polygon to Chase Cart

In this section, the polygon length and yield measurements are compared to the associated chase cart unload length and yield measurements. This comparison will provide information about the percent coverage of the polygons compared to the total area covered between load transfers. Table 5 through Table 9 break down the comparison of the polygons and the chase carts per field.

Each row indicates the target polygon length determined by the minimum and maximum settings used for generating polygons with Yield Analyzer. N is the total number of chase cart unloads detected. The MP_areaCovered column is the mean percentage of the harvested area accounted for by the sum of the polygons for each load transfer.

As seen in Equation 2, the mean percentage differences between the total distance traveled and the sum of the polygons is calculated. This value measures the average magnitude of the differences between the sums of the polygon lengths and the total distance travelled between each load transfer for the entire field. This value should always be positive since the sum of polygon lengths should never exceed the total distance travelled for each load transfer. Then this value is subtracted from 1 in order to calculate the mean percent area accounted for by the polygons as follows:

$$MP_{areaCovered} = 1 - \frac{100}{n} \sum_{t=1}^{n} \frac{TotalDistance_t - PolygonsLength_t}{TotalDistance} \quad (2)$$

Where
n = total number of chase cart unloads detected,
TotalDistance = total distance travelled over a given load transfer, and
PolygonsLength = sum of polygon lengths over a given load transfer.

The MAPD_yield column is the mean absolute percentage difference (MAPD) of the yield measurements. This value is a measure of the average magnitude of the differences between the polygon and the chase cart yield measurements, as seen in Equation 3. No consideration is made to the sign of the difference in yield measurements.

$$MAPD_{yield} = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{PolygonsYield_t - ChaseCartYield_t}{ChaseCartYield_t} \right| \quad (3)$$

Where
n = total number of chase cart unloads detected,
PolygonsYield = sum of polygon weights(Lbs) / sum of the polygon lengths (m), and
ChaseCartYield = weight of chase cart unload(Lbs) / total distance travelled (m).

The MPD_yield column is the mean percentage difference (MPD) between polygon and chase cart unload yield measurements for the entire field. This value measures the average of the differences between the polygon and the chase cart yield measurements with consideration for the direction of the differences, as seen in Equation 4. This value may provide useful calibration offset values. Equation four is calculated as follows:

$$MPD_{yield} = \frac{100}{n} \sum_{t=1}^{n} \frac{PolygonsYield_t - ChaseCartYield_t}{ChaseCartYield_t} \quad (4)$$

Where
n = total number of chase cart unloads detected,
PolygonsYield = sum of polygon weights(Lbs) / sum of the polygon lengths (m), and
ChaseCartYield = weight of chase cart unload(Lbs) / total distance travelled (m).

In Table 7 and Table 8, the mean percentage of the total harvested area accounted for by the polygons is greater than 100% for each test. This means that the calculated total distances for each chase cart unload was less than the sum of the distances of the polygons. The sum of the polygon lengths should never exceed the total distance travelled; therefore, the data in from Tables 7 and 8 would suggest that there was some error in calculating the total distance measured. This explains the increase in the mean percent difference in yield for these fields compared with fields 1, 2, and 5.

The difference between the chase cart to tractor trailer comparisons and the polygon to chase cart comparisons could be attributed to the differences in the operational states that the measurements were taken. Polygon anchor points were selected under dynamic conditions; whereas, most chase cart unloads were measured under static conditions. After evaluation of the rules-based technique employed by Yield Analyzer, several recommendations for additional rules can be made and are discussed in the Recommendations section.

Table 5. Area and yield comparisons between the polygon dataset and the chase cart dataset for Field 1.

Polygon Length	Field 1			
	N	MP_areaCovered	MAPD_yield	MPD_yield
20	13	93.60 %	6.70%	-0.97%
30	13	94.68 %	6.30 %	-3.78 %
40	13	96.03 %	5.80 %	-5.49 %
60	13	95.04 %	6.66 %	-6.66 %
80	13	89.75 %	7.12 %	-7.12 %

Table 6. Area and yield comparisons between the polygon dataset and the chase cart dataset for Field 2.

Polygon Length	Field 2			
	N	MP_areaCovered	MAPD_yield	MPD_yield
20	49	96.35 %	2.42%	16.60%
30	49	96.17 %	13.46 %	-2.01 %
40	49	96.15 %	11.50 %	-8.37 %
60	49	93.20 %	13.91 %	-3.59 %
80	49	91.94 %	36.59 %	-30.76 %

Table 7. Area and yield comparisons between the polygon dataset and the chase cart dataset for Field 3.

Polygon Length	Field 3			
	N	MP_areaCovered	MAPD_yield	MPD_yield
20	20	127.61 %	31.46 %	-20.71 %
30	20	128.80 %	36.53 %	-23.91 %
40	20	131.21 %	34.77 %	-29.21 %
60	20	130.95 %	35. 51 %	-30.45 %
80	20	127.56 %	36.59 %	-30.76 %

Table 8. Area and yield comparisons between the polygon dataset and the chase cart dataset for Field 4.

Polygon Length	Field 4			
	N	MP_areaCovered	MAPD_yield	MPD_yield
20	24	145.84 %	36.41%	-18.78 %
30	24	149.71 %	35.21 %	-17.84 %
40	24	144.01 %	35.46 %	-19.32 %
60	24	143.56 %	-18.26 %	36.03 %
80	24	136.46 %	36.40 %	-18.47 %

Table 9. Area and yield comparisons between the polygon dataset and the chase cart dataset for Field 5.

Polygon Length	Field 5			
	N	MP_areaCovered	MAPD_yield	MPD_yield
20	14	88.89 %	16.4%	7.91%
30	14	90.02 %	13.10 %	4.01 %
40	14	89.99 %	12.37 %	3.11 %
60	14	90.76 %	9.21 %	-1.11 %
80	14	88.53 %	7.87 %	-3.30 %

Polygon to Polygon

Polygon lengths are user-defined; therefore, it is important to test for significance between various integration lengths. Ideally, for a single point in a given field, Yield Analyzer would compute similar yield measurements at various polygon length settings. This concept is illustrated in Figure 18, where the same point in each field is observed. In order to test this theory, 100 points were randomly selected in each of the five fields.

The 100 observation data set for each field was analyzed using one-way repeated measures with the yield measurement as the response variable and polygon length as the within-subject factor. The data violated ANOVA assumptions of normality and equal variance; therefore, ranked transformation was applied. Post hoc multiple comparisons among the different polygon lengths were conducted with Tukey's adjustment and statistical significance was identified at a significance level of 0.05. All analysis was conducted using PROC MIXED in SAS 9.4 TS1M3 from SAS institute Inc. (Cary, NC). A summary of the results in shown in Table 10, and the comprehensive results can be found in Appendix A where analysis is conducted by field.

Table 10. One-way repeated measures for yield data by field.

Field	P-Value	Description
1	0.9380	No significance between polygon lengths.
2	0.0567	No significance between polygon lengths.
3	0.0009	Significance caused by differences between the 20 and 60 m polygons and the 20 and 80 m polygons.
4	0.386	No significance between polygon lengths.
5	0.0090	Significance caused by differences between the 20 and 80 m polygons and the 30 and 80 m polygons.

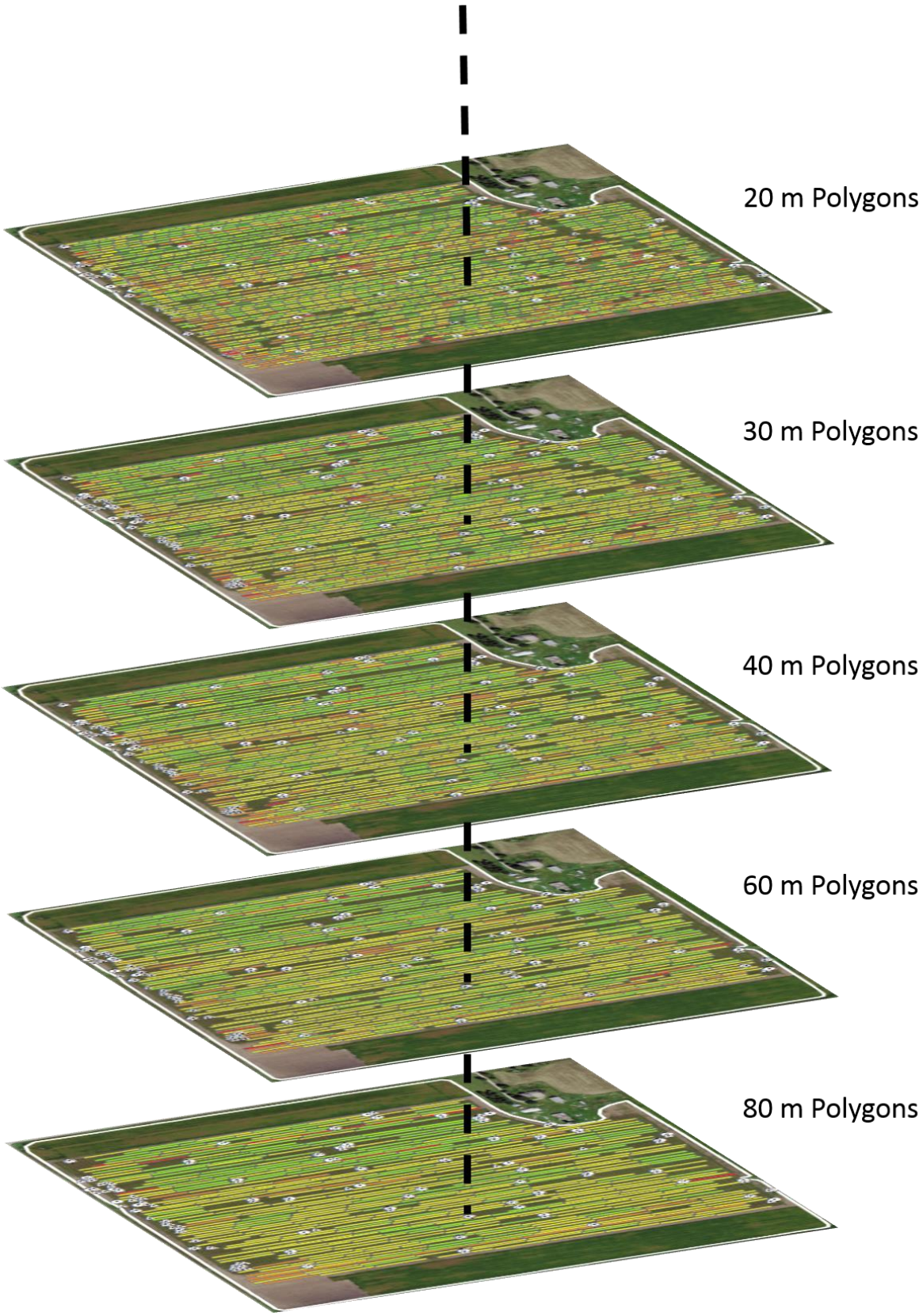


Figure 18. Stacked maps at various polygon lengths.

At the polygon level of acquisition, a significance was analyzed between different integration lengths and the results varied between fields. In Fields 1, 2, and 4 results showed no significant difference in yield between the various polygon lengths. However, in Fields 3 and 5, the results did show significance.

A post hoc multiple comparisons analysis was used to identify the cause for significance. In Field 3, the significance was caused by the differences between two sets of yield measurements: the 20 and 60 m polygons and the 20 and 80 m polygons. In Field 5, the significance was also caused by differences between two set of yield measurements: 20 and 80 m polygons and the 30 and 80 m polygons. For both fields, the significance was caused by the difference in yield measurements between the two minimum integration lengths and the two maximum integration lengths tested.

Recommendations

Yield Analyzer either met or exceed the expectations when comparing tractor trailer weights with the in-field cart weights. However, there is room for improving the rule-based system for detecting error-free anchor points.. The following recommendations are based on the evaluation of Yield Analyzer for five fields at five various anchor point distance settings:

- Accuracy assessment of the calculated total distances measured for each chase cart unload.
- Definition of rules for determining practical distance measurements.
- Definition of rules that minimize the allowable distance between polygons.
- Definition of rules for rejecting physically impossible measurements in weight and distance.
- Implementation of a peripheral system that will associate harvester to chase cart and chase cart to tractor trailer IDs.
- Implementational of a peripheral system for true header width determination.

Chapter 2

A Vision-based Approach for Crop Width Determination

Background & Review of Literature

One the leading sources of error found in yield data is due to inaccuracies associated with measuring the width of crop entering the header during harvest (B. Blackmore & Marshall, 1996). Throughout this study, this width measurement is referred to as the *effective header width*. The effective header width is a necessary measurement for calculating the harvested area component of yield. In many existing systems, the effective header width is handled one of four ways:

1) header width settings are manually updated by the operator (Nielson, 2014),

2) an estimated constant header width is assumed throughout the entire harvesting operation (Joe D Luck & Fulton, 2014) ,

3) post-harvest techniques are used to modify header width (Joe D. Luck, Mueller, & Fulton, 2015), and

4) header widths are automatically adjusted using field coverage maps (Joe D. Luck et al., 2015)

The impact of inaccurate header widths can have a significant influence on yield estimation errors especially when the percentage of changing header width occurrences are high. The most common practical causes for changes in header width are due to field edges, narrow finishes, and point rows (S. Blackmore, 1999). Another cause for header width change is the crop layout in the field. The discovery of hybrid corn, which can be traced back to the beginning of the 20th century, made way for faster growing, disease tolerant, higher yielding crops (Griliches, 1957; Wright, 1980). On hybrid corn fields, male and female plants are planted in patterns. The most widely used schema for planting hybrid corn is a 1 male :4 female row pattern. In order to prevent self-pollination, female tassels are removed giving male plants the opportunity to pollinate the adjacent female rows of corn. After cross-pollination occurs, the male rows are removed prior to harvest leaving behind approximately 80% of the initially planted rows.

Therefore, in the case of commercial-scale, hybrid corn fields where the percentage of missing rows is high, it is necessary to provide producers with the ability to accurately quantify the effective header width throughout the harvesting operation in order to calculate yield. Systems that require operators to manually change the effective header width require an additional responsibility for the operator and add a degree of human error (S. Blackmore, 1999; Reitz & Kutzbach, 1996). Other systems that make assumptions on the header width may assign a constant value which can be anywhere between 70% - 100% of the maximum header width (Beck, Searcy, & Roades, 2001; S. Blackmore, 1999; Reitz & Kutzbach, 1996; Vansichen & De Baerdemaeker, 1992)

Several studies have been dedicated to finding solutions to the issue of unknown header width by developing post-harvest techniques that can be applied to the data after the operation is completed. B. Blackmore and Marshall (1996) introduced the concept of Potential Mapping, a technique used in the post processing of the yield data to overcome this uncertainty caused by unknown crop width. In Yield Editor, a widely used yield data processing software, the Minimum Swath (MINS) filter was designed to remove yield samples with an insignificant header width entry. Point rows and finishing rows are areas where a narrow width is expected. These areas increase noise in the system so significantly that studies such as the one conducted by Beck et al. (2001) have led to suggest avoiding recording data with narrow widths completely. The development of a technique for automated detection of the effective header width will make avoiding these areas unnecessary and will increase the accuracy of yield measurements within fields.

Computer Vision and Machine Learning in Crop Production

With computer vision (CV) methods, the task of object recognition becomes viable, and this technology is being used to accomplish a variety of agricultural tasks such as corn tassel, weed, row, and crop identification (Jiang, Wang, & Liu, 2015; Kurtulmuş & Kavdir, 2014; Montalvo et

al., 2013). Computer vision techniques are used to implement machine learning capabilities by modelling human vision with the use of images. CV is composed of image processing algorithms and pattern recognition techniques. Image processing algorithms are used to process raw images by transformations, filtering, segmentation, etc. Numerous pattern recognition techniques are used for recognizing patterns and trends in a wide variety of datasets.

In a study conducted on blueberry yield monitoring, Swain, Zaman, Schumann, Percival, and Bochtis (2010) uses a color attention method in which the blue pixel index was used as an indication of fruit detection. Another computer vision study used color information as well as morphological features to identify corn tassel locations (Kurtulmuş & Kavdir, 2014). Benalia et al. (2016) used color parameters and principal component analysis to develop a sorter that determines the quality of dried figs. Muscato, Prestifilippo, Abbate, and Rizzuto (2005) used morphological features and neural networks to develop a robotic system for orange harvesting. In each of these studies, results showed a significant correlation between the information extracted from images and the information required from agricultural environments.

Often times, farmers are asked for expert advice on making operational decisions which may be replaced with automated systems. Computer vision technology and machine learning techniques can provide automated solutions for redundant tasks such as header width detection. The focus of this study was to use an experimental dataset to test the performance of a vision-based approach to determine effective header width.

Objectives

The overall objective of this study is to determine the effective header width of a harvester during operation using an image classification approach. In contrast to other computer vision tasks such as recognition, content based image retrieval, and detection, the goal of image classification

is to determine the class of an entire image or a portion of an image. The following two-part experiment tests two separate image classification techniques for determining the status of each cutting region of a header implement as active or inactive. The first part of the experiment uses image color features for classifying the cutting regions of an image. The second part identifies texture features and trains a binary classifier for classifying cutting regions of an image.

Methods & Materials

Data Acquisition

The yield mapping system in this study calculates yield measurements using an assumed constant header width. The constant value was determined based on the full width of the header implement used on each harvester. Maximum header implement widths varied from 12-row to 14-row headers. In this study, header imagery was collected from a 12-row header. A GoPro HERO3+ 1080p (used in 1280 x 720 mode). Action Camera was used to capture two sets of video data during actual harvesting operations in the field under natural lighting conditions.

The video data sets were converted to Portable Network Graphics (.png) files for individual frame analysis. Individual frames were 24-bit images with a size of 1280x720 pixels. Figure 19 shows the extent of the field of view (FOV) at which the videos were acquired. This image also demonstrates the need for a means to measure the effective header width. In this example, it can be seen that the harvester is only operating at 50% of the header's capacity while in mid-field. This situation may be one of many, where harvester operators compromise harvesting at maximum capacity for logistic purposes. It was discovered that, in this scenario, the operator adjusted the rate of harvest so that the towed trailer cart would be filled with crop at the edge of the field.

Figure 19. Example of a situation mid-field where the harvester harvested at 50% of the header capacity.

Digital Image Processing

Digital image processing (DIP) encompasses a broad range of techniques used to manipulate raw images for a variety of objectives. With DIP, images may be transformed into color spaces that accentuate specific parameters not obvious in the raw image format. Segmentation is another DIP process and is used to divide an image into meaningful parts. Other DIP methods include image restoration, pixilation, and many others.

In the following sections, two separate tests were conducted to determine effective header width from images using two distinct DIP methods. In the first test, a color feature approach was implemented in which thresholds were defined for three parameters: hue, saturation, and intensity. The second test used a texture feature description approach in which Speeded Up Robust Features (SURF) were identified and used for training a support vector machine (SVM). Though each test used different DIP and classifications methods, the same image segmentation method is used for both studies.

Segmentation

The camera was mounted such that the FOV of the images contains four main color classes that are of interest: crop, soil, row dividers, and stripper plates. These can be seen in Figure 20. In the images examined, the camera was not positioned such that all twelve sets of stripper plates were visible. For the purposes of this study, only those stripper plate regions that were visible were used as illustrated in Figure 20 parts 2-10. The regions surrounding each set of stripper plates, shown by the extent of the red boundaries in Figure 20, were the areas defined for row detection. Each image was segmented to these nine Regions of Interest (ROIs) for individual image analysis.

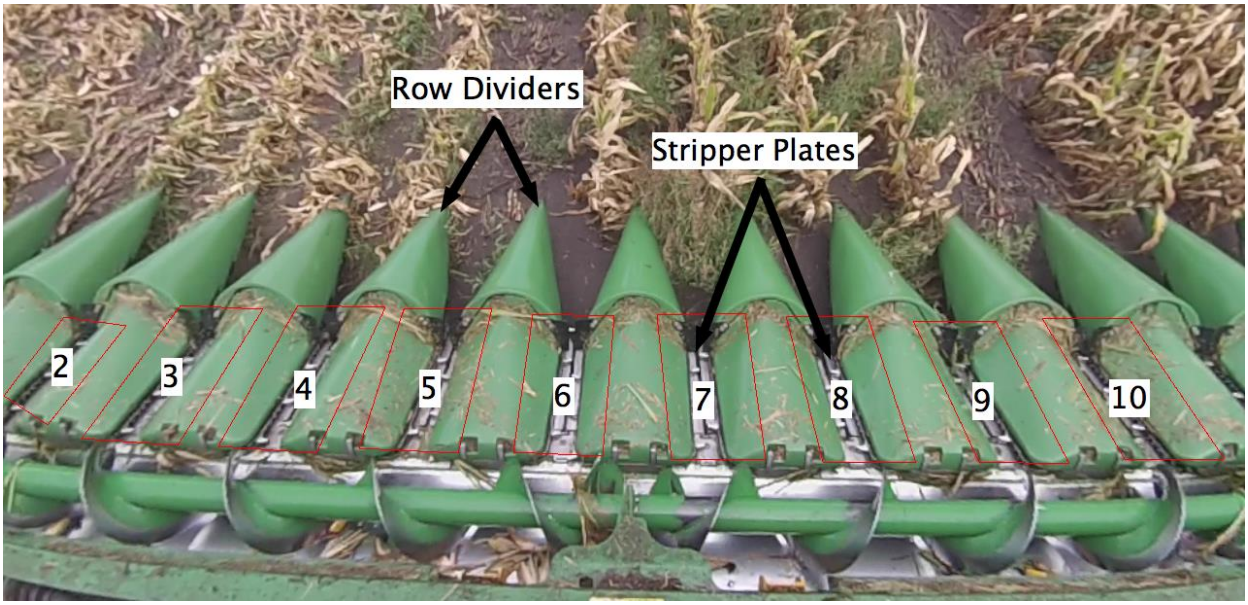


Figure 20. Features and regions of interest used for detecting presence of crop rows.

Two sets of the nine ROI pixel coordinates was manually determined for each of the two videos used in this study. The location of the ROIs remained constant via pixel indexing throughout all images within each video. Because of the dynamics of the harvester and changes in header position caused by variations in the topography throughout the terrain, the header implement was not static

throughout the set of images. Therefore, it was necessary to determine a size for the regions of interest large enough to accommodate the movement of the header. This segmentation process was the initial DIP step for both methods described below.

Method 1: Color-based Image Classification

The color details from an image may provide a significant amount of useful information. These details, also called color descriptors, can simplify the task of object recognition, extraction, and segmentation. Color image processing involves any manipulation to pixel values and can be used to modify images in many different ways such as correcting colors, reducing noise, and sharpening images (Gonzalez & Woods, 2002).

There is a broad range of color image processing applications such as printing, color televisions, and the Internet. Because of this, a method of standardization was needed to facilitate the specification of colors for each application. Color models, also known as color spaces, are defined for this purpose. A color model describes a range of colors in terms of typically 3 or 4 components, and examples of color models include RGB, CMY, CMYK, and HSI (Koschan & Abidi, 2008). The RGB and HSI color models are used here.

RGB Color Model

The RGB color model is the most commonly used color model. It is commonly found in color cameras, and is used to display images on computer monitors. An image in the RGB color model is an MxNx3 array of color pixels, and each pixel is a triplet that corresponds to red, green, and blue color components (Gonzalez, Woods, & Eddins, 2004). The images used in this study were captured in the RGB color space.

Though the images are captured in RGB, this color model is not always suitable for image processing procedures (Liu & Chung, 2011). The red, green, and blue components are highly correlated, making it difficult to use these components to characterize objects by their colors. This

is particularly a challenge when attempting to identify gray objects. As seen in Figure 21, the gray scale in the RGB space is the line where the red, green, and blue components are approximately equal in the 3-dimensional model.

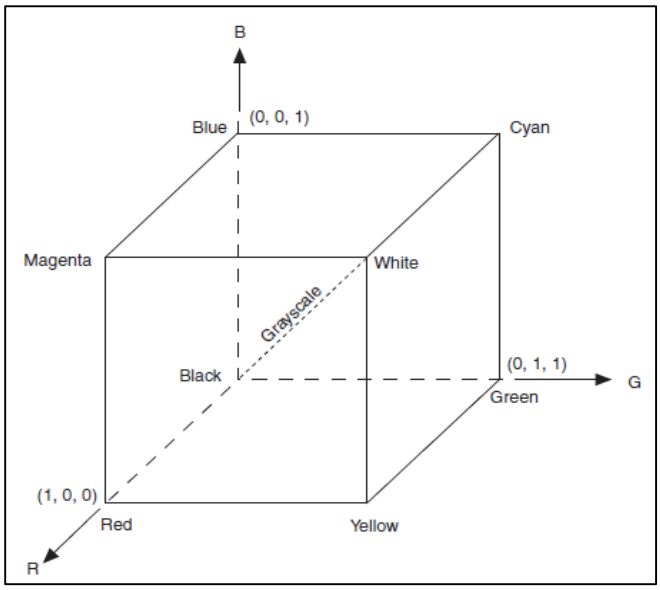


Figure 21. RGB color space model (Instruments, 2016).

HSI Color Model

Characteristics that are generally used to distinguish colors are hue, saturation, and intensity (Koschan & Abidi, 2008). The hue component represents the visible color and is a measure of the wavelength of light on the visible spectrum that produces the most energy (Gonzalez & Woods, 2002). The saturation describes the purity of the color which is influenced by the increased presence of white(Gonzalez & Woods, 2002). The third component in the HSI color model, intensity, does not carry any color information, but is used to describe light that is void of color and ranges from black to grays to white (Gonzalez & Woods, 2002).

The benefit of analyzing images in the HSI color space is that human perception of color corresponds with these three components. In HSI space, color, or hue, is expressed as a single component and not a function of three separate RGB components (Gonzalez & Woods, 2002). Additionally, saturation and intensity components may be useful in providing information on the visibility of the stripper plates within each ROI. The stripper plates in each ROI are distinctly gray which can easily be described with intensity and saturation. In the manual detection of active or inactive ROIs, a correlation was determined between the visibility of the stripper plates and the presence of a crop row. Prior knowledge would suggest that the lack of visibility of the stripper plates would determine an active header status. Likely, the clear visibility of the stripper plates would suggest the lack of a crop row and determine an inactive header status.

Description of Color-based Classification Method

Figure 22 illustrates the image processing pipeline used for this method of extracting color features to determine the state of the region of interest. All images used for threshold determination were first converted to HSI color space, then threshold values were determined for each color component, and finally a simple decision rule was used to classifying ROIs.

Figure 22. Method 1 pipeline using color descriptors for image classification

RGB to HSI Color Transformation

A color transformation is used to transform images from one color model to another. This type of transformation may be useful in extracting more information from the image in terms of a different set of characteristics. The transformation of the images from RGB to HSI is given by the following conversions (Gonzalez & Woods, 2002):

$$H = \begin{cases} \theta, if\ B \leq G \\ 360 - \theta, if\ B > G \end{cases} \quad (1)$$

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{\frac{1}{2}}}\right\} \quad (2)$$

$$S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)] \quad (3)$$

$$I = \frac{1}{3}(R+G+B) \quad (4)$$

where R, G, and B correspond to red, green, and blue pixel values. Each image is represented as an element wise average of the pixels, and the resulting 3x1 feature vector (υ_{image_n}) is used to represent the image during classification.

Threshold Determination

The operational states of the stripper plate regions were determined by significant changes in the pixel distribution for hue, saturation, and intensity. This distribution is determined by examining the HSI histograms of each image such as the ones in Figures 23 and 24. Thresholds were defined based on this pixel distribution on a training set of 160 images that were manually labelled as *active* or *inactive*, indicating the presence or absence of a crop row, respectively. Determining the expected distribution for each class of images required statistical analysis of the distribution of pixels. The two descriptive statistical parameters used to design a decision rule were the mean and standard deviation.

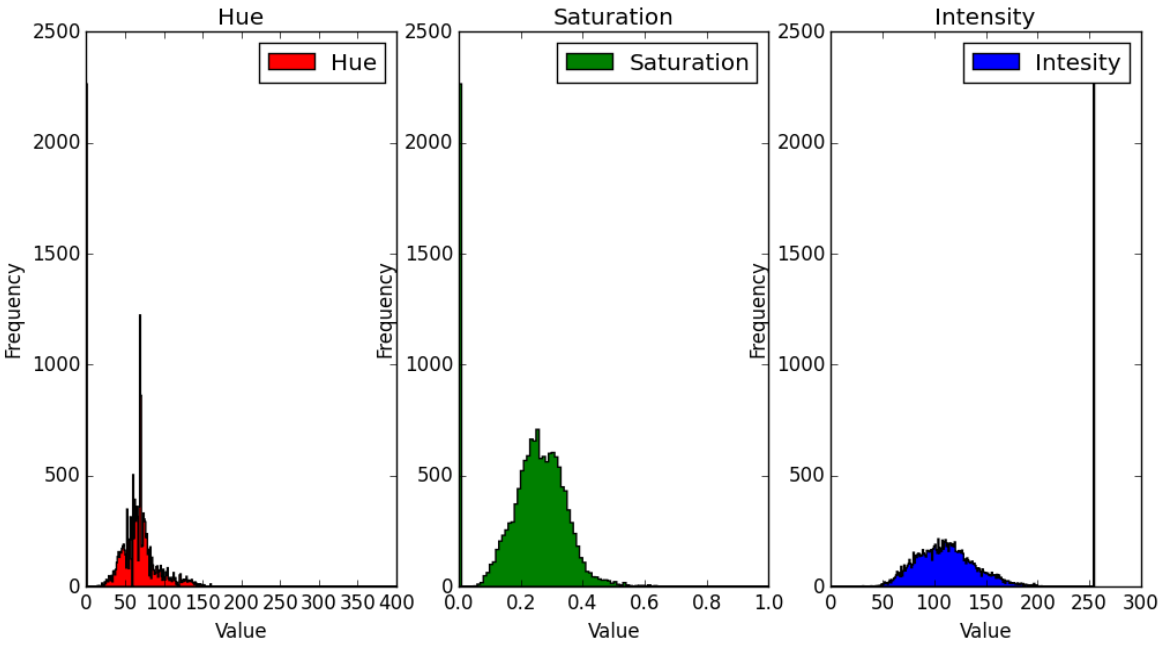


Figure 23. Distribution of pixels for an active ROI.

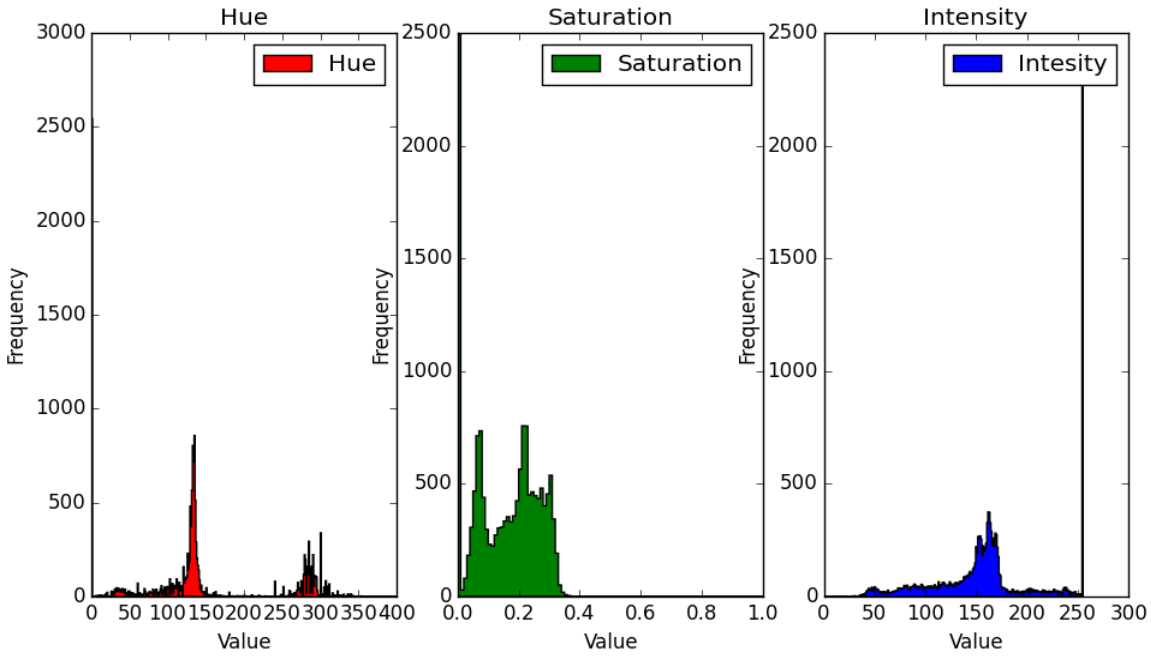


Figure 24. Distribution of pixels for an inactive ROI.

Classification Using Decision Rule

Assigning each of the HSI components with equal weight, thresholds for each component were determined for an active class. The minimum and maximum thresholds are expressed in 3x1 vectors v_{min} and v_{max}, respectively. The decision rule, seen in Equation 5, was a basic component-wise inequality problem where v_{min} and v_{max} were determined at 0.25, 0.5, 0.75, and 1 standard deviations away from the mean values in Table 11. Tests on thresholds greater than 1 standard deviation from the mean resulted in 100 % misclassification of images labelled inactive. For automatic analysis, the mean and standard deviation of pixel values for each image were calculated and written to a Comma Separated Values (.csv) file using the HSI_Histograms.py script found in Appendix B. Equation 5 defines the discriminant function and the decision rule used in the classification scheme illustrated in Figure 25.

$$if\ v_{min}\ \geq\ v_{image_n}\ \leq\ v_{max}\ , then\ image_n\ is\ active$$
$$else,\ \ image_n\ is\ inactive \tag{5}$$

Where
v_{min} is the minimum HSI vector for an active state,
v_{max} is the maximum HSI vector for an active state, and
v_{image_n} is the HSI vector representation for the image.

Table 11. Normalized hue, saturation, and intensity components for classification.

Active State H, S, and I Component Means (Normalized)			
	Hue	**Saturation**	**Intensity**
Mean	0.13	0.23	0.57
Standard Deviation	0.12	0.17	0.28

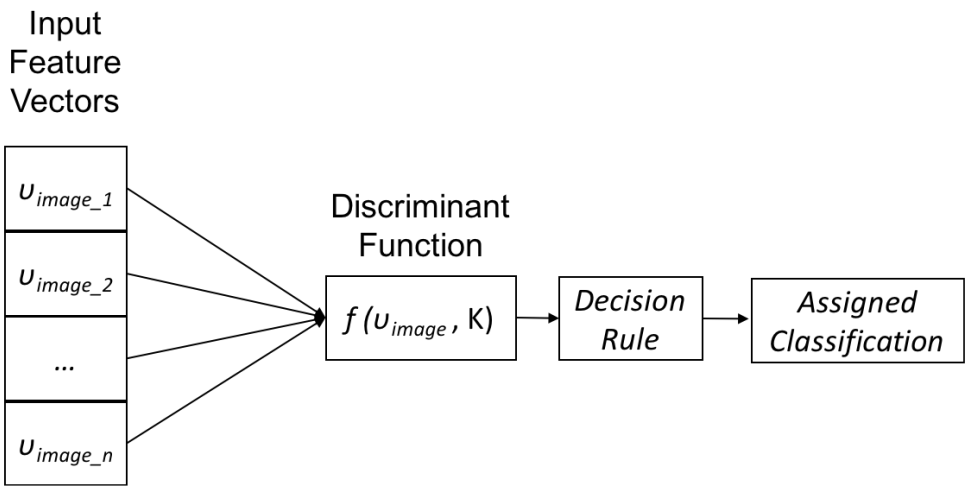


Figure 25. Classification scheme for color-based method, where K is knowledge derived from the training data represented in Equation 5.

Results and Discussion

Method 1 of this experiment implements a simple decision rule based on color parameter thresholds determined from statistical analysis on 160 ROI images. A separate dataset of 160 ROI images was used for testing threshold parameters. Each of these images was manually labelled as active or inactive in order to test the performance of the decision rule. The performance of the decision rule is shown in Table 12 for four separate tests based on the standard deviation coefficient used to determine minimum and maximum thresholds.

The classifier performed very well at a threshold range of 0.5 and 0.75 standard deviations away from the mean values of the hue, saturation, and intensity components. However, slight deviation from 0.5 - 0.75 standard deviations away from the mean caused the frequency of misclassified images to far exceed the number of correctly classified images as seen in Table 12. In conclusion, the proposed color-based model may be a viable means for classifying active from inactive rows. However, color characteristics of hybrid seed corn vary widely from green to beige due to changes

in moisture content throughout the harvesting season. For this reason, a second image classification method was tested that did not rely on color features alone.

Table 12. Confusion matrix for color-based decision rule classification performance.

Threshold	ROI_Class	Active_Actual	Inactive_Actual	Average Accuracy
1 StD	**Active_Predicted**	100 %	77.27 %	60.80 %
	Inactive_Predicted	0 %	21.59 %	
0.75 StD	**Active_Predicted**	100 %	5.68 %	97.16 %
	Inactive_Predicted	0 %	94.32 %	
0.5 StD	**Active_Predicted**	98.61 %	0 %	99.31 %
	Inactive_Predicted	1.39 %	100 %	
0.25 StD	**Active_Predicted**	27.78%	0%	63.89 %
	Inactive_Predicted	72.22 %	100 %	

Method 2: Texture-based Image Classification

For this method, texture features were used for the classification of ROIs as *inactive* or *active*. Unlike the color-based method which only considers the distribution of pixels values, texture features provide information on the spatial arrangement of the pixel values (Shapiro & Stockman, 2001). Examples of the properties that can be measured in terms of texture features include smoothness, coarseness, regularity, and directionality (Gonzalez & Woods, 2002). Texture features provide a more robust means of object recognition or classification because many texture features are scale- and rotation- invariant. The following method extracts the strongest texture features from all the training images in each category. Then for each image, k-nearest neighbors algorithm is used to generate a histogram of distinct features and the frequency of each distinct feature. This histogram is used as a feature vector for representing the ROIs in each image. The feature vector

image representation is used to train a support vector machine classifier. This workflow is illustrated in Figure 26 where the image acquisition and segmentation methods are the same as those used in Method 1. Matlab's Computer Vision Toolbox was used for implementing this approach (The MathWorks, 2016).

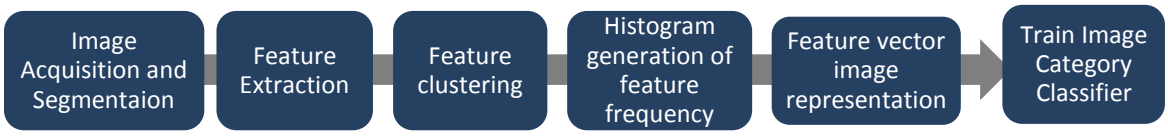


Figure 26. Method 2 pipeline using texture descriptors for image classification.

Bag of Feature Image Classification

The image classification scheme outlined in Figure 26 is prominently used for handling visual classification tasks in computer vision. This process implements a classification model called Bag of Words, the name is derived from the model initial conception in text recognition (Csurka, Dance, Fan, Willamowski, & Bray, 2004). In computer vision, this model may also be referred to as *bag of keypoints* or *bag of features*. Throughout the following sections the process will be referred to as Bag of Features (BoF*).* The BoF approach applied in this study for detecting active header rows closely follows the methods described by Csurka et al. (2004) with few exceptions.

The first step in BoF, illustrated in Figure 27, is feature extraction. For each category of images in the training data set, all detected Speeded Up Robust Features (SURF) are computed. The next section describes SURF descriptors. The training data set consisted of 250 randomly selected images for each classification: active and inactive. For each category of header ROI images used, 4,000 to 16,000 features were detected.

Next, from the training set of images, a bag of features is created for each image. Each bag of features serves to represent each image during the training of a classifier (Csurka et al., 2004). For classifier training, the distance between feature vectors is computed and used to determine the classification of each image. Though many classifiers such as Neural Networks and Naïve Bayes may be used, a support vector machine classifier was chosen for its repeated success in BoF image category classification problems. The BoF method described here is outlined in Figure 27.

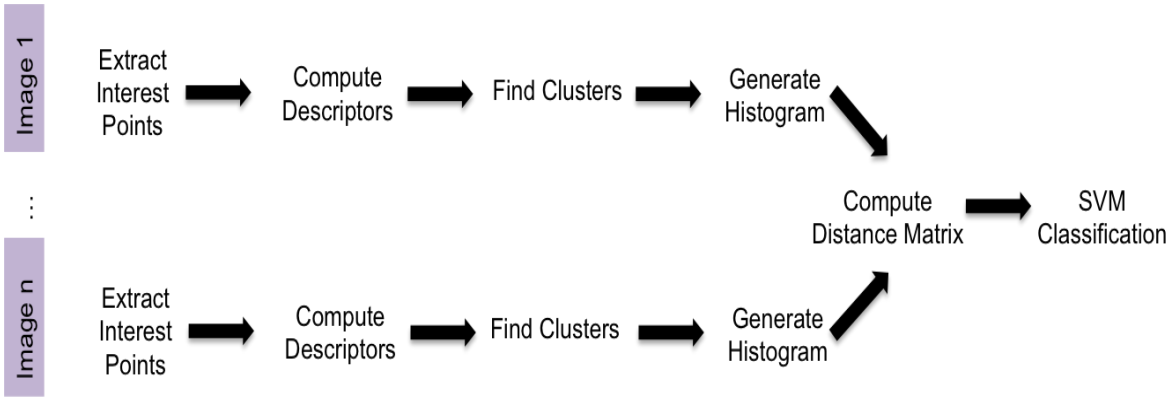


Figure 27. Bag of words image classification method.

Speeded Up Robust Features

There are many types of texture features that can be used for object detection purposes such as moment invariants, blob features, and Gaussian derivatives. Here, the focus was on using a specific feature detector called Speeded Up Robust Features (SURF). SURF are scale- and rotation-invariant descriptors that are highly discriminative and computationally inexpensive (Bay, Tuytelaars, & Van Gool, 2006a).

The process of SURF detection described by Bay, Tuytelaars, and Van Gool (2006b) can be summarized in three main steps: interest point detection, local neighborhood description, and matching. SURF detects distinct, local blob features within an image by using the determinant of

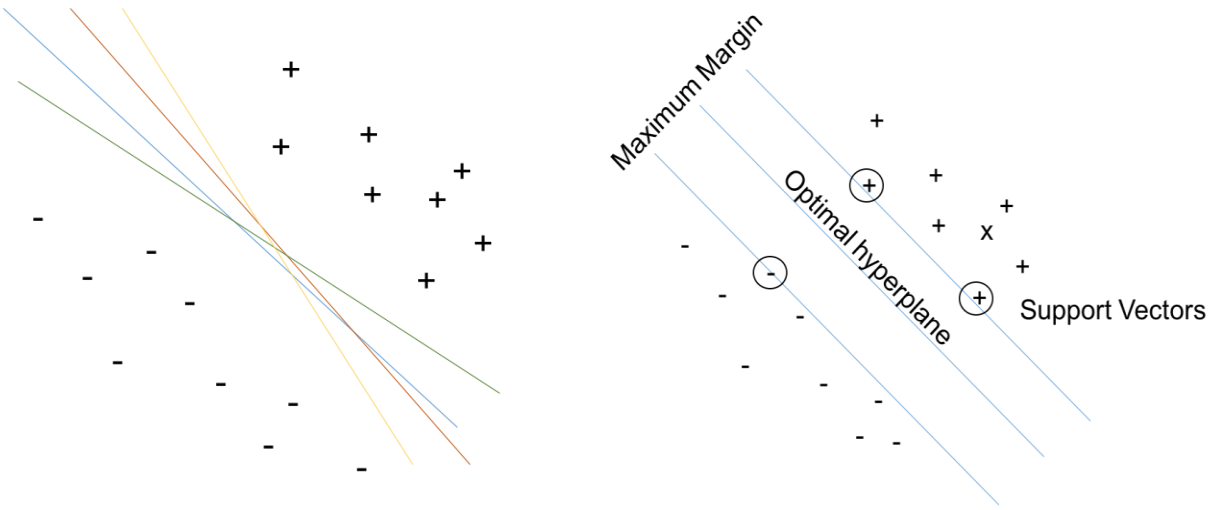
the Hessian matrix of the image. These blobs become points of interest. Next, image features are described by the distribution of pixels that surround each interest point. Finally, for object recognition purposes, these features are used for the processing of other ROIs.

SURF is widely used in image processing problems for object detection and is a patented detector and descriptor that requires a license for use. In this study, SURF tools were accessed through MATLAB's Computer Vision Toolbox.

Support Vector Machines

The classification problem presented was made up of only two classes: active and inactive. Classification problems such as this one can be solved with support vector machines (SVM), which are designed for binary classification. SVMs are a supervised, discriminative classifier that requires a labeled training set of data (Duda, Hart, & Stork, 2012). In this test, the labeled training set of data comes from the bags of features created from the training set of images.

SVMs are maximum margin classifiers. This means that the algorithm finds a hyperplane that totally separates the two classes with the maximum distance from hyperplane to any feature vector from either class. In Figure 28-A, notice how multiple hyperplanes can be fitted to separate the two classes, but the optimal hyperplane in Figure 28-B identifies the maximum margin between classes (Cortes & Vapnik, 1995).



(A) (B)

Figure 28. (A) Various possible decision boundaries (B) Optimal decision boundary using SVM

Methods such as Naïve Bayes and Linear Regression are able to find a decision boundary between classes. However support vector machines use support vectors to find the optimal decision boundary with the greatest marginal distance between classes. For the purpose of training and testing the image category classification techniques, random images were selected from a database of over 16,000 labelled images. Each training set consisted of two categories: active and inactive. Each category contains 250 randomly selected for training. The testing set contained 1,000 randomly selected validation images to evaluate the performance of the SVM classifier (k = 100, linear kernel).

Results and Discussion

Since the images in the dataset were collected under natural lighting conditions, a change in the direction of the harvester could lead to shadow interferences, pixel saturation, and insufficient lighting. Therefore, multiple sets of training images were used to create SVM classifiers. Table 13 reports the average accuracy for the SVM performance on all combinations of training and

testing sets. Two set of video data were used, and the passes indicated the harvester pass in the field. Subsequent passes represent a change in the direction of the harvester in the field. Individual confusion matrices for each test run can be found in Appendix C.

Table 13. Average accuracy for each combination of training and testing data.

Training Sets	Testing Sets	
	Video 1 All	**Video 2 All**
Video 1 All	95 %	88 %
Video 1 Pass 1	83 %	74 %
Video 1 Pass 2	84 %	87 %
Video 2 All	76 %	96 %
Video 2 Pass 1	71 %	94 %
Video 2 Pass 2	82 %	96 %

Overall, the BoF approach for image category classification achieved classification above 83% when using training images from within the same video as testing image. Additionally, the classifier performed above 71% when using any combination of training and testing sets from two separate videos and four different passes in the field. SVMs trained with the images throughout the entirety of the same video as the testing images had the greatest performance of 95 – 96%. The results in Table 13 would suggest that training SVMs with images from a single pass in the field performed significantly less than if training images from segments of the entire video were used.

Pattern recognition models and computer vision techniques provide a powerful tool that can replace the need for expert advice on redundant tasks. Computer vision can provide sight to agricultural machinery and pattern recognition tools can be used to train systems to make decisions based on what the machines see. The success of using computer vision in agricultural environments could lead to many crop management solutions such as time lag determination, weed mapping, and field process automation.

Recommendations

The concept study presented in this chapter is just the start of the development of a vision-based system for effective header width determination. Further research should be conducted to test the performance of the system under extreme conditions. One of the main challenges of using a vision system is that images acquired in an agricultural environment are exposed to the elements. Furthermore, certain crops are not exclusively harvested under daylight conditions and consideration must be made to the change in lighting throughout the day. Suggested recommendations for future research needs are as follows:

- Further development of this study should incorporate automatic detection of the header implement and each set of stripper plates.
- ROIs should automatically adjust to the extent of the stripper plates.
- Additional video data should be recorded under various possible weather conditions considered suitable for harvesting.
- Additional video data should be recorded under various possible lighting conditions.
- Image acquisition systems should have a field of view of the entire span of a header implement.
- Image analysis methods that combine the methods used in this study should be testing.

Conclusions

The results of this study will contribute to the overall effort of increasing the performance the yield monitoring system for seed corn developed by the University of Tennessee. The main objectives of this study were:

1) to evaluate a rule-based technique for measuring site-specific yield variability within a seed corn field,

2) To validate the yield measurement accuracy under field harvest conditions, and

3) To evaluate computer vision techniques for row-crop detection.

The rule-based techniques offers multiple level of yield determination for the users. At the chase cart level of yield determination, weight measurements calculated by Yield Analyzer were within approximately 6.0 % of the tractor trailer loads. Polygon-level performance varied among fields, but for three of the five fields, polygon yield measurements compared mostly < 20.0 % from the chase cart yield measurements. One-way repeated measures analysis resulted in the three of the five fields showing no significance between various polygon length measurements. A post hoc multiple comparisons analysis identified the cause for significance was due to differences between yield measurements at the low polygon lengths (20 and 30 m) and high polygon lengths (60 and 80 m).

The overall performance of both vision-based methods studied in this paper would suggest that a vision-based system can assist in the task of determining effective header width. The color-based method performed > 97.0 % average accuracy when a standard deviation coefficient of 0.75 was used. The texture-based method performed with an average accuracy >70 % for any combination of training images used, and > 95 % average accuracy when training images and testing images from the same video data set were used.

References

Bay, H., Tuytelaars, T., & Van Gool, L. (2006a). Surf: Speeded up robust features *Computer vision–ECCV 2006* (pp. 404-417): Springer.

Bay, H., Tuytelaars, T., & Van Gool, L. (2006b). *Surf: Speeded up robust features*. Paper presented at the European conference on computer vision.

Beck, A., Searcy, S., & Roades, J. (2001). Yield data filtering techniques for improved map accuracy. *Applied Engineering in Agriculture, 17*(4), 423.

Benalia, S., Cubero, S., Prats-Montalbán, J. M., Bernardi, B., Zimbalatti, G., & Blasco, J. (2016). Computer vision for automatic quality inspection of dried figs (Ficus carica L.) in real-time. *Computers and Electronics in Agriculture, 120*, 17-25. doi: http://dx.doi.org/10.1016/j.compag.2015.11.002

Blackmore, B., & Marshall, C. (1996). Yield mapping; errors and algorithms. *Precision Agriculture*(precisionagricu3), 403-415.

Blackmore, S. (1999). Remedial correction of yield map data. *Precision Agriculture, 1*(1), 53-66.

Capehart, T. (2016). Corn. Retrieved July 3, 2016, from http://www.ers.usda.gov/topics/crops/corn.aspx

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning, 20*(3), 273-297.

Csurka, G., Dance, C., Fan, L., Willamowski, J., & Bray, C. (2004). *Visual categorization with bags of keypoints*. Paper presented at the Workshop on statistical learning in computer vision, ECCV.

Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification*: John Wiley & Sons.

Fernades-Cornejo, J. (2004). The Seed Industry in U.S. Agriculture. Washington, D.C.: United States Department of Agriculture.

Gonzalez, R. C., & Woods, R. E. (2002). Digital image processing: Prentice hall Upper Saddle River.

Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2004). *Digital image processing using MATLAB*: Pearson Prentice Hall.

Griffin, T. W. (2010). The Spatial Analysis of Yield Data. In M. A. Oliver (Ed.), *Geostatistical Applications for Precision Agriculture* (pp. 89-116). Dordrecht: Springer Netherlands.

Griliches, Z. (1957). Hybrid Corn: An Exploration in the Economics of Technological Change. *Econometrica, 25*(4), 501-522. doi: 10.2307/1905380

Instruments, N. (2016). Color Spaces. Retrieved May 18, 2016, from http://zone.ni.com/reference/en-XX/help/372916M-01/nivisionconcepts/color_spaces/

Jiang, G., Wang, Z., & Liu, H. (2015). Automatic detection of crop rows based on multi-ROIs. *Expert Systems with Applications, 42*(5), 2429-2441. doi: http://dx.doi.org/10.1016/j.eswa.2014.10.033

Koschan, A., & Abidi, M. (2008). *Digital color image processing*: John Wiley & Sons.

Kurtulmuş, F., & Kavdir, İ. (2014). Detecting corn tassels using computer vision and support vector machines. *Expert Systems with Applications, 41*(16), 7390-7397.

Liu, C.-C., & Chung, P.-C. (2011). Objects extraction algorithm of color image using adaptive forecasting filters created automatically. *International Journal of Innovative Computing, Information and Control, 7*(10), 5771-5787.

Luck, J. D., & Fulton, J. P. (2014). Best Managment Practices for Collecting Accurate Yield Data and Avoiding Errors During Harvest: University of Nebraska Extension

Luck, J. D., Mueller, N., & Fulton, J. P. (2015). Improving Yield Map Quality y Reducing Errors Through Yield Data File Post-Processing: University of Nebraska - Lincoln Extension.

Montalvo, M., Guerrero, J. M., Romeo, J., Emmi, L., Guijarro, M., & Pajares, G. (2013). Automatic expert system for weeds/crops identification in images from maize fields. *Expert Systems with Applications, 40*(1), 75-82.

Muscato, G., Prestifilippo, M., Abbate, N., & Rizzuto, I. (2005). A prototype of an orange picking robot: past history, the new robot and experimental results. *Industrial Robot: An International Journal, 32*(2), 128-138. doi: doi:10.1108/01439910510582255

Nielson, R. L. (2014). Wandering Swath Width Syndrome: Yield Monitor Errors (A. Dept., Trans.): Purdue University.

Ping, J., & Dobermann, A. (2005). Processing of yield map data. *Precision Agriculture, 6*(2), 193-212.

Reitz, P., & Kutzbach, H. (1996). Investigations on a particular yield mapping system for combine harvesters. *Computers and Electronics in Agriculture, 14*(2), 137-150.

Schneider, S., Von Rawlins, S., Han, S., Evans, R., & Campbell, R. (1996). Precision agriculture for potatoes in the Pacific Northwest. *Precision Agriculture*(precisionagricu3), 443-452.

Shapiro, L., & Stockman, G. C. (2001). Computer vision. 2001. *ed: Prentice Hall*.

Simbahan, G., Dobermann, A., & Ping, J. (2004). Screening yield monitor data improves grain yield maps. *Agronomy Journal, 96*(4), 1091-1102.

Sudduth, K. A., & Drummond, S. T. (2007). Yield Editor. *Agronomy Journal, 99*(6), 1471-1482.

Swain, K. C., Zaman, Q. U., Schumann, A. W., Percival, D. C., & Bochtis, D. D. (2010). Computer vision system for wild blueberry fruit yield mapping. *Biosystems Engineering, 106*(4), 389-394.

The MathWorks, I. (2016). Image Category Classification Using Bag of Features (r2016a): The MathWorks, Inc.

Thomas, D., Perry, C., Vellidis, G., Durrence, J., Kutz, L., Kvien, C., . . . Hamrita, T. (1999). Development and implementation of a load cell yield monitor for peanut. *Applied Engineering in Agriculture, 15*(3), 211.

Thylen, L., & Murphy, D. P. (1996). The control of errors in momentary yield data from combine harvesters. *Journal of agricultural engineering research, 64*(4), 271-278.

Vansichen, R., & De Baerdemaeker, J. (1992). *Measuring the actual cutting width of a combine by means of an ultrasonic distance sensor.* Paper presented at the Proceedings of international scientific conference on trends in agricultural engineering.

Walter, J., Hofman, V., & Backer, L. (1996). Site-specific sugarbeet yield monitoring. *Precision Agriculture*(precisionagricu3), 835-844.

Whitney, J. D., Miller, W. M., Wheaton, T., Salyani, M., & Schueller, J. K. (1999). Precision farming applications in Florida citrus. *Applied Engineering in Agriculture, 15*(5), 399.

Wilkerson, J. B. (2015). *Yield Monitoring Update Presentation*. PowerPoint Presentation. University of Tennessee.

Wright, H. (1980). Commercial hybrid seed production. *Hybridization of Crop Plants*(hybridizationof), 161-176.

Appendix

Appendix A – SAS Output

Field 1: One-way repeated measures analysis and multiple comparisons results.

Covariance Parameter Estimates		
Cov Parm	**Subject**	**Estimate**
CS	subject	281006
Residual		130879

Fit Statistics	
-2 Res Log Likelihood	7503.7
AIC (Smaller is Better)	7507.7
AICC (Smaller is Better)	7507.7
BIC (Smaller is Better)	7512.9

Null Model Likelihood Ratio Test		
DF	**Chi-Square**	**Pr > ChiSq**
1	323.70	<.0001

Type 3 Tests of Fixed Effects				
Effect	**Num DF**	**Den DF**	**F Value**	**Pr > F**
distance	4	396	0.20	0.9380

Least Squares Means						
Effect	**distance**	**Estimate**	**Standard Error**	**DF**	**t Value**	**Pr > |t|**
distance	Yield_20m	1023.90	64.1783	396	15.95	<.0001
distance	Yield_30m	1015.23	64.1783	396	15.82	<.0001
distance	Yield_40m	1029.33	64.1783	396	16.04	<.0001

distance	Yield_60m	989.93	64.1783	396	15.42	<.0001
distance	Yield_80m	1001.48	64.1783	396	15.60	<.0001

Differences of Least Squares Means

Effect	distance	_distance	Estimate	Standard Error	DF	t Value	Pr > |t|	Adjustment	Adj P
distance	Yield_20m	Yield_30m	8.6700	51.1624	396	0.17	0.8655	Tukey-Kramer	0.9998
distance	Yield_20m	Yield_40m	-5.4300	51.1624	396	-0.11	0.9155	Tukey-Kramer	1.0000
distance	Yield_20m	Yield_60m	33.9700	51.1624	396	0.66	0.5071	Tukey-Kramer	0.9639
distance	Yield_20m	Yield_80m	22.4200	51.1624	396	0.44	0.6615	Tukey-Kramer	0.9923
distance	Yield_30m	Yield_40m	-14.1000	51.1624	396	-0.28	0.7830	Tukey-Kramer	0.9987

Differences of Least Squares Means

Effect	distance	_distance	Estimate	Standard Error	DF	t Value	Pr > |t|	Adjustment	Adj P
distance	Yield_30m	Yield_60m	25.3000	51.1624	396	0.49	0.6212	Tukey-Kramer	0.9879
distance	Yield_30m	Yield_80m	13.7500	51.1624	396	0.27	0.7883	Tukey-Kramer	0.9989
distance	Yield_40m	Yield_60m	39.4000	51.1624	396	0.77	0.4417	Tukey-Kramer	0.9391
distance	Yield_40m	Yield_80m	27.8500	51.1624	396	0.54	0.5865	Tukey-Kramer	0.9826
distance	Yield_60m	Yield_80m	-11.5500	51.1624	396	-0.23	0.8215	Tukey-Kramer	0.9994

Field 2: One-way repeated measures analysis and multiple comparisons results.

Covariance Parameter Estimates		
Cov Parm	**Subject**	**Estimate**
CS	subject	356453
Residual		187227

Fit Statistics	
-2 Res Log Likelihood	7670.1
AIC (Smaller is Better)	7674.1
AICC (Smaller is Better)	7674.1
BIC (Smaller is Better)	7679.3

Null Model Likelihood Ratio Test		
DF	**Chi-Square**	**Pr > ChiSq**
1	294.72	<.0001

Type 3 Tests of Fixed Effects				
Effect	**Num DF**	**Den DF**	**F Value**	**Pr > F**
distance	4	396	2.32	0.0567

Least Squares Means						
Effect	**distance**	**Estimate**	**Standard Error**	**DF**	**t Value**	**Pr > |t|**
distance	Yield_20m	1572.09	73.7347	396	21.32	<.0001
distance	Yield_30m	1472.93	73.7347	396	19.98	<.0001
distance	Yield_40m	1562.21	73.7347	396	21.19	<.0001

distance	Yield_60m	1412.78	73.7347	396	19.16	<.0001
distance	Yield_80m	1511.88	73.7347	396	20.50	<.0001

Differences of Least Squares Means

Effect	distance	_distance	Estimate	Standard Error	DF	t Value	Pr > |t|	Adjustment	Adj P
distance	Yield_20m	Yield_30m	99.1600	61.1926	396	1.62	0.1059	Tukey-Kramer	0.4851
distance	Yield_20m	Yield_40m	9.8800	61.1926	396	0.16	0.8718	Tukey-Kramer	0.9998
distance	Yield_20m	Yield_60m	159.31	61.1926	396	2.60	0.0096	Tukey-Kramer	0.0716
distance	Yield_20m	Yield_80m	60.2100	61.1926	396	0.98	0.3257	Tukey-Kramer	0.8625
distance	Yield_30m	Yield_40m	-89.2800	61.1926	396	-1.46	0.1454	Tukey-Kramer	0.5898

Differences of Least Squares Means

Effect	distance	_distance	Estimate	Standard Error	DF	t Value	Pr > |t|	Adjustment	Adj P
distance	Yield_30m	Yield_60m	60.1500	61.1926	396	0.98	0.3262	Tukey-Kramer	0.8629
distance	Yield_30m	Yield_80m	-38.9500	61.1926	396	-0.64	0.5248	Tukey-Kramer	0.9690
distance	Yield_40m	Yield_60m	149.43	61.1926	396	2.44	0.0150	Tukey-Kramer	0.1065
distance	Yield_40m	Yield_80m	50.3300	61.1926	396	0.82	0.4113	Tukey-Kramer	0.9236
distance	Yield_60m	Yield_80m	-99.1000	61.1926	396	-1.62	0.1061	Tukey-Kramer	0.4858

Field 3: One-way repeated measures analysis and multiple comparisons results.

Covariance Parameter Estimates		
Cov Parm	**Subject**	**Estimate**
CS	subject	163715
Residual		121350

Fit Statistics	
-2 Res Log Likelihood	7425.1
AIC (Smaller is Better)	7429.1
AICC (Smaller is Better)	7429.1
BIC (Smaller is Better)	7434.3

Null Model Likelihood Ratio Test		
DF	**Chi-Square**	**Pr > ChiSq**
1	220.08	<.0001

Type 3 Tests of Fixed Effects				
Effect	**Num DF**	**Den DF**	**F Value**	**Pr > F**
distance	4	396	4.75	0.0009

Least Squares Means						
Effect	**distance**	**Estimate**	**Standard Error**	**DF**	**t Value**	**Pr > |t|**
distance	Yield_20m	933.49	53.3915	396	17.48	<.0001
distance	Yield_30m	875.35	53.3915	396	16.39	<.0001
distance	Yield_40m	872.10	53.3915	396	16.33	<.0001

distance	Yield_60m	789.33	53.3915	396	14.78	<.0001
distance	Yield_80m	742.52	53.3915	396	13.91	<.0001

Differences of Least Squares Means

Effect	distance	_distance	Estimate	Standard Error	DF	t Value	Pr > |t|	Adjustment	Adj P
distance	Yield_20m	Yield_30m	58.1400	49.2646	396	1.18	0.2386	Tukey-Kramer	0.7628
distance	Yield_20m	Yield_40m	61.3900	49.2646	396	1.25	0.2135	Tukey-Kramer	0.7242
distance	Yield_20m	Yield_60m	144.16	49.2646	396	2.93	0.0036	Tukey-Kramer	0.0297
distance	Yield_20m	Yield_80m	190.97	49.2646	396	3.88	0.0001	Tukey-Kramer	0.0012
distance	Yield_30m	Yield_40m	3.2500	49.2646	396	0.07	0.9474	Tukey-Kramer	1.0000

Differences of Least Squares Means

Effect	distance	_distance	Estimate	Standard Error	DF	t Value	Pr > |t|	Adjustment	Adj P
distance	Yield_30m	Yield_60m	86.0200	49.2646	396	1.75	0.0816	Tukey-Kramer	0.4070
distance	Yield_30m	Yield_80m	132.83	49.2646	396	2.70	0.0073	Tukey-Kramer	0.0563
distance	Yield_40m	Yield_60m	82.7700	49.2646	396	1.68	0.0937	Tukey-Kramer	0.4475
distance	Yield_40m	Yield_80m	129.58	49.2646	396	2.63	0.0089	Tukey-Kramer	0.0669
distance	Yield_60m	Yield_80m	46.8100	49.2646	396	0.95	0.3426	Tukey-Kramer	0.8769

Field 4: One-way repeated measures analysis and multiple comparisons results.

Covariance Parameter Estimates		
Cov Parm	**Subject**	**Estimate**
CS	subject	207086
Residual		127044

Fit Statistics	
-2 Res Log Likelihood	7464.3
AIC (Smaller is Better)	7468.3
AICC (Smaller is Better)	7468.3
BIC (Smaller is Better)	7473.5

Null Model Likelihood Ratio Test		
DF	**Chi-Square**	**Pr > ChiSq**
1	259.50	<.0001

Type 3 Tests of Fixed Effects				
Effect	**Num DF**	**Den DF**	**F Value**	**Pr > F**
distance	4	396	1.05	0.3826

Least Squares Means						
Effect	**distance**	**Estimate**	**Standard Error**	**DF**	**t Value**	**Pr > |t|**
distance	Yield_20m	1671.85	57.8040	396	28.92	<.0001
distance	Yield_30m	1585.95	57.8040	396	27.44	<.0001

distance	Yield_40m	1632.68	57.8040	396	28.25	<.0001
distance	Yield_60m	1583.66	57.8040	396	27.40	<.0001
distance	Yield_80m	1617.69	57.8040	396	27.99	<.0001

Differences of Least Squares Means

Effect	distance	_distance	Estimate	Standard Error	DF	t Value	Pr > |t|	Adjustment	Adj P
distance	Yield_20m	Yield_30m	85.9000	50.4071	396	1.70	0.0891	Tukey-Kramer	0.4326
distance	Yield_20m	Yield_40m	39.1700	50.4071	396	0.78	0.4376	Tukey-Kramer	0.9371
distance	Yield_20m	Yield_60m	88.1900	50.4071	396	1.75	0.0810	Tukey-Kramer	0.4049
distance	Yield_20m	Yield_80m	54.1600	50.4071	396	1.07	0.2833	Tukey-Kramer	0.8196

Differences of Least Squares Means

Effect	distance	_distance	Estimate	Standard Error	DF	t Value	Pr > |t|	Adjustment	Adj P
distance	Yield_30m	Yield_40m	-46.7300	50.4071	396	-0.93	0.3545	Tukey-Kramer	0.8863
distance	Yield_30m	Yield_60m	2.2900	50.4071	396	0.05	0.9638	Tukey-Kramer	1.0000
distance	Yield_30m	Yield_80m	-31.7400	50.4071	396	-0.63	0.5293	Tukey-Kramer	0.9702
distance	Yield_40m	Yield_60m	49.0200	50.4071	396	0.97	0.3314	Tukey-Kramer	0.8675
distance	Yield_40m	Yield_80m	14.9900	50.4071	396	0.30	0.7663	Tukey-Kramer	0.9983
distance	Yield_60m	Yield_80m	-34.0300	50.4071	396	-0.68	0.5000	Tukey-Kramer	0.9617

Field 5: One-way repeated measures analysis and multiple comparisons results.

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
CS	subject	452155
Residual		151775

Fit Statistics

-2 Res Log Likelihood	7530.4
AIC (Smaller is Better)	7534.4
AICC (Smaller is Better)	7534.4
BIC (Smaller is Better)	7539.6

Null Model Likelihood Ratio Test

DF	Chi-Square	Pr > ChiSq
1	405.65	<.0001

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
distance	4	392	3.43	0.0090

Least Squares Means

Effect	distance	Estimate	Standard Error	DF	t Value	Pr > |t|
distance	Yield_20m	1346.90	78.1044	392	17.24	<.0001
distance	Yield_30m	1316.57	78.1044	392	16.86	<.0001
distance	Yield_40m	1242.41	78.1044	392	15.91	<.0001

distance	Yield_60m	1235.25	78.1044	392	15.82	<.0001
distance	Yield_80m	1163.13	78.1044	392	14.89	<.0001

Differences of Least Squares Means

Effect	distance	_distance	Estimate	Standard Error	DF	t Value	Pr > |t|	Adjustment	Adj P
distance	Yield_20m	Yield_30m	30.3333	55.3728	392	0.55	0.5841	Tukey-Kramer	0.9822
distance	Yield_20m	Yield_40m	104.48	55.3728	392	1.89	0.0599	Tukey-Kramer	0.3261
distance	Yield_20m	Yield_60m	111.65	55.3728	392	2.02	0.0445	Tukey-Kramer	0.2601
distance	Yield_20m	Yield_80m	183.77	55.3728	392	3.32	0.0010	Tukey-Kramer	0.0087

Differences of Least Squares Means

Effect	distance	_distance	Estimate	Standard Error	DF	t Value	Pr > |t|	Adjustment	Adj P
distance	Yield_30m	Yield_40m	74.1515	55.3728	392	1.34	0.1813	Tukey-Kramer	0.6669
distance	Yield_30m	Yield_60m	81.3131	55.3728	392	1.47	0.1428	Tukey-Kramer	0.5836
distance	Yield_30m	Yield_80m	153.43	55.3728	392	2.77	0.0059	Tukey-Kramer	0.0460
distance	Yield_40m	Yield_60m	7.1616	55.3728	392	0.13	0.8972	Tukey-Kramer	0.9999
distance	Yield_40m	Yield_80m	79.2828	55.3728	392	1.43	0.1530	Tukey-Kramer	0.6075
distance	Yield_60m	Yield_80m	72.1212	55.3728	392	1.30	0.1935	Tukey-Kramer	0.6899

Appendix B – Image Processing Scripts

Image Segmentation Script

```
# ROI Module
# Written by Fatima Murillo
# Written on November 17, 2015
# Last Updated on July 24, 2015

"""
This module contains functions for reading ROI boundary information from
.txt
files. The location of the files has a defaulted path, but users can
provide a
new path for a new set of ROI Coordinates if they have been adjusted.

setBounds function returns the slopesIntercepts and xyRange dictionaries

splinWin function extracts ROIs from an image and saves them as individual
image files
"""

import re  # Provides regular expression matching operations
import os  # Miscellaneous operating system interfaces
import csv # Implements classes to read and write tabular data in CSV
format
import copy # Provides generic shallow and deep copy operations
import skimage # Collection of algorthms for image processing
from skimage import io # Utilities to read and write images
import random


# select random images from image directory
i = 0
imageList = []
imageDir = '/Users/fatimamurillo/Research/Images/video1Im/v1w6p1Im'
while i < 200:
    randIm = random.choice(os.listdir(imageDir))
    if randIm == '.DS_Store':
        continue
    elif randIm == 'Thumbs.db':
        continue
    elif randIm in imageList:
        continue
    else:
        imageList.append(randIm)
        print(randIm)
    i += 1

# setBounds function
def  setBounds(coordPath):
    # coordPath is the path to the directory containing the ROI coord
files
    # Extract coordinates of 4 corners of each quadrangle (ROI) from
directory containing coordinate .txt files
    xyCoords = {}
```

```
    while True:
        if os.path.exists(coordPath):
            break
        else:
            print("That was not a valid path to coordinates directory.")
            coordPath = input('Please enter valid path to coordinates
directory:')
    for filename in os.listdir(coordPath):
        print(filename)
        if filename == '.DS_Store':
            continue
        else:
            windowNum = re.findall(r'\d+', filename)
            window = windowNum[3]
            nfn = coordPath + '/' + filename
            with open(nfn, 'r') as f:
                reader = csv.reader(f, delimiter = '\t')
                xyCoords["window{0}".format(window)] = list()
                for row in reader:
                    xEntry = float(row[0])
                    yEntry = float(row[1])
                    newEntry = [int(xEntry),int(yEntry)]
                    # For each window, dictionary includes points as
follows: [[A],[B],[C],[D]]
                    # A to D are the four corners of each ROI from top
left to bottom left
                    xyCoords["window{0}".format(window)].append(newEntry)
            f.close()

    # Determine max and min x and y coordinates for use in ROI extraction
    xyRange = {}
    windowsList = list()
    for ROI in xyCoords:
        currentWindow = str(ROI)
        windowNum = re.findall(r'\d+', currentWindow)
        windowsList.append(currentWindow)
        xyRange["window{0}".format(windowNum[0])] = list()
        maxX = xyCoords[ROI][0][0]
        minX = xyCoords[ROI][0][0]
        maxY = xyCoords[ROI][0][1]
        minY = xyCoords[ROI][0][1]
        for xy in xyCoords[currentWindow]:
            if xy[0] > maxX:
                maxX = xy[0]
            elif xy[0] < minX:
                minX = xy[0]
            else:
                continue
            if xy[1] > maxY:
                maxY = xy[1]
            elif xy[1] < minY:
                minY = xy[1]
            else:
                continue
```

```
        rangeEntry = [maxX, minX, maxY, minY]
        xyRange["window{0}".format(windowNum[0])].append(rangeEntry)

    # Determine the slope and y-intercept for each line segment for each
window
    slopesIntercepts = {}
    triangleCentroids = {}
    SITCentroids = {}
    ROICentroids = {}
    assignmentCheck = {}
    thresholdCheckImage = list()
    for each in xyCoords:
        wNum = re.findall(r'\d+', each)
        w = wNum[0]
        slopesIntercepts["window{0}".format(w)] = list()
        triangleCentroids["window{0}".format(w)] = list()
        SITCentroids["window{0}".format(w)] = list()
        ROICentroids["window{0}".format(w)] = list()
        assignmentCheck["window{0}".format(w)] = list()
        xA = xyCoords[each][0][0]
        yA = xyCoords[each][0][1]
        xB = xyCoords[each][1][0]
        yB = xyCoords[each][1][1]
        xC = xyCoords[each][2][0]
        yC = xyCoords[each][2][1]
        xD = xyCoords[each][3][0]
        yD = xyCoords[each][3][1]
        # 1 corresponds to line AB
        slope1 = (yB-yA)/(xB-xA)
        intercept1 = yA - (xA*slope1)
        # 2 corresponds to line BC
        slope2 = (yC-yB)/(xC-xB)
        intercept2 = yB - (xB*slope2)
        # 3 corresponds to line CD
        slope3 = (yD - yC)/(xD-xC)
        intercept3 = yC - (xC*slope3)
        # 4 correcsponds to line DA
        slope4 = (yA-yD)/(xA-xD)
        intercept4 = yD - (xD*slope4)
        entrySI = [[slope1,intercept1],[slope2, intercept2],[slope3,
intercept3],[slope4, intercept4]]
        slopesIntercepts["window{0}".format(w)].append(entrySI)

        # Calculate the centroids of all triangles within the
quadrilateral given coordinates of the corners
        xABC = (xA+xB+xC)/3
        yABC = (yA+yB+yC)/3
        xBCD = (xB+xC+xD)/3
        yBCD = (yB+yC+yD)/3
        xCDA = (xC+xD+xA)/3
        yCDA = (yC+yD+yA)/3
        xDAB = (xD+xA+xB)/3
        yDAB = (yD+yA+yB)/3
```

```
        # Triangle centroids
        xyABC = [round(xABC), round(yABC)]
        xyBCD = [round(xBCD), round(yBCD)]
        xyCDA = [round(xCDA), round(yCDA)]
        xyDAB = [round(xDAB), round(yDAB)]
        entryTC = [xyABC,xyBCD, xyCDA, xyDAB]
        triangleCentroids["window{0}".format(w)].append(entryTC)

        # Determine the slope and y-intercept for each line between
centroids
        # For line between ABC centroid and CDAcentroid
        slope_ABCtoCDA = (yCDA-yABC)/(xCDA-xABC)
        intercept_ABCtoCDA = yABC - xABC*slope_ABCtoCDA
        # BCD centroid to DAB centroid
        slope_BCDtoDAB = (yDAB-yBCD)/(xDAB-xBCD)
        intercept_BCDtoDAB = yBCD - xBCD*slope_BCDtoDAB
        entrySIT =
[[slope_ABCtoCDA,intercept_ABCtoCDA],[slope_BCDtoDAB,intercept_BCDtoDAB]]
        SITCentroids['window{0}'.format(w)].append(entrySIT)

        # Find the coordinates of the intersection of these two lines
        # First calculate x
        xCentroid =  (intercept_BCDtoDAB-
intercept_ABCtoCDA)*(1/(slope_ABCtoCDA-slope_BCDtoDAB))
        yCentroid = slope_ABCtoCDA*xCentroid+intercept_ABCtoCDA
        entryCentroid = [xCentroid,yCentroid]
        ROICentroids["window{0}".format(w)].append(entryCentroid)

        # Standard form: Ax + By = C
        # A = slope, B = 1, C = intercept
        thresholdCheck1 = yCentroid - slope1*xCentroid
        if thresholdCheck1 > intercept1:
            assignment1 = 1
        else:
            assignment1 = 0
        thresholdCheck2 = yCentroid - slope2*xCentroid
        if thresholdCheck2 > intercept2:
            assignment2 = 1
        else:
            assignment2 = 0
        thresholdCheck3 = yCentroid - slope3*xCentroid
        if thresholdCheck3 > intercept3:
            assignment3 = 1
        else:
            assignment3 = 0
        thresholdCheck4 = yCentroid - slope4*xCentroid
        if thresholdCheck4 > intercept4:
            assignment4 = 1
        else:
            assignment4 = 0
        entryAC = [assignment1, assignment2, assignment3, assignment4]
        thresholdCheckImage.append(entryAC)
        assignmentCheck["window{0}".format(w)].append(entryAC)
```

```
    return(slopesIntercepts, xyRange)


#splitWin function
def splitWin(slopesIntercepts, xyRange, folderPath):
    #imageDir = '/Users/fatimamurillo/Research/Images/v2w10p2Im/'
    outputLoc = '/Users/fatimamurillo/Research/Images/v1Label/v1w6p1ROI'
    '''
    imagePath = input('Please enter path to images directory:')
    while True:
        if os.path.exists(imagePath):
            break
        else:
            print('Path invalid.')
            imagePath = input('Please enter path to images directory:')
    '''

    for filename in os.listdir(folderPath):
        if filename == '.DS_Store':
            continue
    #for name in imageList:
       #print(name)
       #imagePath = imageDir + name
        else:
            imagePath = folderPath + '/' + filename
            # Extract image info from image name
            imageNums = re.findall(r'\d+', filename)
            print(imageNums)
            ehwNum = imageNums[0]
            passNum = imageNums[1]
            frameNum = imageNums[2]
           # Create a new folder for each frame
           #newOutputLoc = outputLoc + frameNum
           #if not os.path.exists(newOutputLoc):
            #   os.makedirs(newOutputLoc)
           # Read each image
            image = skimage.io.imread(imagePath)
            for window in slopesIntercepts:
                copyPic = copy.copy(image)
                wNum = re.findall(r'\d+', window)
                windNum = wNum[0]
                dynY = xyRange[window][0][3]
                yMax = xyRange[window][0][2]
                for y in range (dynY, yMax):
                   #dynX = xyRange[window][0][1]
                   xMin = xyRange[window][0][1]
                   xMax = xyRange[window][0][0]
                   #line BC: slopesIntercept[window][0][1]
                   slopeBC = slopesIntercepts[window][0][1][0]
                   interceptBC = slopesIntercepts[window][0][1][1]
                   #line DA: slopesIntercept[window][0][3]
                   slopeDA = slopesIntercepts[window][0][3][0]
                   interceptDA = slopesIntercepts[window][0][3][1]
                   dynXBC = int((y - interceptBC)/slopeBC)
```

```
                    dynXDA = int((y - interceptDA)/slopeDA)
                    for x in range (dynXBC, xMax):
                        copyPic[y, x] = [255,255,255]
                    for x in range (xMin, dynXDA):
                        copyPic[y,x] = [255,255,255]
                windowPortion = copyPic[dynY:yMax, xMin:xMax]
                filename = outputLoc + 'f{0}'.format(frameNum) +
'w{0}'.format(ehwNum) + 'p{0}'.format(passNum) + 'r{0}'.format(windNum) +
'.png'
                io.imsave(filename, windowPortion)
```

Histogram Generation Script

```
# HSI Module
# Written by Fatima Murillo
# Written on December 9, 2015
# Last updated on March 3, 2016

"""
This program retrieves statistical information of images in the HSI
colorspace.
"""

import re
import os # Miscellaneous operating system interfaces
import scipy # Collection of numerical algorithms
from scipy import misc
from matplotlib import pyplot as plt
from matplotlib import colors
import numpy as np # Multi-dimensional container of generic data
import statistics # Provides functions for calculating mathematical
statistics
import csv # Implements classes to read and write tabular data in CSV
format


# img =
scipy.misc.imread('/Users/fatimamurillo/Documents/PythonScripts/rowDetecti
on/templateROIs/NEWwindow6ROI_template.png')


# Generate a csv file that includes statistics report for each image from
splitWinOutput

# Create a new file
filename =
'/Users/fatimamurillo/Documents/PythonScripts/rowDetectII/HSIstats_test.cs
v'
path2RefIm =
'/Users/fatimamurillo/Documents/PythonScripts/rowDetectII/splitWinOutputTe
st'

with open(filename, 'w', newline = '') as openFile:
    csvWriter = csv.writer(openFile, delimiter = ',')
    csvWriter.writerow(['Frame','Window','Class', 'hueMean', 'hueVar',
'hueStDev', 'satMean',
                        'satVar', 'satStDev', 'intMean','intVar',
'intStDev'])

    for frame in os.listdir(path2RefIm):
        if frame == '.DS_Store':
            continue
        else:
            frameDir = path2RefIm + '/' + frame
            for label in os.listdir(frameDir):
```

```
                if label == '.DS_Store':
                    continue
                else:
                    labelDir = frameDir + '/' + label
                    for image in os.listdir(labelDir):
                        if image == '.DS_Store':
                            continue
                        else:
                            path2Im = labelDir + '/' + image
                            windowNum = re.findall(r'\d+',image)
                            img = scipy.misc.imread(path2Im)
                            array = np.asarray(img)
                            arr = (array.astype(float))/255.0
                            img_hsv = colors.rgb_to_hsv(arr[...,:3])

                            # Extract hue information
                            lu1 = img_hsv[...,0].flatten()
                            hueMean = statistics.mean(lu1)
                            hueVar = statistics.pvariance(lu1)
                            hueStDev = statistics.pstdev(lu1)

                            # Extract saturation information
                            lu2 = img_hsv[...,1].flatten()
                            satMean = statistics.mean(lu2)
                            satVar = statistics.pvariance(lu2)
                            satStDev = statistics.pstdev(lu2)

                            # Extract intensity information
                            lu3 = img_hsv[...,2].flatten()
                            intMean = statistics.mean(lu3)
                            intVar = statistics.pvariance(lu3)
                            intStDev = statistics.pstdev(lu3)
                            csvWriter.writerow([frame, windowNum[0],
label, hueMean, hueVar, hueStDev, satMean,
                                    satVar, satStDev, intMean,intVar,
intStDev])
            print('Please wait...')
            print('Working on Frame ' + frame + '...')
openFile.close()




# Plot HSI histogram
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import colors
# Active
#imgPath =
'/Users/fatimamurillo/Research/Images/v1Label/v1w10p1ROI/Active/f4w10p1r7.
png'

# Inactive
```

```
imgPath =
'/Users/fatimamurillo/Research/Images/v1Label/v1w10p1ROI/Inactive/f4w10p1r
4.png'
img = plt.imread(imgPath)
array=np.asarray(img)
arr=(array.astype(float))/255.0
img_hsv = colors.rgb_to_hsv(arr[...,:3])

lu1=img_hsv[...,0].flatten()
plt.subplot(1,3,1)
plt.hist(lu1*360,bins=360,range=(0.0,400.0),histtype='stepfilled',
color='r', label='Hue')
plt.title("Hue")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.legend()


lu2=img_hsv[...,1].flatten()
plt.subplot(1,3,2)
plt.hist(lu2,bins=100,range=(0.0,1.0),histtype='stepfilled', color='g',
label='Saturation')
plt.title("Saturation")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.legend()


lu3=img_hsv[...,2].flatten()
plt.subplot(1,3,3)
plt.hist(lu3*255,bins=256,range=(0.0,255.0),histtype='stepfilled',
color='b', label='Intesity')
plt.title("Intensity")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.legend()
```

Appendix C – Image Classification Tests

Test 1

```
Training Set: v1w6p1ROI
Testing Set: video1

Creating Bag-Of-Features from 2 image sets.
-------------------------------------------
* Image set 1: Active.
* Image set 2: Inactive.

* Extracting SURF features using the Detector selection method.
** detectSURFFeatures is used to detect key points for feature extraction.

* Extracting features from 250 images in image set 1...done. Extracted 14925 features.
* Extracting features from 250 images in image set 2...done. Extracted 8492 features.

* Keeping 80 percent of the strongest features from each image set.

* Balancing the number of features across all image sets to improve clustering.
** Image set 2 has the least number of strongest features: 6794.
** Using the strongest 6794 features from each of the other image sets.

* Using K-Means clustering to create a 100 word visual vocabulary.
* Number of features          : 13588
* Number of clusters (K)      : 100

* Initializing cluster centers...100.00%.
* Clustering...completed 38/100 iterations (~0.12 seconds/iteration)...converged in 38↙
iterations.

* Finished creating Bag-Of-Features


Training an image category classifier for 2 categories.
-------------------------------------------------------
* Category 1: Active
* Category 2: Inactive

* Encoding features for category 1...done.
* Encoding features for category 2...done.

* Finished training the category classifier. Use evaluate to test the classifier on a↙
test set.

Run classifier on training set to see how well it performs.

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 250 images from category 1...done.
* Evaluating 250 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:


                   PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 0.96     0.04
Inactive    | 0.04     0.96

* Average Accuracy is 0.96.

Evaluate classifier performance on validation set

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 500 images from category 1...done.
* Evaluating 500 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:


                   PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 0.72     0.28
Inactive    | 0.05     0.95

* Average Accuracy is 0.83.
```

Test 2

```
Training Set: v1w10p1ROI
Testing Set: video1

Creating Bag-Of-Features from 2 image sets.
-----------------------------------------------
* Image set 1: Active.
* Image set 2: Inactive.

* Extracting SURF features using the Detector selection method.
** detectSURFFeatures is used to detect key points for feature extraction.

* Extracting features from 250 images in image set 1...done. Extracted 12981 features.
* Extracting features from 250 images in image set 2...done. Extracted 12075 features.

* Keeping 80 percent of the strongest features from each image set.

* Balancing the number of features across all image sets to improve clustering.
** Image set 2 has the least number of strongest features: 9660.
** Using the strongest 9660 features from each of the other image sets.

* Using K-Means clustering to create a 100 word visual vocabulary.
* Number of features          : 19320
* Number of clusters (K)      : 100

* Initializing cluster centers...100.00%.
* Clustering...completed 37/100 iterations (~0.15 seconds/iteration)...converged in 37↙
iterations.

* Finished creating Bag-Of-Features


Training an image category classifier for 2 categories.
-------------------------------------------------------
* Category 1: Active
* Category 2: Inactive

* Encoding features for category 1...done.
* Encoding features for category 2...done.

* Finished training the category classifier. Use evaluate to test the classifier on a↙
test set.

Run classifier on training set to see how well it performs.

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 250 images from category 1...done.
* Evaluating 250 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:

                PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 0.99     0.01
Inactive    | 0.11     0.89

* Average Accuracy is 0.94.

Evaluate classifier performance on validation set

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 500 images from category 1...done.
* Evaluating 500 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:

                PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 0.99     0.01
Inactive    | 0.32     0.68

* Average Accuracy is 0.84.
```

Test 3

```
Training Set: video1
Testing Set: video1

Creating Bag-Of-Features from 2 image sets.
-----------------------------------------------
* Image set 1: Active.
* Image set 2: Inactive.

* Extracting SURF features using the Detector selection method.
** detectSURFFeatures is used to detect key points for feature extraction.

* Extracting features from 250 images in image set 1...done. Extracted 16418 features.
* Extracting features from 250 images in image set 2...done. Extracted 12724 features.

* Keeping 80 percent of the strongest features from each image set.

* Balancing the number of features across all image sets to improve clustering.
** Image set 2 has the least number of strongest features: 10179.
** Using the strongest 10179 features from each of the other image sets.

* Using K-Means clustering to create a 100 word visual vocabulary.
* Number of features          : 20358
* Number of clusters (K)      : 100

* Initializing cluster centers...100.00%.
* Clustering...completed 32/100 iterations (~0.16 seconds/iteration)...converged in 32↙
iterations.

* Finished creating Bag-Of-Features


Training an image category classifier for 2 categories.
-------------------------------------------------------
* Category 1: Active
* Category 2: Inactive

* Encoding features for category 1...done.
* Encoding features for category 2...done.

* Finished training the category classifier. Use evaluate to test the classifier on a↙
test set.

Run classifier on training set to see how well it performs.

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 250 images from category 1...done.
* Evaluating 250 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:

                PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 1.00     0.00
Inactive    | 0.05     0.95

* Average Accuracy is 0.97.

Evaluate classifier performance on validation set

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 500 images from category 1...done.
* Evaluating 500 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:


                PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 0.96     0.04
Inactive    | 0.06     0.94

* Average Accuracy is 0.95.
```

Test 4

```
Training Set: v2w6p1ROI
Testing Set: video1

Creating Bag-Of-Features from 2 image sets.
-------------------------------------------
* Image set 1: Active.
* Image set 2: Inactive.

* Extracting SURF features using the Detector selection method.
** detectSURFFeatures is used to detect key points for feature extraction.

* Extracting features from 250 images in image set 1...done. Extracted 10419 features.
* Extracting features from 250 images in image set 2...done. Extracted 4264 features.

* Keeping 80 percent of the strongest features from each image set.

* Balancing the number of features across all image sets to improve clustering.
** Image set 2 has the least number of strongest features: 3411.
** Using the strongest 3411 features from each of the other image sets.

* Using K-Means clustering to create a 100 word visual vocabulary.
* Number of features          : 6822
* Number of clusters (K)      : 100

* Initializing cluster centers...100.00%.
* Clustering...completed 35/100 iterations (~0.10 seconds/iteration)...converged in 35↙
iterations.

* Finished creating Bag-Of-Features


Training an image category classifier for 2 categories.
-------------------------------------------------------
* Category 1: Active
* Category 2: Inactive

* Encoding features for category 1...done.
* Encoding features for category 2...done.

* Finished training the category classifier. Use evaluate to test the classifier on a↙
test set.

Run classifier on training set to see how well it performs.

Evaluating image category classifier for 2 categories.
------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 250 images from category 1...done.
* Evaluating 250 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:

                  PREDICTED
KNOWN       | Active   Inactive
------------------------------
Active      | 0.98     0.02
Inactive    | 0.02     0.98

* Average Accuracy is 0.98.

Evaluate classifier performance on validation set

Evaluating image category classifier for 2 categories.
------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 500 images from category 1...done.
* Evaluating 500 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:


                  PREDICTED
KNOWN       | Active   Inactive
------------------------------
Active      | 0.96     0.04
Inactive    | 0.33     0.67

* Average Accuracy is 0.82.
```

Test 5

```
Training Set: v2w10p1ROI
Testing Set: video1

Creating Bag-Of-Features from 2 image sets.
-----------------------------------------
* Image set 1: Active.
* Image set 2: Inactive.

* Extracting SURF features using the Detector selection method.
** detectSURFFeatures is used to detect key points for feature extraction.

* Extracting features from 250 images in image set 1...done. Extracted 10568 features.
* Extracting features from 250 images in image set 2...done. Extracted 6216 features.

* Keeping 80 percent of the strongest features from each image set.

* Balancing the number of features across all image sets to improve clustering.
** Image set 2 has the least number of strongest features: 4973.
** Using the strongest 4973 features from each of the other image sets.

* Using K-Means clustering to create a 100 word visual vocabulary.
* Number of features          : 9946
* Number of clusters (K)      : 100

* Initializing cluster centers...100.00%.
* Clustering...completed 20/100 iterations (~0.14 seconds/iteration)...converged in 20↙
iterations.

* Finished creating Bag-Of-Features


Training an image category classifier for 2 categories.
-------------------------------------------------------
* Category 1: Active
* Category 2: Inactive

* Encoding features for category 1...done.
* Encoding features for category 2...done.

* Finished training the category classifier. Use evaluate to test the classifier on a↙
test set.

Run classifier on training set to see how well it performs.

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 250 images from category 1...done.
* Evaluating 250 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:
                  PREDICTED
  KNOWN       | Active   Inactive
  ----------------------------------
  Active      | 0.99     0.01
  Inactive    | 0.00     1.00

  * Average Accuracy is 1.00.

  Evaluate classifier performance on validation set

  Evaluating image category classifier for 2 categories.
  -------------------------------------------------------

  * Category 1: Active
  * Category 2: Inactive

  * Evaluating 500 images from category 1...done.
  * Evaluating 500 images from category 2...done.

  * Finished evaluating all the test sets.

  * The confusion matrix for this test set is:


                  PREDICTED
  KNOWN       | Active   Inactive
  ----------------------------------
  Active      | 1.00     0.00
  Inactive    | 0.57     0.43

  * Average Accuracy is 0.71.
```

Test 6

```
    Training Set: v2w10p1ROI
    Testing Set: video2

    Creating Bag-Of-Features from 2 image sets.
    -------------------------------------------
    * Image set 1: Active.
    * Image set 2: Inactive.

    * Extracting SURF features using the Detector selection method.
    ** detectSURFFeatures is used to detect key points for feature extraction.

    * Extracting features from 250 images in image set 1...done. Extracted 10361 features.
    * Extracting features from 250 images in image set 2...done. Extracted 6419 features.

    * Keeping 80 percent of the strongest features from each image set.

    * Balancing the number of features across all image sets to improve clustering.
    ** Image set 2 has the least number of strongest features: 5135.
    ** Using the strongest 5135 features from each of the other image sets.

    * Using K-Means clustering to create a 100 word visual vocabulary.
    * Number of features          : 10270
    * Number of clusters (K)      : 100

    * Initializing cluster centers...100.00%.
    * Clustering...completed 33/100 iterations (~0.09 seconds/iteration)...converged in 33↙
    iterations.

    * Finished creating Bag-Of-Features


    Training an image category classifier for 2 categories.
    -------------------------------------------------------
    * Category 1: Active
    * Category 2: Inactive

    * Encoding features for category 1...done.
    * Encoding features for category 2...done.

    * Finished training the category classifier. Use evaluate to test the classifier on a↙
    test set.

    Run classifier on training set to see how well it performs.

    Evaluating image category classifier for 2 categories.
    -------------------------------------------------------

    * Category 1: Active
    * Category 2: Inactive

    * Evaluating 250 images from category 1...done.
    * Evaluating 250 images from category 2...done.

    * Finished evaluating all the test sets.

    * The confusion matrix for this test set is:
                   PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 0.99     0.01
Inactive    | 0.01     0.99

* Average Accuracy is 0.99.

Evaluate classifier performance on validation set

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 500 images from category 1...done.
* Evaluating 500 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:


                   PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 0.97     0.03
Inactive    | 0.05     0.95

* Average Accuracy is 0.96.
```

Test 7

```
Training Set: v2w6p1ROI
Testing Set: video2

Creating Bag-Of-Features from 2 image sets.
-----------------------------------------------
* Image set 1: Active.
* Image set 2: Inactive.

* Extracting SURF features using the Detector selection method.
** detectSURFFeatures is used to detect key points for feature extraction.

* Extracting features from 250 images in image set 1...done. Extracted 10440 features.
* Extracting features from 250 images in image set 2...done. Extracted 4270 features.

* Keeping 80 percent of the strongest features from each image set.

* Balancing the number of features across all image sets to improve clustering.
** Image set 2 has the least number of strongest features: 3416.
** Using the strongest 3416 features from each of the other image sets.

* Using K-Means clustering to create a 100 word visual vocabulary.
* Number of features          : 6832
* Number of clusters (K)      : 100

* Initializing cluster centers...100.00%.
* Clustering...completed 26/100 iterations (~0.11 seconds/iteration)...converged in 26↙
iterations.

* Finished creating Bag-Of-Features


Training an image category classifier for 2 categories.
-------------------------------------------------------
* Category 1: Active
* Category 2: Inactive

* Encoding features for category 1...done.
* Encoding features for category 2...done.

* Finished training the category classifier. Use evaluate to test the classifier on a↙
test set.

Run classifier on training set to see how well it performs.

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 250 images from category 1...done.
* Evaluating 250 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:

                PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 0.99     0.01
Inactive    | 0.02     0.98

* Average Accuracy is 0.98.

Evaluate classifier performance on validation set

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 500 images from category 1...done.
* Evaluating 500 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:


                PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 0.92     0.08
Inactive    | 0.04     0.96

* Average Accuracy is 0.94.
```

Test 8

```
Training Set: video2
Testing Set: video2

Creating Bag-Of-Features from 2 image sets.
-----------------------------------------
* Image set 1: Active.
* Image set 2: Inactive.

* Extracting SURF features using the Detector selection method.
** detectSURFFeatures is used to detect key points for feature extraction.

* Extracting features from 250 images in image set 1...done. Extracted 10053 features.
* Extracting features from 250 images in image set 2...done. Extracted 5297 features.

* Keeping 80 percent of the strongest features from each image set.

* Balancing the number of features across all image sets to improve clustering.
** Image set 2 has the least number of strongest features: 4238.
** Using the strongest 4238 features from each of the other image sets.

* Using K-Means clustering to create a 100 word visual vocabulary.
* Number of features          : 8476
* Number of clusters (K)      : 100

* Initializing cluster centers...100.00%.
* Clustering...completed 27/100 iterations (~0.13 seconds/iteration)...converged in 27↙
iterations.

* Finished creating Bag-Of-Features


Training an image category classifier for 2 categories.
-------------------------------------------------------
* Category 1: Active
* Category 2: Inactive

* Encoding features for category 1...done.
* Encoding features for category 2...done.

* Finished training the category classifier. Use evaluate to test the classifier on a↙
test set.

Run classifier on training set to see how well it performs.

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 250 images from category 1...done.
* Evaluating 250 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:

                  PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 0.84     0.16
Inactive    | 0.04     0.96

* Average Accuracy is 0.90.

Evaluate classifier performance on validation set

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 500 images from category 1...done.
* Evaluating 500 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:


                  PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 0.98     0.02
Inactive    | 0.07     0.93

* Average Accuracy is 0.96.
```

Test 9

```
Training Set: video1
Testing Set: video2

Creating Bag-Of-Features from 2 image sets.
-----------------------------------------
* Image set 1: Active.
* Image set 2: Inactive.

* Extracting SURF features using the Detector selection method.
** detectSURFFeatures is used to detect key points for feature extraction.

* Extracting features from 250 images in image set 1...done. Extracted 16022 features.
* Extracting features from 250 images in image set 2...done. Extracted 12784 features.

* Keeping 80 percent of the strongest features from each image set.

* Balancing the number of features across all image sets to improve clustering.
** Image set 2 has the least number of strongest features: 10227.
** Using the strongest 10227 features from each of the other image sets.

* Using K-Means clustering to create a 100 word visual vocabulary.
* Number of features          : 20454
* Number of clusters (K)      : 100

* Initializing cluster centers...100.00%.
* Clustering...completed 33/100 iterations (~0.17 seconds/iteration)...converged in 33↙
iterations.

* Finished creating Bag-Of-Features


Training an image category classifier for 2 categories.
-------------------------------------------------------
* Category 1: Active
* Category 2: Inactive

* Encoding features for category 1...done.
* Encoding features for category 2...done.

* Finished training the category classifier. Use evaluate to test the classifier on a↙
test set.

Run classifier on training set to see how well it performs.

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 250 images from category 1...done.
* Evaluating 250 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:

                  PREDICTED
KNOWN       | Active   Inactive
-----------------------------------
Active      | 0.98     0.02
Inactive    | 0.03     0.97

* Average Accuracy is 0.98.

Evaluate classifier performance on validation set

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 500 images from category 1...done.
* Evaluating 500 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:


                  PREDICTED
KNOWN       | Active   Inactive
-----------------------------------
Active      | 0.94     0.06
Inactive    | 0.18     0.82

* Average Accuracy is 0.88.
```

Test 10

```
Training Set: video2
Testing Set: video1

Creating Bag-Of-Features from 2 image sets.
-----------------------------------------------
* Image set 1: Active.
* Image set 2: Inactive.

* Extracting SURF features using the Detector selection method.
** detectSURFFeatures is used to detect key points for feature extraction.

* Extracting features from 250 images in image set 1...done. Extracted 9447 features.
* Extracting features from 250 images in image set 2...done. Extracted 5358 features.

* Keeping 80 percent of the strongest features from each image set.

* Balancing the number of features across all image sets to improve clustering.
** Image set 2 has the least number of strongest features: 4286.
** Using the strongest 4286 features from each of the other image sets.

* Using K-Means clustering to create a 100 word visual vocabulary.
* Number of features          : 8572
* Number of clusters (K)      : 100

* Initializing cluster centers...100.00%.
* Clustering...completed 29/100 iterations (~0.11 seconds/iteration)...converged in 29↙
iterations.

* Finished creating Bag-Of-Features


Training an image category classifier for 2 categories.
-------------------------------------------------------
* Category 1: Active
* Category 2: Inactive

* Encoding features for category 1...done.
* Encoding features for category 2...done.

* Finished training the category classifier. Use evaluate to test the classifier on a↙
test set.

Run classifier on training set to see how well it performs.

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 250 images from category 1...done.
* Evaluating 250 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:
                  PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 0.85     0.15
Inactive    | 0.02     0.98

* Average Accuracy is 0.92.

Evaluate classifier performance on validation set

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 500 images from category 1...done.
* Evaluating 500 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:


                  PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 1.00     0.00
Inactive    | 0.48     0.52

* Average Accuracy is 0.76.
```

Test 11

```
Training Set: v1w6p1ROI
Testing Set: video2

Creating Bag-Of-Features from 2 image sets.
-----------------------------------------------
* Image set 1: Active.
* Image set 2: Inactive.

* Extracting SURF features using the Detector selection method.
** detectSURFFeatures is used to detect key points for feature extraction.

* Extracting features from 250 images in image set 1...done. Extracted 14751 features.
* Extracting features from 250 images in image set 2...done. Extracted 8463 features.

* Keeping 80 percent of the strongest features from each image set.

* Balancing the number of features across all image sets to improve clustering.
** Image set 2 has the least number of strongest features: 6770.
** Using the strongest 6770 features from each of the other image sets.

* Using K-Means clustering to create a 100 word visual vocabulary.
* Number of features          : 13540
* Number of clusters (K)      : 100

* Initializing cluster centers...100.00%.
* Clustering...completed 21/100 iterations (~0.11 seconds/iteration)...converged in 21↙
iterations.

* Finished creating Bag-Of-Features


Training an image category classifier for 2 categories.
-------------------------------------------------------
* Category 1: Active
* Category 2: Inactive

* Encoding features for category 1...done.
* Encoding features for category 2...done.

* Finished training the category classifier. Use evaluate to test the classifier on a↙
test set.

Run classifier on training set to see how well it performs.

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 250 images from category 1...done.
* Evaluating 250 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:

                  PREDICTED
KNOWN       | Active   Inactive
-----------------------------------
Active      | 0.97     0.03
Inactive    | 0.05     0.95

* Average Accuracy is 0.96.

Evaluate classifier performance on validation set

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 500 images from category 1...done.
* Evaluating 500 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:


                  PREDICTED
KNOWN       | Active   Inactive
-----------------------------------
Active      | 0.84     0.16
Inactive    | 0.35     0.65

* Average Accuracy is 0.74.

>>
```

Test 12

```
Training Set: v1w10p1ROI
Testing Set: video2

Creating Bag-Of-Features from 2 image sets.
-----------------------------------------
* Image set 1: Active.
* Image set 2: Inactive.

* Extracting SURF features using the Detector selection method.
** detectSURFFeatures is used to detect key points for feature extraction.

* Extracting features from 250 images in image set 1...done. Extracted 12972 features.
* Extracting features from 250 images in image set 2...done. Extracted 12012 features.

* Keeping 80 percent of the strongest features from each image set.

* Balancing the number of features across all image sets to improve clustering.
** Image set 2 has the least number of strongest features: 9610.
** Using the strongest 9610 features from each of the other image sets.

* Using K-Means clustering to create a 100 word visual vocabulary.
* Number of features          : 19220
* Number of clusters (K)      : 100

* Initializing cluster centers...100.00%.
* Clustering...completed 29/100 iterations (~0.16 seconds/iteration)...converged in 29↙
iterations.

* Finished creating Bag-Of-Features


Training an image category classifier for 2 categories.
-------------------------------------------------------
* Category 1: Active
* Category 2: Inactive

* Encoding features for category 1...done.
* Encoding features for category 2...done.

* Finished training the category classifier. Use evaluate to test the classifier on a↙
test set.

Run classifier on training set to see how well it performs.

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 250 images from category 1...done.
* Evaluating 250 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:

                PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 1.00     0.00
Inactive    | 0.12     0.88

* Average Accuracy is 0.94.

Evaluate classifier performance on validation set

Evaluating image category classifier for 2 categories.
-------------------------------------------------------

* Category 1: Active
* Category 2: Inactive

* Evaluating 500 images from category 1...done.
* Evaluating 500 images from category 2...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:


                PREDICTED
KNOWN       | Active   Inactive
------------------------------------
Active      | 0.97     0.03
Inactive    | 0.23     0.77

* Average Accuracy is 0.87.
```

Vita

Fatima Murillo was born in Santa Clara, CA to the parents of Lyndon and Elizabeth Murillo. She is the eldest of four children: Bernadette, Godwin, and Luke. Fatima moved to Bell Buckle, TN in 2001 and attended Cascade Elementary school in Wartrace, TN. She graduated with honors from Cascade High School in Wartrace, TN in 2010 and was accepted to the University of Tennessee's Institute of Agriculture. In 2014, she received her Bachelor of Science degree from the department of Biosystems Engineering and Soil Science where she met her fiancé, Chance Frana. Upon graduation, she accepted a graduate research assistantship in the same department with Dr. John B. Wilkerson whose focus is on sensor development for agricultural applications. Fatima Murillo graduated with a Master's of Science degree in December 2016.