DESIGN AND IMPLEMENTATIONS OF OPEN-SOURCE AG IOT DEVICES FOR FARM MACHINERY DATA ACQUISITION AND INTEGRATED ANALYTICS

by

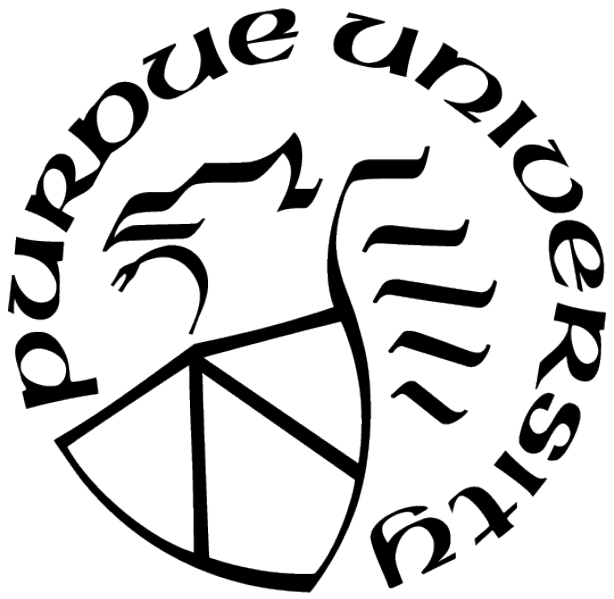
Yang Wang

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy



School of Electrical and Computer Engineering

West Lafayette, Indiana

August 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. James V. Krogmeier, Chair

School of Electrical and Computer Engineering

Dr. Dennis R. Buckmaster

Department of Agricultural and Biological Engineering

Dr. David J. Love

School of Electrical and Computer Engineering

Dr. Mark R. Bell

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

To my granddad and grandpa, Y. Jin and C. Wang

ACKNOWLEDGMENTS

First and foremost, I would like to thank God for providing me with the wisdom and perseverance in pursuing this degree.

I am deeply thankful for my advisors Dr. Krogmeier and Dr. Buckmaster for their guidance and support. They not only dedicated time to give insightful advices on my research but also helped me to see beyond graduate school and grow as a person. I am thankful for the the projects I worked on. They were the primary reasons that I was able to find my future job.

I would like to thank my family. I am only able to come through this far with their endless love and support.

I would like to express gratitude to my fiancé, Tamara Guest. Your self-giving love, touching prayers, and amazing cuisine encouraged me and lifted my spirit through many tough moments.

I would like to thank my friends and colleagues at the OATS center. Graduate school is definitely more enriching and fun with you guys.

Lastly, I would like to thank Ault Farms and Krogmeier Farms for data collection assistance for this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

SAE	Society of Automotive Engineers
ASABE	American Society of Agricultural and Biological Engineers
CAN	Controller Area Network

ABSTRACT

Agricultural machinery is critical in modern farming. With continuous technological advancements in farm machinery, farm machines have evolved from simple mechanical machines to cyberphysical systems that contain rich sources of multimodal sensor data. Effective acquisition and analyses of these data have become essential but challenging tasks in revealing machine-centric and logistical insights to researchers and farmers.

In this dissertation, theses challenge are addressed in two parts. The first part demonstrates successful development and deployment of two open-source telematic devices for collecting machine network, geospatial, and video data. The first, ISOBlue 2.0, was designed to be a logger of both GPS and CAN data with wireless data streaming capabilities. The second, ISOBlue HD, an extension of ISOBlue 2.0, was configured to behave as a network server that interfaced with external cameras for automatic video recording of machine operation contexts. These devices were deployed in a variety of machines in different farming activities. A total of over 1 TB of multimodal machinery data were collected.

The second part presents three problems that focus on analyzing primarily GPS track data collected from past wheat harvests. The first poses an activity classification problem. It involved clustering a 3D feature set generated from both GPS and CAN data from a combine using the Density-Based Spatial Clustering of Applications with Noise algorithm. The resultant clusters between on-road and in-field data samples as well as normal and anomalous activities. The second problem concentrates on combine unloading event detections using GPS tracks of multiple combines in 16 harvest sessions. The identified events from a novel algorithm that couples Interacting Multiple Models filtering and composite rules were utilized to estimate the total yield for each session. The estimated yields had an overall accuracy of over 90% when comparing to the actual weight ticket records. Lastly, two instantaneous metrics, instantaneous area capacity and swath utilization, were proposed and estimated using GPS tracks of multiple combines in 7 different fields during various harvest years. A novel algorithm was created for estimating instantaneous actual harvested area and swath utilization. This enabled exact computations of instantaneous metrics as oppose to conventional rough estimates of area capacity. Harvest performances were evaluated both

temporally and geospatially by machines and years. It was discovered that three contributing factors that lead to high area capacity were wide header attachments, high harvesting speed, and uniform harvesting patterns. Moreover, it was found that the benefit of a wider header might diminish if the harvesting speed was low.

1. INTRODUCTION

At its core, modern agriculture is a business of logistics where farmers find their competitive advantage relative to other farmers in the timeliness and efficiency with which they make informed decisions in preparing the soil, applying crop nutrients, planting the seed, applying crop protection and water, harvesting, and marketing the crop. The choices they make in inputs such as fertilizer, seed genetics, and chemicals, while important, are secondary to timeliness and the weather. Therefore, the first adopted and most widely used tools of precision agriculture were those which improved timeliness and efficiency, such as advanced satellite positioning technologies, auto-steer, larger implements, faster working speeds, and business and market information enabled by the internet. Viewed as a critical part of agriculture-as-logistics, the performance of distributed agricultural systems comprised of multiple interacting machines and human operators is of critical importance. See [1]–[7].

1.1 Farm Machinery Network Data

Much of the relevant agricultural machine data is transported over the wired ISOBUS network [8] running the Controller Area Network (CAN) [9] protocol. An agricultural machine typically has two CAN busses: the tractor bus and the implement bus. Electronic control units (ECUs) use this network to communicate with each other and to receive and transmit data from many sensors. Data of interest include machine status information like engine speed or operation-dependent information like instantaneous crop flow rate, crop moisture readings, application rate, etc.

Specifically, a typical ISOBUS message uses the extended CAN frame format that includes a 29-bit CAN identifier (CAN ID) and a 64-bit data payload. Each CAN ID is a concatenation of data fields with variable length as shown in Figure 1.1. A Parameter Group Number (PGN) refers to the combined value of Extended Data Page (EDP), Data Page (DP), Protocol Data Unit Format (PF), and Protocol Data Unit Specific (PS) fields. The PF can fall into either PDU1 or PDU2 format. A message is addressable if it follows the PDU1 format; otherwise, a message can only be broadcast if it follows the PDU2 format. Moreover, the PGN is a number that defines the decoding scheme of the data payload

within each message. This number is normally proprietary and differs from one machine manufacturer to another.

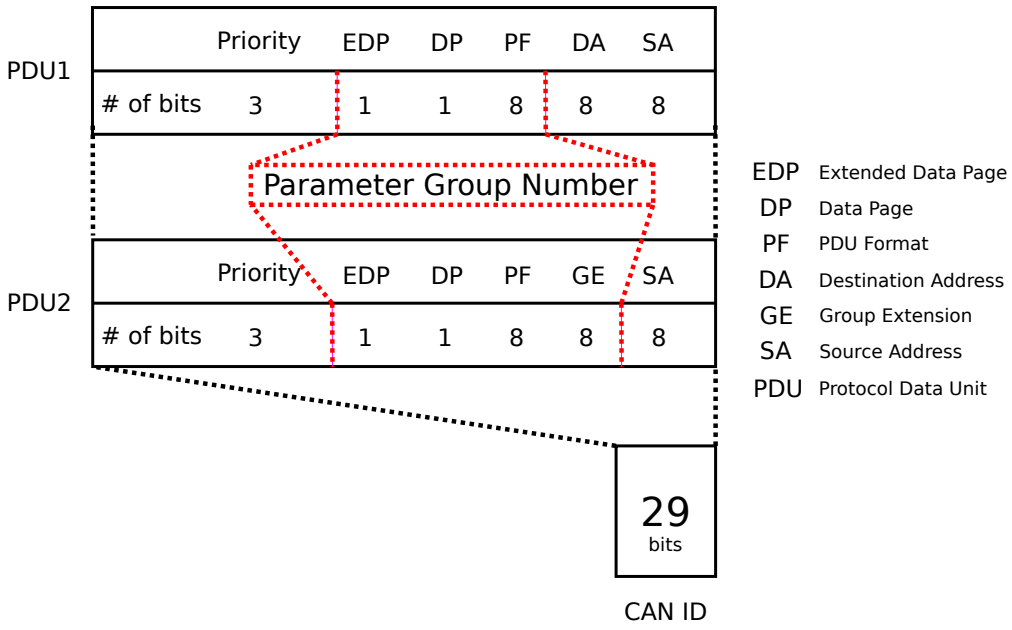


Figure 1.1. The Parameter Group Number (PGN) within the CAN ID of an ISOBUS message contains information for decoding the meaning of a data payload.

1.2 Generic Farm Machinery Telematic Data Pipeline

To effectively obtain and process this data, traditional manual processes [10] for data retrieval are woefully inadequate and outdated to support the continued automation of agricultural machinery in terms of high bandwidth sensors (e.g., video, lidar) as well as technologies like on-machine edge computing and real-time processing [11]–[15].

The manual process has been replaced by vendor-specific machine-to-cloud data pipelines for automating both data acquisition and processing tasks. A typical farm machinery to cloud data pipeline is comprised of three compoments as illustrated in Figure 1.2. The telematic device serves as the starting point for raw data collection and preprocessing. For communicating with the cloud, the device is often equipped with a wireless module (cellular, Bluetooth, etc.) to relay collected data to a nearby mobile device that would then upload the data or forward the data directly to a dedicated cloud instance.

The cloud is an orchestration of proper configurations and applications that would handle the incoming data from telematic devices. A popular way of orchestration is referred to as a Lambda architecture [16]. The core philosophy behind this architecture is that it can manage both batch and stream processing needs. For batch processing, data are typically stored and backed up in one or several append-only and immutable databases using predefined data schemas. They serve as data sources for complex post-processing software to query up to several years worth of data for mining and modeling purposes. On the other hand, the stream processing methods intend to produce real-time analytics such as low fuel alerts. Hence, this part of the pipeline usually employs low-latency messaging busses for allowing fast ingestion and continuous data flow. Furthermore, the endpoint applications are the gateways for users to interact with the stored data. They could be mobile/web applications that provide temporal and geospatial visualizations of machine and device statuses for monitoring and farm equipment management.

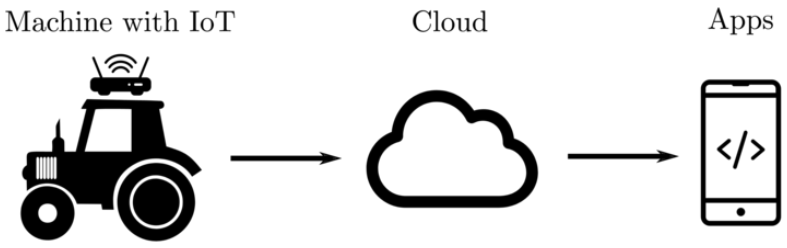


Figure 1.2. A generic farm machinery to cloud data pipeline consists of an IoT device that forwards filtered data to the cloud. These data are processed to generate both real-time and post-processed insights to farm managers.

1.3 Context Mining in Farming Activities

In this dissertation the term "context" is understood to mean labels of datasets, which are relevant for training of machine learning algorithms or for other types of statistical analysis. The formalization of the "context" notion has been in the works for quite some time since the advent of context-aware computing [17]. In this dissertation, "context" refers to the interactions of farm machinery during farming tasks. Taking combine harvesters in wheat harvest as an example, context refers to what the machine-operator is doing at a particular

moment. For example: 1) a combine may be moving back and forth in a field harvesting wheat, 2) a combine may be stopped and unloading on a grain cart or truck, 3) a combine may be unloading on a grain cart while it is simultaneously harvesting grain (called "unloading on the go"), 4) a combine may be cornering or maneuvering outside of the unharvested field area, 5) a combine may be traveling on a road, 6) a combine may be maneuvering around an in-field obstacle, 7) a combine may be idling or otherwise not operational.

These contexts, which are easily understood by human operators, directly affect operators' in-field decisions. One of the key tactical decisions, for instance, is the choice of path. Proprietary work by OEMs and research studies have made great progress towards route optimizations as seen in [18]–[21]. While often such optimization algorithms may suffice, this is not always the case. The use of real-time and recent-past data can facilitate performance by reducing wasted time, travel distance, and fuel usage. To truly optimize, it will require full integration of CAN (yield, speed, engine load, etc.), Global Navigation Satellite System (GNSS) tracks, video, and additional information like topography and audio. In the move toward autonomy, such comprehensive datasets are required as they contribute to robustness in AI developments but, more immediately, enable improved decision-making assistance (e.g., route suggestion, anomaly detection) for operators. Hence, while the combine is operating, sensor data is typically being captured for which the meta-data about context is needed to explain and properly label the datasets. In this dissertation, a dataset that contains sufficient information for algorithms to label, fuse, mine, and infer a full suite of contexts related to machine and surroundings, is referred to as a context-rich dataset. This dissertation concentrates on the acquisitions and the processing of such datasets for mining contextual knowledge from various wheat harvests.

1.4 Organization of Dissertation

This rest of this dissertation is organized as follows:

1. Chapter 2 details the development of two open-source farm machinery IoT devices for collecting context-rich machinery data which implements a rudimental version of a machine-to-cloud data pipeline. A series of exemplary data processing steps for

extracting operational contexts in wheat harvest was presented. This chapter was a synthesized version of past publications in [22] and [23].

2. Chapter 3 dives into the classification of machine maneuvers and anomalies using GPS and CAN data from a single harvester in wheat harvest. This chapter is an edited version of a past conference paper [24].

3. Chapter 4 focuses on the identificaition of combine harvester unloading events using GPS tracks of multiple machines during 16 wheat harvest sessions. A rule-based algorithm was discussed in detail and the accuracy of the identified events were presented. This chapter was based on a past conference paper [25].

4. Chapter 5 details a workflow that estimates instantaneous efficiency metrics using GPS tracks of combine harvesters during wheat harvest years. Harvest performances were evaluated using these metrics for both temporal and geospatial comparisons by machines and years. This chapter was written to be submitted as a journal article.

5. Chapter 6 concludes this dissertation and provides a outlook for future works.

2. DEVELOPMENT OF OPEN-SOURCE PLATFORMS FOR FARM MACHINERY DATA ACQUISITION AND STREAMING

Acknowledgement: this chapter is a synthesized version of two past publications. One was published as a paper [22] at the 2017 Annual ASABE International Meeting and the other was published as an article [23] in the *Sensors* journal.

2.1 Introduction

This chapter introduces two open-source platforms, ISOBlue 2.0 and ISOBlue HD, for acquisition and streaming of farm machinery data. A review of existing data acquisition platforms is given in Section 2.2. The detailed discussion on ISOBlue 2.0 and HD are given in Section 2.3. A summary of past data acquisition using both platforms is given in Section 2.4. Lastly, a series of exemplary data processing steps in analyzing a context-rich dataset from a past deployment is given in Section 2.5.

2.2 Review of Existing Data Acquisition Platforms

A widely accepted method to collect data from farm machinery is to use an ISOBUS diagnostic port [26] shown in Figure 2.1. The port provides CAN bus data connections as well as 12 V unswitched power from the machine battery. The port is commonly utilized for attaching service tools for machine repair and maintenance [27].

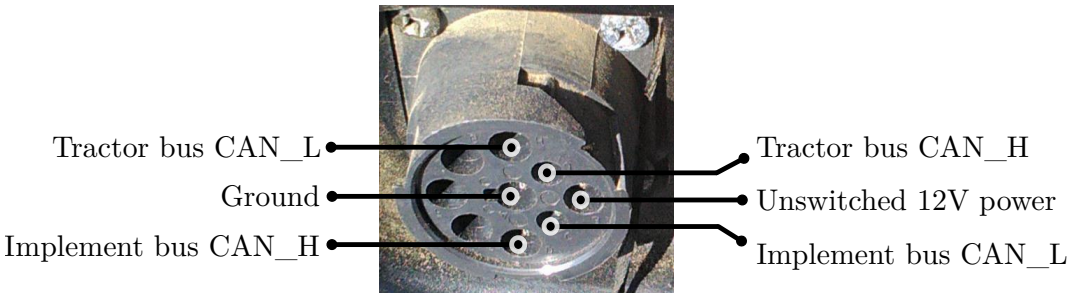


Figure 2.1. Photo of a 9-pin ISOBUS diagnostic port with labels of pinout.

Proprietary telematic devices, typically bundled with powerful vendor-specific software backends, are favorable among farmers because of the ease of installation and integration with their existing machine fleets. On the other hand, with the increasing affordability of high-performance computing platforms, researchers have created a number of custom platforms for collecting data from farm machinery. A selection of both commercial-grade and custom platforms are listed in Table 2.1.

Table 2.1. A selection of both proprietary and custom platforms for farm machinery telematics is listed.

Name	Vendor/Creator	Features	Type
Modular Telematics Gateway (MTG)	John Deere [28]	CAN, GNSS, motion sensors, wireless modules	Proprietary
FieldView Drive	Climate Corp [29]	CAN	Proprietary
PUC	Farmobile [30]	CAN, GNSS, motion sensors, wireless modules	Proprietary
Cropinfra	Backman et al. [31]	CAN, GNSS	Custom
FieldSafe	Kragh et al. [32]	GNSS, ranging, video, motion sensors	Custom
CyCAN	Darr [33]	CAN	Custom
ISOBlue	Layton et al. [34]	CAN, wireless modules	Custom
PolyCAN	Fite et al. [35]	CAN	Custom

Proprietary solutions are well-tested and powerful as they come with a wide range of peripherals and wireless capability for sensor fusion and data streaming. However, they also

come with several shortcomings. First, commercial-grade telematics are expensive. The price point of these systems renders it difficult for researchers or hobbyists to purchase and use them for experiment. Second, although commercial devices offer a wide range of peripherals, they rarely offer customizability or expandability. Consequently, there is no convenient way for farmers to add or experiment with custom hardware or software features. In addition, the collected data typically ends up in vendor-specific clouds. In other words, a mixed fleet of telematic units from different brands results in potential data interoperability issues. Third, raw machinery data are usually inaccessible. Although there exist open-source efforts such as MyJohnDeere [36] that allow third-party users to retrieve vendor-specific data, the retrieved data are typically filtered or processed versions of the raw data. In other words, vendors are in control of what data users could see and retrieve.

Custom platforms, on the other hand, benefit from the flexibilities in hardware selection and the freedom in software implementations for tailored data collection needs. For instance, CyCAN [33] was deployed in past experiments in [37] and [38] for collecting CAN data. In addition, PolyCAN [35] was utilized as a open-source platform for machine fault diagnosis. Moreover, Cropinfra [31] and FieldSafe [32] platforms were employed to collect multimodal machine datasets for improving crop production and in-field obstacle avoidance. Nevertheless, since custom platforms are created for specific tasks, their functionalities are often limited. For example, none of the listed custom platforms except ISOBlue [34] has data streaming capabilities. ISOBlue is able to forward the collected data to a nearby mobile device via a Bluetooth connection. Subsequently, the mobile device can forward the data to the cloud. However, ISOBlue only collects CAN data.

2.3 Open-Source Farm Machinery Data Platform Development

The shortcomings in proprietary and custom farm machinery data platforms call for the need to create improved platforms for logging farm machinery datasets. To create such platforms, there are a few design and implementation contraints. The first requirement is that a suitable computing hardware platform needs to be carefully selected to provide a wide range of peripherals for automatic data acquisition, wireless communication, and future customiza-

tions. Moreover, the hardware components need to be protected by a ruggedized enclosure to withstand heat, dust, water spills, and vibration. In terms of software implementations, a power manager is needed to automatically wake up and suspend the platform according to the machine power state. Furthermore, an opportunistc data streaming scheme needs to be considered to address intermittent network issues in rural areas. Last but not least, the development process needs to be open-source and ready for community adoption.

These constraints became the underlying design philosophies for two new open-source farm machinery sensing platforms, ISOBlue 2.0 and ISOBlue HD. They are presented and discussed in next sections for their hardware selections and software implementations.

2.3.1 ISOBlue 2.0

ISOBlue 2.0 was created to automatically log CAN, GNSS, and diagnostic data. It was powered directly from the ISOBUS diagnostic port power pins. In addition, it was designed to automatically suspend and wake up according to machine CAN activities. Moreover, it was made to stream data opportunistcally to the cloud via an open-source streaming tool called Apache Kafka [39].

Hardware Components

A single board computer shown in Figure 2.2 was chosen as the core computing hardware. Its specifications [40] are highlighted in Table 2.2. The board was chosen for a few reasons. First, the on-board quad-core processor, coupled with 2 GB of RAM, provided plentiful computing power for logging tasks. Second, the on-board power regulator supported a wide input power range from 7 to 27 V, which is compatible with the in-vehicle diagnostic port's power without extra power conversion. Third, the board includes peripherals and connectors for interfacing with additional hardware components. This not only satisfies the peripheral needs for making this platform but also opens up possibilities for bidirectional communication with new sensors or even in-cab information systems like yield monitors in the long run.

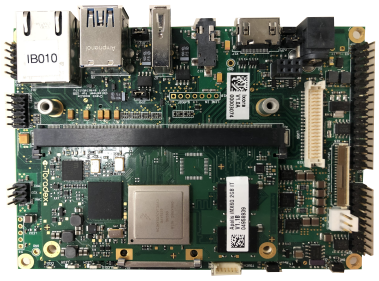


Figure 2.2. Photo of single board computer used in ISOBlue 2.0 with industrial I/O options.

Table 2.2. Highlights of single board computer specifications.

Feature	Description
Processor	ARM® Cortex-A9, 800 MHz, quad-core
RAM	2 GB DDR3
Flash	4 GB eMMC flash
Interfaces	mSATA, miniPCIe, Ethernet, CAN, USB, GPIO
Input power	7 to 27 V

A systematic overview on how the single board computer interfaces with other hardware components is illustrated in Figure 2.3. ISOBlue 2.0 connects directly to an ISOBUS diagnostic port for both power and CAN bus connections. Sensors, cellular module, and hard drives are either built-in or connected to the single board computer for sensing, storing, and streaming data. In regard to power distribution, most components are powered directly through connected peripherals (mSATA, USB, and miniPCIe) from the single board computer. However, there is one exception: the on-board Real Time Clock (RTC) is powered by a standalone 3 V battery. The dedicated battery allows the RTC to keep time even when the board is completely off.

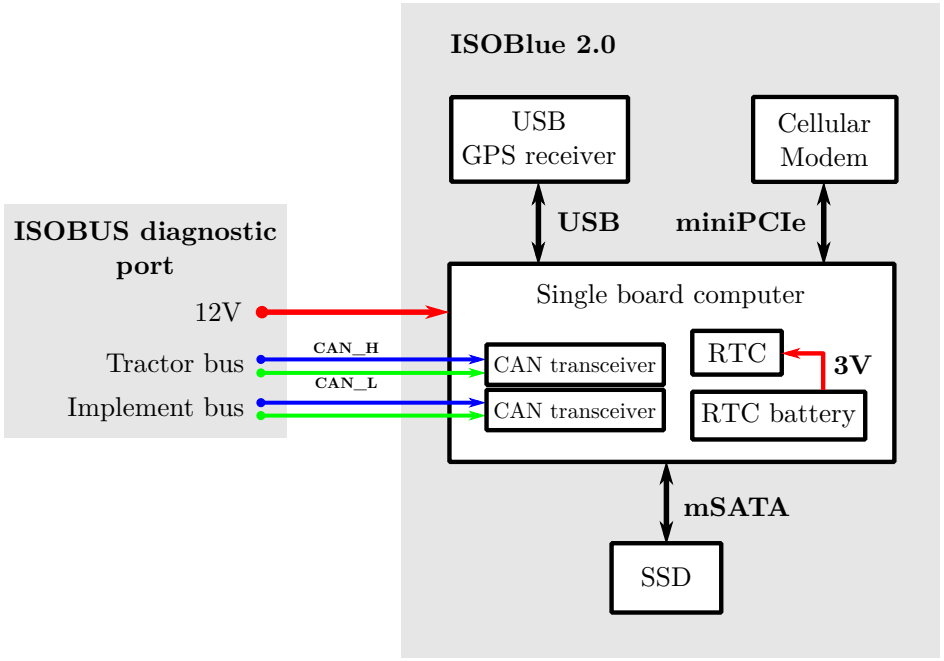
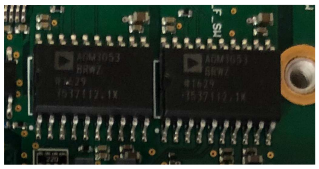


Figure 2.3. System connection diagram of ISOBlue 2.0.

Data Sources

Two types of electronic components illustrated in Figure 2.4 were utilized as data sources for CAN and GNSS data. Their part model names and nominal data rates are provided in Table 2.3. For sensing CAN signals, the single board computer came with two identical CAN transceivers. Each transceiver was configured to have a baud rate of 250 kbps for converting CAN bus signals into bitstreams. The bitstreams were further processed by custom programs to construct CAN frames. Each CAN frame was roughly 12 bytes (with the extended CAN frame format). Hence, with a nominal acquisition rate of 700 frames per second, approximately 8.4 kB of CAN data was written to the disk each second.

For receiving GNSS data, a USB GPS module was utilized. Upon the reception of stable satellite signals, the GPS module reported new GPS fixes at 1 Hz with a reported accuracy of 2.5 meters [41]. The fixes included both geospatial and accuracy data which totaled to around 26 bytes. In addition, the GPS module sent Pulse Per Second (PPS) signals over USB. These signals were used for keeping system time accurate.



(a) On-board CAN transceivers.

(b) USB GPS module.

Figure 2.4. Photos of CAN transceiver and USB GPS module used as primary data sources for collecting CAN and GNSS data.

Table 2.3. Specifications of data sources illustrated in Figure 2.4.

Component	Quantity	Model	Nominal Data Rate
CAN transceiver	2	Analog Device ADM3053 [42]	700 frames/second
USB GPS module	1	Navisys GR-701W [41]	1 message/second

Additional Components

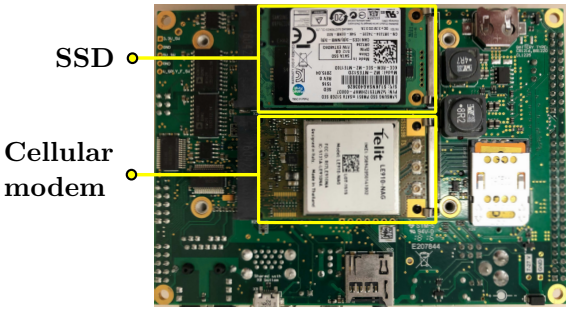


Figure 2.5. A cellular modem and a SSD were installed into the corresponding slots for providng network access and data storage.

Additional components were installed to enable network access and provide disk space for data storage as shown in Figure 2.5. A Telit LE910 [43] cellular modem was installed with two patch antennas for 4G/LTE network access. In terms of data storage, a 512 GB

mSATA Solid State Drive (SSD) was included to supply dedicated disk space for CAN and GNSS data storage.

Enclosure

All hardware components except the USB GPS module were securely housed in dust- and water-resistant enclosure as shown in Figure 2.6. A custom-made plexiglass plate was placed at the bottom of the enclosure as the base for mounting hardware components. For CAN and USB connections, both the installed ISOBUS diagnostic and the USB coupler provided sealed connections for interconnecting internal and external components. Although both the enclosure and the couplers are rated IP67, the platform as a whole does not meet strictly to any protection standard. It is a viable solution for this platform as the primary focus for having an enclosure is to minimize the chance of hardware damage which could lead to data corruption during deployment.

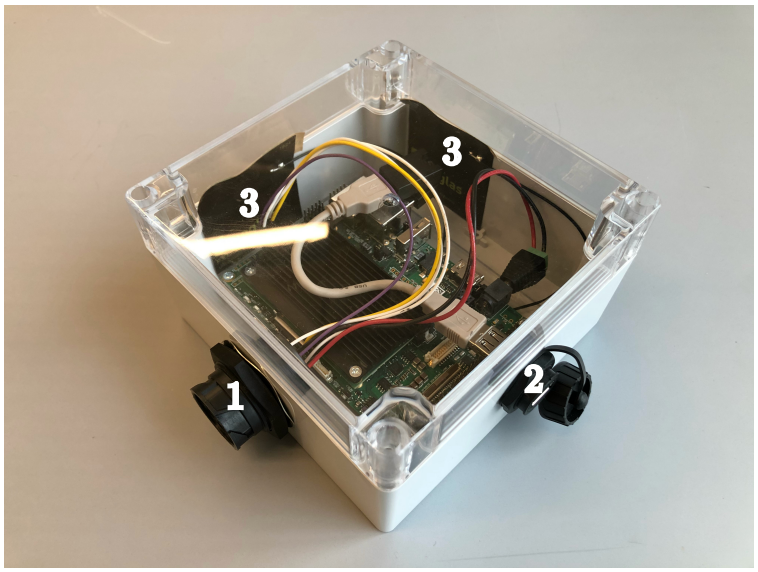


Figure 2.6. Photo of ISOBlue 2.0 hardware in an IP67-rated enclosure. The numbered components are 1) ISOBUS coupler, 2) USB coupler, and 3) low-profile patch antennas.

Software Development

The software implementations focus on the customiza67tions of an open-source Board Support Package (BSP) provided by the board manufacturer. As overviewed in Figure 2.7, the BSP contains Ångström [44]—an lightweight operating system, applications that encapsulate system management programs, power management programs, a Kafka cluster, and data loggers with additional device drivers and middlewares. Custom applications were written in a generic fashion that could be easily ported to another platform if needed. All source code was developed under the Apache 2.0 license and is available on GitHub [45].

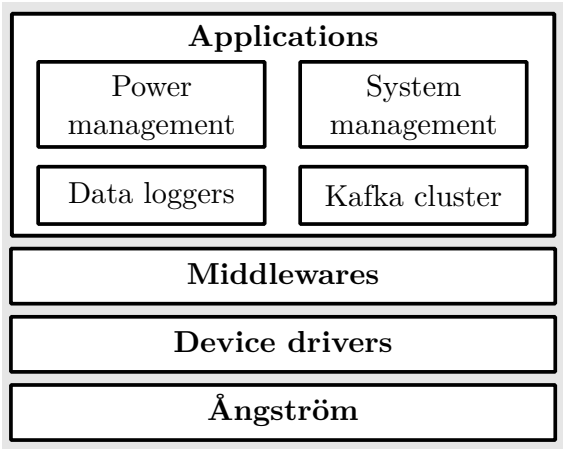


Figure 2.7. The custom Board Support Package (BSP) contains four areas of applications running on an operating system called Ångström: system management, power management, data cluster, and data logging.

System Management

Pre-existing open-source applications listed in Table 2.4 were configured for managing system hardware and application executions. For managing hardware, *udev* [46] employed custom rules to automatically monitor hardware and triggered actions upon hardware changes. Three custom *udev* rules were written to: 1) bring up two CAN interfaces with the correct baud rates, 2) enable wake-on-CAN feature, and 3) establish cellular connections with an Access Point Name (APN). Moreover, a GPS service daemon called *gpsd* [47] was installed for interfacing with the GPS module. This daemon periodically fetched data from the GPS

module and made them available on a TCP port. For overseeing and automating program executions, *systemd* [48] was utilized. Applications used dedicated *systemd* service files to specify execution orders, dependencies, start/restart policies.

On the network side, *openssh* [49] was utilized to port-forward a local port to a remote desktop. This enabled remote access to the platform for debugging purposes.

For system time-keeping, *chrony* [50] was configured to correct system time according to reliable time sources. The application was supplied with two time sources: 1) a list of Network Time Protocol (NTP) servers and 2) the PPS source from the USB GPS module. These two sources provided a fail-safe way to synchronize time regardless of Internet availability.

Table 2.4. A list of open-source system management applications for monitoring overall system, network configurations, and system devices.

Application	Description
`udev` [46]	System device manager
`systemd` [48]	System service daemon
`openssh` [49]	Networking tools using Secure Shell (SSH)
`chronyd` [50]	Clock synchronization daemon
`gpsd` [47]	GPS service daemon

Power Management

An application called *can-watchdog* was written to control the power of the hardware according to the machine power state. The workflow of the applications is visualized in Figure 2.8. Specifically, *can-watchdog* employs a combination of a Linux timer and SocketCAN [51]—a kernel driver that implements CAN protocol for Linux and provides CAN interfaces as network sockets. Once the application starts, it initializes two CAN sockets and a 10-second timer. Then, it enters into a loop that continuously listens for incoming CAN frames and resets the timer if there are new CAN frames. If the application sees no incoming CAN frame for 10 seconds, it would issue a command to suspend the single board computer.

After the suspend, the single board computer saves all system states in its memory; leaving one working CPU core and the CAN transceivers on. Whenever the machine is turned back on, CAN activity would resume. The working core is sent an interrupt by the CAN transceiver, waking up the board.

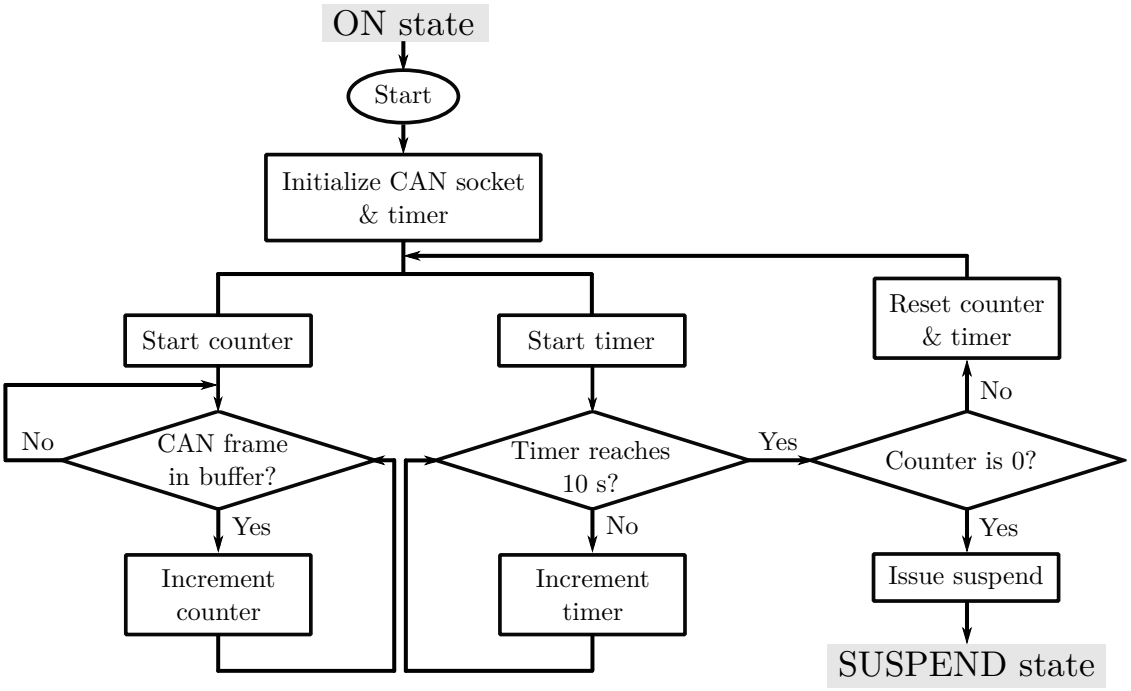
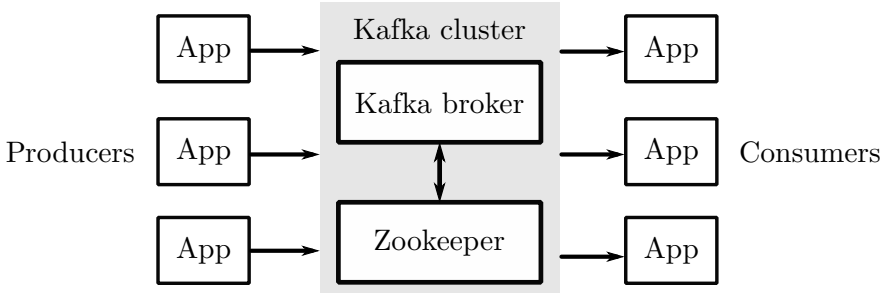


Figure 2.8. Flowchat of the custom application *can-watchdog* for system power management.

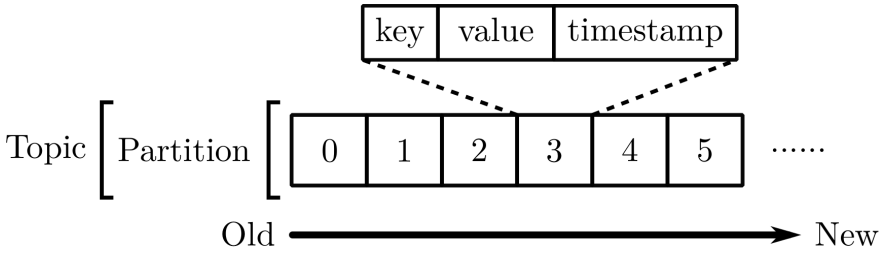
Kafka Cluster

Apache Kafka [39] was chosen as the data exchange system to log CAN, GNSS, and diagnostic data for its robust data exchange between applications and systems [52], [53]. The basic components of Kafka contain a Kafka cluster and Kafka clients (producers and consumers) as shown in Figure 2.9a. A Kafka cluster consists of two components: a *Kafka broker* and a *Zookeeper*. A *Kafka broker* handles incoming and outgoing messages from producers and consumers while a *Zookeeper* manages the topology within a *Kafka broker*. They work jointly to coordinate with clients for data storing and distribution. Data, referred to as records in Kafka's convention, are stored in different topics. The topics for storing CAN, GPS, and diagnostic data are listed in Table 2.5. Moreover, all records in Kafka topics are

saved in an immutable and append-only sequence. The immutability guarantees the order of the data on the disk. As the data of interest are all time-series, this feature is particularly helpful as it prevents potential data shuffling.



(a) A generic setup for a Kafka platform.



(b) Anatomy within a Kafka topic.

Figure 2.9. The setup in **(a)** shows a basic Kafka setup that consists of a Kafka cluster and clients. The Kafka cluster is comprised of a *Kafka broker* and a *Zookeeper*. Kafka clients either push (producers) or pull (consumers) data to or from a Kafka broker. The data are stored as immutable records within a Kafka topic as shown in **(b)**.

The Kafka software was downloaded from the Apache Kafka's archive [54]. Running instances of a *Kafka broker* and a *Zookeeper* forms a Kafka cluster. The cluster manages four Kafka topics in Table 2.5. Data loggers described in the next section connect to the cluster as Kafka producers to publish sensor data to these topics. Moreover, the data streaming is enabled via a Kafka software called *MirrorMaker* synchronizes data stored in the *remote* topic to a remote Kafka cluster via a protected SSH tunnel. The synchronized data provides diagnostic and geospatial information of the platform for visualization and debugging purposes.

Table 2.5. Data stored in different Kafka topics.

Topic	Description
`imp`	Implement bus data
`tra`	Tractor bus data
`gps`	GPS data
`remote`	GPS and diagnostic data

Data Loggers

Different data loggers were developed to record four types of data as shown in Figure 2.10. For logging CAN, GNSS, and diagnostic data, three custom Kafka producers were implemented: *kafka-can-log*, *kafka-gps-log*, and *heartbeat*. Two of these producers (*kafka-can-log* and *heartbeat*) were written in C and *kafka-gps-log* was written in Python. As a result, a mixture of middlewares in Table 2.6 were installed to provide runtime libraries to these loggers.

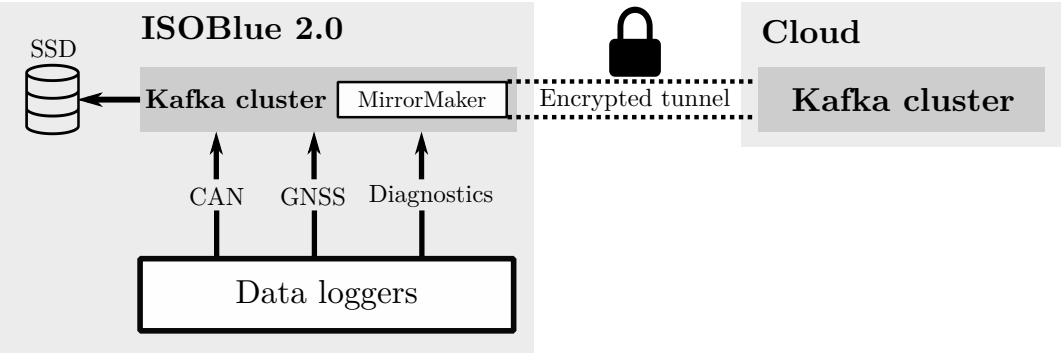


Figure 2.10. The Kafka cluster was utilized for logging CAN, GNSS, and diagnostic data onto the SSD. The cluster communicated with a remote Kafka cluster via an encrypted tunnel for mirroring diagnostic information.

Table 2.6. Preinstalled middlewares for the Kafka data loggers.

Name	Description
`librdkafka` [55]	Apache Kafka client C API
`kafka-python` [56]	Apache Kafka client Python API
`avro-c` [57]	Apache Avro serialization C library
`avro-python` [58]	Apache Avro serialization Python library
`gps3` [59]	Client Python library for `gpsd`

These applications share a common workflow as illustrated in Figure 2.11. Each application starts by connecting to the Kafka cluster. It then attempts to connect to a data source and subsequently enters a loop for reading, serializing, and eventually publishing data to the Kafka cluster via Kafka client APIs. The differences stem from the data sources. Specifically, *kafka-can-log* creates a *SocketCAN* network socket for listening on a CAN bus; *kafka-gps-log* connects to *gpsd* via *gps3* for fetching GPS data. Meanwhile, *heartbeat* directly queries Internet connection status and cellular strength once per second using a system command. Once the data from different loggers are published to the Kafka cluster, they are stored in the SSD. It is noteworthy that after data serialization, the original data is transformed into binary data that uses significantly less disk space than storing the original data. For example, a 64-bit long CAN data payload is stored as a 64-bit long binary number after serialization in comparison to a 16-character long hexadecimal string which requires a total of 1024 bit of disk space assuming one character needs 1 byte of disk space.

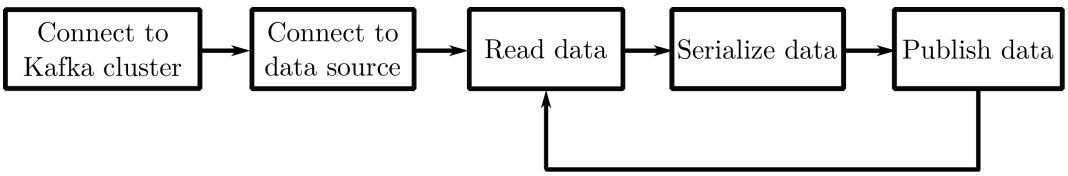


Figure 2.11. The Kafka data loggers share a similar workflow yet differ in data sources. Each logger initializes a loop that continuously reads, serializes, and publishes data for data collection.

2.3.2 ISOBlue HD

ISOBlue HD was an extension of ISOBlue 2.0 which demonstrated the original platform's expandability. The main purpose of ISOBlue HD was to collect context-rich farm machinery datasets that encapsulate CAN, GNSS, and video data. No previous platform has been created to collect such datasets that contain both detailed machine information (CAN and GNSS) with minable content (video) for inferring contexts in farming tasks.

Hardware Additions

ISOBlue HD utilized the same single board computer specified in Table 2.2 as the computing platform. Nevertheless, ISOBlue HD was built with a few additional components as shown in Figure 2.12. These components are 1) three IP cameras, 2) a Power-over-Ethernet (PoE) network switch, 3) a custom relay Printed Circuit Board (PCB), and 4) a USB hard drive.

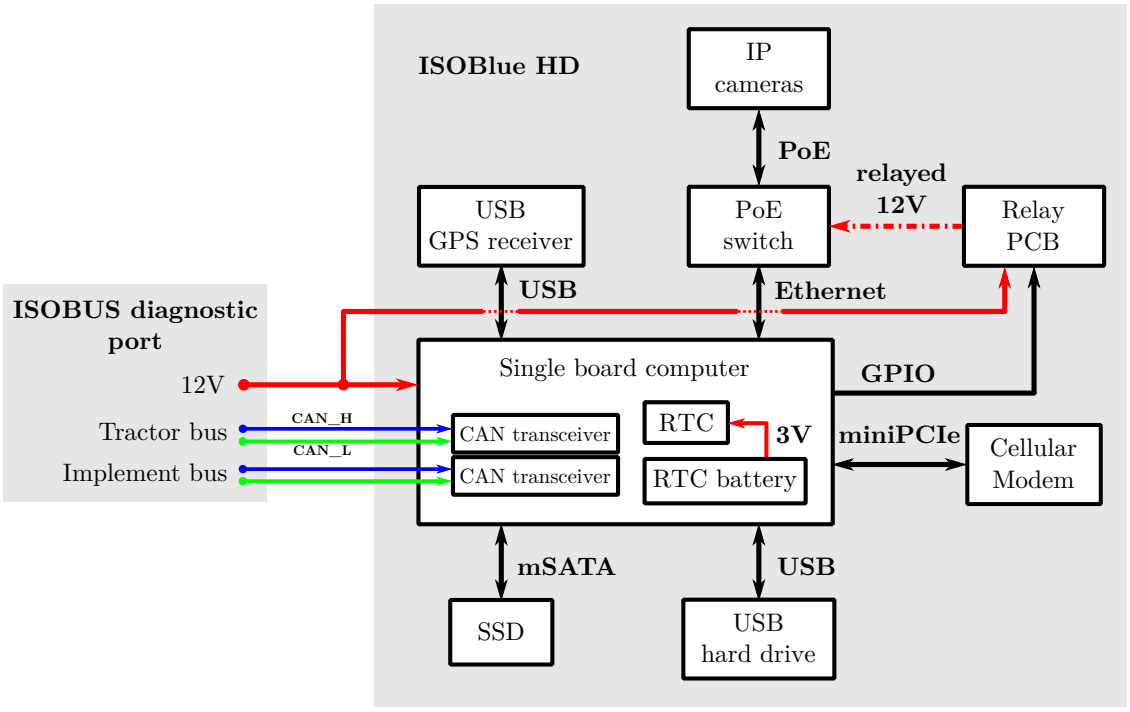


Figure 2.12. System connection diagram of ISOBlue HD.

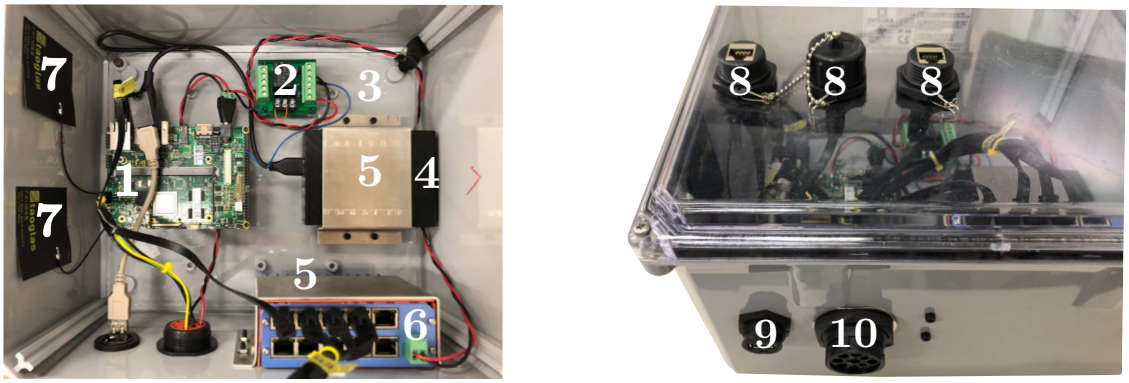
First, three Ubiquiti IP cameras [60] (Figure 2.13b) were added for capturing high definition videos. The reason for choosing cameras is that there is no similar sensor that offers both cost-effectiveness and direct visual perception of operational contexts. The tri-camera setup aimed to capture video data from various viewing angles. The cameras supported Real Time Streaming Protocol (RTSP) [61] and they were configured as Dynamic Host Configuration Protocol (DHCP) clients to make video streams accessible via IP addresses. Each video stream had a resolution of 1920 $\times$ 1080 at 30 fps (1080p) with an encoding rate of 6000 kbps. Moreover, the cameras and the single board computer established a Local Area Network (LAN) through a Veracity [62] PoE network switch. It supplied IEEE 802.3af [63] compliant electric power to the cameras and bridged data connection for the LAN via Ethernet cablings. By using wired connections, higher bandwidths and therefore better video quality could be achieved. In order to save captured video files, a 4 TB USB hard drive was utilized as the dedicated storage device.

(a) Custom relay PCB. (b) Internet Protocol (IP) camera.

Figure 2.13. A custom relay PCB in **(a)** was created to relay the unswitched 12 V power from the diagnostic port to the PoE network switch which subsequently turned on the IP cameras shown in **(b)**.

Moreover, a custom relay PCB (Figure 2.13a) was created for controlling power to the PoE network switch and the cameras. The PCB was controlled by a GPIO pin from the single board computer. The toggling of the pin was based on the power state of the single board computer. Each time the board went to suspend, the GPIO pin was set to low which signaled the relay PCB to cut the 12 V off to the network switch. Conversely, the relay PCB resumed the power to the network switch when the board was on and the pin was set to high.

Furthermore, an appropriately sized enclosure was selected for housing all the components as shown in Figure 2.14a. Since the IP cameras require Ethernet connections, three sealed Ethernet couplers were added onto the new enclosure along with the ISOBUS and USB couplers as shown in Figure 2.14b. The enclosure as well as the couplers are all IP67-rated. The goal is to minimize the chance of device damage that could cause data corruption in a machine deployment scenario.



(a) Enclosed components.

(b) Sealed couplers.

Figure 2.14. Photo of ISOBlue HD Hardware components housed in a sealed enclosure to withstand dust and vibrations as shown in **(a)**. Couplers shown in **(b)** provide sealed connections between internal and external components. The annotated components are: (1) single board computer (2) relay PCB (3) plexiglass base (4) USB hard drive (5) anti-vibration straps (6) PoE switch (7) cellular antennas (8) Ethernet couplers (9) USB coupler and (10) ISOBUS coupler.

Software Additions

The single board computer was configured as a DHCP server to communicate with the cameras via the PoE switch via an open-source tool called *dnsmasq* [64]. This tool utilized a configuration file that specifies: 1) a server IP address, 2) a list of MAC addresses of the cameras, and 3) an IP address range. Whenever the cameras were on, they automatically sent DHCP requests for becoming clients. Once *dnsmasq* received the requests, it would automatically assign IP addresses using the specified range. As soon as this process finished, video loggers could access camera streams via their IP addresses.

In terms of power management, a script called *sleep-hook* was written to work in conjunction with *can-watchdog* described earlier in Section 2.3.1. In case of no CAN activity, *can-watchdog* would issue a suspend command which also triggers *sleep-hook* to set the GPIO pin to low. This scheme both suspends the single board computer and signals the relay to shut off power to the PoE switch and cameras. When the board wakes back up, it triggers *sleep-hook* to set the state of the GPIO pin to high which turns the PoE switch and the cameras back on.

For logging the video streams, three *systemd* services were written to automatically record the RTSP streams via an open-source tool called *ffmpeg* [65]. Each service was responsible for connecting to the dedicated RTSP stream using an IP address and saving the stream to the USB HDD in 10-minute blocks as Audio Video Interleave (AVI) files. Each video file is named using the epoch timestamp when the recording starts. Moreover, it is worth noting that *ffmpeg* records raw video streams with no resizing or resampling. Hence, each resultant AVI file is able to retain the original video quality.

2.4 Platform Deployment and Data Acquisition

A fleet of ten ISOBlue 2.0s was made and deployed into different machines at various farms and research facilities in three states in the contiguous United States. A deployed ISOBlue 2.0 is shown in Figure 2.15. Moreover, a visualization that summaries ISOBlue 2.0 deployment locations and collected data sizes is illustrated in Figure 2.16.

Figure 2.15. Photo of an ISOBlue 2.0 in typical placement at the corner of the machine cab to minimize intrusions to the machine operator.

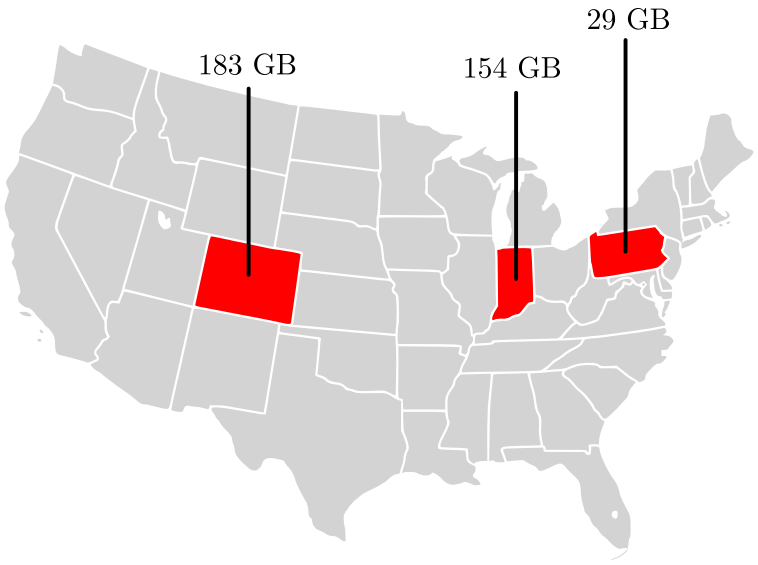


Figure 2.16. Map showing magnitude of data collected using ISOBlue 2.0s.

Moreover, one ISOBlue HD was made and deployed in a CASE IH Axial-Flow® [66] 6088 combine harvester (Figure 2.17a) during wheat harvest in July 2019. The harvest took place in Rochester, Indiana, USA. The device was connected to the ISOBUS diagnostic port and placed in a nonintrusive location in the machine cab. The IP cameras were mounted in three in-cab positions: two on the front windshield and one on the back window behind the operator seat. The goal of the front cameras was to capture video on the header status while

the back camera was to record operator actions on a joystick and control panel buttons. The cameras were secured onto glass surfaces using heavy-duty suction cups as shown in Figure 2.17b.

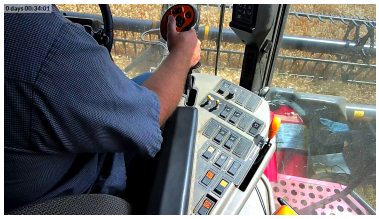
The platform was retrieved at the end of the harvest for data offloading. The offloaded data contained a total of over 230 thousand GPS points, 10 million video frames, and 69 million CAN frames which added up to 437 GB of data covering approximately 84 hours of harvest time. Figure 2.18a to Figure 2.18c illustrate sample frames captured from the three cameras.

(a) A combine harvester for deployment.

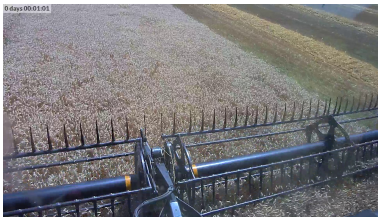
(b) IP cameras were mounted using suction cups.

Figure 2.17. ISOBlue HD was deployed into a combine harvester for collecting a wheat harvest dataset. IP cameras were fastened onto heavy-duty suction cups for mounting on different surfaces.

(a) Left side header view.



(b) Operator camera view.



(c) Right side header view.

Figure 2.18. The tri-camera configuration captured header statuses and operator actions.

2.5 Contextual Knowledge Mining

This section examines an hour-long ISOBlue HD data (CAN, GNSS, and video) from July 15, 2019. The purpose of the following subsections is to highlight exploratory find-

ings concerning the harvest and more importantly, present an extendable data processing template that integrates labeling, mining, and merging data sources within a context-rich dataset, as opposed to giving conclusive solutions or results on operational logistics and decision-making.

2.5.1 Contextual Label Generation

The video data were labeled for context according to two sets of contextual labels defined in Table 2.7. The first set encapsulates the header position of the combine harvester from the front camera views. On the other hand, the second set focuses on frequent joystick and control panel buttons that can be observed from the operator view.

Table 2.7. Header position and operator action contextual labels.

Label	**Description**
Header position	
Header up	Header is at up position.
Transition	Header transition (up to down or vice versa).
Header down	Header is at down position.
Operator actions	
None	No operator action observed.
Joystick 1	Header height and tilt.
Joystick 2	Reel height adjustment.
Joystick 3	Unload auger swing out/in.
Joystick 4	Resume header preset or raise header.
Joystick 5	Unload auger on/off.
Panel 1	Header rotation on/off.

Each video was labeled by an individual with domain knowledge. This was achieved via an open-source tool called MuViLab [67]. The tool uses a set of labels and a video data file as inputs and automatically splits the video data into short clips. These clips are then played in a loop with a matrix-like format in an application interface demonstrated in Figure 2.19. The individual could then iteratively label each clip until the clips run out. After this process, the tool generates a file that contains a sequence of clip-label mappings. This file was further processed to output a file that associates a sequence of labels to corresponding epoch timestamps. This process was repeated for video files from three cameras. As a result, three time-series label files were generated.

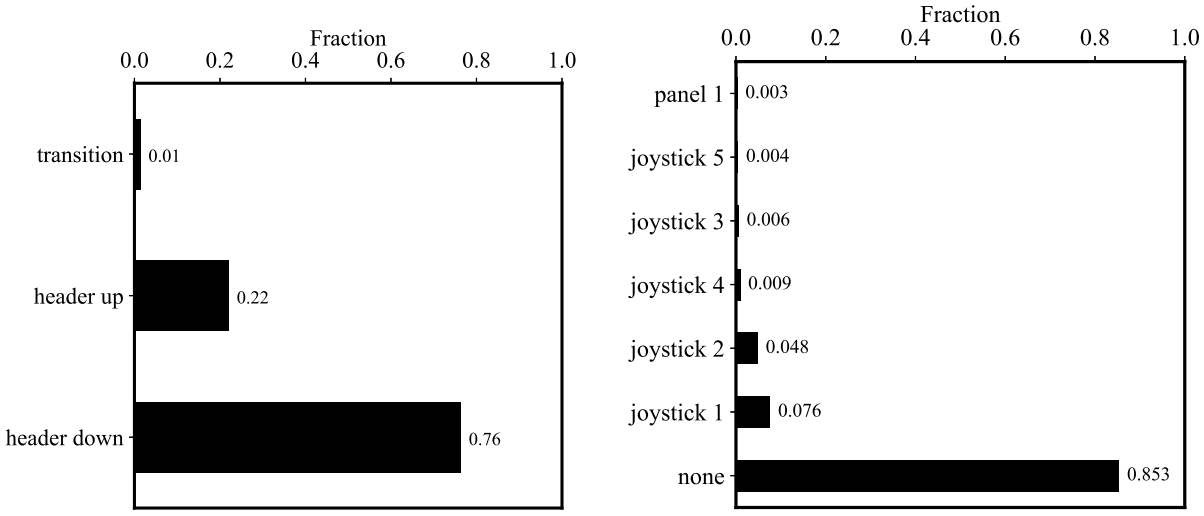
Figure 2.19. Screen capture of MuViLab open-source tool for simultaneously labeling multiple frames of a video In this instance, blue, red, green boxes each stand for "header up", "transition", and "header down" labels, respectively

2.5.2 Preliminary Contextual Knowledge

Preliminary contextual knowledge can be extracted from the time-series label files. Plots in Figure 2.20 visualize the fractions of the number of occurrences for header positions and operator actions. From Figure 2.20a, "header down" occurs more frequently than "header up" and "transition" as the combine header usually stays down during harvest. Moreover,

apart from the "none" label in Figure 2.20b, "joystick 1" (header height and tilt) and "joystick 2" (reel height) are the two most likely labels. This is likely due to the fact that the operator actively adjusts reel and header height throughout the harvesting process according to the crop status and the terrain.

Besides, plots in Figure 2.21 reveal more information by examining the Cumulative Distribution Functions (CDFs) of the inter-arrival times of header or operator action events specified by the labels. Figure 2.21a shows that both "header up" and "header down" labels have consistent yet short inter-arrival times. However, "header up" has a maximum of inter-arrival times of over 350 seconds while "header down" has a maximum of about 225 seconds. This difference in maximum is likely due to the difference in activities associated with these two labels. When the header is up, the combine is opted to transition to the next pass, unload crop, or park. On the other hand, when the header is down, the combine is normally limited to harvest crops. Hence, the additional flexibilities in activity with the "header up" label have increased its maximum inter-arrival times.

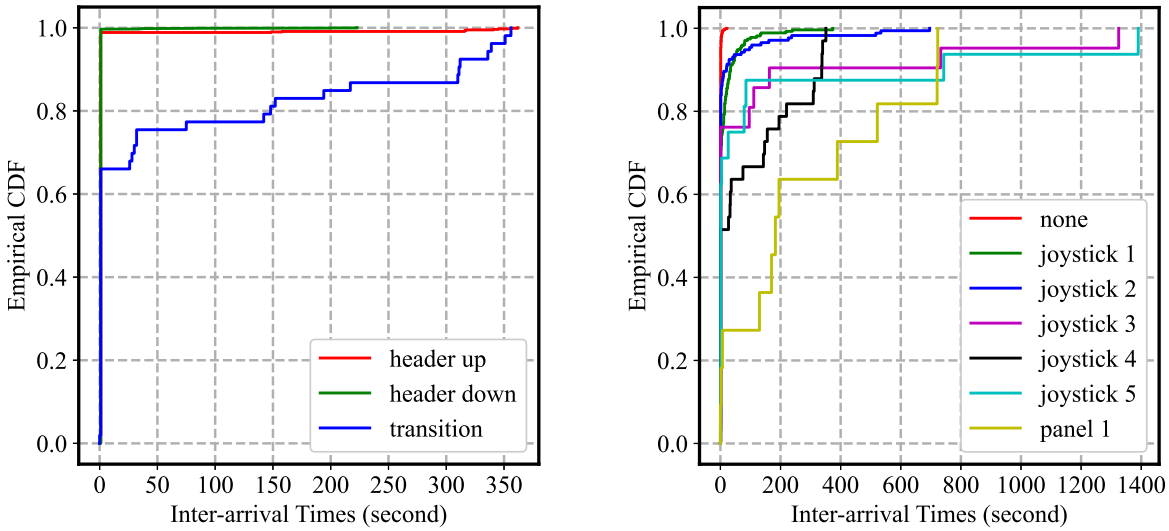


(a) Distribution of header position labels.

(b) Distribution of operator action labels.

Figure 2.20. Fractions of labels gives an idea of overall operations of the combine harvester. From **(a)**, "header down" is the logical status for header position as the combine needs to lower the header to harvest. For **(b)**, "joystick 1" and "joystick 2" has the second and third biggest fractions due to operator's consistent adjustment of header reel height and header height when cutting wheat.

Furthermore, the smoothness of the CDF curves in Figure 2.21b suggests the discrepancies in the frequencies of operator actions on various joystick and control panel buttons. The relative smoothness of "joystick 1" and "joystick 2" CDFs suggests these two actions are more frequent than the other actions. This not only corresponds to the relative high fraction of these two labels in Figure 2.20b but also aligns with the heuristic experience that an operator consistently makes subtle adjustment on both header reel and header height according to the height of wheat during harvest.



(a) Inter-arrival times for header positions.

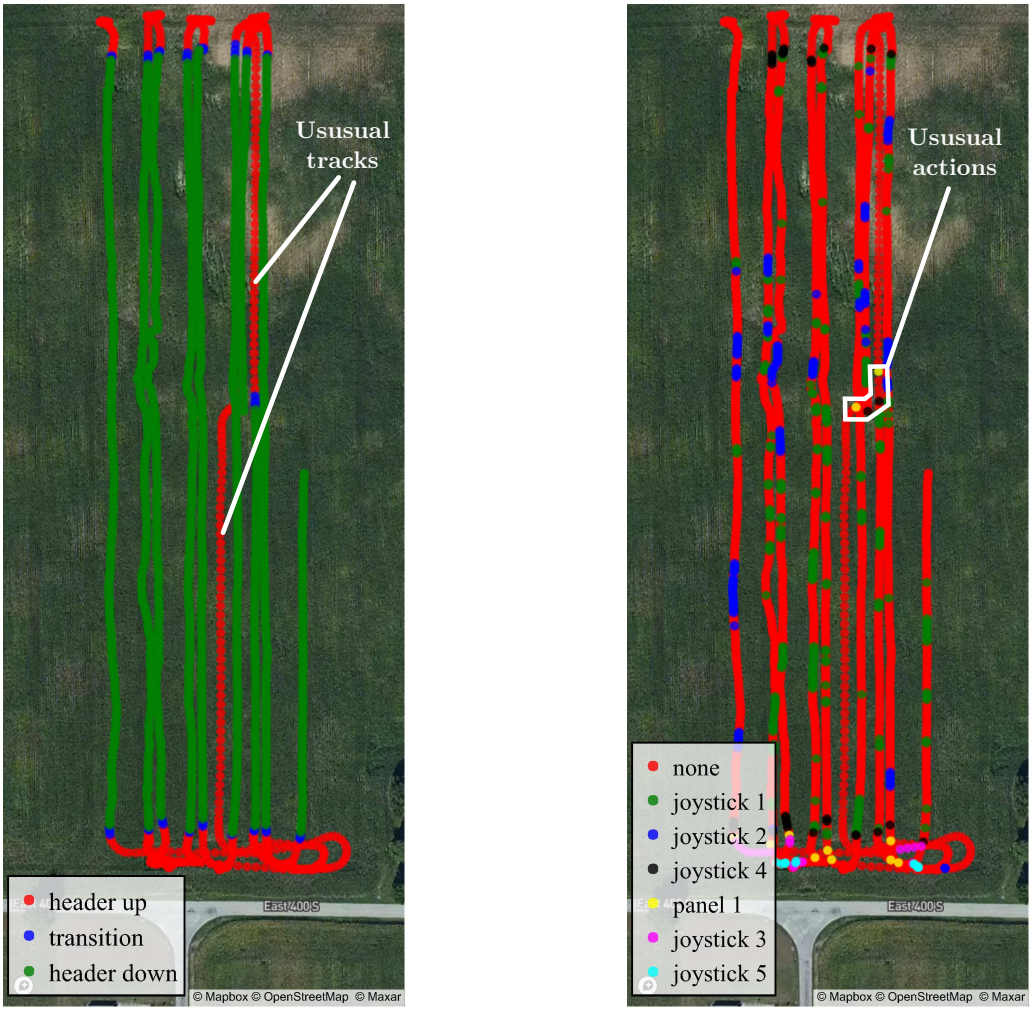
(b) Inter-arrival times for operator actions.

Figure 2.21. The empirical CDFs were estimated for the inter-arrival times for both sets of labels. Domain knowledge can be applied to explain the maximum of the inter-arrival times and the smoothness of the CDFs.

2.5.3 GPS Tracks with Contexts

The combine's GPS track visualizes where the machine has been during harvest. Analyzing the track alone reveals limited information on the navigation choice done by the operator. On the other hand, by incorporating contextual labels, the reasons behind certain choices could be inferred. For example, as shown in Figure 2.22, the one-hour GPS track comprising approximately 9 passes for harvesting was merged with contextual labels based on timestamps. GPS coordinates in Figure 2.22a are highlighted using the header position labels. The "header down" coordinates indicate where the wheat is cut. Moreover, the "tran-

sition" coordinates provide a clear cut-off where the combine harvester stops harvesting. In addition, the "header up" coordinates suggest maneuvers such as turning, reversing, etc., in headlands. Apart from these obvious observations, by looking at Figure 2.22b jointly, the two unusual passes map to a sequence of unusual operator actions (header up/down and reel on/off) in the middle of the field which could potentially be caused by a full grain bin or machine anomaly; an inference that could not be made without the contextual labels.



(a) GPS track with header position contexts. (b) GPS track with operator action contexts.

Figure 2.22. GPS track data, paired with contextual labels, can reveal information that cannot be easily obtained from GPS track only. For example, header position contexts provide a clear separation between harvest and non-harvest area in **(a)**.

Figure 2.22b also highlights the relations between operator actions and geospatial coordinates. For instance, "joystick 1" (header height and tilt) and "joystick 2" (header reel height) occur frequently within each pass. Moreover, "joystick 4" (header reset/resume) only occurs during headland transitions. Actions "joystick 3" (unload auger extension) and "joystick 5" (auger on/off) are only triggered in headlands where a tractor-hauled grain cart is likely to park and wait for crop transfer. Moreover, the frequent usage of header height and reel height adjustments indicate the heavy involvement of human inputs during the harvesting process which could be sources of error and fatigue.

2.5.4 Extracted CAN Signals with Contexts

CAN data typically contain straightforward information about machine and operation status. However, the proprietary nature of CAN data renders most research effort exclusively to few OEM engineers and ones who have access to the decoding information. Nevertheless, by leveraging labels generated in Section 2.5.1, the following example offers a possibility to understand unknown CAN data from both a contextual and non-proprietary perspective or even reverse-engineer the unknowns for security concerns.

In this example, the CAN data were first examined by their payloads before merging with contextual labels. A logged CAN frame consists of a timestamp, a CAN ID, and a 64-bit payload. There are 29 unique CAN IDs in the dataset. These CAN IDs are usually associated with decoding schemes for extracting sensory data from the payloads—a critical intermediate step in studying machine status. However, CAN IDs and their underlying decoding schemes for their payloads are mostly proprietary. Various recent works [68]–[73] provide methods that compute the bit-flip rate and the bit-flip magnitude of each bit in the payload for estimation of payload boundaries without the reliance of proprietary information. Once boundaries have been identified, sensor data could be subsequently extracted. Following previous works, the bit-flip rate b for a particular bit of the 64-bit payload was obtained by dividing the total number of bit-flips over the total number of payloads for a CAN ID, which is:

$$b = \frac{\text{\# of times a bit flips}}{\text{\# of times a particular message is seen}}.$$

The bit-flip magnitude for a particular bit is defined as $\lceil \log_{10}(b) \rceil$. Using these definitions, bit-flip rates were first computed for all 29 CAN IDs. Bit-flip magnitudes were then evaluated and visualized in Figure 2.23. In this figure, the x axis specifies the bit number while the y axis specifies a particular CAN ID.

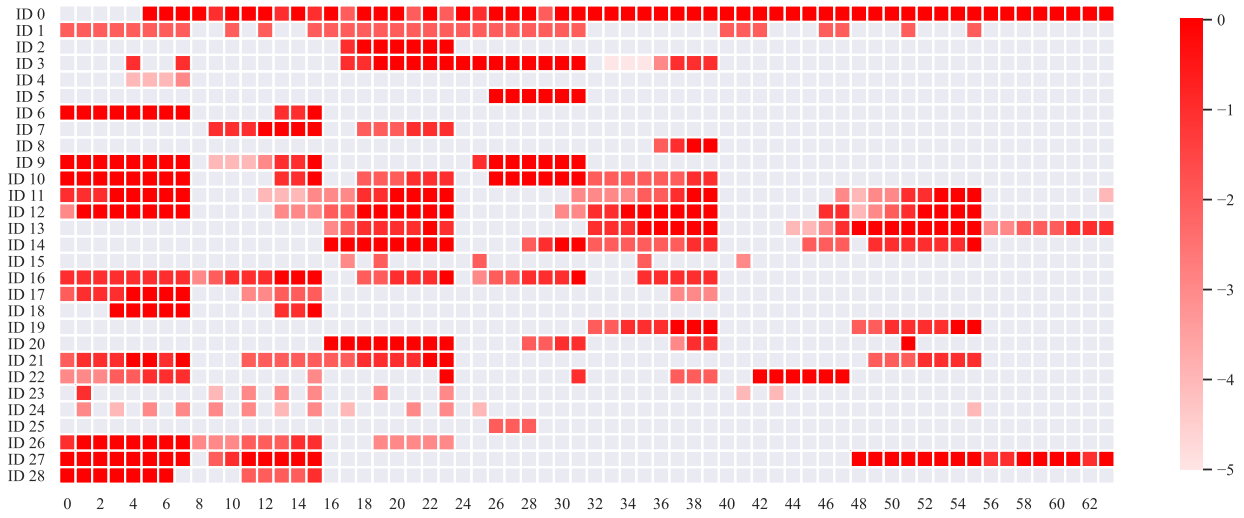


Figure 2.23. Bit-flip magnitudes for 29 unique CAN IDs are visualized. A larger magnitude value represents a more frequent change for a particular bit.

Physical signals can be extracted by visually inspecting and locating the payload boundaries from Figure 2.23. The extracted signals were further merged with contextual labels to uncover signal semantics with no prior knowledge about what the signal means. For instance, Figure 2.24 plots a series of CAN ID 27 signal traces (by concatenating byte 0 and 1) 20 seconds before and after the header down-to-up transition occurs. The figure also plots the average time instances when the operator presses "joystick 4" and when the header finishes transitioning to a down position. It can be observed that there is consistently approximately a one-second delay in between "joystick 4" presses and the start of each transition. In addition, the time needed for the header up to transition to down is on average 3 seconds. Moreover, after the header is fully in a down position, the signal traces gradually fall to 0 after a roughly 12-second delay. This phenomenon suggests that information encapsulated in byte 0 and 1 of CAN ID 27 could originate from a yield or flow sensor since the operator typically keeps the thresher on at the end of each pass for an extended time to thresh the

last bit of crop. Although this suggestion is heuristic and speculative, it could not have been generated without combining contextual labels with CAN data.

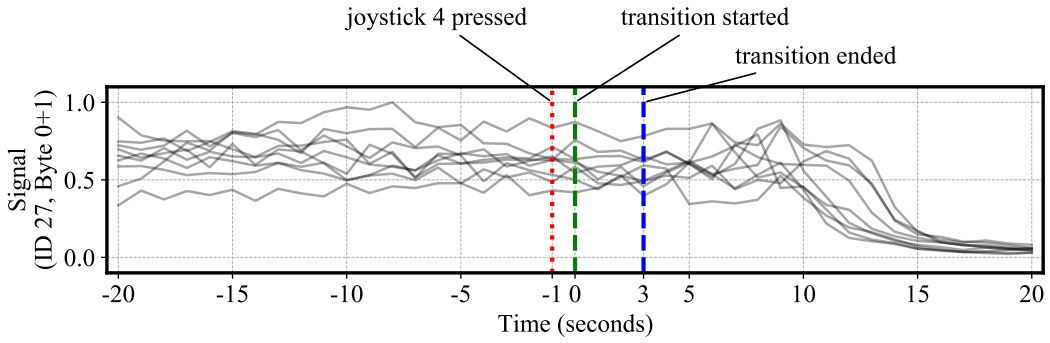


Figure 2.24. Extracted signal traces from CAN ID 27 are visualized with both selected header position and operation action labels. By incorporating context, the meaning of the signal traces can be interpreted as a flow or yield sensor readings.

2.6 Conclusions

Both ISOBlue 2.0 and ISOBlue HD were deployed in past farming activities to collect multimodal farm machinery datasets. Specifically, ISOBlue HD successfully collected CAN, GPS, and video data from a combine harvester during a 2019 wheat harvest. The video data captured header status and operator actions. The dataset was processed to generate preliminary context-rich results. The video data were first labeled with header position and operation action labels to generate time-series contextual label files. These contextual labels were analyzed to generate preliminary knowledge of the distributions and frequencies of different labels. Moreover, the contextual labels were merged with CAN and GPS data based on timestamps. Both CAN and GPS data reveal unique perspectives with contextual labels. GPS coordinates highlighted with header position labels provided clear distinctions between harvest vs. non-harvest areas. On the other hand, the extracted CAN signal was paired with both header position and operator action labels to infer its meaning with no prior knowledge of the signal.

Despite the success in deployment and data collection, both platforms and the post-processing pipeline discussed in Section 2.5 have room for improvement in future works.

First, the enclosure platform was bulky and inconvenient to move around in the cab. Its size could be reduced and more efforts are needed to make a sealed enclosure for in-cab or even cabless environment. Meanwhile, the platform experienced inconsistent Internet access which caused delayed data streaming and failed remote login attempts on multiple occasions. Although sporadic network access is a common problem in rural areas [74], an intelligent data streaming policy needs to be implemented to batch data during sub-optimal network conditions and send data opportunistically whenever the network becomes available. Moreover, although the Kafka cluster performed well for storing data, it was cumbersome to fetch data from the cluster during post-processing as the logs could not be queried like a traditional database. Hence, it would be reasonable to replace Kafka with a more compact database that provides both indexed data logging as well as querying capabilities. In addition, for scaling up the experiment, multiple ISOBlue HDs could be made and deployed in different machines for setting up a distributed context-rich sensing network that enables real-time inference of operational contexts as well as potential context sharing. This potentially needs more minable data from other operations (tilling, spraying, etc.) from a fleet of different machines. Lastly, to realize context-sharing, an efficient data transmission scheme that involves proper compression, error control, etc. is essential in ensuring reliable sharing of inferred contexts among machines in rural areas. Limited network connectivity in these areas requires, for instance, eliminations of unwanted redundancy of CAN payload bits to save transmission bandwidth which could be an extension of the work discussed in Section 2.5.4.

In terms of post-processing, the manual data processing steps in Section 2.5 could be enhanced in a couple of aspects; much of them could better be addressed with specific applications. For instance, one application of improving machine function comes directly from the data discussed in Section 2.5.3; the extremely frequent adjustments of joysticks 1 and 2 (header height and tilt, reel height) are a major source of fatigue. Artificial Intelligence developments to capitalize on video processing and other available layers (perhaps aerial imagery, topography, knowledge of the neighboring pass) to automate these adjustments could improve safety along with operation effectiveness. This requires automatic labeling and processing of video data from different camera views using computer vision algorithms

as highlighted in works like [75] and [76] for studying the relations among hand/finger movements, header positions/motions, and overall harvesting efficiency. This knowledge could be further combined with geospatial and CAN data for pushing machine performance "to the limit" in terms of optimized path planning and engine capacity without sacrificing threshing and cleaning quality. All of these insights rely on a comprehensive data pipeline—the focus of future works, which encapsulates research topics like GNSS track clustering, signal extraction/interpretation from CAN data, and sensor signal classification [77].

3. DATA-DRIVEN ACTIVITY ANOMALY DETECTION AND CLASSIFICATION

Acknowledgement: this chapter is a revised version of the paper [24] published in the Proceedings of the 14th International Conference on Precision Agriculture.

3.1 Introduction

In modern agriculture, machinery has become the one of the necessities in providing safe, effective, and economical farming operations. In a typical farming operation, different machines perform different tasks, and sometimes are used together for collaborative farming activities. In such cases, different machines are associated with representative activity patterns. For example, in a harvest scenario, combines move through a field following regular swaths while grain carts follow irregular paths as they ferry grain from combines to trucks. Sometimes unusual conditions due to soil status, machine status, weather, or human factors may cause anomalous activity patterns. Detecting and classifying anomalous patterns can be used for planning and efficiency improvements. This main purpose of this chapter is to explore the feasibility of applying the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [78] on a 3-Dimensional feature dataset created by Kalman filter residuals, machine speed, and engine load data from a combine harvester during wheat harvest. The results showed clear delineations between in-field and on-road data points. Moreover, the results also demonstrated the algorithm was able to distinguish normal (harvesting, on-road travels) vs. abnormal combine activities (in-field turns, sudden stops, etc.)

3.2 Related Works

Zhang performed rudimentary work [79] utilizing GPS paths for classifying different machines' activity patterns using a rule-based algorithm to understand what has happened in the field for farm logistics improvements. Recent works [80]–[82] suggested that the DBSCAN

algorithm can be used to cluster point clouds for anomaly detection, pattern recognition, and performance analysis.

Nevertheless, this type of data mining has yet to be performed for agricultural datasets. For a harvest scenario, a fleet of machines are utilized for accomplishing different tasks. Among these tasks, combine harvesters are one of the most important pieces of equipment as they are capable of efficiently harvesting crops, temporarily storing crops, and finally unloading crops to designated locations. Learning about combine activities in terms of their patterns and anomalies is the essential step in studying their correlations to as-harvested data like crop yield. These became the incentives to 1) develop a feature dataset that reflect the operation mode of a combine harvester and 2) apply a suitable clustering algorithm such as DBSCAN to test its feasibility for classifying combine operational patterns.

3.3 Data Collection and Preprocessing

In normal combine operations, the combine usually travels in near constant speed in a straight line for wheat cutting. On the other hand, changes in combine speed typically indicate in-field or on-road maneuvers like turning and accelerating. These information are usually encapsulated in GPS data stream. Moreover, engine status such as engine speed or load from machinery CAN data might also reflect these sudden maneuvers. Hence, the data acquisition in this work include both CAN and GPS data. These data were collected from a CASE 6130 [83] combine harvester using an ISOBlue 2.0 described in Chapter 2 during two wheat harvest sessions in 2017. The location of wheat harvest was in Amherst, Colorado, USA. The GPS data had a frequency of 1 Hz and it consisted a total of 52116 samples. Each GPS data sample contains an epoch timestamp (second), vehicle speed (meters/second), latitude, and longitude. The latitude and longitude data pairs were converted into Universal Transverse Mercator (UTM) coordinates (easting and northing) for later processing needs. Figure 3.1 illustrates an annotated version of the complete GPS track data.

Moreover, engine load data were selected and parsed from the collected CAN data. Engine load data were selected because it is a direct indicator of power usage in a combine harvester [84]. In other words, the changes in combine operation modes would exhibit fluctuations in

the engine load measurements. In addition, the engine load data had a frequency of 10 Hz and each engine load sample came with an epoch timestamp. The data were decimated to 1 Hz to match with the GPS data rate. This was done by averaging every 10 samples in the engine load data. The engine load as well as the GPS samples were joined based on their timestamps. Figure 3.2 shows a time-series representation of the aligned combine speed and engine load data. In time intervals $t \in [0, 1\times10^4]$ and $[4.2\times10^4, 5\times10^4]$ seconds, the combine harvester seemed to be harvesting; indicated by the consistent speed around 2.5 m/s and an average engine load about 75%.

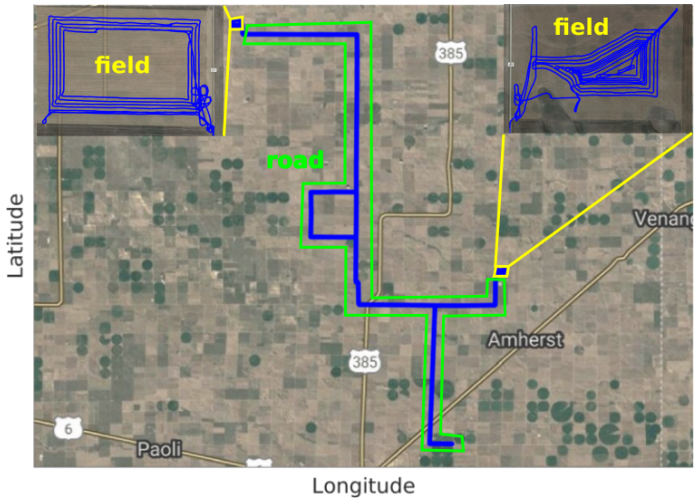


Figure 3.1. The GPS track of the 6130 combine harvester consists of both in-field and on-road data points. For the GPS data of interest, the combine worked on portions of two different fields.

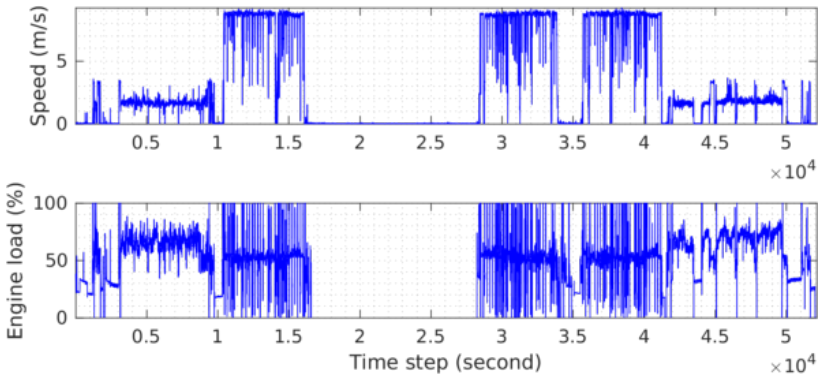


Figure 3.2. Sample time-series representation of the engine load and the speed signal show different modes of the combine harvester.

Furthermore, a scatter plot of the same engine load and the speed data could be created. Figure 3.3 illustrates an annotated version of the scattered plot by performing a visual clustering of the data points. Several observations could be drawn:

- Data points in the green oval encapsulate low speed and moderate engine load. These points may represent the scenario when the combine was harvesting in the field.
- On the other hand, the points in the red oval are associated with high speed values which could potentially link to combine's on-road travels.
- The data points outside of either the red or green ovals are "noisy" readings. It is likely that these points are related to the scenario when the machine was making maneuvers (turns, sudden speed changes, etc.).

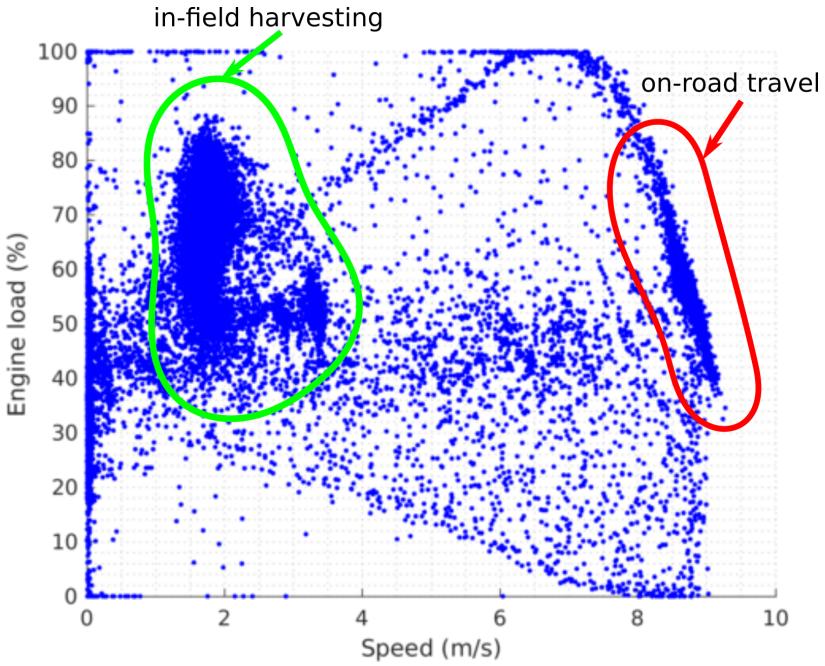


Figure 3.3. Scatter plot of engine load vs. combine speed which illustrates clusters associated with operating modes.

3.4 Generation of a 3D Feature Set

The engine load the speed data discussed in Figure 3.3 serves as a 2D feature dataset that would be extended in this section. This section describes a Kalman filter approach for

generating another feature: Kalman filter residual. This feature is further integrated into the 2D feature set to form a 3D feature set for clustering.

A Kalman filter is a well-known algorithm to track targets that move according to well-defined motion models. In this work, it is used to track combine position and tell how much the motion deviates from a constant velocity (CV) model via a parameter called Kalman filter residual at each time step. In a normal operation setting (either in harvesting or moving between fields), a combine usually travels at a constant speed. In this case, Kalman filter residual tends to be small as the CV model captures this motion well. On the other hand, if a combine changes its speed drastically during turning, sudden stops, accelerating, and decelerating, the CV model tends to perform poorly and the Kalman filter estimates would have a large but eventually self-corrected error. In other words, the residual serves as an indicator to show the smoothness of the combine motion. The state space representation of the model is well-known and can described as:

$$X(k+1) = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot X(k) + v(k), \tag{3.1}$$

$$Z(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot X(k) + w(k) \tag{3.2}$$

where $X(k)$ is the state vector defined as $\begin{bmatrix} x & \dot{x} & y & \dot{y} \end{bmatrix}^T$, $Z(k)$ is the measured state estimates, $v(k)$ is the process noise characterized by noise covariance Q, and $w(k)$ is the measurement noise characterized by noise covariance R. Noise covariances (Q and R), initial error covariances $\hat{P}(0)$ and initial state estimates $\hat{X}(0)$ are initialized in series of equations in Equation (3.3) and Equation (3.6). The process noise levels are selected to be small rel-

ative to measurement noise levels. This informs the filter to trust the filter predicted state estimates more than the measured state.

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}, \tag{3.3}$$

$$R = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \tag{3.4}$$

$$\hat{X}(0) = \begin{bmatrix} x(0) & 0 & y(0) & 0 \end{bmatrix}^T, \tag{3.5}$$

$$\hat{P}(0) = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

The Kalman filter was run and the corresponding residual values for positions in easting and northing (x and y) were saved. Figure 3.4 gives a time-series plot of the filter-generated Kalman filter residuals.

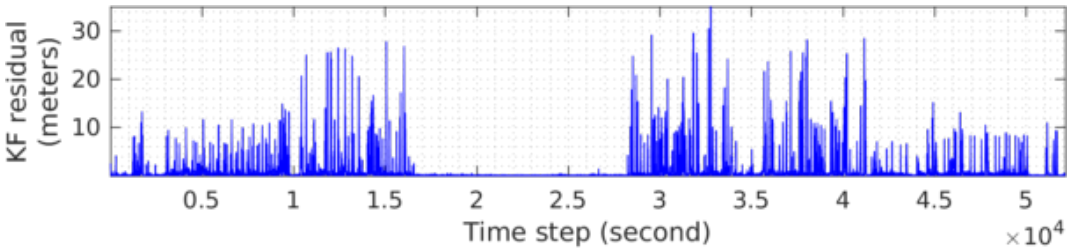


Figure 3.4. The Kalman filter residuals have a number of spikes throughout the dataset. The same speculations for combine operations as in Figure 3.2 could be drawn from observing the average magnitudes of the residual values in $[0, 1 \times 10^4]$ and $[4.2 \times 10^4, 5 \times 10^4]$ time intervals.

Finally, the residual values were joined with the engine load and the speed data. This resulted a 3D feature dataset which is visualized in Figure 3.5.

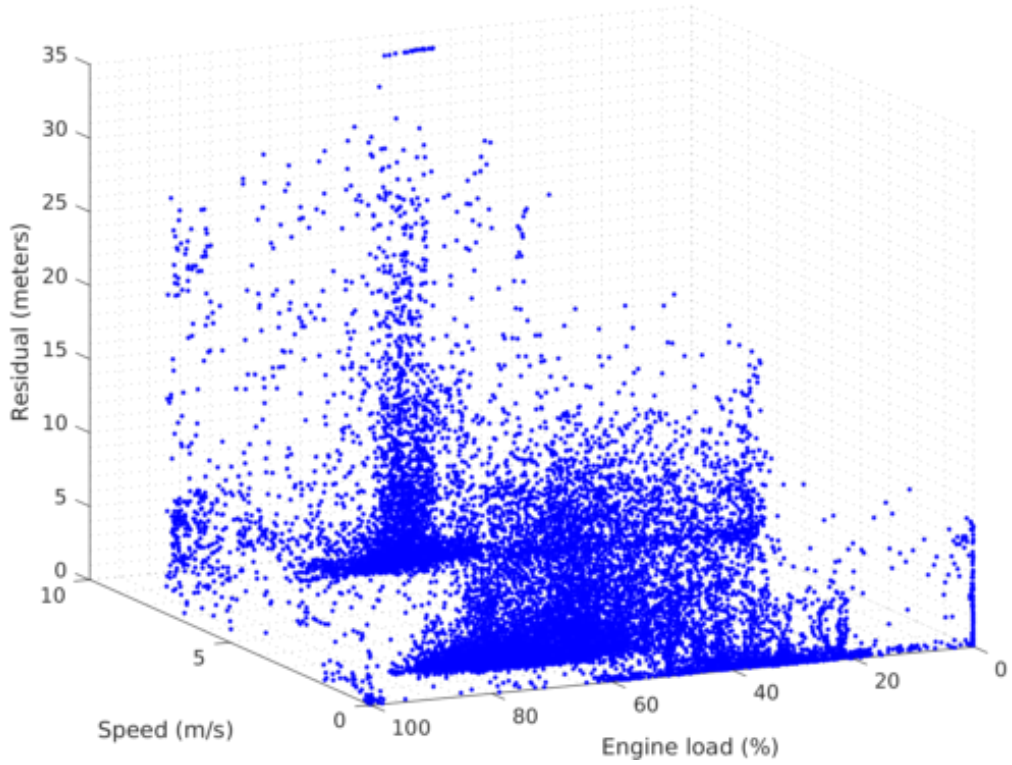


Figure 3.5. 3D scatter chart of Kalman filter residual values for many values of speed and engine load.

3.4.1 Spatial Clustering using DBSCAN

There are multiple clustering algorithms available for spatial data minining [85]. DBSCAN is selected as the clustering algorithm for classifying the feature dataset in Figure 3.5. There are two main reasons. First, it is a unsupervised method, which means that the algorithm can run without an expectation of how many clusters it needs to form. Second, its density-based approach performs well [86], [87] in discovering clusters with arbitrary shapes. Both reasons are applicable for the feature dataset of this work.

The algorithm requres two input parameters: ϵ and *minPts*. ϵ is the minimum distance between two data samples that could be considered as neighbors (i.e., within the same cluster). Moreover, *minPts* is the minimum number of data samples that is the required to form a cluster. As a example, Figure 3.6 illustrates the initial stage of the algorithm workflow. In this figure, r is defined as the minimum Euclidean distance parameter and *minPts* is defined as 3. The algorithm starts by visiting the first point of interest (POI)

and performs a range query to fetch all the samples that fall within the distance r. After this step, the two green seeds and the POI are considered to form a cluster. Afterwards, the POI is labeled as "visited" and then the algorithm repeats the same procedure on the two seeds for further expanding the cluster. This procedure goes on until all the points have been visited. Sometimes a range search returns no samples. In this case, the algorithm will then label the point as noise. Detailed explanations of how DBSCAN algorithm works can be found in **ester˙1996**.

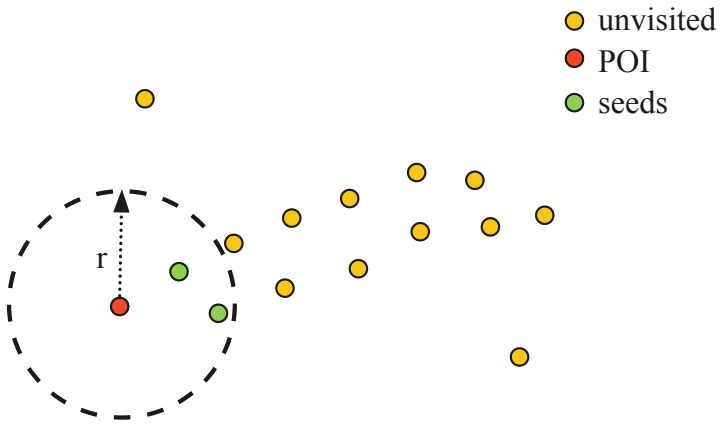


Figure 3.6. The DBSCAN algorithm visits each point in the dataset and perform a range search with input parameter ϵ. It then continues the grow the cluster if the range search returns more seed samples. The algorithm terminates after it has visited all the points.

The selections of the two input parameters are important part of the algorithm design. If ϵ is too small, DBSCAN will result in redundant clusters that carry similar traits. On the other hand, an exaggerated ϵ value means that the algorithm would falsely merge points together and output very few clusters. A heuristic method called the "elbow method" [88] is commonly known in selecting these two parameters through a basic grid search. Hence, a range of ϵ and *minPts* values were supplied for searching the suitable parameters with the feature dataset as illustrated in Figure 3.7 . It could be observed that the "elbow" is located at when ϵ is 2.0 and *minPts* is 120 as the number of clusters stablizes for $\epsilon > 2$.

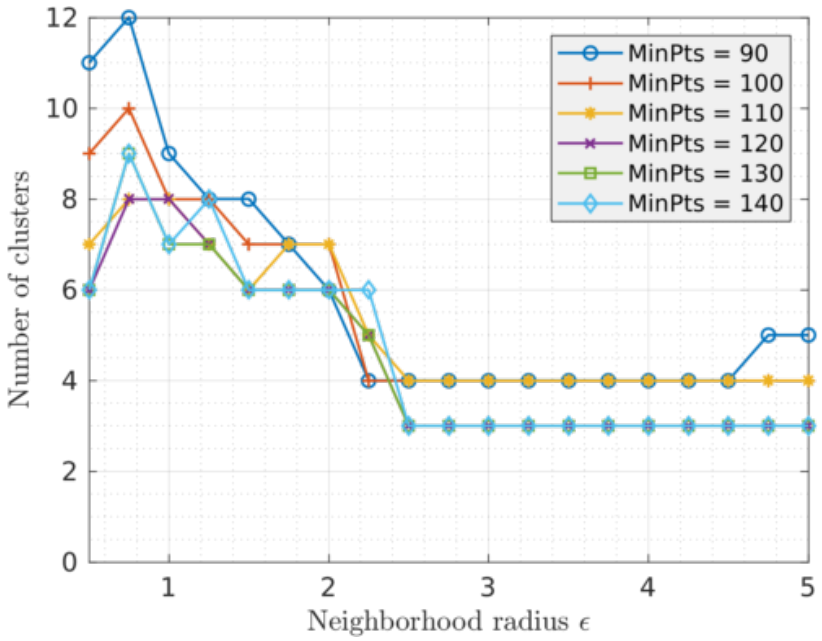


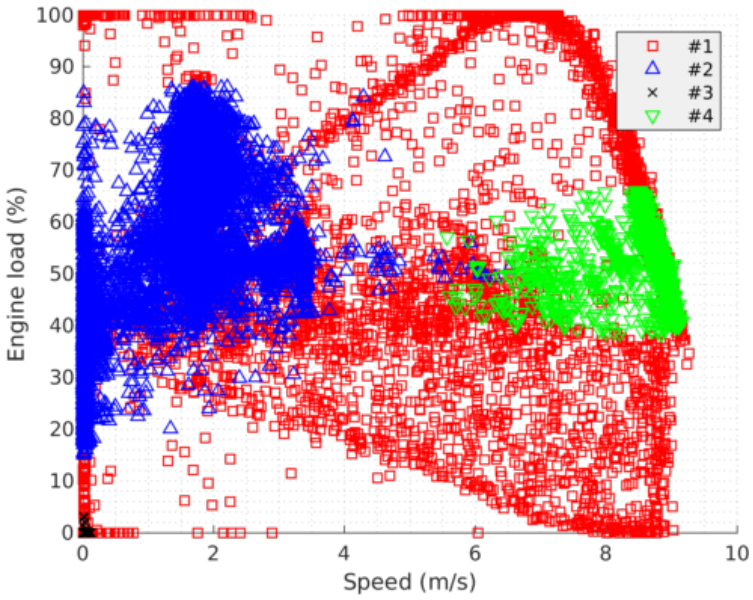
Figure 3.7. A heuristic elbow method was utilized to perform a grid search on a range of input parameters. The resultant "elbow" can observed when ϵ is 2 and *minPts* is 120 since the number of clusters stablizes when $\epsilon > 2$.

3.5 Results

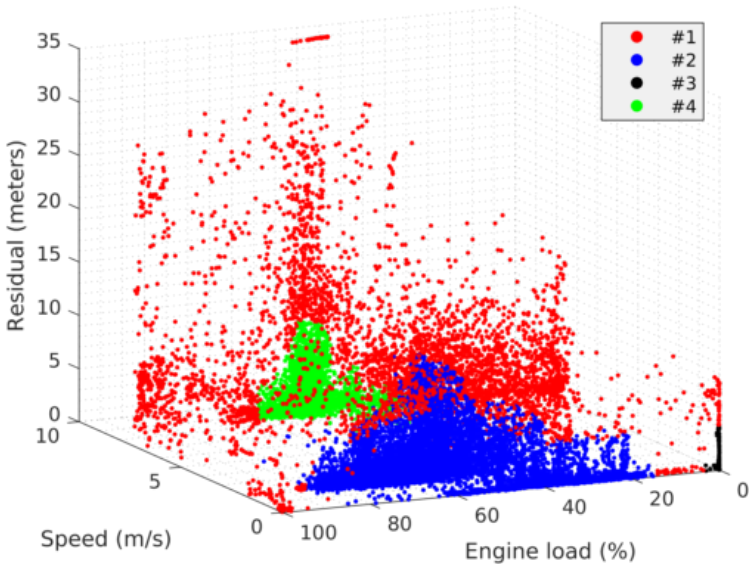
The clustering algorithm was performed on the 3D feature dataset with $\epsilon = 2$ and $minPts = 120$. The clustered results are visualized in both Figure 3.8a and Figure 3.8b. The number of points that corresponds to each cluster is listed in Table 3.1.

Table 3.1. Number of points within each cluster and their percentages.

Cluster Number	Num. of Points	Percentage (%)
1 (Noise)	16558	31.8
2	22485	43.1
3	12920	24.8
4	153	0.3

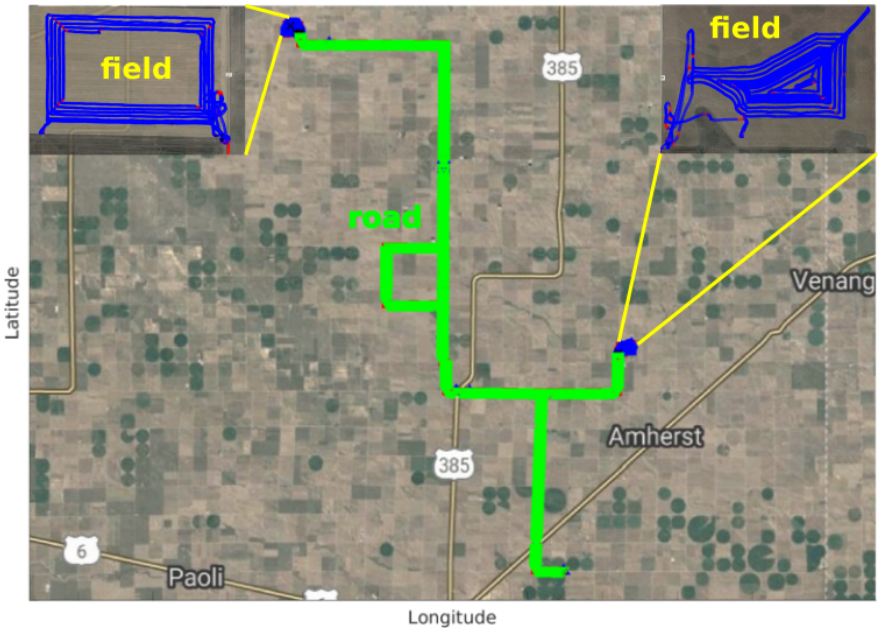


(a) Clustered feature data in 2D.

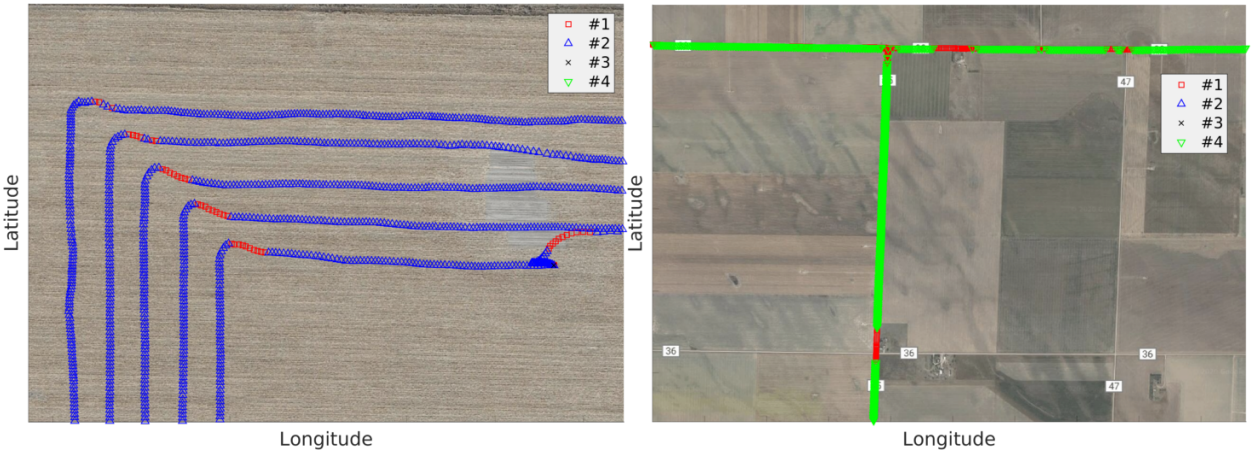


(b) Clustered feature data in 3D.

Figure 3.8. The DBSCAN algorithm clustered the feature dataset into four clusters. Each cluster represents a different combine operation patterns. Cluster #2 and #4 indicate in-field and on-road combine motions while cluster #3 represents idling of the combine. In addition, the noise cluster, cluster #1 encapsulates combine maneuvers. These speculations can be further confirmed by geospatial contexts in Figure 3.9.



(a) In-field and on-road samples are delineated from the clustering.



(b) Noise samples in in-field turns.

(c) Noise samples in road maneuvers.

Figure 3.9. The combine GPS track data provides contextual information for intepreting the cluster results in Figure 3.8. Plot **(a)** shows a clear distinction between in-field and on-road samples from cluster #2 and #4. Moreover, plots **(b)** and **(c)** show that the presence of noise samples mostly occur within in-field and on-road maneuvers.

Several speculations could be made from observing the clusters. First, cluster #2 is characterized by moderate engine load with substantial variance and relative moderate speed range (0 to 4 m/s). This cluster is likely to correspond to when combine was cutting wheat. On the other hand, samples cluster #3 have engine load and speed samples that are close to 0; this might indicate the combine was idling. Furthermore, cluster #4 is represented

by high engine load with little variance and high speed samples (6 to 9 m/s). It is natural to assume that these samples belong to when the combine was on the road. Surprisingly, the algorithm were also able to pick out noisy samples in cluster #1 that do not correspond to any dense clusters. These samples might be related to combine maneuvers that could potentially cause drastic changes in speed and engine load.

The GPS track data provides a contextual information to both intepret the cluster meanings and confirm earlier speculations. The series of plots in Figure 3.9 provides visualizations of an overall view of the GPS data colorcoded by the clusters as well as detailed plots of in-field and on-road tracks. It is evident that the algorithm was able to delineate in-field and on-road samples from as shown in Figure 3.9a. Furthermore, the meanings of noise samples in cluster #1 can also be intepreted. Figure 3.9b shows the noise samples are interlaced with cluster #2 samples (harvesting) during turning. This is likely due to high Kalman filter residual during turns. In addition, on-road motion discontinuities are visualized in Figure 3.9c. The noise samples in the case are interlaced with cluster #4 samples (road travels) since the combine needs to stop at various road intersections and residential areas to ensure road safety and avoid potential obstacles.

3.6 Conclusions

In this work, a feature dataset was formed by using Kalman filter residual, speed, and engine load data generated from the collected CAN and GPS data from a combine during wheat harvest. This dataset was clustered using DBSCAN for classifying combine operation modes. The resultant clusters successfully distinguished differences in combine operational modes for in-field harvesting, on-road travels, and maneuvers for both in-field and on-road scenarios. Nevertheless, this work could be improved in a few areas. First, the accuracy of the algorithm should be evaluated by using ground truth data which could be potentially obtained from CAN data. Secondly, the selection of input parameters should be more rigorously determined. Thirdly, the methodology for formulating a proper data cloud should also be more rigorously defined. Methods such as Principle Component Analysis (PCA) should be considered for determining critical data components from CAN and GPS streams.

Lastly, instead of relying on single model Kalman filtering and using residual as a parameter, a more adaptive multiple model Kalman filtering algorithm needs to be utilized for capturing different motion models and generating more useful features.

4. COMBINE HARVESTER UNLOADING EVENT IDENTIFICATION USING GPS DATA

Acknowledgement: this chapter is a revised version of the paper [25] published at the 2019 Annual ASABE International Meeting.

4.1 Introduction

For typical wheat harvesting scenarios in modern agriculture, unloading events, either initiated by the combine harvester or the grain cart, serve as an important gateway to acquire insights on field efficiency and overall field logistics, if the events are properly identified. When available, they enable the possibility for product tracing and calibrations of yield monitors. Ideally, researchers could collect processed data from the yield monitor or raw data from the ISOBUS diagnostic port to extract auger status or auger flow rate for unloading events identification. However, interpreting messages from these sources are extremely challenging due to the proprietary nature of these messages. In this work, Global Position System (GPS) data were utilized instead for automatic identification of in-field unloading events from combine harvesters to tractor-hauled grain carts. A workflow that comprised Interacting Multiple Models (IMM) filtering and a rule-based algorithm to extract unloading events from GPS tracks was introduced and discussed. To evaluate the workflow performance, unloading events from 16 wheat harvest sessions were identified. The total time duration of the identified events in each session was multiplied by a fixed auger unloading rate to obtain an estimated weight of harvested wheat. These estimates were compared with actual weight ticket records. It was shown that the estimated weights from the identified unloading events achieved an overall accuracy over 90%.

4.2 Related Works

The utilizations of Global Positioning System (GPS) technologies are ubiquitous in the present era of precision agriculture. Agricultural applications like advanced phenotyping [89], [90], remote sensing [91], and agricultural machinery guidance and control systems rely

on either low-cost or high-precision Real-Time Kinematic (RTK) GPS location data for data collection and analytics purposes. For agricultural machinery, GPS data are particularly important as they are essential for making possible precision farming operations and technologies like precision seeding, Variable Rate Application, yield mapping, and autosteering. Moreover, with GPS-equipped machines, farmers could navigate through low-visibility field conditions such as rain, fog, and darkness with more confidence. These precision operations not only minimize potential human-induced error such as overlaps or gaps in operations like agrochemical application but also boosts the efficiency of the whole farming cycle of planting, application, and harvesting. Apart from GPS data combined with other sensor measurements like mass flow rate, soil moisture, etc., data from GPS receivers themselves are rich data streams that contain not only geospatial coordinates but also carry information like timestamps, speed, heading, positional error, etc. Hence, by collecting and post-processing GPS data sets, one could reveal knowledge and discover patterns in machine operations, inter-machine interactions, and operator behaviors in order to gain knowledge for high-level farm management. For instance, machinery management information from GPS data were extracted in [92] to evaluate the potential for improving harvest efficiency [92]. In addition, simulated GPS coordinates from their proposed path planning algorithm were generated and compared with the recorded GPS readings for avoiding in-field obstacles in [93]. Moreover, novel approaches for multi-machine activity recognition and product traceability generation using GPS tracks only were discussed in and [79] and [94]. Furthermore, a multiple-model filtering algorithm for classification of in-field combine harvester operation modes based on collected GPS tracks were explored in [95]. Nevertheless, no prior works have been conducted to investigate the inference or the idenfication of unloading events between combine harvesters and tractor-hauled grain carts.

4.3 Preliminaries

For typical harvesting sessions, unloading harvested crops usually involves interactions between combines, grain carts, and grain trucks. The scope of this work is limited to in-field unloading scenarios between combines and grain carts. Typical unloading scenarios between

combines and grain carts are explained in Section 4.3.1. Moreover, GPS data acquisition were conducted in two combines and one tractor during wheat harvest season of 2018 for supplying the GPS data. The GPS data acquisition method and preprocessing steps are discussed Section 4.3.2.

4.3.1 Typical Unloading Scenarios

There are two typical unloading modes between combines and tractor-hauled grain carts in a wheat harvest scenario:

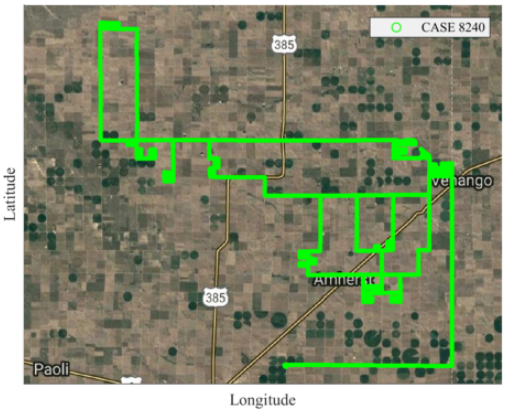
1. **Unloading On-the-Go:** it is a common practice nowadays for combines to unload grain on-the-go as it minimizes the combine idle times during harvesting sessions [96]. When a combine operator is ready to unload the grain, he or she swings out the auger spout to signal a nearby grain cart operator. While the combine is still moving, the grain cart operator pulls the grain cart next to the combine and makes sure the cart stays underneath the auger spout by maintaining a constant speed. Once the combine operator believes both machines are in alignment of speeds and auger spout is above the grain cart, the combine starts unloading. When the combine finishes unloading, the operator pulls the auger back. Meanwhile, the grain cart operator drives the grain cart away from the combine for transfering the crop unto a grain truck.

2. **Unloading When Parked:** This scenario is less common but still takes place during harvesting sessions. A combine parks at some location in the field with the auger swung out and the grain cart parks right next to the combine and aligns the cart underneath the auger. The combine proceeds to unload the grain to the cart. Once the unloading finishes, both the combine and the grain cart either drive off or remain parked at the same location.

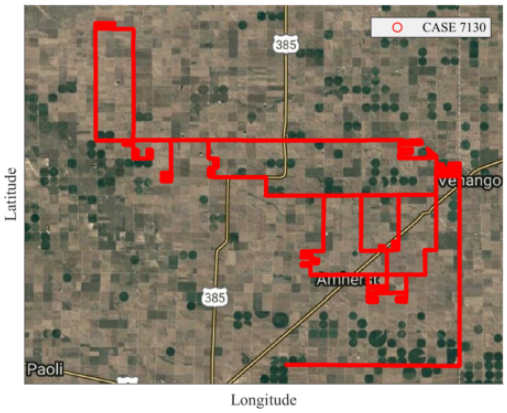
For simplifcation purposes, only the first scenario was considered as it is more commonly adapted for unloading crops between combines and grain carts.

4.3.2 GPS Data Collection and Preprocessing

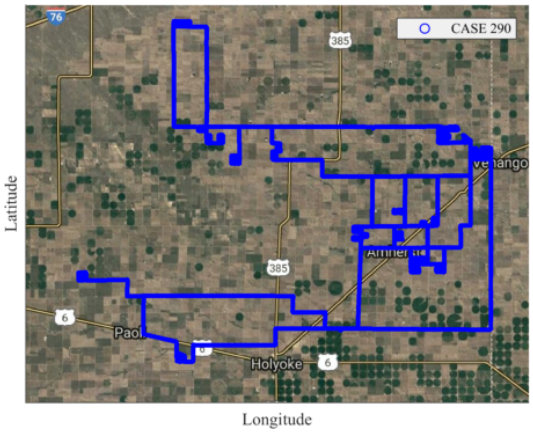
A custom Android application [97] running on Nexus 7 tablets was utilized to collect GPS data during 2018 wheat harvest in Amherst, Colorado, USA. The tablets were deployed in the cabs of three machines of interest which worked jointly in the same harvesting sessions. These machines were 1) a CASE 8240 combine [98], 2) a CASE 7130 [99] combine, and 3) a CASE 290 [100] tractor pulling a Crustbuster 1075 [101] grain cart. The collected GPS data had a frequency of 1 Hz and an accuracy of 5 meters. They were stored as Comma-Separated Values (CSV) files. Each file had four columns: *gpsTime*, *latitude*, *longitude*, and *heading*. The complete GPS track data for each machine are visualized in Figure 4.1.



(a) GPS data from the 8240 combine.



(b) GPS data from the 7130 combine.



(c) GPS data from the 290 tractor.

Figure 4.1. Data collection was conducted from two combines and one tractor working in the same fields. The collected GPS track data shows where the machines have been. The tractor GPS data had additional tracks since it helped other fleets for crop transfer in other fields.

Two preprocessing steps were performed using MATLAB [102]. First, *latitude* and *longitude* were converted to Universal Transverse Mercator (UTM) coordinates (i.e., x and y). It is noteworthy that the recorded GPS coordinates reflected the tablet locations. In other words, post-processing is needed to estimate the locations of the auger spout center as well as the grain cart center for better identifications of unloading events. This post-processing step is discussed in Section 4.4.2.

The second step involved extracting in-field GPS coordinates. Only in-field GPS data were considered since it was assumed that unloading events could only occur within field boundaries. To extract the in-field GPS data, field boundaries were first generated using the method described in [79]. A total of 16 field boundaries were produced. These field boundaries were subsequently utilized to extract the in-field coordinates. The extracted in-field GPS data for each machine were stored as three individual MAT files. Each MAT file consisted 16 data structs which corresponded to preprocessed data in 16 fields. The field definitions for each data struct are given in Table 4.1.

Table 4.1. Field definitions for preprocessed GPS data of each machine from each field.

Field Name	Description (units)
gpsTime	GPS epoch timestamp (ms)
x	Easting (m)
y	Northing (m)
heading	Truth north heading (degree)

4.4 Workflow

The preprocessed combine and grain cart GPS data go through the workflow shown in Figure 4.2 for unloading event identifications. The workflow consists of four building blocks:

1. **IMM filtering:** this step runs IMM filtering on the positions (easting and northing) of preprocessed GPS data and generates two parameters for each time step: 1) a model

probability that tells how likely a machine follows a constant speed motion and 2) a speed estimate that specifies how fast a machine travels.

2. **Grain cart center estimation:** preprocessed cart GPS data are further processed to estimate the grain cart center locations at each time step with pre-measured offset parameters.

3. **Auger spout center estimation:** similar to grain cart center estimation, auger spout center locations at each time step are computed based on preprocessed combine GPS data.

4. **Rule-based unloading event identification:** this step takes in the parameters computed from last three steps: model probabilities, speed estimates, grain cart center locations, and auger spout center locations. It uses threshold rules on differences of the computed parameters for identification of unloading events. The identified unloading event specifies a collection of time steps an unloading event might occur from a combine's perspective.

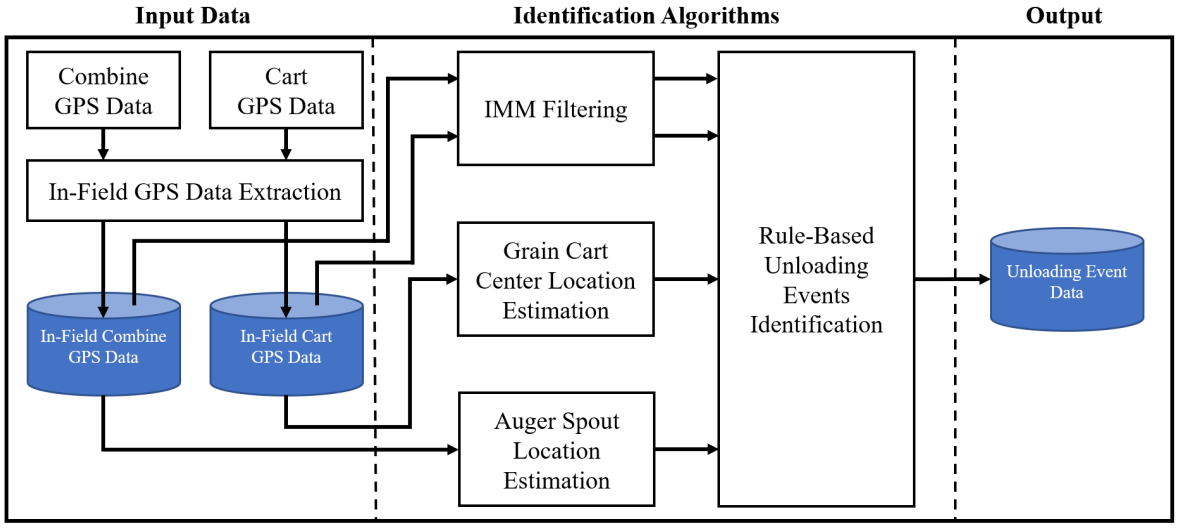


Figure 4.2. The proposed workflow consists of four components: 1) IMM filtering, 2) auger center estimations, 3) grain cart center estimations, and 4) a rule-based algorithm for unloading event identification.

4.4.1 Interacting Multiple Models (IMM) Filtering

The IMM algorithm is a Kalman-filter based algorithm for estimating the state of a maneuverable target [103], [104]. The main goal is to estimate the state of a linear system with Markovian switching coefficients. Such a linear system is assumed to be dynamic and the dynamics are described by multiple models, which are assumed to be correct. This type of linear system can be expressed as:

$$X(k+1) = F_{\theta(k)} \cdot X(k) + v(k), \tag{4.1}$$

$$Z(k) = H_{\theta(k)} \cdot X(k) + w(k), \tag{4.2}$$

where $X(k)$ is the system state obtained from the process matrix $F_{(\cdot)}$, $Y(k)$ is the system state predicted from the measurement matrix $H_{(\cdot)}$, and $\theta(k)$ is a finite state Markov chain that takes values in $\{1, \cdots, N\}$ with the model transition probability from model i to state j being p_{ij}. $v(k)$ and $w(k)$ are assumed to be white Gaussian process and measurement noise. As illustrated in a two-model case in Figure 4.3, the algorithm consists of a filter, typically a Kalman filter or its extensions, for each model, an estimate mixer at the input of the filters, a model probability updater, and a final estimate combiner at the output of the filters. The main benefit of this multiple-model approach, as investigated in [95], is that the algorithm could soft-switch among the most-likely state space models instead of relying on one single state-space model.

In addition, $\hat{X}^{\mathrm{j}}(k|k)$ is the state estimate from model j, $\mu(k)$ is the vector of updated model probabilities, and $\hat{X}(k|k)$ is the combined state estimate at time k. With the assumption that the model switching is governed by a Markov chain, the input estimate mixer uses the model probabilities from the previous time step $\mu(k-1|k-1)$ and the model transition probabilities p_{ij} to compute a mixed estimate for each filter. Each filter then uses a mixed estimate and an actual measurement $z(k)$ to compute a new estimate and a likelihood value $\Lambda^{\mathrm{j}}(k)$ according to each filter's model. The likelihood values and the model transition probabilities are then combined to output the new model probabilities. Finally, the combined state estimate is computed with the new state estimates from each filter as well as their updated

model probabilities. The goal of utilizing this algorithm is to not only estimate the target's states but also use the algorithm-generated model probabilities to obtain information on the target's motions.

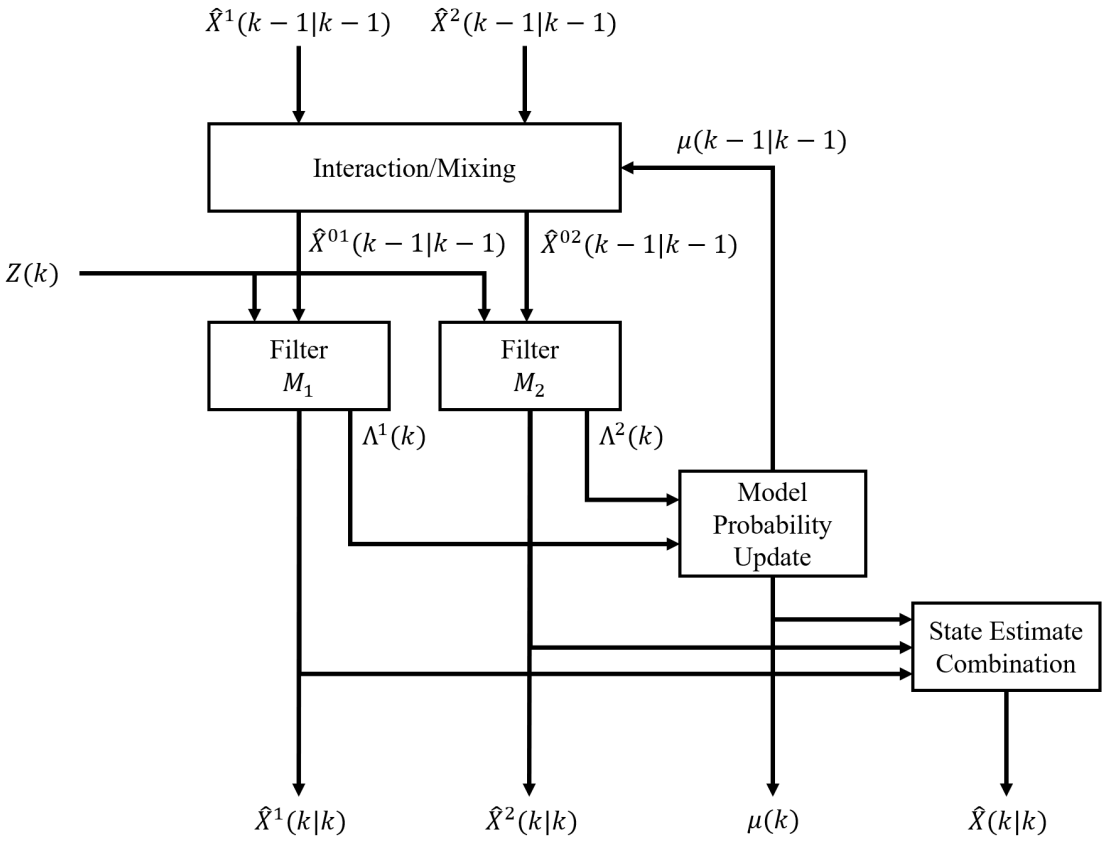


Figure 4.3. The IMM filtering algorithm has three main steps that cycle through the algorithm: interaction/mixing, model probability update, and combined state estimation. The goal for utilizing this algorithm in the scope of this paper is to generate model probabilities and speed estimates as two of the input parameters for the rule-based algorithm.

Model Selection

To obtain the best possible filter outputs, the IMM algorithm has to be properly designed to model machine motions correctly. In a wheat harvest scenario, a combine harvester, either driven by a human operator or autosteering technology, tends to follow a uniform motion with a relatively constant speed. In addition, the heading of the combine harvester doesn't experience drastic change either. During unloading, this uniform motion still applies; the

machine intends to stay at a uniform speed for unloading to the cart. Therefore, the uniform motions of machines need to be modeled in the IMM algorithm. On the other hand, the non-uniform machine motions such as sudden stops and turning likely implies that the machines were not unloading. A model for this type of motions also needs to be integrated so that the IMM algorithm can output model probabilities that tell which machine motion is more likely at each time step.

A suitable model for modeling uniform machine motions is the Nearly Constant Velocity (NCV) model as discussed in [105]. Its state space representation is listed as follows:

$$X(k+1) = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot X(k) + v(k), \tag{4.3}$$

$$Z(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot X(k) + w(k) \tag{4.4}$$

where $X(k)$ is the state vector at time step k defined as $\begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^T$ and x, y, $\dot{x}$, $\dot{y}$ are easting, northing, and speed estimates (in meters per second) in easting and northing directions. $Y(k)$ is the predicted measurement vector. Moreover, $w(k)$ is the measurement noise covariance matrix R_{NCV} and $v(k)$ is the process noise covariance matrix Q_{NCV} given by:

$$Q_{\text{NCV}}(k) = \begin{bmatrix} \frac{1}{3}\Delta t^3 & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{3}\Delta t^3 & 0 & \frac{1}{2}\Delta t^2 \\ \frac{1}{2}\Delta t^2 & 0 & \Delta t & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 & \Delta t \end{bmatrix} \cdot q_{\text{NCV}}. \tag{4.5}$$

Note that the state transition matrix F and the measurement matrix H has different dimensions (4×4 and 2×4). This is because at each time step k, the measurement $Z(k)$, i.e., the GPS data, only contains positions in x and y directions. This data vector

is compared with the computed $Z(k)$ at each time step to help the algorithm determine whether to trust the supplied data $z(k)$ more or the measured state $Z(k)$. Since this model is linear, a traditional Kalman filter was utilized in the IMM algorithm.

For modeling non-uniform machine motions, a "trap" model called the Nearly Coordinated Turn (NCT) model is employed. The intention is to capture motions that are out of ordinary (i.e., headland turning, sudden speed changes, etc.). According to [105], this model is given by:

$$X(k+1) = \begin{bmatrix} 1 & 0 & \frac{\sin(\omega(k)\Delta t)}{\omega(k)} & \frac{\cos(\omega(k)\Delta t)-1}{\omega(k)} & 0 \\ 0 & 1 & \frac{1-\cos(\omega(k)\Delta t)}{\omega(k)} & \frac{\sin(\omega(k)\Delta t)}{\omega(k)} & 0 \\ 0 & 0 & \cos(\omega(k)\Delta t) & -\sin(\omega(k)\Delta t) & 0 \\ 0 & 0 & \sin(\omega(k)\Delta t) & -\cos(\omega(k)\Delta t) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot X(k) + v(k), \tag{4.6}$$

$$Z(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \cdot X(k) + w(k) \tag{4.7}$$

Similar in Equation (4.3), $X(k)$ is the state vector defined as $\begin{bmatrix} x & y & \dot{x} & \dot{y} & \omega \end{bmatrix}^T$. The state vector is the same as the one in the NCV model except for ω—the target turning rate in radian per second. The dimension mismatch between the state vectors in NCV and NCT models is solved by augmenting the earlier state vector with a component that is zero. This simply amounts to a zero turn rate in the NCV motion. Similarly, the covariance matrix in Equation (4.5) is augmented by a column and a row of zeros. Similar to the NCV model,

NCT model's measurement noise is defined by $w(k)$ and its noise covariance matrix R_{NCT}. Moreover, $v(k)$ is the process noise defined by the covariance matrix Q_{NCT} given by:

$$Q_{\text{NCT}}(k) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \Delta t \end{bmatrix} \cdot q_{\text{NCT}}. \tag{4.8}$$

Note that the NCT model in Equation (4.6) is a nonlinear model. The state estimates are predicted using an Extended Kalman filter (EKF) within the IMM algorithm. The usage of the EKF algorithm requires linearization of the original NCT model representations. As derived in [105], the linearized NCT model is expressed as:

$$X(k+1) = \begin{bmatrix} 1 & 0 & \frac{\sin(\omega(k)\Delta t)}{\omega(k)} & \frac{\cos(\omega(k)\Delta t)-1}{\omega(k)} & X_1 \\ 0 & 1 & \frac{1-\cos(\omega(k)\Delta t)}{\omega(k)} & \frac{\sin(\omega(k)\Delta t)}{\omega(k)} & Y_1 \\ 0 & 0 & \cos(\omega(k)\Delta t) & -\sin(\omega(k)\Delta t) & X_2 \\ 0 & 0 & \sin(\omega(k)\Delta t) & -\cos(\omega(k)\Delta t) & Y_2 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot X(k) + v(k), \tag{4.9}$$

$$Z(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \cdot X(k) + w(k) \tag{4.10}$$

where X_1, Y_1, X_2, Y_2 are defined as:

$$X_1 = \frac{\omega(k)\Delta t\cos(\omega(k)\Delta t) - \sin(\omega(k)\Delta t)}{\omega(k)^2}\dot{x}(k) - \frac{\omega(k)\Delta t\sin(\omega(k)\Delta t) - \cos(\omega(k)\Delta t)}{\omega(k)^2}\dot{y}(k),$$

$$Y_1 = \frac{\omega(k)\Delta t\sin(\omega(k)\Delta t) + \cos(\omega(k)\Delta t)}{\omega(k)^2}\dot{x}(k) - \frac{\omega(k)\Delta t\cos(\omega(k)\Delta t) - \sin(\omega(k)\Delta t)}{\omega(k)^2}\dot{y}(k),$$

$$X_2 = -\Delta t\sin\omega(k)\Delta t\cdot\dot{x}(k) - \Delta t\cos\omega(k)\Delta t\cdot\dot{y}(k),$$

$$Y_2 = -\Delta t\cos\omega(k)\Delta t\cdot\dot{x}(k) - \Delta t\sin\omega(k)\Delta t\cdot\dot{y}(k).$$

Input Parameter and Noise Level Determination

With the established state space models for modeling machine motions, a set of input parameters need to be selected for the algorithm. The initial model probabilities $\mu(0)$ and p_{ij} were initialized as the following:

$$\mu(0) = \begin{bmatrix} 0.90 & 0.10 \end{bmatrix}, \tag{4.11}$$

$$p_{\text{ij}} = \begin{bmatrix} 0.90 & 0.10 \\ 0.10 & 0.90 \end{bmatrix}. \tag{4.12}$$

Moreover, the initial state estimates for both models were initialized as the first data point from each field's data. The initial noise covariances were selected heuristically as the IMM algorithm is not sensitive to the choices of either parameters. They can be expressed as:

$$\hat{X}_{\text{NCV}}(0) = \begin{bmatrix} x(0) & y(0) & 0 & 0 \end{bmatrix}^T, \tag{4.13}$$

$$\hat{P}_{\text{NCV}}(0) = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}, \tag{4.14}$$

$$\hat{X}_{\text{NCT}}(0) = \begin{bmatrix} x(0) & y(0) & 0 & 0 & 0 \end{bmatrix}^T, \tag{4.15}$$

$$\hat{P}_{\text{NCT}}(0) = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0.1 \end{bmatrix}. \tag{4.16}$$

Furthermore, the selection of noise levels for each model is an essential yet important part of the estimator design. In this work, it is assumed that both the process and measurement noise under the NCV model is small relative to those of the NCT model. The intention

for this design choice is to allow the IMM algorithm to be sensitive to changes in machine motions for delineating any non-uniform motions from relatively smooth motions. For this reason, the noise levels for the two models were initialized as follows:

$$q_{\text{NCV}}(k) = 0.01, \hat{R}_{\text{NCV}}(0) = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.15 \end{bmatrix}, \tag{4.17}$$

$$q_{\text{NCT}}(k) = 0.15, \hat{R}_{\text{NCT}}(0) = \begin{bmatrix} 0.15 & 0 \\ 0 & 0.45 \end{bmatrix}. \tag{4.18}$$

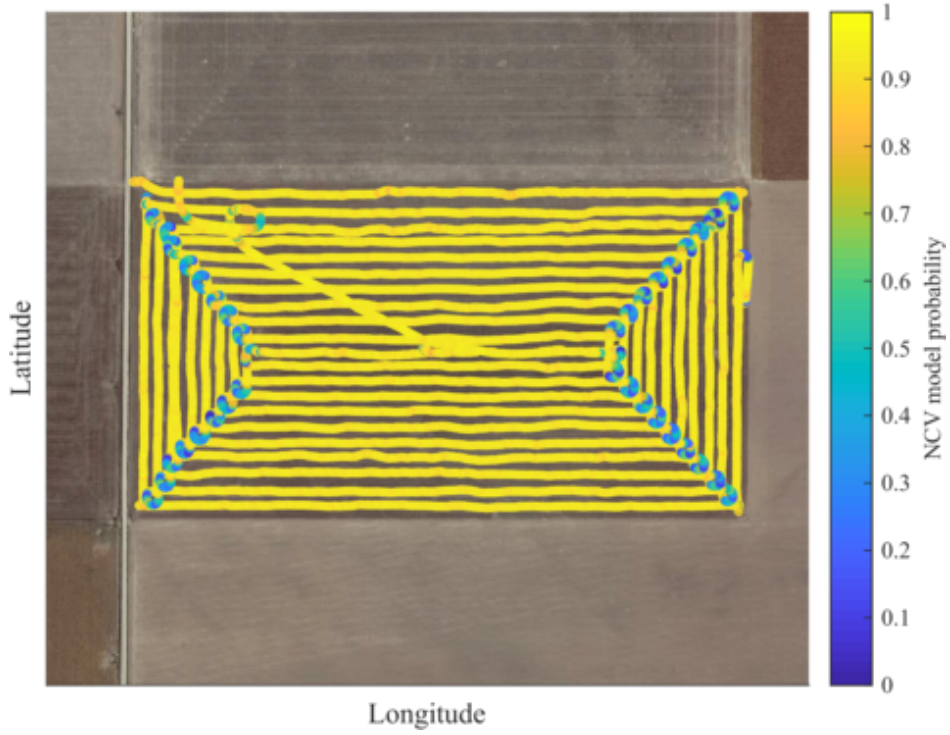
IMM Filter Outputs

The IMM algorithm was run with the design parameters by iterating through each machine's data structs. The input data for each run of the algorithm were the coordinates in easting and northing direction. The NCV model probability and the speed estimate in x and y directions at each time step were saved. The overall speed at each time step were computed as $\sqrt{\dot{x}^2 + \dot{y}^2}$. Both the NCV model probabilities and overall speed estimates were saved for later workflow steps.

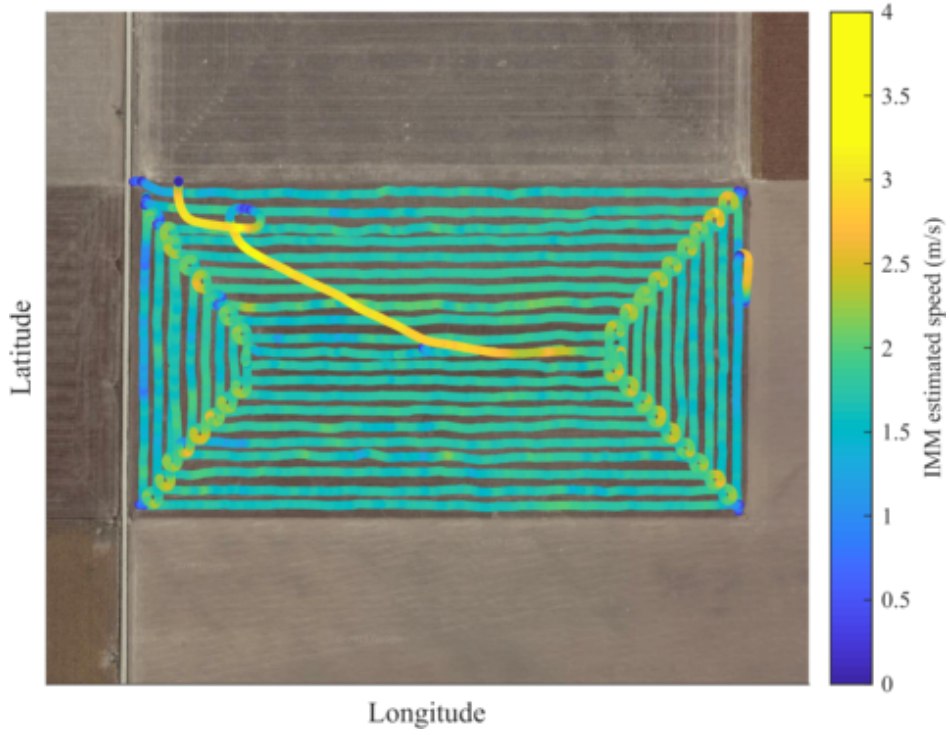
An example IMM output of the NCV model probability and estimate speed values of the 7130 combine from one field are visualized geospatially in Figure 4.4a and Figure 4.4b. Several observations can be drawn:

- The combine followed a loop-style pattern that followed the coutour of the field for harvesting.
- Due to the harvest pattern, the NCV model probability values remain close to the value 1 when the combine was harvesting wheat.
- The NCV model probability values drop to lower values at corners of the field since the IMM algorithm "believes" that it is no longer likely for the target to follow the NCV model.

- Similar patterns can be observed for speed estimates. The estimated speed remain constant when the combine was harvesting; however, the estimated speed rose to higher speed during turning.



(a) NCV model probability map.



(b) Estimated speed map.

Figure 4.4. Both NCV model probability and speed estimates from the IMM algorithm are visualized geospatially on a map using one field of 7130 combine's GPS data.

4.4.2 Position Estimation

As mentioned in Section 4.3.2, the GPS data for both combines and the cart need to be corrected to properly reflect the positions of auger spout center and grain cart center locations so that the unloading events could be better identified. The next two sections describe the method for estimating the auger spout and grain cart center locations from the current combine and tractor GPS data.

Auger Spout Center

There are two assumptions in estimating the auger spout center position. First, for simplifying the estimation process, it is assumed that the auger is extended at all times. Moreover, the second assumption is that the tablet location aligns horizontally with the auger spout center. Figure 4.5 gives a visual representation of these assumptions as well as the auger spout location in relative to the tablet location. In this figure, $x_{\text{offset}}^{(c)}$ represents the distance from the tablet to the auger spout center in x direction for a particular combine model number c. This values were measured for each combine and are listed in Table 4.2.

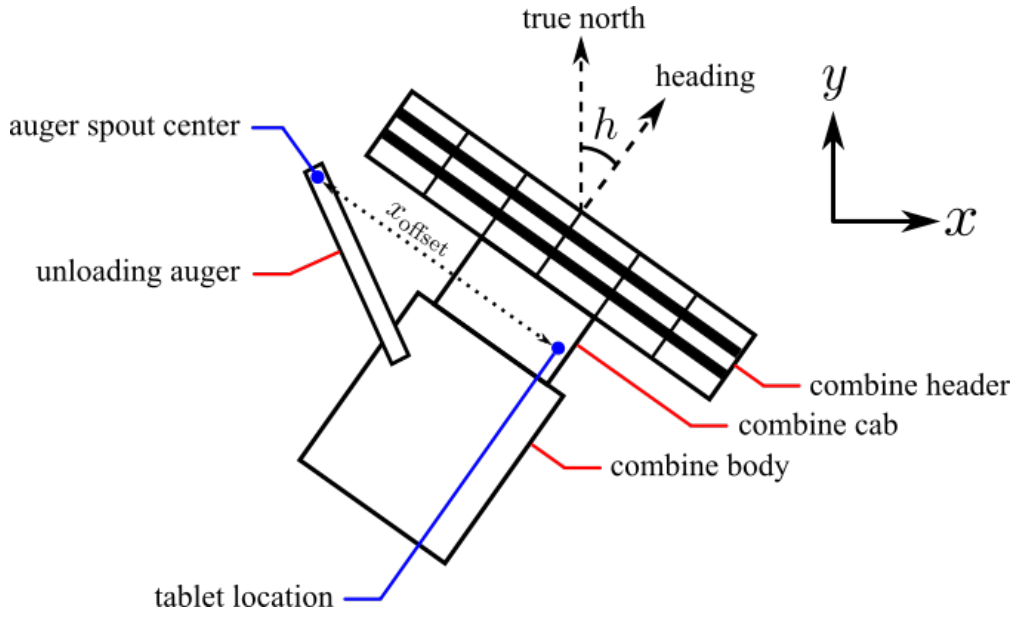


Figure 4.5. The tablet location has a fixed offset to the auger spout center. This offset is measured and given in Table 4.2 for each combine model.

Table 4.2. Measured distances in x between the tablet and the auger spout center for each combine model.

Model	$x^{(c)}_{\textbf{offset}}$ **(meters)**
7130	8.84
8240	9.75

The auger spout centers at each time step k were estimated by adding the tablet location by offsets in both x and y directions. This can be expressed as:

$$\begin{bmatrix} x_{\text{ac}}(k) \\ y_{\text{ac}}(k) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \end{bmatrix} + R\left[h(k)\right] \cdot \begin{bmatrix} x^{(c)}_{\text{offset}} \\ 0 \end{bmatrix}, \tag{4.19}$$

where $x^{(c)}_{\text{ac}}(k)$ and $y^{(c)}_{\text{ac}}(k)$ are the estimated auger spout centers at time step k for a combine model c, $x(k)$ and $y(k)$ is the tablet's location, and $h(k)$ is the heading angle. $R[\,\cdot\,]$ is a clockwise rotation matrix that rotates the offset value by the heading angle. This rotation matrix can be expressed as:

$$R\left[h(k)\right] = \begin{bmatrix} \cos\left[h(k)\right] & \sin\left[h(k)\right] \\ -\sin\left[h(k)\right] & \cos\left[h(k)\right] \end{bmatrix} \tag{4.20}$$

Using Equation (4.19), the auger spout centers for each combine could be computed for all time steps. These positions were appended as two new fields, $x^{(c)}_{\text{ac}}$ and $y^{(c)}_{\text{ac}}$, into each combine's GPS data struct.

Grain Cart Center

The estimation of grain cart center locations is different from the auger spout location estimation because the tractor-cart motions are complicated by the trailer motions. Figure 4.6 shows an visualization of the tablet position in regard to the grain cart center location with

offsets in both x and y direction. The offset values are assumed to be fixed and they are measured as the following:

$$x_{\text{offset}} = 0.91 \text{ meters}, y_{\text{offset}} = 7.92 \text{ meters}. \tag{4.21}$$

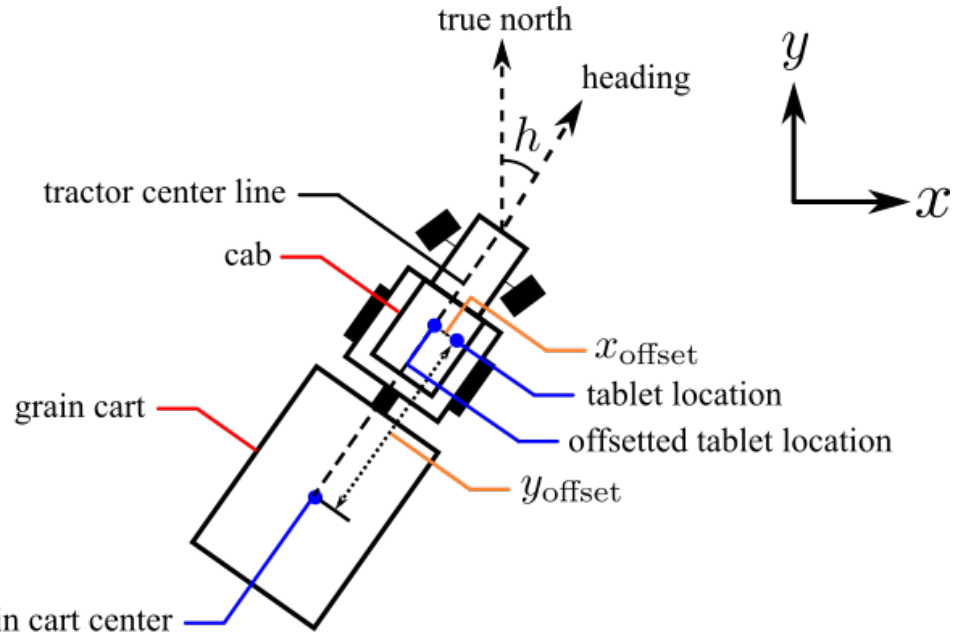


Figure 4.6. The procedure for finding the grain cart center is slightly different than finding the auger spout center. The offsetted tablet location is first estimated. Afterwards, this position is used to search for the most feasible grain cart center location.

There are two steps in estimating the grain cart center given the tablet's location. First, the offsetted tablet location at each time step k is found using these equations:

$$x_{\text{ot}}(k) = x(k) + x_{\text{offset}} \cdot \cos(180^\circ - h(k)), \tag{4.22}$$

$$y_{\text{ot}}(k) = y(k) + y_{\text{offset}} \cdot \sin(180^\circ - h(k)), \tag{4.23}$$

where $h(k)$ is the heading of the tractor at time step k.

Given the offsetted tablet location, the second step involves a range search method demonstrated in Figure 4.7. The method first retrieves a collection of GPS data points P that fall

within the circle with a radius of $1.25 \cdot y_{\text{offset}}$. Subsequently, the method estimates the current grain cart center location by finding an intersection between the connected line segments using collection P and a circle with a radius of y_{offset}. There are three assumptions for this method to work:

- Any connected line segments created using collection P must be a straight line.
- The estimated grain cart center locations must fall on the line segments created from point collection.
- The intersection point must be from the past given a current tablet location.

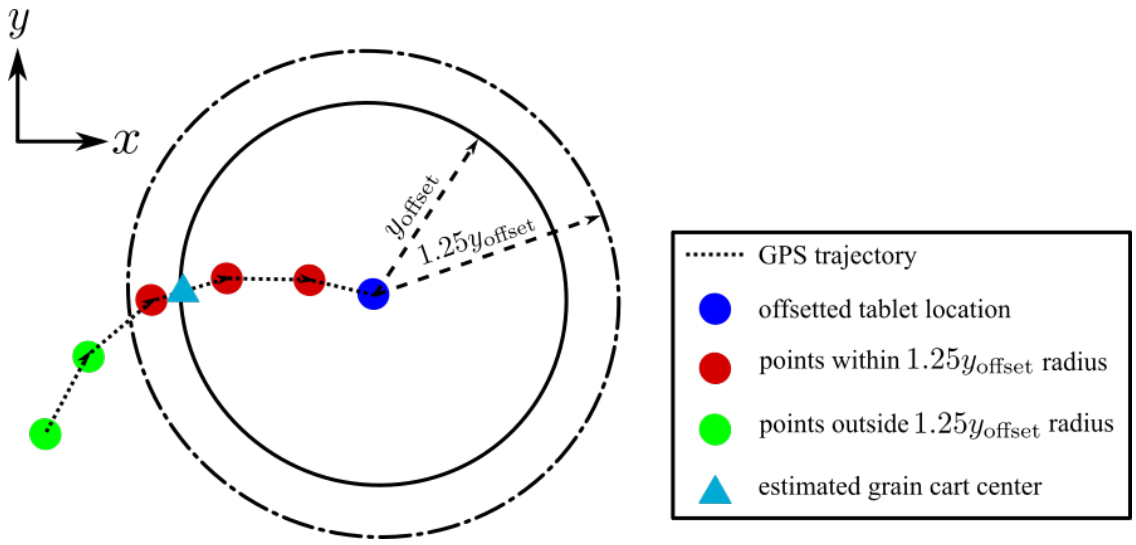


Figure 4.7. This provides a toy example that estimates the grain cart center location from candidate points.

In case of multiple intersections, the method selects the intersected point that is the nearest in time. The selected point is the estimated grain cart center location. This method was applied for the tractor GPS data to estimate grain cart centers. All grain cart center locations were stored as two new fields x_{gc} and y_{gc} in the tractor data struct.

4.4.3 Rule-Based Unloading Events Extraction

After the previous two steps, updated data struct for the combine and tractor data were prepared. Table 4.3 and Table 4.4 lists the updated data struct for each type of machine.

In this section, a rule-based algorithm is introduced for identifying combine unloading events using the updated data. The pseudocode is listed in Algorithm 1. The goal of this algorithm is to obtain the data vector U that has the unloading event labels (0 for not unloading, 1 for unloading) for each time step and the data vector T that contains the total unloading durations for each field. The algorithm is comprised of three main parts:

1. Computation of unloading event parameters.

2. Generation of initial unloading event indices using thresholds and finding consecutive unloading event indices.

3. Filtering of initial unloading event indices.

Algorithm 1 require three input tolerance parameters: ϵ_d, ϵ_μ and ϵ_v. These parameters impose threshold rules for checking whether a particular point should be considered as unloading. The rest of the algorithm is centered around these three parameters by computing necessary data from loaded GPS data structs and labeling identified events. After loading the necessary data from line 1 to 2, the algorithm starts with a loop that iterates through the data for unloading event identification. The unloading event parameters are first computed using the routine *computeUleParameters* in line 6. The pseudocode along with the input/output definitions for this routine are provided in Algorithm 2.

The initial unloading events indices are found by using a set of constraints specified by the algorithm inputs in line 9 and 10. These indices are then used to label parts of vector U to 1 (unloading). Afterwards, a subroutine called *findConsecutiveSubSeq* is written to find all the start and the end indices of consecutive unloading events. This subroutine's pseudocode is listed in Algorithm 3.

The unloading labels U after Algorithm 3 usually contain false labels which cause potential overestimation of total unloading duration in later steps. Lines from 13 to 20 from Algorithm 1 specify the corrections of false labels caused by initial positions of a combine and a cart. At the beginning a typical harvesting session, a combine and a grain cart sometimes park next to each other. It is possible that the computed unloading parameters associated with this time period fall within the constraints in line 9 of *uleIdentify*. However, no unload-

ing event has occurred yet at this time. To remove these false labels, a simple rule is used: if the median speed associated with these labels is less than a specified threshold (0.25 m/s). The indices of these labels are found and set back to 0 (not unloading) accordingly. This will also set *firstUlFlag* to true because it is no longer necessary to correct future labels after identification of the first false unloading event.

The rule stated in line 23 to 25 correct any false labels when the total unloading duration indicated by the indices falls under or surpasses the lower or the upper bound. The lower bound (15 seconds) was selected heuristically as the time is too short for a combine to unload properly. The upper bound was selected by referencing the typical unloading durations for the two combines in this work. The minimum unloading durations were computed using the manufacturer specifications [98], [99] on combines' grain tank capacities and maximum unloading rates. The specifications and the computed unloading durations are listed in Table 4.5.

Table 4.3. Updated MAT data struct for each combine's data.

Field Name	**Description (units)**
gpsTime	GPS epoch timestamp (ms)
x_{ac}	Auger spout center location in Easting (m)
y_{ac}	Auger spout center location in Northing (m)
v	IMM estimated speed (m/s)
p_{NCV}	IMM estimated NCV model probability
heading	Truth north heading (degree)

Table 4.4. Updated MAT data struct for the tractor data.

Field Name	Description (units)
gpsTime	GPS epoch timestamp (ms)
x_{gc}	Grain cart center location in Easting (m)
y_{gc}	Grain cart center location in Northing (m)
v	IMM estimated speed (m/s)
p_{NCV}	IMM estimated NCV model probability
heading	Truth north heading (degree)

Table 4.5. Bin capacity and maximum unloading auger rate for each combine are listed. The unloading duration column specifies the computed minimum amount of time in seconds each combine need to unload a full bin of grain.

Model Name	Bin Capacity (bu)	Max Unloading Rate (bu/sec)	Unloading Duration (sec)
7130	300	3.2	94
8240	410	4.0	103

Algorithm 1: *uleIdentify* - Combine Unloading Events Identification

Input:
ϵ_d (distance tolerance)
ϵ_μ (NCV model probability difference tolerance)
ϵ_v (speed difference tolerance)

Output:
T (total unloading duration in seconds)
U (unloading labels)

cb $\leftarrow$ **Read** combine GPS data;
gc $\leftarrow$ **Read** grain cart GPS data;
numFields = length(cb);
for $k = 1$ **to** numFields **do**
  firstUlFlag $\leftarrow$ **false**;
  Initialize U with the same length as cb(k) and fill it with 0's;
  `// compute unloading event parameters`
  [D, μ_{diff}, V_{diff}] $\leftarrow$ computeUleParameters(cb(k), gc(k));
  $V \leftarrow$ **Read** combine's speed data from cb(k);
  `// use rules to find initial unloading event candidate indices`
  $M \leftarrow$ **Find** indices where
    $-\epsilon_d \leq D \leq \epsilon_d$ **and** $-\epsilon_\mu \leq \mu_{\text{diff}} \leq \epsilon_\mu$ **and** $-\epsilon_v \leq V_{\text{diff}} \leq \epsilon_v$;
  `// label events with 1's`
  $U(M) = 1$;
  `// find start and end indices for consecutive elements in U`
  $[U_{\text{start}}, U_{\text{end}}] \leftarrow$ findConsecutiveSubSeq(U, 1);
  `// find and label false events with 0's`
  for $l = 1$ **to** length(U_{start}) **do**
    $L \leftarrow$ **Read** indices from $U_{\text{start}}(l)$ to $U_{\text{end}}(l)$;
    if firstUlFlag == false **then**
      if median($V(L)$) < 0.25 meters/second **then**
        $U(L) = 0$;
      else
        firstUlFlag $\leftarrow$ **true**;
      end
    end
    $\tau = U_{\text{end}}(l) - U_{\text{start}}(l) + 1$;
    if firstUlFlag == true **and** ($\tau < 15$ seconds **or** $\tau > 200$ seconds) **then**
      $U(L) = 0$;
    end
    $T(k) \leftarrow$ **Compute** unloading total duration by summing all the elements in U
  end
end
Return T, U

Algorithm 2: *computeUleParameters* - Unloading Events Parameters Calculation

Input:

cb (combine GPS data struct)

gc (grain cart GPS data struct)

Output:

D (distance vector between auger spout locations and grain cart center locations)

μ_{diff} (difference vector for two machines' NCV model probabilities)

V_{diff} (difference vector for two machines' speeds)

`// read in combine and grain cart data`

$[T_{\text{cb}}, P_{\text{cb}}, \mu_{\text{cb}}, V_{\text{cb}}] \leftarrow$ **Read** timestamps, auger spout locations, NCV model probabilities, and speed data from *gc* data struct;

$[T_{\text{gc}}, P_{\text{gc}}, \mu_{\text{gc}}, V_{\text{gc}}] \leftarrow$ **Read** timestamps, grain cart center locations, NCV model probabilities, and speed data from *gc* data struct;

`// compute parameters`

for $k = 1$ **to** length(T_{cb}) **do**

  `// find match timestamp between two machines`

  $M \leftarrow$ **Find** indices where $T_{\text{gc}} == T_{\text{cb}}(k)$;

  `// compute results based on found index and loop index`

  $D(k) = \text{norm}(P_{cb}(k) - P_{gc}(M))$;

  $\mu_{\text{diff}}(k) = \mu_{\text{cb}}(k) - \mu_{\text{gc}}(M)$;

  $V_{\text{diff}}(k) = V_{\text{cb}}(k) - V_{\text{gc}}(M)$;

end

Return D, μ_{diff}, V_{diff}

Algorithm 3: *findConsecutiveSubSeq* - Consecutive Subsequences Identification

Input:

parentSeq (parent sequence)

element (element of interest)

Output:

S (vector that contains consecutive sequences' start indices)

E (vector that contains consecutive sequences' end indices)

Initialize s with the same length as *parentSeq* and fill with -1's;

`// find indices that is not equal to the element specified from the parent sequence, and set elements of these indices to 0`

$I \leftarrow$ **Find** indices where $parentSeq \neq element$;

$s(I) = 0$;

`// append two 0 elements to s; one at the beginning, one at the end`

$s \leftarrow \{0, s, 0\}$;

`// compute the difference between elements`

$s_{\text{diff}} \leftarrow$ **Compute** the differences between adjacent elements of s;

`// find the start and the end indices`

$S \leftarrow$ **Find** indices where $s_{\text{diff}} == -1$;

e $\leftarrow$ **Find** indices where $s_{\text{diff}} == 1$;

$E \leftarrow$ **Add** 1 for every element in e;

Return S, E

4.5 Results

Algorithm *uleIdentify* was run with the following heuristically chosen input tolerance parameters:

$$\epsilon_d = 7 \text{ m}, \epsilon_\mu = 0.1, \epsilon_v = 0.5 \text{ m/s}.$$

The resultant unloading event labels U and total unloading time T for each field were identified. Two examples of geospatial representations of identified in-field unloading events are given in Figure 4.8.

The accuracy of the identified unloading events were compared indirectly by comparing the estimated yields and the weight ticket records for each field. The total yield for each field was computed with the total unloading duration T and the maximum unloading auger rates specified in Table 4.5. It can be expressed as:

$$\text{Total yield (bu)} = \text{max unloading rate (bu/sec)} \times \text{total unloading duration (sec)}. \quad (4.24)$$

Equation (4.24) is an estimation of the total yield with the underlying assumption that the unloading rate was fixed at the maximum rate during each unloading. The computed total yields are listed in Table 4.6 along with the weight ticket records from these fields. Percentage error for each field were also calculated for comparisons.

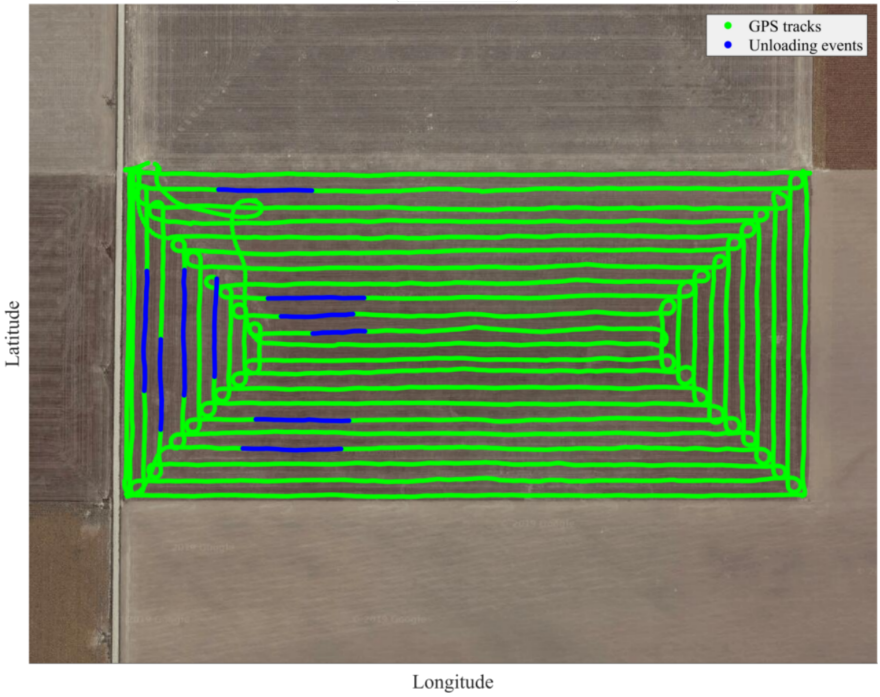
The estimated weights show both over and underestimations as indicated by the percentage error. The maximum absolute percentage error is 21.1% and the minimum is 0.4%. The maximum error traces back to field 14. By observing the original GPS data tracks for the three machines in this field, it was found that there were more occurrences when a combine and a grain cart parked next to each other at the same time instances. In these scenarios, as long as the computed machines' unloading parameters fit the rules specified in Algorithm 1, the identification algorithm would label these indices as unloading events. This could create false unloading event labels and incorrect yield estimations which further lead to large percentage error. Nevertheless, the average absolute percentage error was computed as 9.4%. It indicates that the algorithm performs well overall in identifying in-field combine

unloading events using only GPS data. The errors are reasonable due to assumptions and the simplifications during the estimation process.

Table 4.6. List of weight ticket records, estimated yields, and corresponding percentage error for 16 fields in this work.

Field ID	**Weight Ticket Records (bu)**	**Estimated Yields (bu)**	**Percentage Error (%)**
1	4255	4589	+7.9
2	4038	4211	+4.3
3	6310	5314	-16.2
4	8556	7632	-10.8
5	2511	2242	-10.7
6	6045	5782	-4.4
7	4884	4268	-12.6
8	4525	4915	+8.6
9	4615	4258	-7.7
10	9890	10832	+9.5
11	13698	15086	+10.1
12	4880	4546	-6.8
13	4429	4445	+0.4
14	16842	20401	+21.1
15	22027	23164	+5.2
16	6459	5576	-13.7

(a) Identified unloading events for the 7130 combine.



(b) Identified unloading events for the 8240 combine.

Figure 4.8. Geospatial representation of status of the 7130 and 8240 combines in the same field.

4.6 Conclusions

By using the unloading event identification algorithm discussed in this work, in-field unloading events were identified from GPS data collected from 16 fields of interest. The accuracy of the identified events were assessed comparing the estimated yields for each yield and the weight ticket records. The average absolute percentage error between weight ticket records and estimated yields using algorithm outputs was within 10%. The results demonstrated the applicability of using GPS data only to discover and identify useful insights for inter-machine. In addition, it is possible to improve and lower the estimation error by incorporating more sophisticated motion models and inclusions of more unloading scenarios. Furthermore, a more precise way to evaluate the identified unloading events could be explored. For instance, CAN bus data could be parsed for generating auger unloading labels. These labels could be utilized for assessing and improving the current algorithm performance.

5. WHEAT HARVEST PERFORMANCE COMPARISON USING MULTI-YEAR GPS DATA

5.1 Introduction

Combine harvesters are one of the most important machinery for wheat harvesting. Selecting the right combine harvesters according to factors like header sizes, autosteer capabilities, and logistical challenges are considerably the most critical decisions farmers need to make during harvest for achieving the best possible harvest performance. One way for evaluating harvest performance is through the productivities of combine harvesters. According to [106], productivity, also referred as area capacity C_{a}, is defined as follows:

$$C_{\mathrm{a}} = \frac{s \cdot w \cdot E_{\mathrm{f}}}{c}, \tag{5.1}$$

where

s is average speed,

w is full implement width,

E_{f} is field efficiency,

c is a normalization constant for unit conversion.

Nonetheless, there are several shortcomings with the metric of area capacity in Equation (5.1). First, the theoretical maximum area capacity is defined by assuming maximum field efficiency (i.e., $E_{\mathrm{f}} = 1$). In addition, nominal C_{a} values can be computed with speed values specified in [107]. These nominal values only provide a coarse estimation of machine performance. Moreover, these values are likely to change as the harvesting speed of combine harvesters tend to increase with the advancement in machine mechanics and improved route planning. Furthermore, the metric does not encapsulate a temporal or geospatial aspect which prevents farmers and researchers from performing time-motion analysis of harvest activities. Last but not least, the metric could not fully reflect a joint harvest scenario where

multiple combines work together with different productivities. Hence, the current way of computing area capacity could only provide a coarse assessment of harvest performance.

These drawbacks with the current metric could lead incorrect selection or paring of machine fleets, which could further result in decreased productivity and increased machine costs. Hence, improved metrics that could better describe both average and instantaneous aspects of harvest performance are needed.

5.2 Related Works

Various past works have investigated the topics of effective implement width, coverage area, and field efficiency. This section hopes to provide a brief literature review on these past works.

5.2.1 Effective Implement Width and Coverage Area Estimation

Various past works attempted to determine the both effective implement width and area coverage in various farming tasks from a single machine's perspective. In [108], the effective cut width of a combine harvester during wheat harvest was estimated using a bitmap-based method using GPS data. The estimated cut width was further utilized to adjust the yield map. However, this method assumed a fixed rectangular grid for computing the cut width. In addition, the actual harvested area was not computed. These disadvantages were discussed in [109], which proposed a polygon-based method for estimating the actual harvested area for a combine harvester during soybean harvest. Neither works considered the header working state (i.e., header up or down) which could lead to inaccuracy in their estimation. In addition, neither work mentioned any tactics for running the algorithm with noisy GPS data as both works utilized high-precision GPS data.

Moreover, a harvest area measurement system that incorporates ultrasonic sensors and DGPS was developed in [110] to accurately measure the effective cut width during a wheat harvest. The total harvested area was also estimated. The downside of this work is that the measurement system was complex and difficult to implement. Another work in [111] described a method that fused both GPS and CAN data to determine the total cultivated

area, overlaps, and field boundary for a plowing operation. By parsing CAN data, the plow working state can be accurately recorded. Hence, the total actual plowing area could be better estimated along with the overlapping area. However, the estimation of the effective plow width was not mentioned.

None of the mentioned works discuss the situation where multiple machines are present in field for cooperative operations. The multiple-machine scenario is common in various activities like wheat and corn harvesting. Hence, the effective swath width of one machine with the presence of another should be investigated.

5.2.2 Field Efficiency and Capacity Estimation

The definition of field efficiency discussed in Section 5.1 and Equation (5.1) were frequently employed in past works to evaluate machine and field performance. For instance, four traffic pattern indices were developed using GPS and yield monitor data in [112] and [113] for comparing field efficiencies in fields with different shapes and finding correlations between field efficiencies and average steering angles. Moreover, both field efficiency and field machine index (FMI) were estimated from yield monitor data for a citrus canopy shaker in [114] for evaluating harvest performance. Moreover, a similar approach was taken for evaluating grape harvester [115] and biomass balers [116].

Nevertheless, the disadvantage associated with these works is that the theoretical maximum area capacity is assumed not measured. Hence, the accuracy of the resultant field efficiency E_{f} cannot be evaluated. This issue is circumvented by employing slightly modified field efficiency measures as discussed in [117] and [118]. The modified measures can be summarized as:

$$E_{\mathrm{f}} = \frac{t_{\mathrm{p}}}{t_{\mathrm{p}} + t_{\mathrm{np}}} \tag{5.2}$$

where t_{p} represents the total productive time and t_{np} is the total non-productive time. It aims to account for the time when the implement was actually engaged during field operation and this value could be easily aggregated from yield monitor data.

Nonetheless, time-based measurement in Equation (5.2) still focuses on estimating and comparing overall field efficiency and capacity. For farming operations, field efficiency and

capacity are likely to vary with time and location. Although comparing the overall field efficiency and/or capacity gives an estimate on how a machine (or operator) performed, it does not offer finer details in either temporal or geospatial sense. A new efficiency index called field traversing efficiency was proposed in [119] to address this issue. However, the computation of this index was applied to simulated paths instead of real GPS data. Moreover, the temporal aspect of the efficiency index was not investigated. This calls for the need to estimate instantaneous metrics. It not only gives finer details on temporal machine performance but also improves the ability to identify suboptimal or high efficiency regions if paried with geospatial data.

5.3 Contributions

The contributions of this work are listed as follows:

1. Proposed a novel method for classifying states of GPS data using Interacting Multiple Models (IMM) filtering and Spatial-Temporal Density-Based Spatial Clustering of Applications with Noise (ST-DBSCAN). The advantage of this method is that it only needs GPS data to infer machine states instead of relying on yield monitor data in past works discussed in Section 5.2.

2. Proposed a novel method that incorporates both track smoothing and offset correction via optimization to automatically correct GPS positions. This method mitigates the reliance on high-precision GPS data for estimating effective swath widths and actual harvested areas.

3. Proposed and computed two instantaneous metrics (instantaenous header utilization and instantaenous area capacity) for comparing harvesting performances among machines and years in different fields.

5.4 GPS Data Selection

Since the primary objective of this work is to perform harvest efficiency analyses of across multiple combines and various years, a fair comparison can only be achieved by collating effi-

ciency metrics of the work done by multiple combines within the same fields across different years. Hence, a comprehensive GPS dataset that encapsulates these is needed.

With this requirement, the Combine Kart Truck GPS Data Archive [120] was selected as the data source for this work. It contains GPS data of mixed fleets of combine harvesters collected using Android tablets during the wheat harvest season from 2014 to 2019 in Colorado, USA. In addition, metadata related to combine harvesters—model names and header width configurations, were collected from the farm personnel prior to this work. Using the machine identifiers (IDs) provided from the dataset, data from 5 different combine harvesters were selected and stored as individual Comma-Separated Values (CSV) files. Each file was named as `machineId-year.csv` to specify a combine's data for a particular year. The mappings between the combine model names and machine IDs for these combines are listed in Table 5.1.

Table 5.1. Combine harvester models and machine IDs mappings.

Model Name	Machine ID
CASE IH Axial-Flow® 2388	`2388`
CASE IH Axial-Flow® 6088	`6088`
CASE IH Axial-Flow® 6130	`6130`
CASE IH Axial-Flow® 7130	`7130`
CASE IH Axial-Flow® 8240	`8240`

Each CSV file contains GPS samples at a frequency of 1 Hz with following data columns: `ts`, `lat`, `lon`, `speed`, and `heading`. Their definitions are given in Table 5.2.

Table 5.2. Combine harvester GPS CSV file data columns and their definitions.

Column Name	Definition (units)
`ts`	GPS epoch timestamp (s)
`lat`	Latitude (degree)
`lon`	Longitude (degree)
`speed`	Machine speed (m/s)
`heading`	Heading from true north (degree)

Additional preprocessing steps were performed on each CSV file. First, each CSV file was checked for any null and duplicated GPS samples. These samples were subsequently removed. Secondly, the `lat` and `lon` columns of these files were converted to Universal Transverse Mercator (UTM) coordinates (easting and northing in meters) as later processing steps would require working with coordinates in an Euclidean plane. The UTM coordinates were appended as two new columns called `x` and `y` to each CSV file. Thirdly, GPS samples within each file were filtered using field boundaries. There are 7 field boundary polygons created using the Google Maps tool [121]. In addition, total field areas in acre were gathered from the same tool. Since the field boundaries were created manually, the total field areas served as approximate numbers and were not utilized in later part of this work for evaluating different metrics. Morover, each field was assigned with an abbreviated name using the full name given by the farm personnel. The reason for selecting these fields was that combine harvesters worked jointly in these fields during two harvest years, offering an appropriate dataset for later comparisons.

After the filtering process, only in-field GPS samples of each CSV file were kept and saved as `machineId-fieldName-year.csv`. The detailed information of the selected fields, machine usages, header configurations, and total numbers of GPS samples, are populated in Table B.1. It can be observed that both machine fleets and header configurations sometimes vary over two harvest years.

5.5 GPS Error Characterizations

Estimations of effective swath widths, harvested areas, and harvest efficiency metrics rely on first knowing the positions of combine header centers. The current GPS samples do not reflect where the header is due to various GPS errors. These errors need to be first modeled and corrected; otherwise, the estimations of the aforementioned parameters could be badly biased.

In this work, it is assumed that tracks were associated with three main types of GPS errors: 1) a fixed offset from the GPS receiver antenna to the header center, 2) GPS receiver biases, and 3) GPS receiver random noises. The first error was introduced since the data collection devices were installed in the machine cabs. This created a machine-specific distance between the devices and the header centers. Moreover, biases and random noises are caused by various time-varying factors [122] that affect a GPS receiver's ability to report the true positions of the receiver.

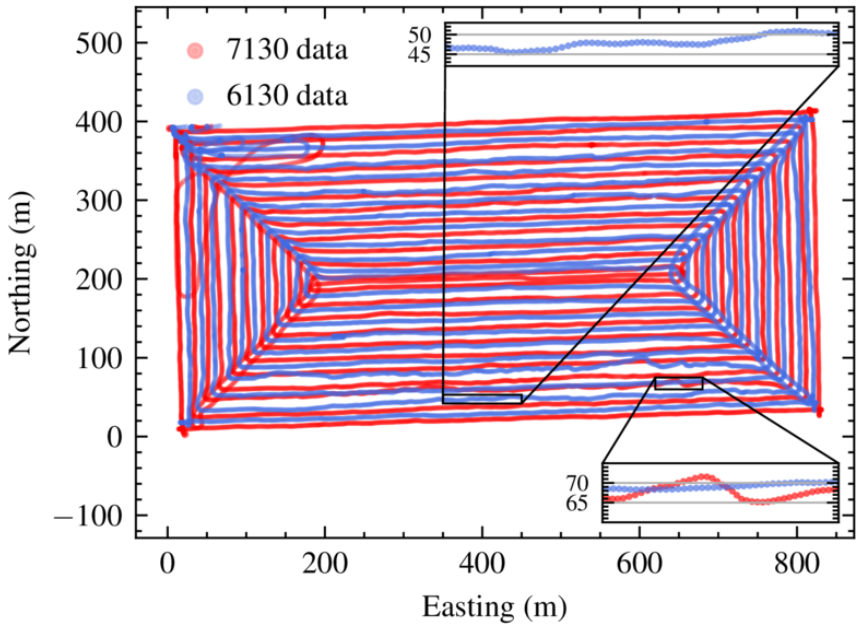


Figure 5.1. GPS tracks for field `RU` in 2017.

As an example, tracks driven by the 6130 and 7130 combines in field `RU` during 2017 wheat harvest are shown in Figure 5.1 to illustrate the effects of biases and random noises.

The top right figure shows possible error caused by time-varying random noises as the 6130 combine should drive a smooth path during harvesting. Moreover, the bottom right figure illustrates possible GPS biases as the paths of two combines overlapped.

To model these errors, two assumptions were made. The first assumption was that for a given year, the GPS receiver bias was field-dependent only. In other words, the biases for all GPS samples for a combine in a certain field were assumed to be fixed. Moreover, it was assumed that the receiver random noise could be modeled as a Gaussian distribution with independent means and variances for easting and northing directions. Hence, a simple model for describing the GPS errors for a combine c in a specific field f can be developed as follows:

$$\begin{aligned} \mathcal{E}^{(c,f)}(k) &= \mathcal{O}^{(c,f)} + \mathcal{B}^{(c,f)} + \mathcal{H}^{(c,f)}(k) \qquad (5.3) \\ &= \begin{bmatrix} o_x^{(c,f)} \\ o_y^{(c,f)} \end{bmatrix} + \begin{bmatrix} \beta_x^{(c,f)} \\ \beta_y^{(c,f)} \end{bmatrix} + \begin{bmatrix} \eta_x^{(c,f)}(k) \\ \eta_y^{(c,f)}(k) \end{bmatrix} \\ &= \begin{bmatrix} o_x^{(c,f)} + \beta_x^{(c,f)} \\ o_y^{(c,f)} + \beta_y^{(c,f)} \end{bmatrix} + \begin{bmatrix} \eta_x^{(c,f)}(k) \\ \eta_y^{(c,f)}(k) \end{bmatrix} \qquad (5.4) \end{aligned}$$

where

$$\begin{aligned} \mathcal{E}^{(c,f)}(k) &= \text{the overall error at time step } k, \\ \mathcal{O}^{(c,f)} &= \text{the fixed antenna to header center offset}, \\ \mathcal{B}^{(c,f)} &= \text{the fixed receiver bias}, \\ \mathcal{H}^{(c,f)}(k) &= \text{the receiver random noise at time step } k, \end{aligned}$$

and the random noise for easting and northing direction is assumed to be independent Gaussian noises:

$$\eta_{(\cdot)}(k) \sim \mathcal{N}(\mu_{(\cdot)}, \sigma^2_{(\cdot)}). \qquad (5.5)$$

If a GPS track sample of a combine c in a field f is denoted as $Z^{(c,f)}(k)_{\text{meas}}$ at time step k, the combine header center can be expressed as:

$$Z^{(c,f)}_{\text{meas}}(k) = Z^{(c,f)}_{\text{h}}(k) + \mathcal{E}^{(c,f)}(k). \tag{5.6}$$

The rest of this work describes a workflow that first estimates the combine header center positions via a series of smoothing and correction and then computes instantaneous effective swath width, actual harvested area, and instantaneous metrics in regard to harvester performances.

5.6 Algorithm

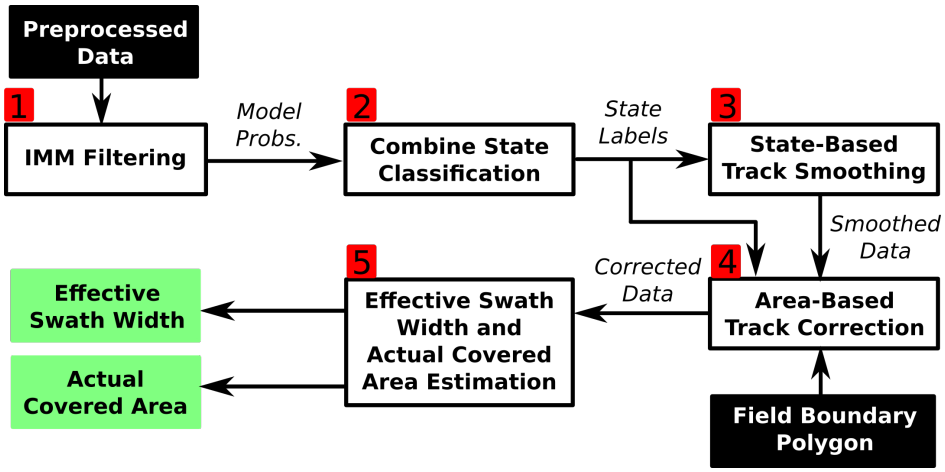


Figure 5.2. Workflow of the algorithm for correcting the GPS tracks and computing the instantaneous parameters.

The algorithm consists of 5 steps. The workflow of these steps is visualized in Figure 5.2. The algorithm uses a combination of filtering, smoothing, and optimization techniques to mitigate the effects of the GPS errors on each track data before estimating the effective swath width and actual harvested area at each time step. The summary for each step is listed as follows:

1. Interacting Multitple Models (IMM) filtering: GPS data were analyzed using a IMM filter with two models (Nearly Constant Velocity and Nearly Coordinated Turn) for

generating model probabilities. The model probabilities were utilized in later steps for data classification and track smoothing purposes.

2. Operation State Classification: this step coupled Spatial-Temporal Density Based Spatial Clustering of Applications with Noise (ST-DBSCAN) [123] and heuristic rules to segment GPS samples into four states: harvesting, turning, idling, and speeding. These states are utilized in later steps for further smoothing and corrections.

3. State-based smoothing: GPS samples were smoothed using a Constant Acceleration (CA) model based Rauch-Tung-Striebel (RTS) smoother [124] to mitigate the effect of random noise $\mathcal{H}$.

4. Area-Based Track Correction: this step finds a correction vector to compensate the effect of the fixed antenna error $\mathcal{O}$ and the fixed receiver bias error $\mathcal{B}$.

5. Effective swath width and actual harvested area estimation: this step uses the corrected GPS samples to estimate the effective swath width and actual harvested area. These quantities are further utilized for computing efficiency metrics.

To have a consistent flow of explanation, the visual outputs of each algorithm step are based on the 7130 combine track data in field `RU` during 2017 wheat harvest.

5.6.1 Interacting Multiple Model (IMM) Filtering

The IMM filtering introduced in Section 4.4.1 was employed with the same state-space models (Nearly Constant Velocity and Nearly Coordinated Turn models) for filtering the GPS tracks. The running of the IMM algorithm followed the same initializations and selections of input parameters and noise levels specified in Section 4.4.1. Moreover, the Extended Kalman filter (EKF) for handling the nonlinear NCT model in the IMM algorithm was replaced by the Unscented Kalman filter (UKF) [125].

Different aspects of performance improvement of UKF over EKF were discussed in various past publications [126]–[128]. The major flaw of EKF comes from the linearization of a nonlinear process model $f(\cdot)$ using a first-order Taylor series expansion, which only provides

an approximation to the optimal terms. The UKF, on the other hand, generates a set of points called sigma points, denoted as $\mathcal{X}$, with their corresponding weights via a sigma function and a weight function. The sigma points are propagated forward in time using the unscented transform—which essentially is passing the sigma points through the nonlinear process model (i.e., $f(\mathcal{X})$) of a given problem. The propagated sigma points as well as the corresponding weights are utilized for computing the new state estimates and covariance. In this work, the generation of sigma points and the weights were based on the algorithm in [125], which are,

$$\begin{aligned}
\mathcal{X}_0 &= X, \\
\mathcal{X}_{\mathrm{i}} &= X + \left[\sqrt{(n+\lambda)P}\right]_{\mathrm{i}} \quad \text{for i} = 1, \cdots, n, \\
\mathcal{X}_{\mathrm{i}} &= X - \left[\sqrt{(n+\lambda)P}\right]_{\mathrm{i}} \quad \text{for i} = n+1, \cdots, 2n,
\end{aligned}$$

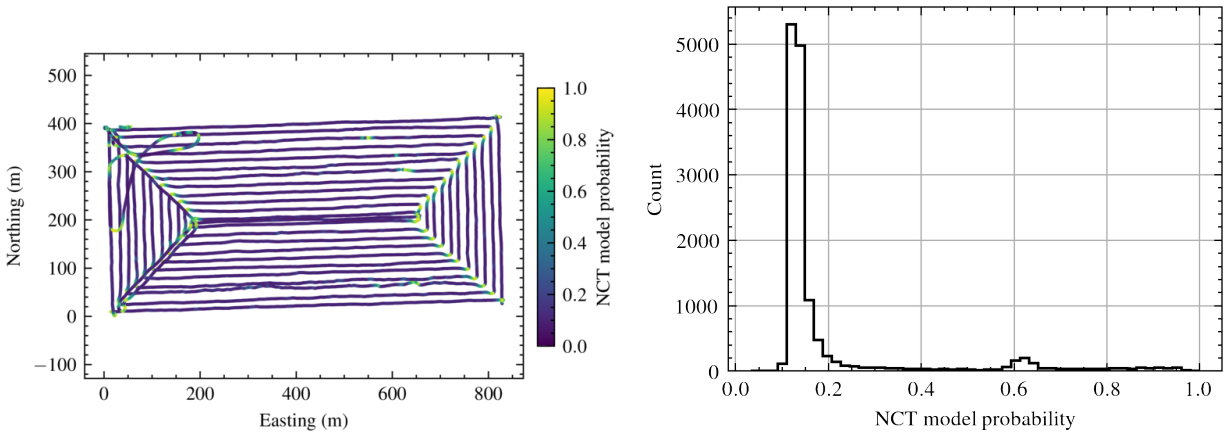
and

$$\begin{aligned}
w_0^{(m)} &= \frac{\lambda}{n+\lambda}, \\
w_0^{(c)} &= \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \nu), \\
w_{\mathrm{i}}^{(m)} &= w_{\mathrm{i}}^{(c)} = \frac{1}{2(n+\lambda)} \quad \text{for i} = 1, \cdots, 2n,
\end{aligned}$$

where X and P are the state and covariance estimates. The dimensionality of the state vector n is 5 as described by Equation (4.6) and $\lambda = \alpha^2(n+\kappa) - n$ is a scaling parameter. The choice of scaling factor α, κ, and ν is based on discussions in [125]. These parameters were selected accordingly as follows:

$$\begin{aligned}
\alpha &= 0.1, \\
\nu &= 2, \\
\kappa &= -2.
\end{aligned}$$

Using the updated IMM algorithm, NCV and NCT model probabilities at each time step k were generated for each GPS track. Figure 5.3a shows the NCT model probability map and the corresponding NCT model probability histogram for the 7130 combine in field RL during 2017 wheat harvest. In Figure 5.3a, the map shows the NCT model probability increases whenever the combine made turns. In addition, the distribution of NCT model probability values suggests that the combine spent most of its time harvesting.



(a) Map of NCT model probability.

(b) Histogram of NCT model probability.

Figure 5.3. Results of NCT model probability of 7130 tracks in field RL during 2017 wheat harvest.

5.6.2 Combine State Classification

This step uses the position, speed, and the NCT model probability data of each GPS track for classifying combine operational states. The classified states enables the state-based track smoothing step in Section 5.6.3 and allows computations of efficiency metrics in Section 5.7.

There are four operational states of interest: harvesting, turning, idling and speeding. These states represent both productive and non-productive combine activities during wheat harvest. The only productive state, harvesting, represents the instances when the combine was actively cutting wheat. In terms of non-productive states, the idling and speeding states represent the times when the combine was either at rest or traveling at high speeds; both states pertain to maneuvers that do not produce harvested wheat. Moreover, the turning

state only corresponds to non-productive turns as oppose to turns that amount to wheat cutting. The list of states and their abbreivated labels are provided in Table 5.3.

Table 5.3. List of operational states and their labels.

State	Label
Harvesting	`H`
Turning	`R`
Idling	`I`
Speeding	`S`

The classification workflow is overviewed in Figure 5.4. It is based on the assumption that each GPS sample can only be associated with one state label. The workflow first separates the samples from each GPS track G into preliminary `R` and `H` samples ($G_{\mathrm{R,p}}$ and $G_{\mathrm{H,p}}$) based on their NCT model probability values (p_{NCT}): if a sample has a NCT probability that exceeds 0.5, it would be labeled as `R`; otherwise, it would be labeled as `H`. Subsequently, the preliminary samples are filtered via two refinement processes to generate final classified GPS samples for each state category.

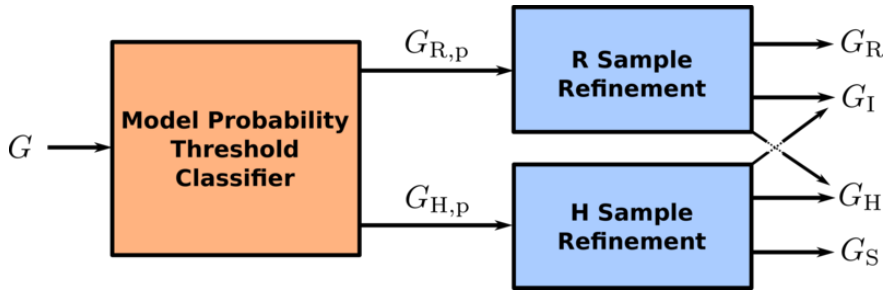


Figure 5.4. The state classification includes a basic classifier and two refinement proccess for identifying state labels for a GPS track dataset G.

The samples in $G_{\mathrm{R,p}}$ and $G_{\mathrm{H,p}}$ normally contain misclassified samples as depicted in Figure 5.5. For example, the turns at corners should be classified as `H` as the combine was simultaneously turning and harvesting wheat; however, these samples were mistaken as `R`

samples instead. In addition, the preliminary `H` samples might also contain both `I` and `S` samples. Hence, these samples need to be further refined.

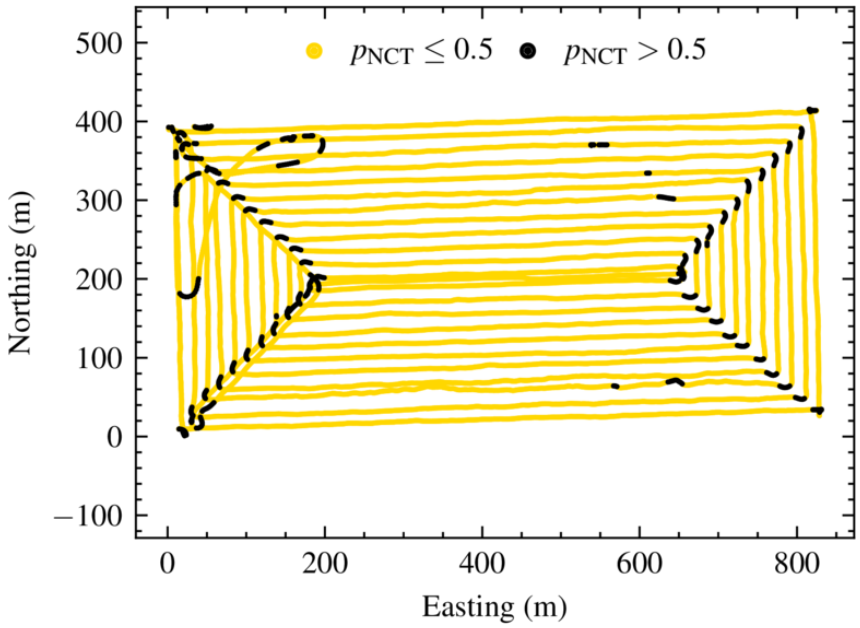


Figure 5.5. Results of preliminary `R` and `H` samples of 7130 tracks in field `RU` during 2017 using model probability threshold classifier.

The refinement of samples in $G_{\text{R,p}}$ was conducted by examining average speeds, traveled distances, and maximum heading changes of GPS segments created from these samples. The GPS segments were created using the Spatial-Temporal DBSCAN (ST-DBSCAN) algorithm [123], which extends the DBSCAN algorithm described in Section 3.4.1 by incorporating temporal distances (i.e., time differences) in addition to spatial distances among data samples. This extension allows clustering of scattered GPS samples that are both spatially and temporally close. For running the ST-DBSCAN algorithm, a maximum temporal distance ϵ_t was selected in addition to the maximum spatial distance ϵ_s and the minimum number of samples *minPts* for forming clusters. These parameters are summarized from Equation (5.7)

to Equation (5.9). Figure 5.6 illustrates a series of clustered GPS segments using the preliminary R samples in Figure 5.5.

$$\epsilon_t = 10 \text{ seconds}, \tag{5.7}$$

$$\epsilon_s = 5 \text{ meters}, \tag{5.8}$$

$$minPts = 2. \tag{5.9}$$

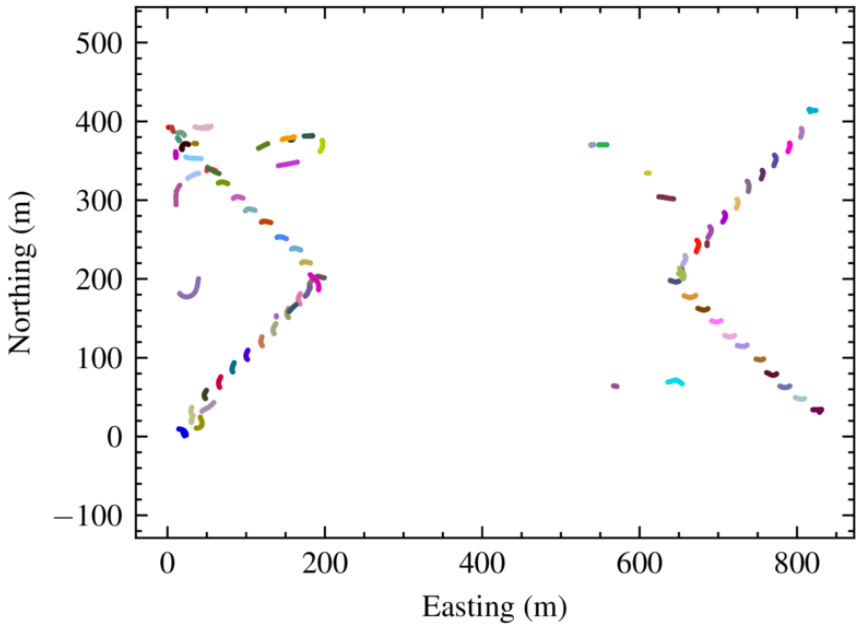


Figure 5.6. Results of ST-DBSCAN generated segments using preliminary R samples in Figure 5.5.

The ST-DBSCAN algorithm returned both noise and GPS segments. The noise samples were relabeled as H since they didn't belong to any R segments. Moreover, the segments were checked iteratively with a composite rule specified in line 5 to 13 from Algorithm 4. The rule was constructed with three thresholds, γ_{h}, γ_{s}, and γ_{d}, which corresponds to the maximum heading change, average speed, and total traveled distance of a particular segment. In Algorithm 4, line 5 checks whether the maximum heading change within a segment exceeds γ_{h}. This better distinguishes whether a segment is a non-productive turn or a turn that amounts to harvesting. As a result, samples of a segment would be relabeled as H if its

maximum heading change falls below γ_{h}. For segments that exceed this threshold, they were checked for average speeds and the total traveled distances in line 6 to rule out any idling scenarios. If a segment's average speed and total traveled distance is less than γ_{s} and γ_{d}, the samples of the segment would be relabeled as `I`.

The selections of γ_h, γ_s, and γ_d were based on heuristic wheat harvest experience. The chosen thresholds are listed as follows:

$$\gamma_{\mathrm{h}} = 90^{\circ}, \tag{5.10}$$

$$\gamma_{\mathrm{s}} = 0.1 \text{ meters/second}, \tag{5.11}$$

$$\gamma_{\mathrm{d}} = 1 \text{ meter}. \tag{5.12}$$

Algorithm 4: Refinement of `R` Segments

Input:
S — `R` segments collection
Output:
S_{R} — refined `R` segments collection
S_{H} — `H` segments collection
S_{I} — `I` segments collection

```
// Initialize collections
```
1 $S_{\mathrm{R}} \leftarrow S$;
2 $S_{\mathrm{H}} \leftarrow \emptyset$;
3 $S_{\mathrm{I}} \leftarrow \emptyset$;
```
// Rule-based method for refining S
```
4 **for** s **in** S **do**
5 **if** maxHeadingChg(s) $> \gamma_h$ **then**
6 **if** meanSpeed(s) $< \gamma_s$ and distTravel(s) $< \gamma_d$ **then**
7 $S_{\mathrm{I}} \leftarrow S_{\mathrm{I}} \cup s$;
8 $S_{\mathrm{R}} \leftarrow S_{\mathrm{R}} \setminus s$;
9 **end**
10 **else**
11 $S_{\mathrm{H}} \leftarrow S_{\mathrm{H}} \cup s$;
12 $S_{\mathrm{R}} \leftarrow S_{\mathrm{R}} \setminus s$;
13 **end**
14 **end**
15 **return** S_{R}, S_{H}, S_{I}

The refinement of $G_{\mathrm{H,p}}$ samples has two parts. The first part coarsely segments the samples into `I` and `S` samples using two speed thresholds. Samples with speed less than

γ_s defined in Equation (5.11) were relabeled as `I`. Furthermore, samples with speed greater than a threshold $\gamma_{s,max}$ were relabeled as `S`. This threshold was selected to be the overall mean speed plus one standard deviation determined from all samples in a GPS track. It was assumed that any speed that exceeded γ_s indicated the speeding motion of a combine.

The second part of the refinement used a similar ST-DBSCAN approach to determine if the `S` samples were valid. The `S` samples were clustered into segments using ST-DBSCAN. The noise samples were relabeled as `H` samples. Furthermore, each segment was checked for its sample size; any segments that had less than 5 samples were regarded as noisy speed variations during harvesting and relabeled as `H`. The resultant classification after these steps is demonstrated in Figure 5.7.

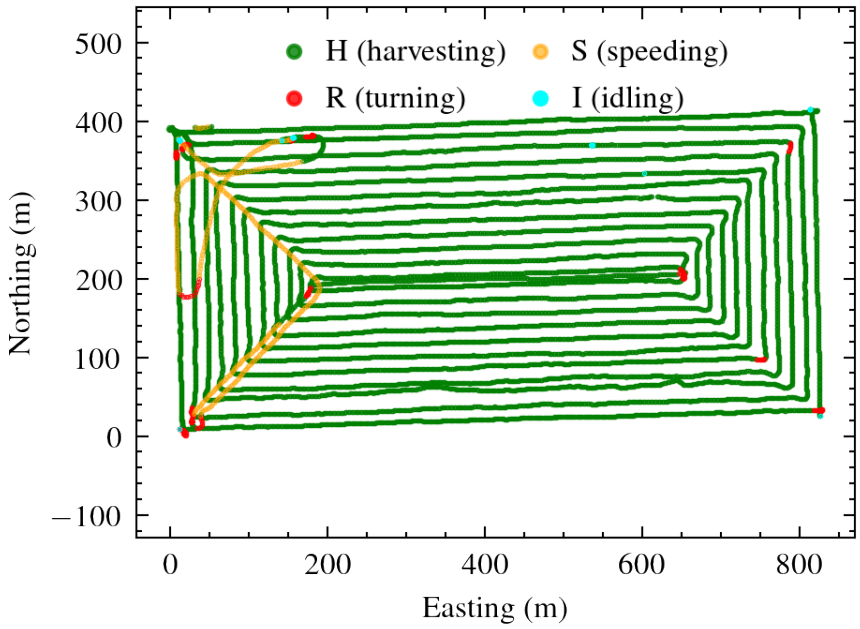


Figure 5.7. Results of the state classification result for 7130 track in field `RU` during 2017 wheat harvest.

5.6.3 State-Based Track Smoothing

The goal of this step is to minimize of the effect of random noise $\mathcal{H}(k)$ in Equation (5.4) for a given GPS track data. Two simplifications were made. First, the idling GPS samples identified in last step were removed and not included in the smoothing process. The reason

is that these samples don't contribute to harvest in terms of area traversing. In addition, these samples adds unwanted noises to later steps for finding the bias vector.

The second simplication is that the turning GPS samples were assumed to have negligible random noises and thus not included in the smoothing process. In other words, their original coordinates were kept after this step. The reason for this simplifications is that these samples typically constituent a small portion of all samples. Hence, the assumption would not be detrimental to later estimation steps.

Moreover, it is assumed that the effect of the time-varying random noise $\mathcal{H}(k)$ is minimized after track smoothing. Hence, the GPS error in Equation (5.4) can be rewritten as:

$$\mathcal{E}^{(c,f)} = \mathcal{O}^{(c,f)} + \mathcal{B}^{(c,f)} \tag{5.13}$$

$$= \begin{bmatrix} o_x^{(c,f)} + \beta_x^{(c,f)} \\ o_y^{(c,f)} + \beta_y^{(c,f)} \end{bmatrix} \tag{5.14}$$

A Rauch-Tung-Striebel (RTS) smoother [124] was employed to perform the smoothing of tracks. An RTS smoother consists of a forward pass and a backward pass. The forward pass generates the state estimates and error covariances via a Kalman filter with a predefined state space model. The outputs of the forward pass are inputs for the backward pass which also contains a Kalman filter. The backward pass starts at the last time step and proceed in reverse time to compute the smoothed state estimates.

The Constant Acceleration (CA) model was selected as the state space model for smoothing. This model was chosen instead of a more conventional Constant Velocity (CV) model because a combine harvester doesn't travel at a constant speed at all times. In other words, it could accelerate or deccelerate during stops and turns. The CA model is better suited for modeling this motion. Its state transition as well as the measurement matrices are listed as the following:

$$X(k+1) = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot X(k) + v(k), \tag{5.15}$$

$$Z(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \cdot X(k) + w(k) \tag{5.16}$$

where $X(k)$ is the state vector at time step k defined as $\begin{bmatrix} x & \dot{x} & \ddot{x} & y & \dot{y} & \ddot{y} \end{bmatrix}^T$. x and y are easting and northing. $Z(k)$ is the predicted measurement vector. $w(k)$ is the measurement noise covariance matrix $R_{\text{CA}}(k)$ and $v(k)$ is the process noise covariance matrix $Q_{\text{CA}}(k)$. According to [129], $R_{\text{CA}}(k)$ and $Q_{\text{CA}}(k)$ are given by:

$$Q_{\text{CA}}(k) = \begin{bmatrix} \frac{\Delta t^5}{20} & \frac{\Delta t^4}{8} & \frac{\Delta t^3}{6} & 0 & 0 & 0 \\ \frac{\Delta t^4}{8} & \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ \frac{\Delta t^3}{6} & \frac{\Delta t^2}{2} & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^5}{20} & \frac{\Delta t^4}{8} & \frac{\Delta t^3}{6} \\ 0 & 0 & 0 & \frac{\Delta t^4}{8} & \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & \frac{\Delta t^3}{6} & \frac{\Delta t^2}{2} & \Delta t \end{bmatrix} \cdot q_{\text{CA}}(k). \tag{5.17}$$

$$R_{\text{CA}}(k) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot r_{\text{CA}}(k). \tag{5.18}$$

Two noise levels $q_{\text{CA}}(k)$ and $r_{\text{CA}}(k)$ were assumed to be constant throughout the smoothing process and they were initialized in Equation (5.19). The choice of a large $r_{\text{CA}}(k)$ with a small q_{CA} value tells the smoother to assume a large measurement noise. In other words,

the smoother will try its best to filter out any erratic movements that do not conform to a straight pass.

$$q_{\text{CA}} = 0.01, \tag{5.19a}$$

$$r_{\text{CA}} = 9. \tag{5.19b}$$

For applying the smoother, the harvesting samples were taken out of each GPS track and splitted into segments. Afterwards, the smoother was applied to each segments to obtain a smooth trajectory. Figure 5.8 depicts an example of the smoothed version of a GPS track. It is evident that the smoothed coordinates are more aligned with how an operator would drive during a straight pass.

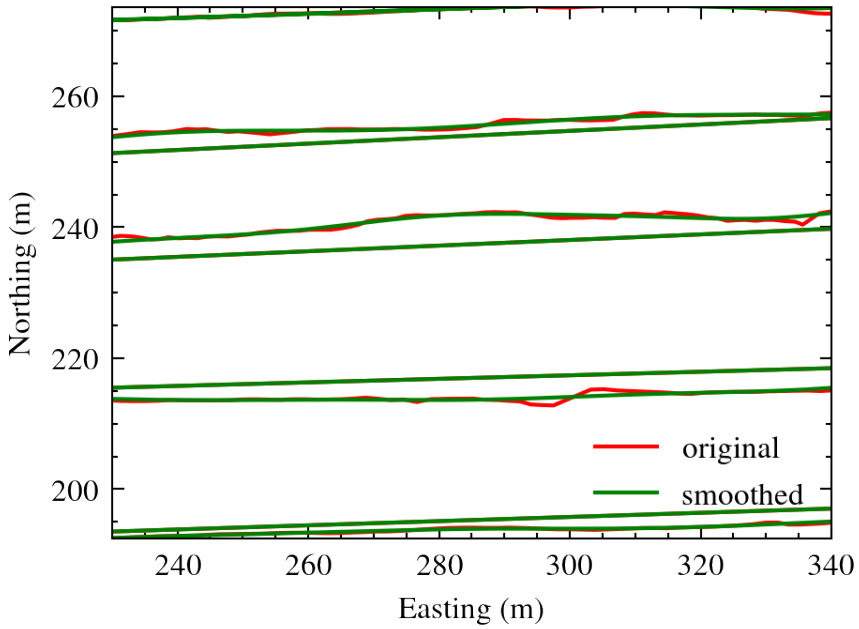


Figure 5.8. Comparison between the original and the smoothed version for a portion of the 7130 track in field `RL` from 2017.

5.6.4 Area-Based Track Correction

The goal of this step is to find a correction vector that would compensate the GPS bias error vector defined in Equation (5.4). Denote this correction vector as $\mathcal{V}^{(c,f)}$, the corrected GPS sample $Z_{\text{corr}(k)}$ at time k could be expressed as:

$$Z_{\text{corr}}^{(c,f)}(k) = Z_{\text{meas}}^{(c,f)}(k) + \mathcal{V}^{(c,f)}, \tag{5.20}$$

where $Z_{\text{meas}}(k)$ was defined in Equation (5.6).

Several preliminary concepts are first described to serve as foundations to the optimization problem in later part of this section. Once these concepts are established, an area-based optimization is introduced to estimate the correction vector for each GPS track.

Swath Line and Coverage Polygon

For a particular combine's GPS track data, denoted as G, consists of a sequence of GPS samples $g(k)$ where k ranges from 0 to N. By connecting consecutive GPS samples, a collection of paths, denoted as $l(k)$, can be created. With the connected paths and a known swath width w, two types of shapes, namely swath lines and coverage polygons, could be created as illustrated in Figure 5.9.

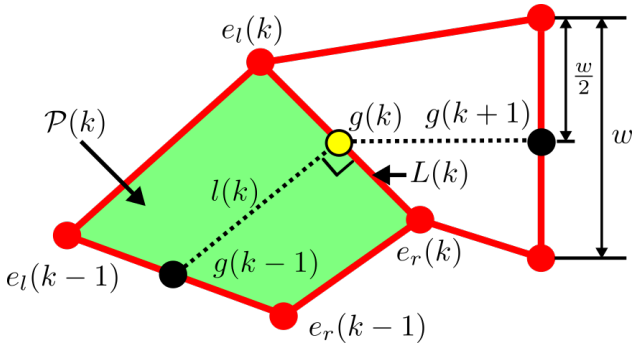


Figure 5.9. Illustration of coverage shapes created using consecutive GPS coordinates.

In Figure 5.9, sets of edge points $\mathrm{e}_l(k)$ and $\mathrm{e}_r(k)$ could be created by extending a path $l(k)$ perpendicularly to the left and right by one half swath width. By connecting the edge

points, swath lines L at time step k could be created. Moreover, a coverage polygon $\mathcal{P}(k)$ is formed by two sets of edge points from time step k and $k-1$. Note that for a total of N time steps, $(N-1)$ coverage polygons could be formed.

Actual Harvested Area vs. Traversed Area

At a given time step k, the actual harvested area by a combine is typically not the area of the coverage polygon as depicted in Figure 5.10. At a time step k, the area of the coverage polygon that is associated with a combine's GPS coordinate $g^{(1)}(k)$ is the traversed area, denoted as $a_{\text{trv}}(k)$. The actual harvested area at a time step k, denoted as $a_{\text{act}}(k)$, is actually the difference between the traversed and the overlapping area $a_{\text{olp}}(k)$ caused by past traversed areas either from the same or the other combine. Mathematically, this could be expressed as:

$$a_{\text{act}}(k) = a_{\text{trv}}(k) - a_{\text{olp}}(k) \tag{5.21}$$

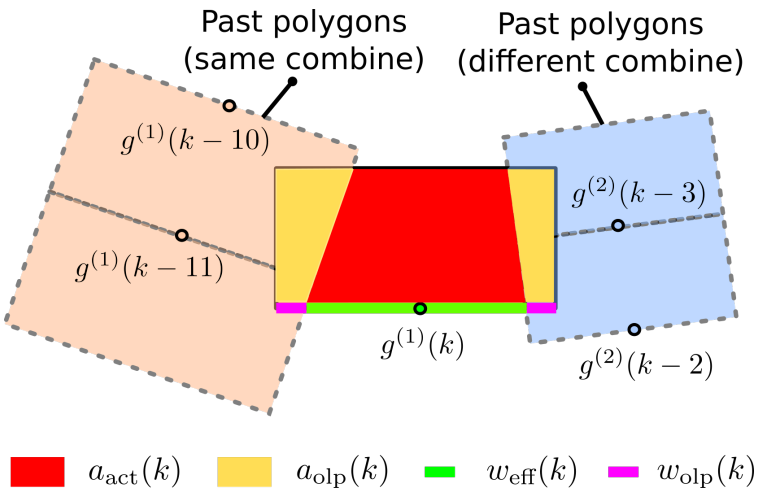


Figure 5.10. Illustration of actual harvested area and overlapping area.

Likewise, the effective swath width at time step k, denoted as $w_{\text{eff}}(k)$, is the length of the swath line (i.e., full swath width) minus the overlapping length, $w_{\text{olp}}(k)$, from the intersected area.

Correction Vector Estimation via an Area-Based Optimization

The intuition behind this step is that during normal wheat harvest, a combine operator typically intends to minimize potential overlapping and skipped area when traversing through a field. The skipped area is the area where an operator missed during harvest. It is highly unlikely to happen in real life but it could occur with noisy GPS tracks in this work. This idea serves as the underlying guideline for designing the optimization problem. For finding the correction vector for each GPS track, the original GPS samples first are shifted to a set of new coordinates. Based on the shifted coordinates, coverage polygons could be generated to compute the overlapping and the skipped area, which are encapsulated in a cost function. The goal of the optimization is to find the optimal shifts (i.e., correction vector) to the original samples that would incur the lowest cost.

The computation of the overlapping area is broken down to three steps. First, the actual harvested area needs to be computed. Denote A_{act} as the total actual harvested area by two combines, the computation could be expressed as:

$$A_{\text{act}} = \mathcal{A}\left\{\left[\bigcup_{m=0}^{M-1} \mathcal{P}_{\mathcal{V}}^{(1)}(m)\right] \bigcup \left[\bigcup_{n=0}^{N-1} \mathcal{P}_{\mathcal{V}}^{(2)}(n)\right]\right\}, \tag{5.22}$$

where

$\mathcal{A}\{\cdot\}$ denotes an operator for computing areas of polygons,

$\mathcal{P}_{\mathcal{V}}^{(\cdot)}(k)$ denotes a coverage polygon created using the corrected coordinate at time step k,

$\bigcup$ performs the union of polygons,

M, N are data sizes of the two GPS track data of interest.

The second step involves computing the total traversed area, denoted as A_{trv}. This area could be calculated from summing up the areas from all the coverage polygon areas, which could be expressed as:

$$A_{\text{trv}} = \sum_{m=0}^{M-1} \mathcal{A}\left[\mathcal{P}_{\mathcal{V}}^{(1)}(m)\right] + \sum_{n=0}^{N-1} \mathcal{A}\left[\mathcal{P}_{\mathcal{V}}^{(2)}(n)\right] \tag{5.23}$$

Finally, the total overlapping area A_{olp} could be obtained as the difference between A_{trv} and A_{act}:

$$A_{\text{olp}} = A_{\text{trv}} - A_{\text{act}}. \tag{5.24}$$

On the other hand, the computation of total skipped area is straightforward since A_{act} is already known:

$$A_{\text{skp}} = \mathcal{A}(\mathcal{P}_{\text{f}}) - A_{\text{act}}, \tag{5.25}$$

where P_{f} denotes a field boundary polygon described in Section 5.4.

With these quantities established, the optimization problem could be formally defined as:

$$\min_{(\mathcal{V}^{(1)}, \mathcal{V}^{(2)})} f(\mathcal{V}^{(1)}, \mathcal{V}^{(2)}) \tag{5.26}$$

where

$$\begin{aligned} &\mathcal{V}^{(\cdot)} = (v_x^{(\cdot)}, v_y^{(\cdot)}), \\ &f(\mathcal{V}^{(1)}, \mathcal{V}^{(2)}) = a \cdot A_{\text{olp}}(\mathcal{V}^{(1)}, \mathcal{V}^{(2)}) + b \cdot A_{\text{skp}}(\mathcal{V}^{(1)}, \mathcal{V}^{(2)}). \end{aligned} \tag{5.27}$$

In Equation (5.27), $\mathcal{V}^{(\cdot)}$ is the correction vector for each combine's track data. a and b are weights assigned to area costs with $a + b = 1$. The choice of weights a and b was selected based on heuristic experience on combine operation. Intuitively, a combine operator intends to minimize both overlaps and skips. Hence, it makes sense to assign the same weights to a and b to equally penalize the cost associated with the overlapping and skipped areas. For this reason, a and b are both chosen to be 0.5. As a result, Equation (5.27) could be rewritten as:

$$f(\mathcal{V}^{(1)}, \mathcal{V}^{(2)}) = 0.5 \cdot A_{\text{olp}}(\mathcal{V}^{(1)}, \mathcal{V}^{(2)}) + 0.5 \cdot A_{\text{skp}}(\mathcal{V}^{(1)}, \mathcal{V}^{(2)}). \tag{5.28}$$

To implement the cost function programatically, the main challenge was to generate the coverage shapes (swath lines and coverage polygons) efficiently with large number of GPS samples. For this reason, each GPS track data was splitted into data chunks of size 500. Each data chunk was processed in parallel for generating both swath lines and coverage polygons.

The routine in Algorithm 5 shows pseudocode for parallerizing the shape generation and cost computation process.

To Equation (5.27), an iterative search process was conducted on the cost function with Powell's method [130]. This method is applicable since it is a conjugate direction method that does not need the cost function to be differentiable. Each run of the optimization was started with the same set of initial guesses for $(v_x^1, v_y^1, v_x^2, v_y^2)$ as $(0, 0, 0, 0)$. The optimized parameters (i.e., the error vectors) that minimize the cost function for each set of GPS track data are listed in Table 5.4.

Algorithm 5: Cost Computation

Input:
$\mathcal{V}^1, \mathcal{V}^2$ (error vectors for each combine)
w^1, w^2 (Full swath widths for each combine)
D^1, D^2 (Partitioned data chunks for each combine)
P_f (Field boundary polygon)

Output:
c (Cost)

`// Generate coverage shapes in parallel for each data set`

$\mathcal{P}^1 \leftarrow \text{parGenShps}(\mathcal{V}^1)$;

$\mathcal{P}^2 \leftarrow \text{parGenShps}(\mathcal{V}^2)$;

`// Compute total travsered area`

$A_\text{tot} = \text{sum}(\mathcal{A}(P^1)) + \text{sum}(\mathcal{A}(P^2))$;

`// Compute actual harvested area`

$P^U = \text{union}(\mathcal{P}^1, \mathcal{P}^2))$;

$A_\text{act} = \mathcal{A}(P^U)$;

`// Compute overlapping area`

$A_\text{overlap} = A_\text{trv} - A_\text{act}$;

`// Compute skipped area`

$A_\text{skip} = \mathcal{A}(P_\text{f}) - A_\text{act}$;

`// Compute cost`

$c = 0.5 \cdot A_\text{overlap} + 0.5 \cdot A_\text{skip}$;

return c

Table 5.4. GPS error correction vector obtained from the optimization step.

Field Name	**Year**	**Model**	v_x **(meter)**	v_y **(meter)**
SL	2016	2388	-1.8	0.10
		6088	-0.48	0.04
	2019	2388	-0.89	0.08
		6088	-1.53	0.15
BR	2014	6088	-0.17	0.01
		6130	-0.05	0.01
	2017	2388	-1.42	0.11
		6088	-1.25	0.09
CH	2014	6088	-0.40	0.05
		6130	-0.11	0.02
	2018	7130	0.16	0.02
		8240	-0.76	0.02
PL	2014	6088	-0.04	0.01
		6130	-0.06	0.01
	2017	6130	0.02	0.02
		7130	-0.05	0.00
LL	2017	2388	-0.37	0.04
		6088	-0.89	0.06
	2019	2388	0.04	0.01
		6088	-0.83	0.07
RU	2017	6130	1.07	0.05
		7130	-0.83	0.07
	2019	7130	0.00	1.14
		8240	-0.41	0.05
MU	2017	6130	-0.02	0.14
		7130	-0.02	0.00
	2019	7130	0.33	0.04
		8240	-0.62	0.06

5.6.5 Effective Swath Width and Actual Harvested Area Estimation

In this step, samples within a GPS track were first corrected using the correction vector obtained from the previous step. The corrected samples were further processed to generate coverage shapes (swath lines and coverage polygons). Subsequently, these shapes allow the computation the effective swath widths as well as the actual harvested areas at each time step. This two parameters allow further computation of instantaneous metrics defined in Section 5.7.2.

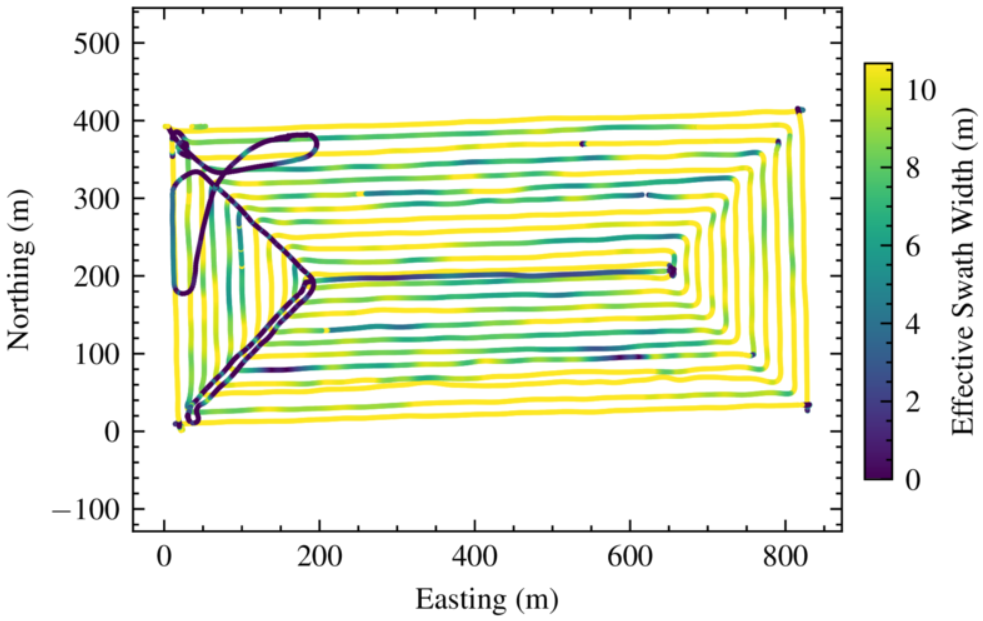


Figure 5.11. Visualization of 7130 combine header's instantaneous effective swath width in field `RU` from 2017.

Algorithm 6 estimates the effective swath width by iterating through the swath line array of a particular combine. For each iteration, it first searches for intersections between a swath line and coverage polygons from past time steps using the *findPastPolyInt* function. The function returns the indices of intersected polygons if it finds any. If such intersection exists, the full header swath width is subtracted from the intersected length. Furthermore, a similar strategy is utilized on the searching intersections from coverage polygons from the other combine. The final effective swath width at a particular index is appended to the output collection. Line 13 to 15 is in place to take account for the situation when the

effective swath width is negative; meaning that the swath line has close to complete overlaps from past coverage polygons. Since a negative swath width does not entail any physical meanings, it is set to 0 instead when this situation occurs. Figure 5.11 shows the estimated effective swath width for the 7130 combine in field `RU` during 2017. It is evident that the effective swath width varies throughout the harvest process, especially in inner harvest loops. Furthermore, a similar strategy was employed to compute the actual harvested area at each time step. The difference is that instead of computing the intersected length, the intersected area was utilized to substract from the traversed area to obtain the actual harvested area.

Algorithm 6: Effective Swath Width Estimation

Input:
- $\mathcal{P}$ (Coverage polygon collection for the combine of interest)
- $\mathcal{L}$ (Swath line collection for the combine of interest)
- w (Full swath width for the combine of interest)
- $\mathcal{P}^o$ (Coverage polygon collection for the other combine)

Output:
- W_{eff} (Effective swath width collection)

```
// First initialize with full swath width
```
$w_{\text{eff}} = w$;
```
// Iterate through each swath line
```
for idx, L in $\mathcal{L}$ **do**
```
    // Find intersections of polygons from same combine
```
    idx = findPastPolyInt($L, \mathcal{P}$);
```
    // Obtain the length of the intersected portion
```
    if len(idx) > 0 **then**
        l = union($\mathcal{P}$(idx)).intersection(L).length;
```
        // Subtract the intersected length
```
        $w_{\text{eff}} = w_{\text{eff}} - l$;
    end
```
    // Search intersections from the other combine's polygons
```
    idx = findPastPolyInt($L, \mathcal{P}^o$);
```
    // Obtain the length of the intersected portion
```
    if len(idx) > 0 **then**
        l_{o} = union($\mathcal{P}^o$(idx)).intersection(L).length;
```
        // Subtract the intersected length
```
        $w_{\text{eff}} = w_{\text{eff}} - l_{\text{o}}$;
    end
    if $w_{eff} < 0$ **then**
        $w_{\text{eff}} = 0$;
    end
```
    // Assign the estimated effective swath width to the output array
```
    W_{ff}[idx] = w_{eff};
end
return W_{e}

5.6.6 Algorithm Outputs

The algorithm was applied to all sets of GPS track data. The samples within each GPS track were classified, smoothed, and corrected using the algorithm. The results of the state classfication are visualized from Figure A.1 to Figure A.7. Preliminary observations of harvest patterns are given in the captions of each figure. Moreover, the error correction vectors for each GPS track are given in Table 5.4. Furthermore, the corrected tracks were further utilized for computations of effective swath width and actual harvested area. These two quantities serve as two key parameters for comparing overall and instantaneous harvest performances in the next section.

5.7 Harvest Performance Comparisons

The algorithm outputs from Section 5.6.6 allow further computations of various efficiency metrics. This section is dedicated to compare harvest performances using both overall and newly proposed instantaneous metrics. It is found that with instantaneous metrics, finer details as well as contextual information in regard to harvest performances could be inferred and extracted.

5.7.1 Overall Efficiency Metrics

Several overall efficiency metrics were considered. The first metric of consideration is based on time. The classification of combine operation states in Section 5.6.2 enables the counting of total harvest time, t_{h}, which is essentially the total number of `H` samples. On the other hand, the total non-productive time that includes turning, idling, and speeding could also be calculated. This leads to the computation of total field time t_{f}, which is the sum of harvest time and non-productive time. With these established quantities, individual time efficiency for each machine could be defined as:

$$\mathrm{e_t} = \frac{t_{\mathrm{h}}}{t_{\mathrm{f}}} \tag{5.29}$$

This metric is suitable for estimating the proportion between the productive and the non-productive time for a single machine. Similarly, the overall time efficiency E_{t} for a particular field can be further defined as:

$$E_{\mathrm{t}} = \frac{\sum_{c} t_{\mathrm{h}}^{(c)}}{\sum_{c} t_{\mathrm{f}}^{(c)}} \tag{5.30}$$

where $c \in \mathcal{C}$ and $\mathcal{C}$ is the collection of combine model numbers in a particular field.

The second metric, coverage efficiency, denoted as E_{g}, is computed to evaluate how well a field was traversed by both combines. The computations of A_{act} and A_{trv} in previous sections make the computation of this metric straightforward. Mathematically, the overall coverage efficiency of a particular field can be expressed as:

$$E_{\mathrm{g}} = \frac{A_{\mathrm{act}}}{A_{\mathrm{trv}}}. \tag{5.31}$$

The third metric focuses on the computation of overall capacity C_{a} for a particular field. It can be expressed as:

$$C_{\mathrm{a}} = \frac{A_{\mathrm{act}}}{T_{\mathrm{H}}}, \tag{5.32}$$

where $T_{\mathrm{H}} = \sum_{c} t_{\mathrm{h}}^{(c)}$.

Equation (5.32) provides an exact solution in comparison to the approximated solution by Equation (5.1) since both A_{act} and T_{H} have already been estimated in previous steps. This improvement could give better evaluations of harvest performance in terms of productivity.

All three efficiency metrics were computed for each machine in fields during different years. In addition, actual harvested area A_{act} and harvest average speed $s_{h,avg}$ were computed to help inferring contexts. These results are listed in Table B.2. Several observations could be made. It could be observed that for all fields except field `BR`, the C_{a}values have increased from the earlier year to the later year. However, three of the fields (`LL`, `RU`, `MU`) with increased C_{a} values have lower E_{t} and E_{g} values from earlier to later years. One possible explanation is the attachment of wider header for one of the combines (2388 in 2019, 8240 in 2019, and 8240 in 2019) in all three fields. A wider header typically restrict on how a combine would turn. This is depicted in Figure A.6 (c) and Figure A.7 (d) as the combines with wider

header often do loop turns as oppose to a right-angle turn for preventing the header from tampering unharvested wheat. Besides, it is probable that the wider headers caused more overlapping area when the combine was traversing the field, which could lead to decrease in E_{g}.

Nonetheless, although preliminary contextual information could be inferred from various overall metrics, these contexts are speculations that could not be confirmed. Moreover, fair comparison using merely overall efficiency metrics is tough to achieve. The variations in A_{act} from year to year render interpretations of the overall efficiency metrics difficult. Furthermore, it is impossible to perform either temporal or geospatial analysis or comparison and pinpoint harvest inefficiencies for various harvest patterns. These issues are addressed by computing and analyzing instantaneous efficiency metrics in the next section.

5.7.2 Instantaneous Efficiency Metrics

This section proposes three new instantaneous efficiency metrics to better evaluate harvest performance of the same dataset. The computations of these metrics also enabled both temporal and geospatial analyses of harvest performance.

The first instantaneous metric focuses on the swath utilization of combines over time. At any given time step k, the instantaneous swath efficiency is defined as:

$$u(k) = \frac{w_{\mathrm{e}}(k)}{w_{\mathrm{f}}}, \tag{5.33}$$

where $w_{\mathrm{e}}(k)$ is the effective swath width estimated in Section 5.6.5 and w_{f} is the full header width. This metric measures how much of a swath was utilized at a time step and tends to be noisy since the effective swath changes consistently throughout the harvesting process. Hence, a smoothed version of this metric, namely, average swath efficiency, is given by computing the instantaneous swath utilization average over time:

$$u_{\mathrm{avg}}(k) = \frac{\sum_{l=0}^{k} u(l)}{k}. \tag{5.34}$$

The third metric is the instantaneous version of the area capacity in Equation (5.32). It is defined as:

$$C_{\mathrm{a}}(k) = \frac{a_{\mathrm{act}}(k)}{0.89}, \tag{5.35}$$

where 0.89 is a constant for converting $a_{\mathrm{act}}(k)$ to acre per hour. Moreover, since each time step k is associated with a GPS coordinate (x, y), Equation (5.35) could be further expressed as:

$$C_{\mathrm{a}}(k) \coloneqq C_{\mathrm{a}}(x, y). \tag{5.36}$$

For geospatial comparison of harvest performance, $C_{\mathrm{a}}(x, y)$ was interpolated onto a set of uniform grid points. The reason is that the locations of the generated uniform grids are fixed, which enables one-to-one mapping for instantaneous area capacities of different years. For a given field, an extended version of the field boundary polygon is first created by extending the minimum and maximum easting and northing coordinates by an offset value of 20 meters. This offset is needed so that the resultant grid centers can cover irregular-shaped fields. Afterwards, a set of grid center points are generated to fill this new polygon with grid spacing with 1 meter. Figure 5.12 shows an example of grid points of 1 meter grid spacing using the field boundary polygon of field `RU`.

Linear interpolation using a Delaunay triangulation query [131] was selected for interpolating $C_{\mathrm{a}}(x, y)$ onto grid points. This interpolant is constructed by triangulating the input data with Delaunay triangulation, and on each triangle performing linear interpolation. Contrary to common interpolation techniques like Nearest Neighbor interpolation, if the interpolant encounters a grid point that falls outside of the convex hull triangle, the interpolated value is simply set to null. This helps eliminate unwanted interpolated value at grid points that are too far away from the original GPS coordinates.

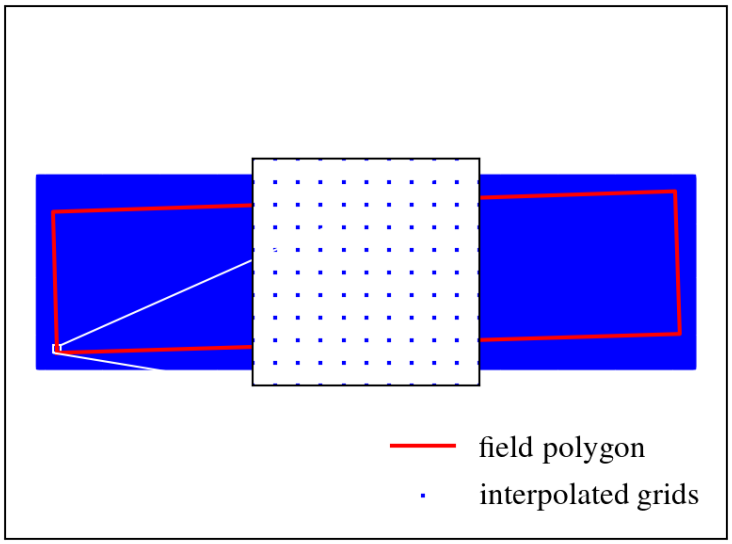


Figure 5.12. Demonstration of generated uniform grids of 1 meter spacing using field boundary polygon of field `RU`.

This interpolation was performed on instantaneous area capacity values of all GPS tracks. The interpolated values were visualized as contour maps from Figure A.15 to Figure A.21 for evaluating harvest performances.

Furthermore, difference of interpolated area capacity at each grid location could be computed for each field using. Mathematically, this could be expressed as:

$$d_{\mathrm{c}}(x_{\mathrm{i}}, y_{\mathrm{i}}) = C_{\mathrm{a}}^{(r_1)}(x_{\mathrm{i}}, y_{\mathrm{i}}) - C_{\mathrm{a}}^{(r_2)}(x_{\mathrm{i}}, y_{\mathrm{i}}), \tag{5.37}$$

where $(x_{\mathrm{i}}, y_{\mathrm{i}})$ is a grid location. In addition, r_1 and r_2 indicate an earlier year and a later year. These differences were computed for each field visualized as contour maps in plots from Figure A.29 to Figure A.35.

Using these instantaneous metrics, finer details on harvest timelines can be revealed. For example, the average swath utilizations for each combine in different years are visualized in plots (a) and (b) from Figure A.8 to Figure A.14. Several observations could be made. First, the variations of swath utilization of each machine could be easily observed. Secondly, time-related information could be easily approximated. For example, for field `CH`, two combines used two harvest sessions to finish harvesting the entire field in both years. In another

instance, field PL was harvested in two sessions in 2014 while the same field was harvested in one session in 2017.

In addition, more observations could be made using contour maps of interpolated area capacities in Figure A.15 to Figure A.21. First, machines saw improvements of area capacities with wider header attachment in later years. This is particularly pronounced in field SL, LL, RU, and MU. Moreover, different harvest patterns could also result in area capacity variations. For instance, contour loop patterns were employed for field CH in both 2014 and 2018. Nevertheless, the divide-and-conquer approach in 2019, coupled with the wider header attachment on the 8240 combine, offered significant increment in area capacities towards the center of the field. The same characteristic could be observed in field MU.

Furthermore, harvest speeds plays an important role in area capacity. Two interesting cases are field BR and field PL. Both fields were harvested with different harvest patterns in both years as shown in Figure A.4. In 2016, field BR was harvested in a mix of divide-and-conquer and contour loops while in 2019, field was a mix of divide-and-conquer and AB patterns. From the average swath utilizations plots in Figure A.11, coupled with the more uniform harvest patterns in 2017, it appears that combines in 2017 should perform better as swath utilization was consistently higher. However, as listed in Table B.2, combines were traveling faster on average during harvesting for both fields, which resulted in higher area capacities even though the harvest pattern and swath utilization appeared to be inferior.

Quantitative analyses on the interpolated instantaneous area capacities could be performed. First, empirical cumulative density functions (CDFs) of the instantaneous area capacity values could also be generated from the interpolated samples. These functions are depicted in plots from Figure A.22 to Figure A.28. Emprical CDFs from fields BR, CR, LL, RU, and MU shows the later year performed better as their CDF curve lie under the one from the earlier year. Meanwhile, the earlier year performed better than the later year in field BR. One exception is field SL. Although it is less clear which year performed better from inspecting the CDF alone, two empirical CDF curves for field SL coincide at around a cumulative probability of 0.5. The curves suggest that the combines in 2019 performed more consistent than the ones in 2016, but on average, combines in 2016 performed better.

5.8 Conclusions

This chapter presented a novel algorithm for computations of instantaneous effective swath width and actual harvested area by combine fleets during wheat harvests via GPS track smoothing and correction. This enabled exact calculations of overall efficiency metrics in related to time, harvested area, and productivity. More importantly, the instantaneous parameters also motivated both temporal and geospatial comparisons and analyses of joint harvest efforts performed by multiple combines. Harvest performance was evaluated using two new instantaneous metrics of instantaneous swath utilization and instantaneous area capacity in 7 wheat fields that occurred in two different years. The first metric, instantaneous swath utilization, was computed for visualizing harvest schedules. The second metric, instantaneous area capacity, was computed and interpolated onto uniform grids for allowing comparisons of instantaneous area capacity at grid locations from year to year. It was discovered that both the usage of wider header and more uniform harvest paths lead to productivity increment in wheat harvest.

This work has room for improvement. In terms of algorithm development, the classified combine states should be compared with ground truth data for measuring the accuracy of the classifier. Moreover, the track smoothing could be improved by using more accurate smoother such as a IMM smoother [132]. Furthermore, the optimization step could be further improved by correcting harvest paths by directions. This means that in addition to operation state classification, the GPS samples need to first be clustered by their directions. This also implies that the optimization would potentially need to expand its number of parameters depending on the number of directions. This could potentially be challenging if the field shape is irregular. Moreover, high-precision field boundary polygons, as suggested in [133], should be generated and incorporated into the algorithm for estimating the correction vectors.

For harvest performance evaluations, the instantaneous metrics could be paired with yield measurement (i.e., mass flow rate) to provide a more comprehensive assessment and comparisons of harvest performance by harvesters and years. Besides, the harvest performance

comparisons could also be expanded by incorporating additional GPS track data from other vehicles/machines (i.e., trucks, grain carts, etc.) for mining insights in harvest logistics.

6. SUMMARY

This dissertation provided four individual studies on both farm machinery data acquisition and analytics. Chapter 2 described two open-source agricultural IoT devices that were deployed in various farming tasks for collecting context-rich datasets. The collected data included CAN, GPS, and video data that totaled over 1 TB. The next three chapters focused on mining and extracting contextual knowledges within the scope of past wheat harvests. In Chapter 3, a feature dataset was created from CAN and GPS data collected from a combine harvester. This dataset was clustered using DBSCAN to identify patterns and anomalies. The semantics behind the clusters were inferred via visual analyses of the clusters on the GPS track. Moreover, in Chapter 4, a rule-based algorithm was developed to identify combine unloading events using track data from tractor-hauled grain carts and harvesters in 16 wheat harvest sessions. The results indicated that the accuracy of the identification was over 90%. Finally, it was proposed in Chapter 5 that instantaneous metrics such as instantaneous area capacity were needed to properly evaluate harvest performances. The computations of this metric as well as the comparisons were performed with a multi-year GPS track data collected from multiple combine harvesters. It was shown that by incorporating instantaneous metrics in additional to overall efficiency metrics, fine details on temporal and geospatial harvest performance could be quantified and assessed.

Nonetheless, these studies could be improved. The data collected from Ag IoT should be cleaned and labeled for potential machine learning and statistical analysis. For CAN data, more comprehensive data forensics could be performed to decipher the meanings of different data payloads using methods discussed in various past works [68]–[73], [77]. In terms of contextual knowledge mining and extraction in wheat harvest, both clustering and activity recognition could incorporate other vehicles (e.g. truck) and machines (e.g., grain carts) for inference of operational logistics. Moreover, more accurate GPS data such as Real Time Kinematic (RTK) track data should be collected to serve as the ground truth to evaluate the filtering and smoothing steps performed in Chapter 4 and Chapter 5. More importantly, yield data such as mass flow rate and moisture content should be incorporated in the analyses performed in the last three chapters. This type of data is especially important to the analyses

performed in Chapter 5 since yield could be the dominant factors in determining harvesting speed and travel patterns.

REFERENCES

[1] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *Journal of Field Robotics*, vol. 26, no. 8, pp. 651–668, Aug. 2009, ISSN: 15564959, 15564967. DOI: 10.1002/rob.20300. [Online]. Available: http://doi.wiley.com/10.1002/rob.20300.

[2] M. Gonzalez-de-Soto, L. Emmi, I. Garcia, and P. Gonzalez-de-Santos, "Reducing fuel consumption in weed and pest control using robotic tractors," *Computers and Electronics in Agriculture*, vol. 114, pp. 96–113, Jun. 2015, ISSN: 01681699. DOI: 10.1016/j.compag.2015.04.003. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0168169915001106.

[3] D. Kortenbruck, H. W. Griepentrog, and D. S. Paraforos, "Machine operation profiles generated from ISO 11783 communication data," *Computers and Electronics in Agriculture*, vol. 140, pp. 227–236, Aug. 2017, ISSN: 01681699. DOI: 10.1016/j.compag.2017.05.039. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0168169916311589.

[4] S. E. Marx, J. D. Luck, R. M. Hoy, S. K. Pitla, E. E. Blankenship, and M. J. Darr, "Validation of machine CAN bus J1939 fuel rate accuracy using Nebraska Tractor Test Laboratory fuel rate data," *Computers and Electronics in Agriculture*, vol. 118, pp. 179–185, Oct. 2015, ISSN: 01681699. DOI: 10.1016/j.compag.2015.08.032. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0168169915002586.

[5] J. Gomez-Gil, R. Ruiz-Gonzalez, S. Alonso-Garcia, and F. Gomez-Gil, "A Kalman Filter Implementation for Precision Improvement in Low-Cost GPS Positioning of Tractors," *Sensors*, vol. 13, no. 11, pp. 15 307–15 323, Nov. 2013, ISSN: 1424-8220. DOI: 10.3390/s131115307. [Online]. Available: http://www.mdpi.com/1424-8220/13/11/15307.

[6] M. Finner and R. Straub, *Farm Machinery Fundamentals*. American Publishing Company, 1985, ISBN: 9780895340160. [Online]. Available: https://books.google.com/books?id=FKAdAQAAMAAJ.

[7] *Automated agriculture for the 21st century : proceedings of the 1991 symposium, 16-17 December 1991, Chicago, Illinois.* eng, ser. ASAE publication ; 11-91. St. Joseph, Mich., USA: American Society of Agricultural Engineers, 1991, ISBN: 0929355210.

[8] "Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 1: General standard for mobile data communication," International Organization for Standardization, Standard, Dec. 2017.

[9] "Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling," International Organization for Standardization, Standard, Dec. 2015.

[10] W. Romans, B. Poore, and J. Mutziger, "Advanced instrumentation for agricultural equipment," *IEEE Instrumentation & Measurement Magazine*, vol. 3, no. 1, pp. 26–29, Mar. 2000, ISSN: 10946969. DOI: 10.1109/5289.823820. [Online]. Available: http://ieeexplore.ieee.org/document/823820/.

[11] T. Torii, "Research in autonomous agriculture vehicles in Japan," *Computers and Electronics in Agriculture*, vol. 25, no. 1-2, pp. 133–153, Jan. 2000, ISSN: 01681699. DOI: 10.1016/S0168-1699(99)00060-5. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0168169999000605.

[12] R. Eaton, J. Katupitiya, K. W. Siew, and B. Howarth, "Autonomous Farming: Modeling and Control of Agricultural Machinery in a Unified Framework," in *2008 15th International Conference on Mechatronics and Machine Vision in Practice*, Auckland, New Zealand: IEEE, Dec. 2008, pp. 499–504, ISBN: 978-1-4244-3779-5. DOI: 10.1109/MMVIP.2008.4749583. [Online]. Available: http://ieeexplore.ieee.org/document/4749583/.

[13] K. Steen, P. Christiansen, H. Karstoft, and R. Jørgensen, "Using Deep Learning to Challenge Safety Standard for Highly Autonomous Machines in Agriculture," *Journal of Imaging*, vol. 2, no. 1, p. 6, Feb. 2016, ISSN: 2313-433X. DOI: 10.3390/jimaging2010006. [Online]. Available: http://www.mdpi.com/2313-433X/2/1/6.

[14] T. Duckett, S. Pearson, S. Blackmore, B. Grieve, W.-H. Chen, G. Cielniak, J. Cleaversmith, J. Dai, S. Davis, C. Fox, P. From, I. Georgilas, R. Gill, I. Gould, M. Hanheide, A. Hunter, F. Iida, L. Mihalyova, S. Nefti-Meziani, G. Neumann, P. Paoletti, T. Pridmore, D. Ross, M. Smith, M. Stoelen, M. Swainson, S. Wane, P. Wilson, I. Wright, and G.-Z. Yang, "Agricultural Robotics: The Future of Robotic Agriculture," *arXiv:1806.06762 [cs]*, Aug. 2018, arXiv: 1806.06762. [Online]. Available: http://arxiv.org/abs/1806.06762.

[15] S. Fountas, N. Mylonas, I. Malounas, E. Rodias, C. Hellmann Santos, and E. Pekkeriet, "Agricultural Robotics for Field Operations," *Sensors*, vol. 20, no. 9, p. 2672, May 2020, ISSN: 1424-8220. DOI: 10.3390/s20092672. [Online]. Available: https://www.mdpi.com/1424-8220/20/9/2672.

[16] M. Kiran, P. Murphy, I. Monga, J. Dugan, and S. S. Baveja, "Lambda architecture for cost-effective batch and speed big data processing," in *2015 IEEE International Conference on Big Data (Big Data)*, 2015, pp. 2785–2792. DOI: 10.1109/BigData.2015.7364082.

[17] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context Aware Computing for The Internet of Things: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 414–454, 2014, ISSN: 1553-877X. DOI: 10.1109/SURV.2013.042313.00197. [Online]. Available: http://ieeexplore.ieee.org/document/6512846/.

[18] J. Conesa-Muñoz, J. M. Bengochea-Guevara, D. Andujar, and A. Ribeiro, "Route planning for agricultural tasks: A general approach for fleets of autonomous vehicles in site-specific herbicide applications," *Computers and Electronics in Agriculture*, vol. 127, pp. 204–220, Sep. 2016, ISSN: 01681699. DOI: 10.1016/j.compag.2016.06.012. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0168169916303908.

[19] M. Spekken and S. de Bruin, "Optimized routing on agricultural fields by minimizing maneuvering and servicing time," *Precision Agriculture*, vol. 14, no. 2, pp. 224–244, Apr. 2013, ISSN: 1385-2256, 1573-1618. DOI: 10.1007/s11119-012-9290-5. [Online]. Available: http://link.springer.com/10.1007/s11119-012-9290-5.

[20] Norbert Diekhans, Jochen Huster, Andreas Brunnert, and Lars-Peter Meyer Zu Helligen, "Method for creating a route plan for agricultural machine systems," US8170785B2, May 2012. [Online]. Available: https://patents.google.com/patent/US8170785B2/en.

[21] Norbert Diekhans and Andreas Brunnert, "Route planning system for agricultural working machines," US7756624B2, Jul. 2010. [Online]. Available: https://patents.google.com/patent/US7756624B2/en.

[22] Y. Wang, A. D. Balmos, A. W. Layton, S. Noel, A. Ault, J. V. Krogmeier, and D. R. Buckmaster, "An Open-Source Infrastructure for Real-Time Automatic Agricultural Machine Data Processing," in *2017 ASABE Annual International Meeting*, American Society of Agricultural and Biological Engineers, 2017, p. 1.

[23] Y. Wang, H. Liu, J. Krogmeier, A. Reibman, and D. Buckmaster, "ISOBlue HD: An open-source platform for collecting context-rich agricultural machinery datasets," *Sensors*, vol. 20, no. 20, p. 5768, Oct. 12, 2020, ISSN: 1424-8220. DOI: 10.3390/s20205768. [Online]. Available: https://www.mdpi.com/1424-8220/20/20/5768.

[24] Y. Wang, A. D. Balmos, D. R. Buckmaster, and J. V. Krogmeier, "Data-Driven Agricultural Machinery Activity Anomaly Detection and Classification," in *14th International Conference on Precision Agriculture*, International Society of Precision Agriculture, 2018, p. 1.

[25] Y. Wang, Y. Zhang, J. V. Krogmeier, and D. R. Buckmaster, "Combine Harvester Unloading Event Inference Using GPS Data," in *2019 ASABE Annual International Meeting*, American Society of Agricultural and Biological Engineers, 2019, p. 1. DOI: 10.13031/aim.201901286.

[26] Truck Bus Control and Communications Network Committee, "Off-Board Diagnostic Connector," SAE International, Tech. Rep. [Online]. Available: https://bit.ly/3uI71rz.

[27] M. Jensen, "Diagnostic tool concepts for iso11783 (isobus)," *SAE Transactions*, vol. 113, pp. 415–419, 2004, ISSN: 0096736X, 25771531. [Online]. Available: http://www.jstor.org/stable/44718838.

[28] Deere & Company, *MTG 4G LTE Complete Specification*, 2021. [Online]. Available: https://www.deere.com/assets/pdfs/common/industries/electronic-solutions/mtg-4g-lte.pdf.

[29] Climate Corporation, *Data Connectivity*, 2021. [Online]. Available: https://climate.com/features/data-connectivity.

[30] Farmobile LLC, *PUC Specification*, 2021. [Online]. Available: https://farmobile.dozuki.com/c/PUC4.

[31] J. Backman, R. Linkolehto, M. Koistinen, J. Nikander, A. Ronkainen, J. Kaivosoja, P. Suomi, and L. Pesonen, "Cropinfra research data collection platform for ISO 11783 compatible and retrofit farm equipment," *Computers and Electronics in Agriculture*, vol. 166, p. 105 008, Nov. 2019, ISSN: 01681699. DOI: 10.1016/j.compag.2019.105008. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0168169918317381.

[32] M. Kragh, P. Christiansen, M. Laursen, M. Larsen, K. Steen, O. Green, H. Karstoft, and R. Jørgensen, "FieldSAFE: Dataset for Obstacle Detection in Agriculture," *Sensors*, vol. 17, no. 11, p. 2579, Nov. 2017, ISSN: 1424-8220. DOI: 10.3390/s17112579. [Online]. Available: http://www.mdpi.com/1424-8220/17/11/2579.

[33] Matthew Darr, "CAN Bus Technology Enables Advanced Machinery Management," *Resource Magazine*, vol. 19, no. 5, pp. 10–11, 2012, Journal Abbreviation: Resource Magazine Place: St. Joseph, MI Publisher: ASABE, ISSN: 1076-3333. DOI: 10.13031/2013.42312. [Online]. Available: http://elibrary.asabe.org/abstract.asp?aid=42312&t=11.

[34] A. W. Layton, A. D. Balmos, S. Sabpisal, A. C. Ault, J. V. Krogmeier, and B. D. R, "ISOBlue: An Open Source Project to Bring Agricultural Machinery Data into the Cloud," in *2014 ASABE Annual International Meeting*, American Society of Agricultural and Biological Engineers, Jul. 2014, pp. 1–8. DOI: 10.13031/aim.20141929380.

[35] A. Fite, A. McGuan, and T. Letz, *Tractor Hacking*, https://tractorhacking.github.io/, Accessed: 2020-08-03.

[36] Deere & Company, *Develop with Deere*, 2021. [Online]. Available: https://developer-portal.deere.com/.

[37] B. R. Covington, "Assessment of utilization and downtime of a commercial level multi-pass corn stover harvesting systems," Pages: 4250803, Master of Science, Iowa State University, Digital Repository, Ames, 2013. DOI: 10.31274/etd-180810-3156. [Online]. Available: https://lib.dr.iastate.edu/etd/13154/.

[38] J. C. Askey, "Automated logistic processing and downtime analysis of commercial level multi-pass corn stover harvesting systems," Pages: 5777390, Master of Science, Iowa State University, Digital Repository, Ames, 2014. DOI: 10.31274/etd-180810-1337. [Online]. Available: https://lib.dr.iastate.edu/etd/13697/.

[39] Apache Software Foundation, *Kafka - A distributed streaming platform*, https://github.com/apache/kafka, Accessed: 2020-08-06, 2020.

[40] Toradex Inc., *NXP® i.MX 6 Computer on Module - Apalis iMX6*, https://www.toradex.com/computer-on-modules/apalis-arm-family/nxp-freescale-imx-6, Accessed: 2020-08-05.

[41] Navisys Technology, *Navisys GR-701, u-blox7 Ultra-High Performance GPS Mouse Receiver*, https://bit.ly/34DrraK, Accessed: 2020-08-05.

[42] Analog Device, *Signal and Power Isolated CAN Transceiver with Integrated Isolated DC-to-DC Converter*, https://www.analog.com/media/en/technical-documentation/data-sheets/ADM3053.pdf, Accessed: 2020-08-05.

[43] Telit, *LE910 Cat. 1 Series*, https://www.telit.com/le910-cat-1-le910b1/, Accessed: 2020-08-05.

[44] K. Raj, *The Ångström Distribution*, https://github.com/Angstrom-distribution, Accessed: 2020-08-05.

[45] Y. Wang, *ISOBlue 2.0 Software GitHub*, https://github.com/ISOBlue/isoblue2/tree/master, Accessed: 2020-08-05.

[46] G. Kroah-Hartman and K. Sievers, *udev - Dynamic Device Management*, https://www.freedesktop.org/software/systemd/man/udev.html, Version 231, 2020.

[47] T. Remco and D. Brashear, *gpsd - a GPS Service Daemon*, https://gpsd.gitlab.io/gpsd/index.html, Version 3.16, 2020.

[48] L. Poettering, K. Sievers, H. Hoyer, D. Mack, T. Gundersen, and D. Herrmann, *systemd - System and Service Manager*, https://www.freedesktop.org/wiki/Software/systemd/, Version 234, 2020.

[49] The OpenBSD Project, *openssh*, https://www.openssh.com/, Version 7.5p1, 2020.

[50] R. Curnow, *chrony*, https://chrony.tuxfamily.org/documentation.html, Version 2.4, 2020.

[51] M. Kleine-Budde and O. Hartkopp, *Socketcan - linux-can / socketcan user space applications*, https://github.com/linux-can/, Accessed: 2020-08-06, 2020.

[52] S. Qian, G. Wu, J. Huang, and T. Das, "Benchmarking modern distributed streaming platforms," in *2016 IEEE International Conference on Industrial Technology (ICIT)*, Taipei, Taiwan: IEEE, Mar. 2016, pp. 592–598, ISBN: 978-1-4673-8075-1. DOI: 10.1109/ICIT.2016.7474816. [Online]. Available: http://ieeexplore.ieee.org/document/7474816/.

[53] A. Akanbi and M. Masinde, "A Distributed Stream Processing Middleware Framework for Real-Time Analysis of Heterogeneous Data on Big Data Platform: Case of Environmental Monitoring," *Sensors*, vol. 20, no. 11, p. 3166, Jun. 2020, ISSN: 1424-8220. DOI: 10.3390/s20113166. [Online]. Available: https://www.mdpi.com/1424-8220/20/11/3166.

[54] Apache Software Foundation, *Kafka Software Archive*, https://bit.ly/3vGw6Vb, Version: 0.10.1.0, 2020.

[55] M. Edenhill, *Librdkafka - the apache kafka c/c++ library*, https://github.com/edenhill/librdkafka, Version: 0.9.3, 2020.

[56] D. Powers, *Kafka-python - a python client for apache kafka*, https://github.com/dpkp/kafka-python, Version: 1.3.1, 2020.

[57] Apache Software Foundation, *Avro C*, https://avro.apache.org/docs/current/api/c/index.html, Version: 1.8.1, 2016.

[58] Apache Software Foundation, *Apache avro getting started (python)*, https://avro.apache.org/docs/current/gettingstartedpython.html, Accessed: 2020-08-05, 2012.

[59] Moe, *gps3 - python 2.7 to 3.5 interface to gpsd*, Version: 0.33.3, 2016.

[60] Ubiquiti Inc., *Ubiquiti Network UniFi G3 Video Camera*, https://www.ui.com/unifi-video/unifi-video-camera-g3/, Accessed: 2020-08-05.

[61] The Internet Society, *Real Time Streaming Protocol (RTSP)*. [Online]. Available: https://tools.ietf.org/html/rfc2326.

[62] Veracity, *CAMSWITCH 8 Mobile*, https://www.veracityglobal.com/products/networked-video-integration-devices/camswitch-mobile.aspx, Accessed: 2020-08-05.

[63] "IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Data Terminal Equipment (DTE) Power Via Media Dependent Interface (MDI)," 2003. [Online]. Available: https://bit.ly/3fYVw9W.

[64] S. Kelley, *dnsmasq*, http://www.thekelleys.org.uk/dnsmasq/doc.html, Version 2.78, 2020.

[65] F. Bellard, *Ffmpeg - a complete, cross-platform solution to record, convert and stream audio and video*, https://www.ffmpeg.org/, Version: 3.3.3, 2020.

[66] CNH Industrial America LLC, *Axial-flow combines*, https://www.caseih.com/northamerica/en-us/products/harvesting/axial-flow-combines, Accessed: 2020-08-05, 2020.

[67] A. Masullo and L. Dalgarno, *Multiple videos labelling tool*, https://github.com/ale152/muvilab, Version: master, 2020.

[68] M. D. Pesé, T. Stacer, C. A. Campos, E. Newberry, D. Chen, and K. G. Shin, "Librecan: Automated can message translator," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19, London, United Kingdom: ACM, 2019, pp. 2283–2300, ISBN: 978-1-4503-6747-9. DOI: 10.1145/3319535.3363190. [Online]. Available: http://doi.acm.org/10.1145/3319535.3363190.

[69] M. Zago, S. Longari, A. Tricarico, M. Gil Pérez, M. Carminati, G. Martínez Pérez, and S. Zanero, "Recan - dataset for reverse engineering of controller area networks," *Data in Brief*, vol. 29, p. 105 149, 2020. DOI: 10.1016/j.dib.2020.105149.

[70] M. Markovitz and A. Wool, "Field classification, modeling and anomaly detection in unknown CAN bus networks," *Vehicular Communications*, vol. 9, pp. 43–52, Jul. 2017, ISSN: 22142096. DOI: 10.1016/j.vehcom.2017.02.005. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2214209616300869.

[71] M. Verma, R. Bridges, and S. Hollifield, "ACTT: Automotive CAN Tokenization and Translation," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA: IEEE, Dec. 2018, pp. 278–283, ISBN: 978-1-72811-360-9. DOI: 10.1109/CSCI46756.2018.00061. [Online]. Available: https://ieeexplore.ieee.org/document/8947694/.

[72] B. C. Nolan, S. Graham, B. Mullins, and C. S. Kabban, "Unsupervised Time Series Extraction from Controller Area Network Payloads," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Chicago, IL, USA: IEEE, Aug. 2018, pp. 1–5, ISBN: 978-1-5386-6358-5. DOI: 10.1109/VTCFall.2018.8690615. [Online]. Available: https://ieeexplore.ieee.org/document/8690615/.

[73] M. Marchetti and D. Stabili, "READ: Reverse Engineering of Automotive Data Frames," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1083–1097, Apr. 2019, ISSN: 1556-6013, 1556-6021. DOI: 10.1109/TIFS.2018.2870826. [Online]. Available: https://ieeexplore.ieee.org/document/8466914/.

[74] E. Yaacoub and M.-S. Alouini, "A Key 6G Challenge and Opportunity—Connecting the Base of the Pyramid: A Survey on Rural Connectivity," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 533–582, Apr. 2020, ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2020.2976703. [Online]. Available: https://ieeexplore.ieee.org/document/9042251/.

[75] H. Liu, A. R. Reibman, A. C. Ault, and J. V. Krogmeier, "Video Classification of Farming Activities with Motion-Adaptive Feature Sampling," in *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, Vancouver, BC: IEEE, Aug. 2018, pp. 1–6, ISBN: 978-1-5386-6070-6. DOI: 10.1109/MMSP.2018.8547117. [Online]. Available: https://ieeexplore.ieee.org/document/8547117/.

[76] H. Liu, A. R. Reibman, A. C. Ault, and J. V. Krogmeier, "Video-Based Prediction for Header-Height Control of a Combine Harvester," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, San Jose, CA, USA: IEEE, Mar. 2019, pp. 310–315, ISBN: 978-1-72811-198-8. DOI: 10.1109/MIPR.2019.00062. [Online]. Available: https://ieeexplore.ieee.org/document/8695327/.

[77] T. Huybrechts, Y. Vanommeslaeghe, D. Blontrock, G. Van Barel, and P. Hellinckx, "Automatic Reverse Engineering of CAN Bus Data Using Machine Learning Techniques," in *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*, F. Xhafa, S. Caballé, and L. Barolli, Eds., vol. 13, Series Title: Lecture Notes on Data Engineering and Communications Technologies, Cham: Springer International Publishing, 2018, pp. 751–761. DOI: https://bit.ly/3g4cSSN. [Online]. Available: https://bit.ly/3cdA0wY.

[78] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," KDD'96, pp. 226–231, 1996.

[79] Y. Zhang, A. Balmos, J. V. Krogmeier, and D. Buckmaster, "Dynamic High-Precision Field Shape Generation via Combine GPS Tracks," American Society of Agricultural and Biological Engineers, 2017. DOI: 10.13031/aim.201700809.

[80] R. Ranjith, J. J. Athanesious, and V. Vaidehi, "Anomaly detection using DBSCAN clustering technique for traffic video surveillance," IEEE, Dec. 2015, pp. 1–6, ISBN: 978-1-5090-1933-5. DOI: 10.1109/ICoAC.2015.7562795. [Online]. Available: http://ieeexplore.ieee.org/document/7562795/.

[81] M. Celik, F. Dadaser-Celik, and A. S. Dokuz, "Anomaly detection in temperature data using DBSCAN algorithm," IEEE, Jun. 2011, pp. 91–95, ISBN: 978-1-61284-919-5. DOI: 10.1109/INISTA.2011.5946052. [Online]. Available: http://ieeexplore.ieee.org/document/5946052/.

[82] Tran Manh Thang and Juntae Kim, "The Anomaly Detection by Using DBSCAN Clustering with Multiple Parameters," IEEE, Apr. 2011, pp. 1–5, ISBN: 978-1-4244-9222-0. DOI: 10.1109/ICISA.2011.5772437. [Online]. Available: http://ieeexplore.ieee.org/document/5772437/.

[83] CASE IH, *CASE IH Axial-Flow® 6130 Specifications*, Jun. 2017. [Online]. Available: https://bit.ly/3uPqpmy.

[84] J. De Baerdemaeker and W. Saeys, "Advanced control of combine harvesters," *IFAC Proceedings Volumes*, vol. 46, no. 18, pp. 1–5, 2013, 4th IFAC Conference on Modelling and Control in Agriculture, Horticulture and Post Harvest Industry, ISSN: 1474-6670. DOI: https://doi.org/10.3182/20130828-2-SF-3019.00069. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667015349491.

[85] A. Nagpal, A. Jatain, and D. Gaur, "Review based on data clustering algorithms," in *2013 IEEE CONFERENCE ON INFORMATION AND COMMUNICATION TECHNOLOGIES*, Thuckalay, Tamil Nadu, India: IEEE, Apr. 2013, pp. 298–303, ISBN: 978-1-4673-5758-6 978-1-4673-5759-3 978-1-4673-5757-9. DOI: 10.1109/CICT.2013.6558109. [Online]. Available: http://ieeexplore.ieee.org/document/6558109/.

[86] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady, "DBSCAN: Past, present and future," in *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, Feb. 2014, pp. 232–238. DOI: 10.1109/ICADIWT.2014.6814687.

[87] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Transactions on Database Systems*, vol. 42, no. 3, pp. 1–21, Aug. 24, 2017, ISSN: 0362-5915, 1557-4644. DOI: 10.1145/3068335. [Online]. Available: https://dl.acm.org/doi/10.1145/3068335.

[88] A. E. Zambelli, "A data-driven approach to estimating the number of clusters in hierarchical clustering," *F1000Research*, vol. 5, p. 2809, Dec. 1, 2016, ISSN: 2046-1402. DOI: 10.12688/f1000research.10103.1. [Online]. Available: https://f1000research.com/articles/5-2809/v1.

[89] S. Sun, C. Li, A. H. Paterson, Y. Jiang, R. Xu, J. S. Robertson, J. L. Snider, and P. W. Chee, "In-field high throughput phenotyping and cotton plant growth analysis using LiDAR," *Frontiers in Plant Science*, vol. 9, p. 16, Jan. 22, 2018, ISSN: 1664-462X. DOI: 10.3389/fpls.2018.00016. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fpls.2018.00016/full.

[90] S. Das Choudhury, A. Samal, and T. Awada, "Leveraging image analysis for high-throughput plant phenotyping," *Frontiers in Plant Science*, vol. 10, p. 508, Apr. 24, 2019, ISSN: 1664-462X. DOI: 10.3389/fpls.2019.00508. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpls.2019.00508/full.

[91] N. Wang, N. Zhang, and M. Wang, "Wireless sensors in agriculture and food industry—Recent development and future perspective," *Computers and Electronics in Agriculture*, vol. 50, no. 1, pp. 1–14, Jan. 2006, ISSN: 01681699. DOI: 10.1016/j.compag.2005.09.003. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0168169905001572.

[92] Randal K. Taylor, Mark D. Schrock, and Scott A. Staggenborg, "Extracting Machinery Management Information from GPS Data," American Society of Agricultural and Biological Engineers, 2002. DOI: 10.13031/2013.13933.

[93] M. Pala, N. Eraghi, F. López-Colino, A. Sanchez, A. de Castro, and J. Garrido, "HCTNav: A Path Planning Algorithm for Low-Cost Autonomous Robot Navigation in Indoor Environments," *ISPRS International Journal of Geo-Information*, vol. 2, no. 3, pp. 729–748, Aug. 2013, ISSN: 2220-9964. DOI: 10.3390/ijgi2030729. [Online]. Available: http://www.mdpi.com/2220-9964/2/3/729.

[94] Y. Zhang, J. V. Krogmeier, A. Ault, and D. Buckmaster, "APT3: Automated product traceability trees generated from GPS tracks," *Transactions of the ASABE*, vol. 63, no. 3, pp. 571–582, 2020, ISSN: 2151-0040. DOI: 10.13031/trans.13384.

[95] Y. Wang, A. D. Balmos, J. V. Krogmeier, and D. R. Buckmaster, "An Optimized Adaptive Kalman Filtering Algorithm for Agricultural GPS Data Processing," in *2018 ASABE Annual International Meeting*, American Society of Agricultural and Biological Engineers, 2018, p. 1. DOI: 10.13031/aim.201800081.

[96] Huisman, W., "Optimum cereal combine harvester operation by means of automatic machine and threshing speed control," PhD thesis, Wageningen University, 1983. [Online]. Available: http://library.wur.nl/WebQuery/wurpubs/77529.

[97] Y. Zhang and A. Balmos, *CKT Android App*, https://github.com/oats-center/android-logger, 2017.

[98] CASE IH, *CASE IH Axial-Flow® 8240 combine Specifications*, Jun. 2017. [Online]. Available: https://www.caseih.com/northamerica/en-us/products/harvesting/axial-flow-combines/axial-flow-8240.

[99] CASE IH, *CASE IH Axial-Flow® 7130 combine Specifications*, Jun. 2017. [Online]. Available: https://bit.ly/2SMYkiu.

[100] CASE IH, *CASE IH Magnum® 290 tractor Specifications*, 2018. [Online]. Available: https://www.ritchiespecs.com/model/case-ih-magnum-290-4wd-tractor.

[101] Crustbuster Speed King Inc., *Grain Cart 1075 Bu Owner's Manual*, 2010. [Online]. Available: https://bit.ly/3wS87T3.

[102] MATLAB, *version 9.4.0 (R2018a)*. Natick, Massachusetts: The MathWorks Inc., 2018.

[103] X. Li and Y. Bar-Shalom, "Design of an interacting multiple model algorithm for air traffic control tracking," *IEEE Transactions on Control Systems Technology*, vol. 1, no. 3, pp. 186–194, 1993, ISSN: 10636536. DOI: 10.1109/87.251886. [Online]. Available: http://ieeexplore.ieee.org/document/251886/.

[104] N. Cui, L. Hong, and J. R. Layne, "A comparison of nonlinear filtering approaches with an application to ground target tracking," *Signal Processing*, vol. 85, no. 8, pp. 1469–1492, Aug. 2005, ISSN: 01651684. DOI: 10.1016/j.sigpro.2005.01.010. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0165168405000307.

[105] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation*. New York: Wiley, 2001, ISBN: 978-0-471-41655-5.

[106] American Society of Agricultural and Biological Engineers, "ASAE EP496.3 (R2020)," Tech. Rep., Feb. 2006. [Online]. Available: https://elibrary.asabe.org/abstract.asp?aid=47748.

[107] American Society of Agricultural and Biological Engineers, *ASAE D497.7 (R2020)*, Mar. 2011. [Online]. Available: https://elibrary.asabe.org/abstract.asp?aid=36431.

[108] S. Han, S. M. Schneider, S. L. Rawlins, and R. G. Evans, "A Bitmap Method for Determining Effective Combine Cut Width in Yield Mapping," *Transactions of the ASAE*, vol. 40, no. 2, pp. 485–490, 1997, ISSN: 2151-0059. DOI: 10.13031/2013.21267.

[109] S. T. Drummond, C. W. Fraisse, and K. A. Sudduth, "Combine Harvest Area Determination By Vector Processing of GPS Position Data," *Transactions of the ASAE*, vol. 42, no. 5, pp. 1221–1228, 1999, ISSN: 2151-0059. DOI: 10.13031/2013.13287.

[110] C. Zhao, W. Huang, L. Chen, Z. Meng, Y. Wang, and F. Xu, "A harvest area measurement system based on ultrasonic sensors and DGPS for yield map correction," *Precision Agriculture*, vol. 11, no. 2, pp. 163–180, Apr. 2010, ISSN: 1385-2256, 1573-1618. DOI: 10.1007/s11119-010-9157-6. [Online]. Available: http://link.springer.com/10.1007/s11119-010-9157-6.

[111] A. Heiß, D. Paraforos, and H. Griepentrog, "Determination of Cultivated Area, Field Boundary and Overlapping for A Plowing Operation Using ISO 11783 Communication and D-GNSS Position Data," *Agriculture*, vol. 9, no. 2, p. 38, Feb. 2019, ISSN: 2077-0472. DOI: 10.3390/agriculture9020038. [Online]. Available: http://www.mdpi.com/2077-0472/9/2/38.

[112] R. D. Grisso, P. J. Jasa, and D. E. Rolofson, "Analysis of Traffic Patterns And Yield Monitor Data For Field Efficiency Determination," *Applied Engineering in Agriculture*, vol. 18, no. 2, 2002, ISSN: 1943-7838. DOI: 10.13031/2013.7782.

[113] R. D. Grisso, M. F. Kocher, V. I. Adamchuk, P. J. Jasa, and M. A. Schroeder, "Field Efficiency Determination Using Traffic Pattern Indices," *Applied Engineering in Agriculture*, vol. 20, no. 5, pp. 563–572, 2004, ISSN: 1943-7838. DOI: 10.13031/2013.17456. [Online]. Available: http://elibrary.asabe.org/abstract.asp??JID=3&AID=17456&CID=aeaj2004&v=20&i=5&T=1.

[114] R. Shamshiri, R. Ehsani, J. M. Maja, and F. M. Roka, "Determining Machine Efficiency Parameters for a Citrus Canopy Shaker Using Yield Monitor Data," *Applied Engineering in Agriculture*, vol. 29, no. 1, pp. 33–41, 2013, ISSN: 1943-7838. DOI: 10.13031/2013.42526.

[115] F. Pezzi and R. Martelli, "Technical and Economic Evaluation of Mechanical Grape Harvesting in Flat and Hill Vineyards," *Transactions of the ASABE*, pp. 297–303, Apr. 2015, ISSN: 21510032, 21510040. DOI: 10.13031/trans.58.10997.

[116] R. ". Grisso, E. G. Webb, and J. S. Cundiff, "In-field performance of biomass balers," *AgriEngineering*, vol. 2, no. 4, pp. 568–580, Dec. 4, 2020, ISSN: 2624-7402. DOI: 10.3390/agriengineering2040038. [Online]. Available: https://www.mdpi.com/2624-7402/2/4/38.

[117] L. M. Griffel, V. Vazhnik, D. S. Hartley, J. K. Hansen, and M. Roni, "Agricultural field shape descriptors as predictors of field efficiency for perennial grass harvesting: An empirical proof," *Computers and Electronics in Agriculture*, vol. 168, p. 105 088, Jan. 2020, ISSN: 01681699. DOI: 10.1016/j.compag.2019.105088.

[118] S. Pitla, N. Lin, S. Shearer, and L. J.D., "Use of Controller Area Network (CAN) Data To Determine Field Efficiencies of Agricultural Machinery," *Applied Engineering in Agriculture*, pp. 829–838, Dec. 2014, ISSN: 08838542, 19437838. DOI: 10.13031/aea.30.10618.

[119] K. Zhou, D. Bochtis, A. L. Jensen, D. Kateris, and C. G. Sørensen, "Introduction of a new index of field operations efficiency," *Applied Sciences*, vol. 10, no. 1, p. 329, Jan. 1, 2020, ISSN: 2076-3417. DOI: 10.3390/app10010329. [Online]. Available: https://www.mdpi.com/2076-3417/10/1/329.

[120] Y. Zhang and J. Krogmeier, *Combine kart truck gps data archive*, May 2020. DOI: doi:/10.4231/GMH9-8X88. [Online]. Available: https://purr.purdue.edu/publications/3083/2.

[121] *Google maps for custom field boundaries for colorado wheat harvest*, May 2021. [Online]. Available: https://bit.ly/3gS5ydl.

[122] M. Karaim, M. Elsheikh, and A. Noureldin, "GNSS error sources," in *Multifunctional Operation and Application of GPS*, R. B. Rustamov and A. M. Hashimov, Eds., InTech, May 30, 2018, ISBN: 978-1-78923-214-1 978-1-78923-215-8. DOI: 10.5772/intechopen.75493. [Online]. Available: http://www.intechopen.com/books/multifunctional-operation-and-application-of-gps/gnss-error-sources.

[123] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial–temporal data," *Data & Knowledge Engineering*, vol. 60, no. 1, pp. 208–221, 2007, Intelligent Data Mining, ISSN: 0169-023X. DOI: https://doi.org/10.1016/j.datak.2006.01.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169023X06000218.

[124] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, Aug. 1965, ISSN: 0001-1452, 1533-385X. DOI: 10.2514/3.3166. [Online]. Available: https://arc.aiaa.org/doi/10.2514/3.3166.

[125] E. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000, pp. 153–158. DOI: 10.1109/ASSPCC.2000.882463.

[126] S. Julier, J. Uhlmann, and H. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control*, vol. 45, no. 3, pp. 477–482, 2000. DOI: 10.1109/9.847726.

[127] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004. DOI: 10.1109/JPROC.2003.823141.

[128] R. van der Merwe, E. Wan, and S. Julier, "Sigma-Point Kalman Filters for Nonlinear Estimation and Sensor-Fusion: Applications to Integrated Navigation," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Aug. 16, 2004, ISBN: 978-1-62410-073-4. DOI: 10.2514/6.2004-5120. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2004-5120.

[129] R. A. Singer, "Estimating optimal tracking filter performance for manned maneuvering targets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-6, no. 4, pp. 473–483, 1970. DOI: 10.1109/TAES.1970.310128.

[130] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, vol. 7, no. 2, pp. 155–162, Jan. 1964, ISSN: 0010-4620. DOI: 10.1093/comjnl/7.2.155. [Online]. Available: https://doi.org/10.1093/comjnl/7.2.155.

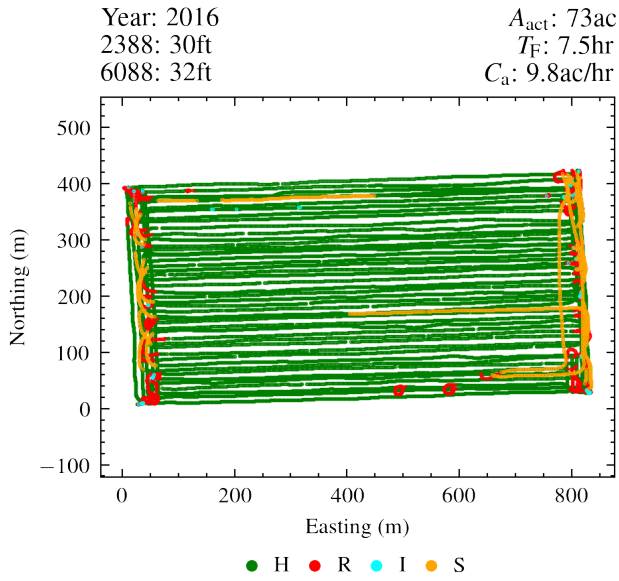
[131] E. Gross and D. Wagner, "KD trees and Delaunay-based linear interpolation for function learning: a comparison to neural networks with error backpropagation," *IEEE Transactions on Control Systems Technology*, vol. 4, no. 6, pp. 649–653, 1996. DOI: 10.1109/87.541694.

[132] N. Nadarajah, R. Tharmarasa, M. McDonald, and T. Kirubarajan, "IMM Forward Filtering and Backward Smoothing for Maneuvering Target Tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 3, pp. 2673–2678, 2012. DOI: 10.1109/TAES.2012.6237617.

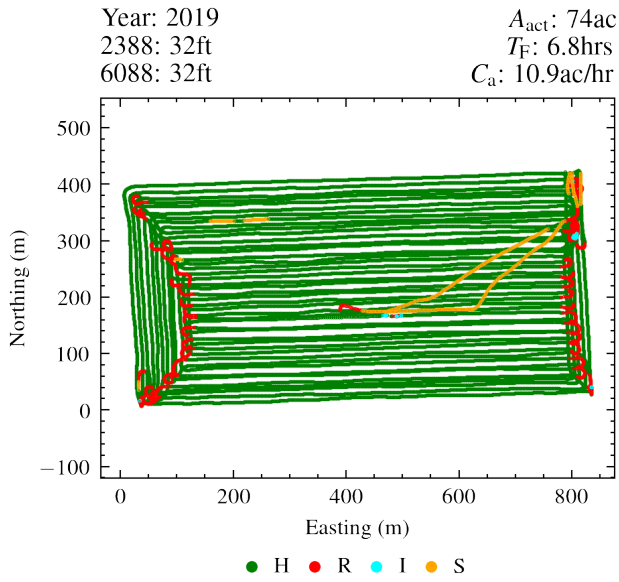
[133] Y. Zhang, A. Balmos, J. V. Krogmeier, and D. Buckmaster, "Dynamic high-precision field shape generation via combine GPS tracks," American Society of Agricultural and Biological Engineers, 2017. DOI: 10.13031/aim.201700809.

A. FIGURES

A.1 State Classification Maps for Chapter 5

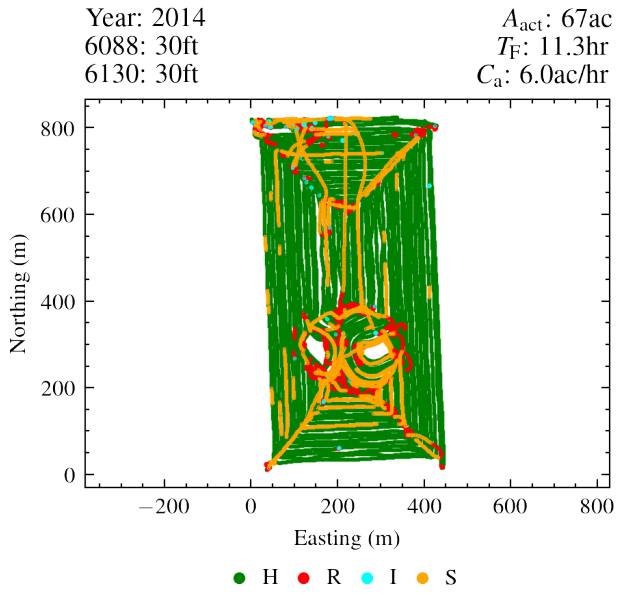


(a) Classified tracks in 2016.

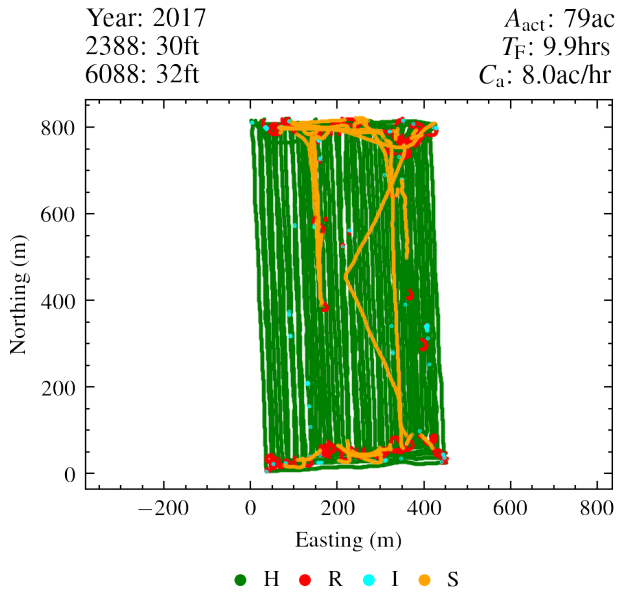


(b) Classified tracks in 2019.

Figure A.1. State classification results for field SL in 2016 and 2019.

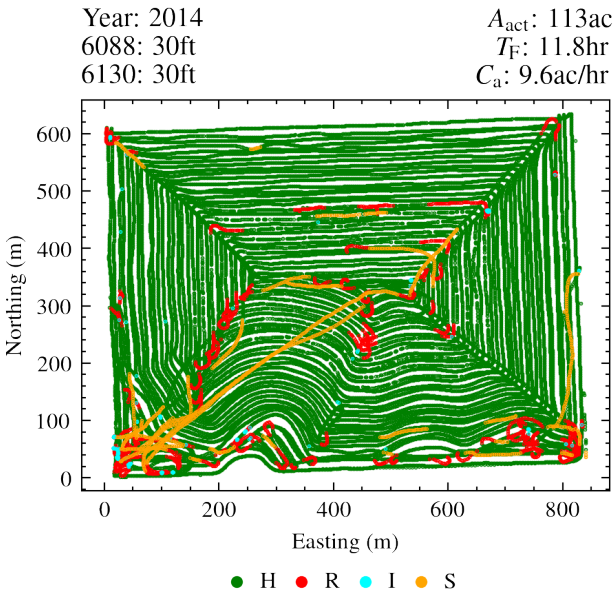


(a) Classified tracks in 2014.

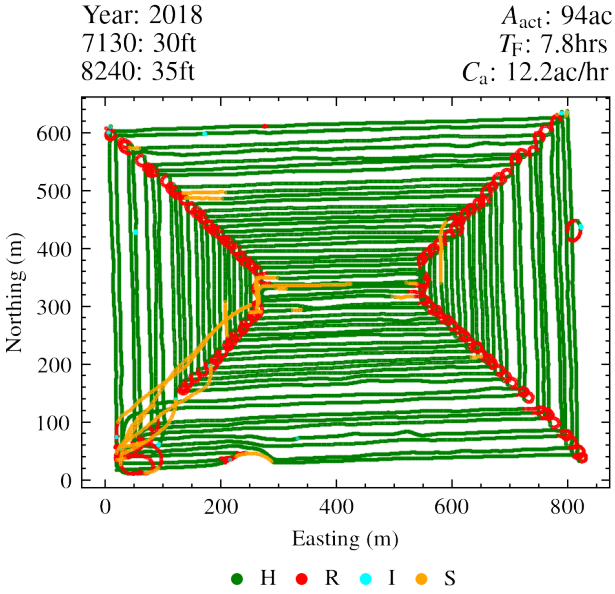


(b) Classified tracks in 2017.

Figure A.2. State classification results for field `BR` in 2014 and 2017. Different harvest patterns were utilized with different combine fleets. In 2014, the fleet traversed the field in loop. The skipped spot in 2014 indicates there might be legumes or wet spots that prevented combines traversing the whole field. In 2017, the field was harvested by sections and the entire field was covered.

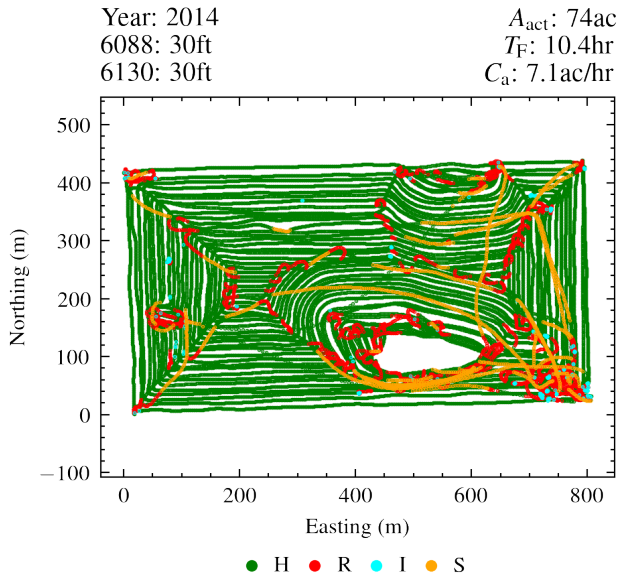


(a) Classified tracks in 2014.

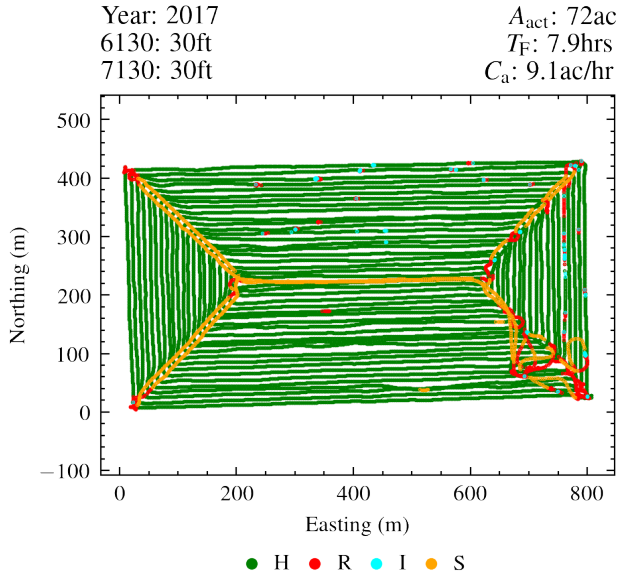


(b) Classified tracks in 2018.

Figure A.3. State classification results for field `CH` in 2014 and 2018. For this field, harvest patterns in both years are similar.

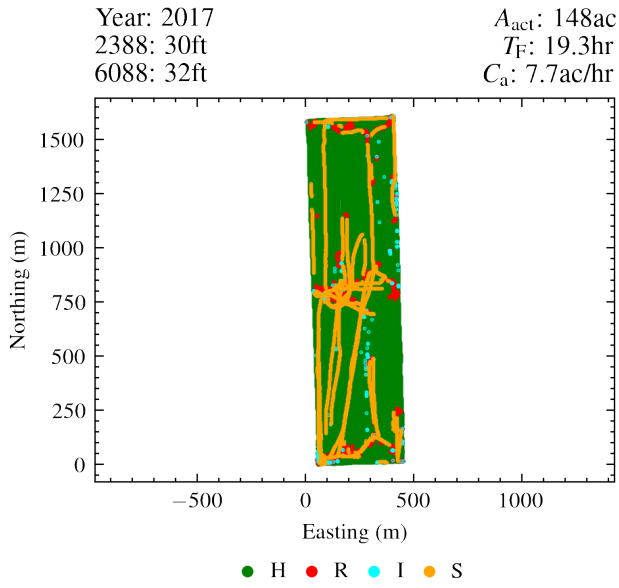


(a) Classified tracks in 2014.

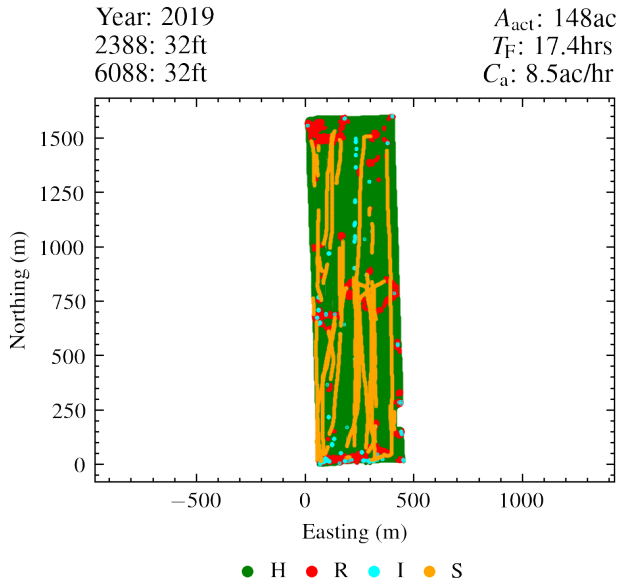


(b) Classified tracks in 2017.

Figure A.4. State classification results for field PL in 2014 and 2017. For this field, the harvest patterns in 2014 and 2017 are vastly different. In 2014, the combine fleet traversed most of the field jointly but also harvested individually for parts of the field. In addition, the skipped area in 2014 suggests that there might be anomalous field conditions. In 2017, the combine fleet followed a uniform harvesting pattern throughout the field and there was no anomalous field conditions.

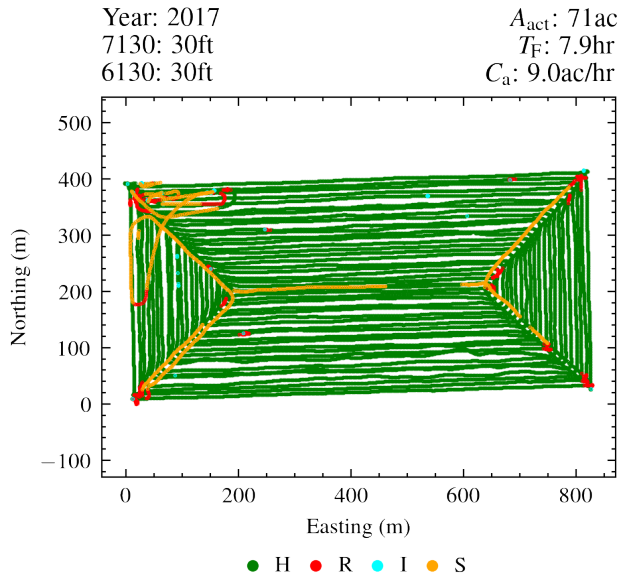


(a) Classified tracks in 2016.

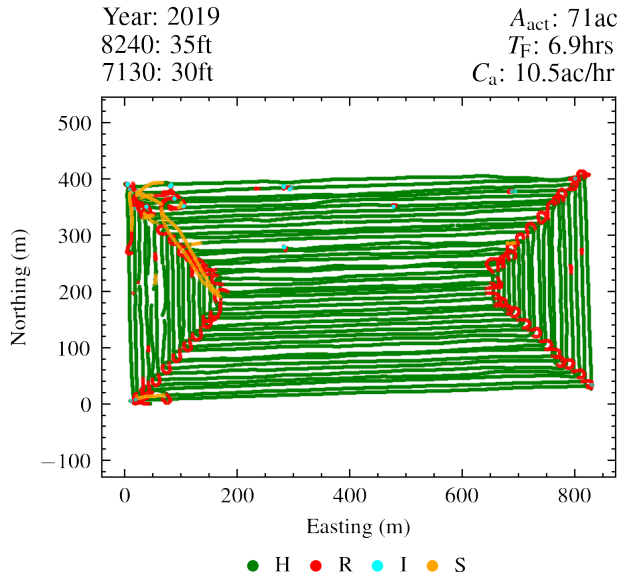


(b) Classified tracks in 2019.

Figure A.5. State classification results for field LL in 2017 and 2019. Although the GPS tracks show different harvest paths, the overall harvest patterns in both years are similar.

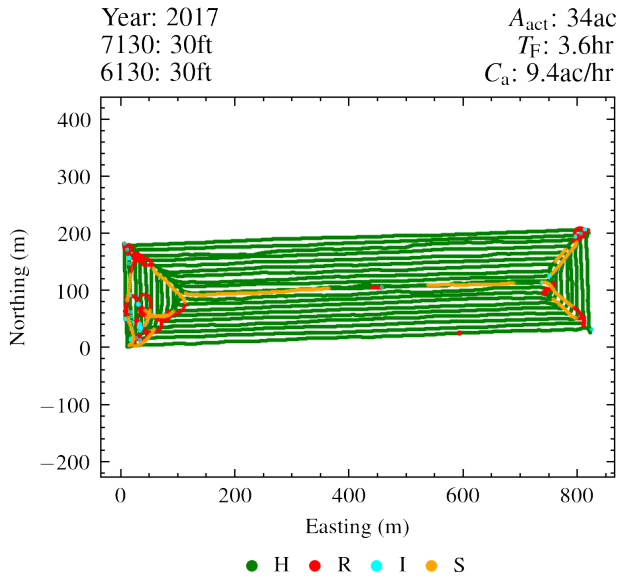


(a) Classified tracks in 2017.

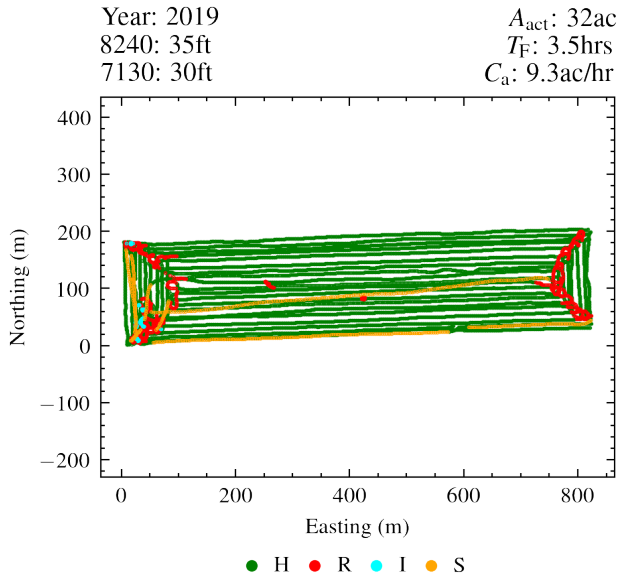


(b) Classified tracks in 2019.

Figure A.6. State classification results for field `RU` in 2017 and 2019. For this field, the harvest patterns in both years follow a loop style pattern for traversing the whole field. The difference is that there are more non-productive turning maneuvers at corners of harvest loops in 2019 than in 2017.



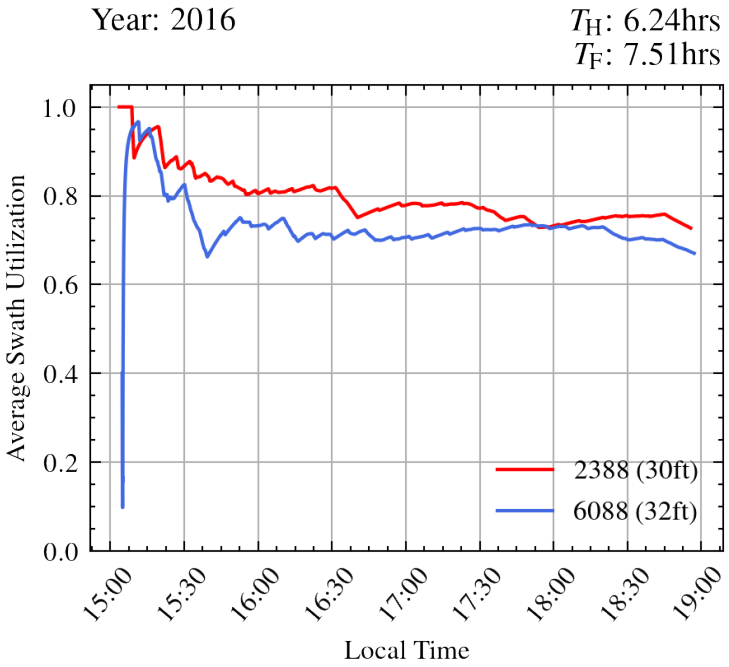
(a) Classified tracks in 2017.



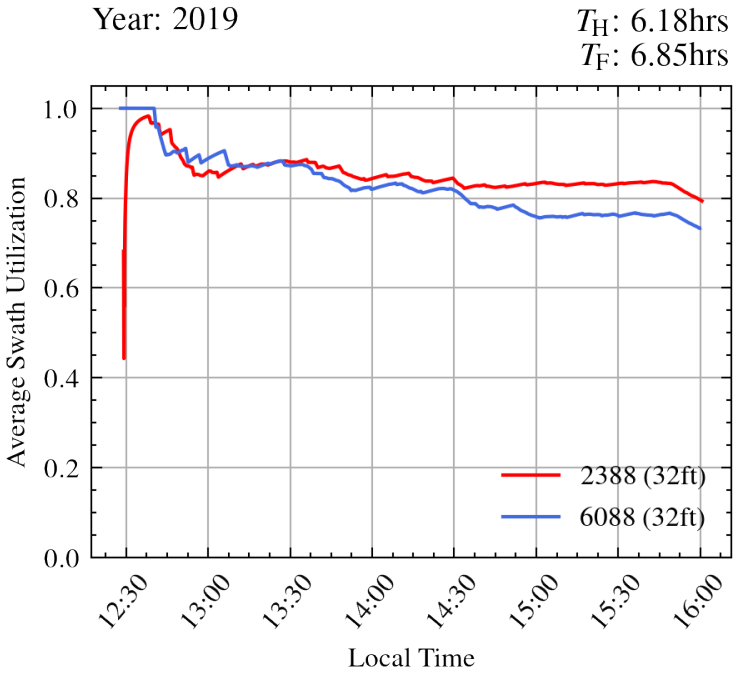
(b) Classified tracks in 2019.

Figure A.7. State classification results for field `MU` in 2017 and 2019. For this field, the harvest patterns for both years follow a similar loop style pattern. The same increment in non-productive turns observed in fig. A.6 could be observed in this plot.

A.2 Average Swath Utilizations Over Time for Chapter 5

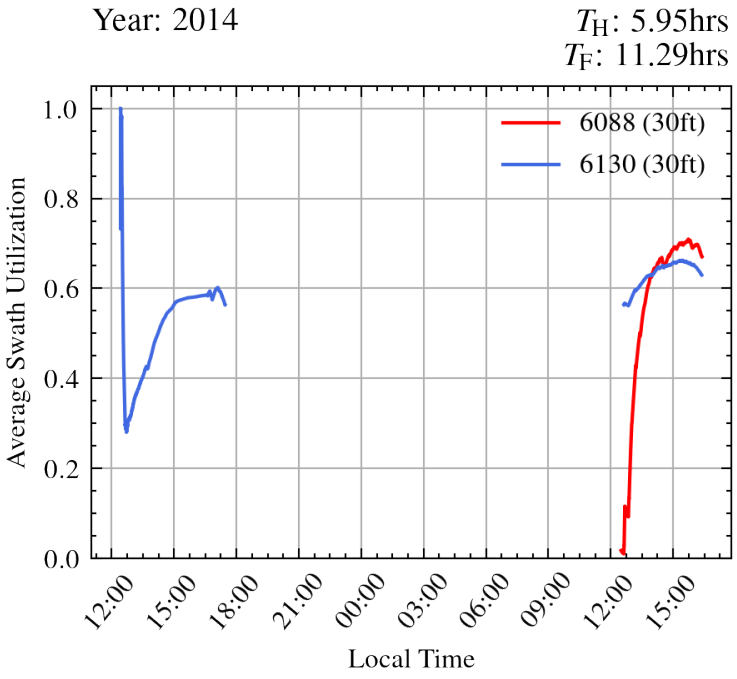


(a) Average swath utilization in 2016.

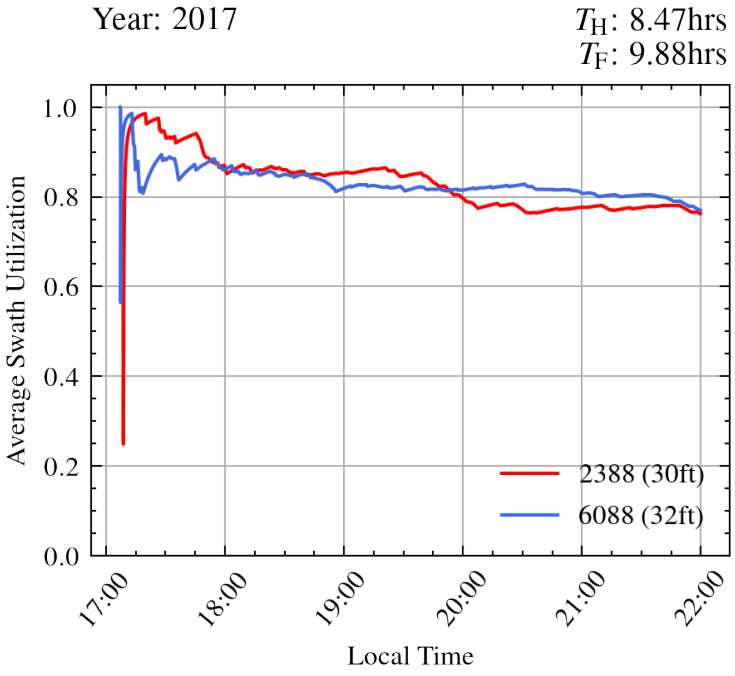


(b) Average swath utilization in 2019.

Figure A.8. Average swath utilizations for field SL.

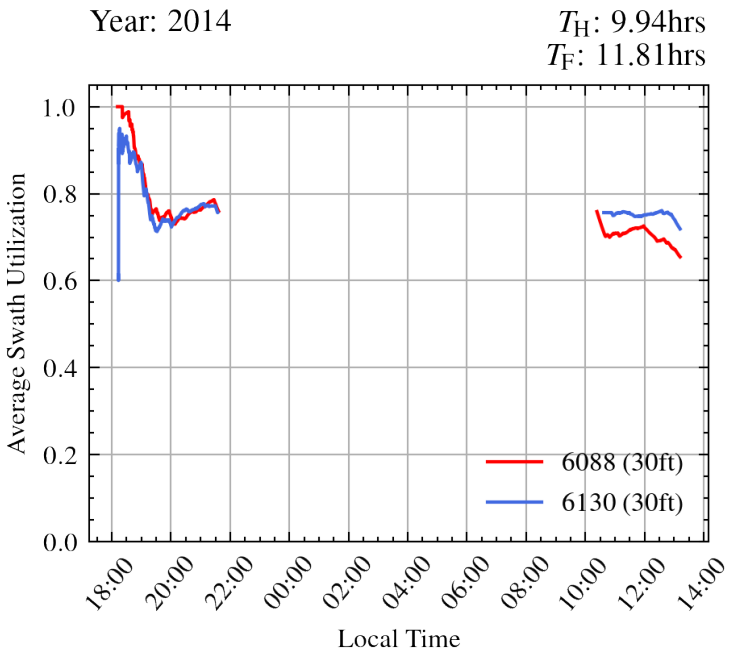


(a) Average swath utilization in 2014.

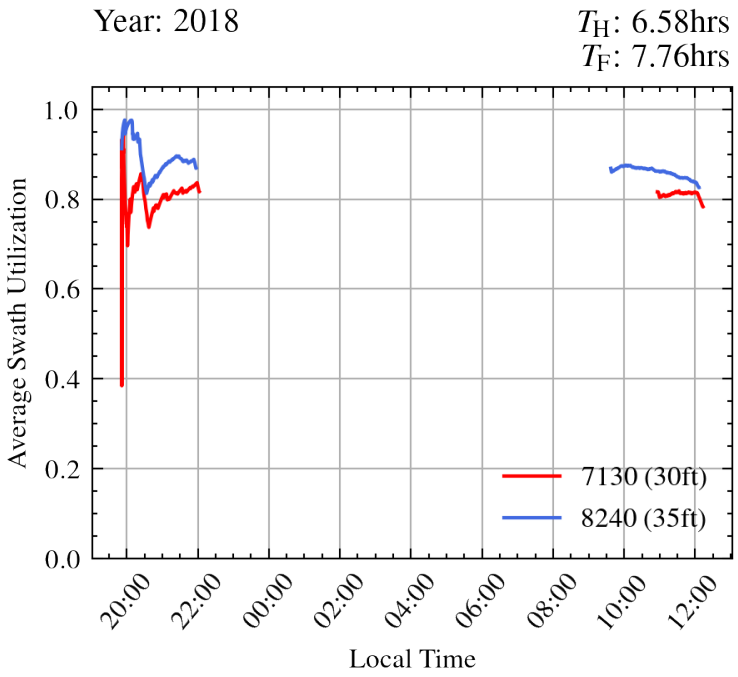


(b) Average swath utilization in 2017.

Figure A.9. Average swath utilizations for field BR.

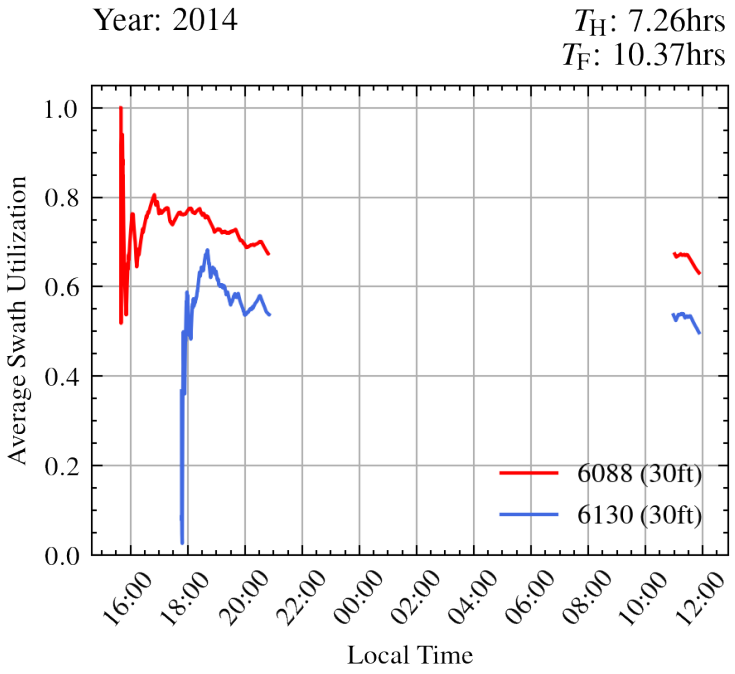


(a) Average swath utilization in 2014.

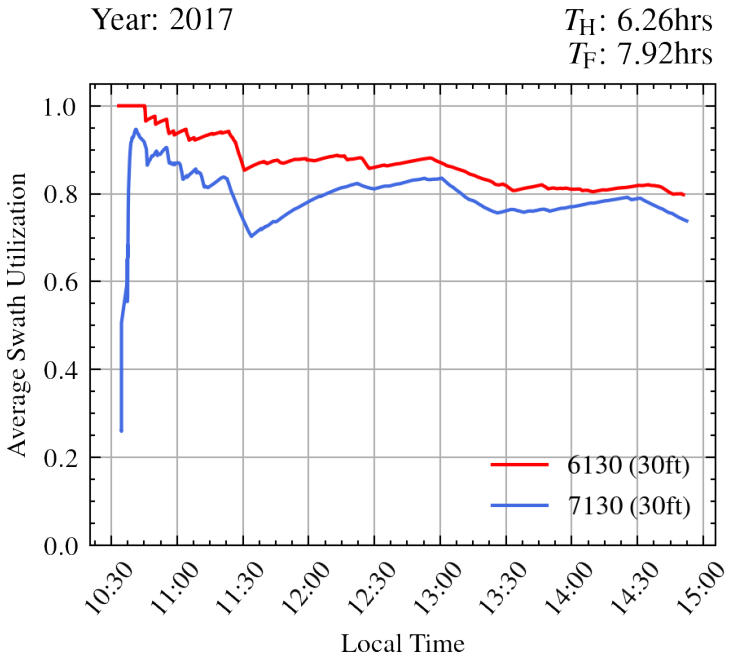


(b) Average swath utilization in 2018.

Figure A.10. Average swath utilizations for field `CH`.

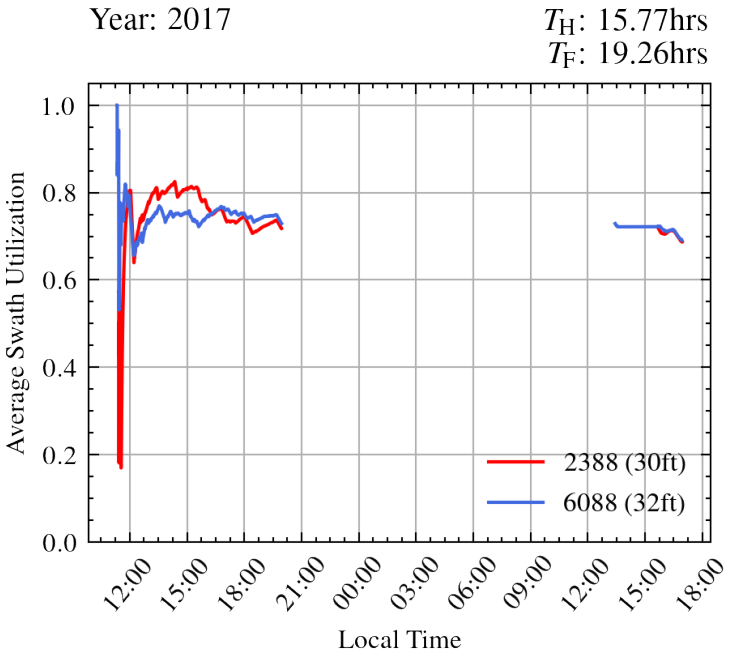


(a) Average swath utilization in 2014.

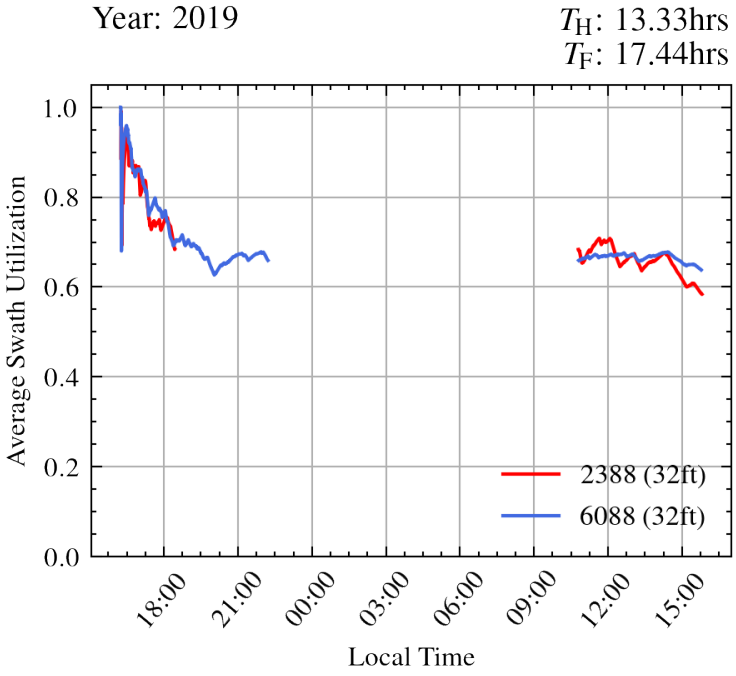


(b) Average swath utilization in 2017.

Figure A.11. Average swath utilizations for field PL.

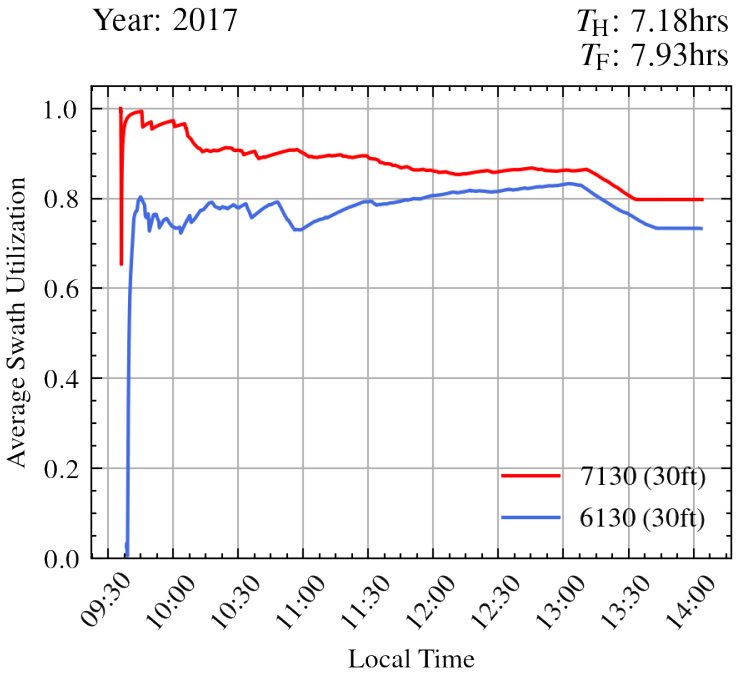


(a) Average swath utilization in 2017.

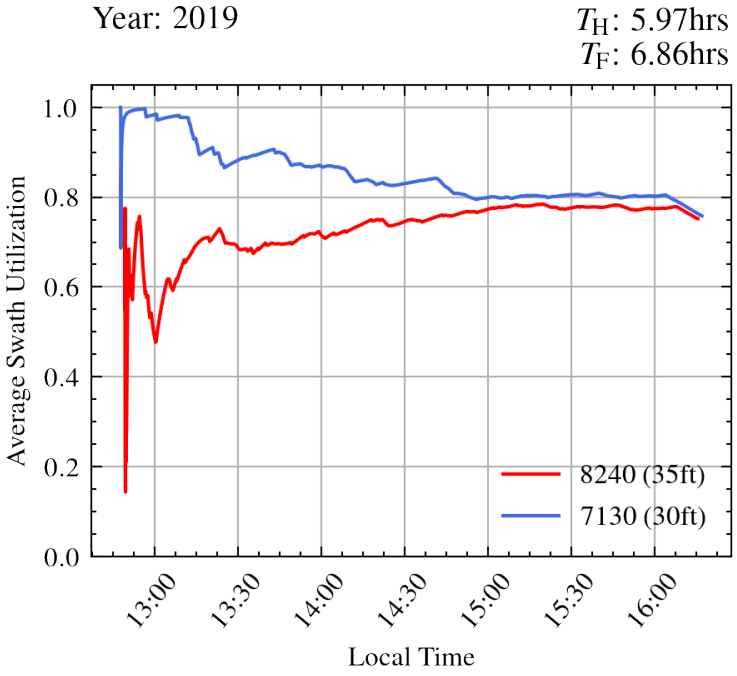


(b) Average swath utilization in 2019.

Figure A.12. Average swath utilizations for field LL.

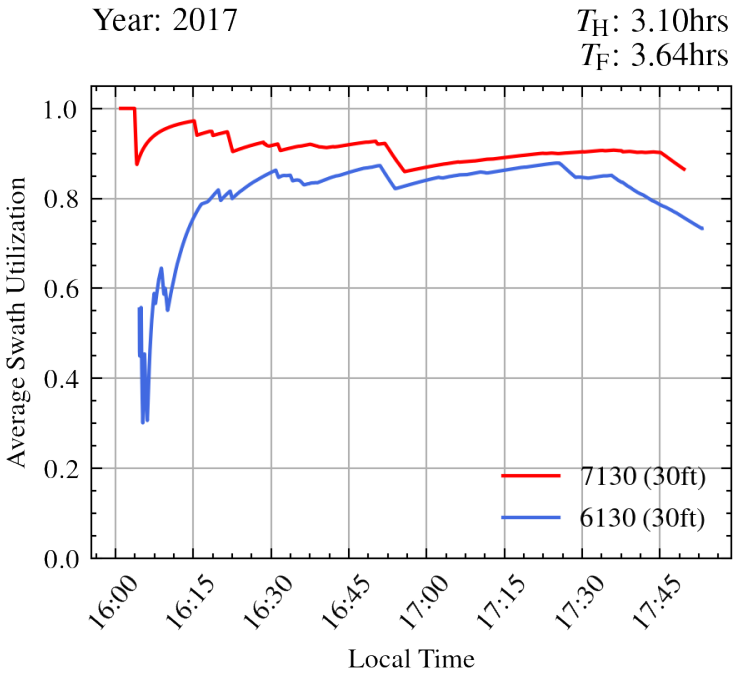


(a) Average swath utilization in 2017.

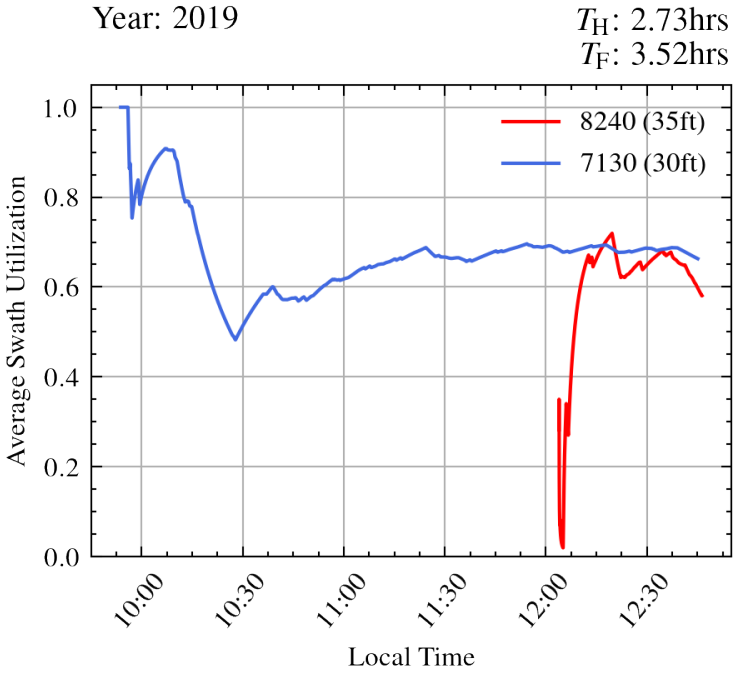


(b) Average swath utilization in 2019.

Figure A.13. Average swath utilizations for field `RU`.



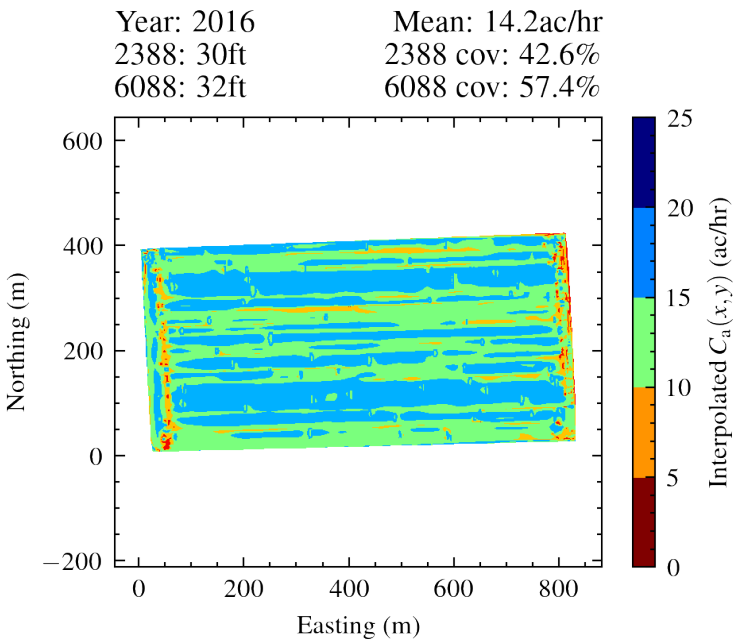
(a) Average swath utilization in 2017.



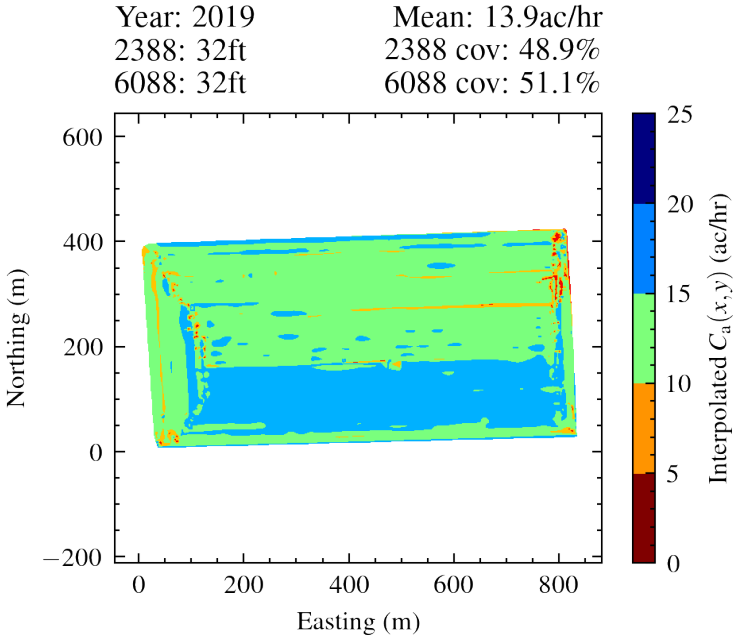
(b) Average swath utilization in 2019.

Figure A.14. Average swath utilizations for field `MU`.

A.3 Interpolated Area Capacity Contour Maps for Chapter 5

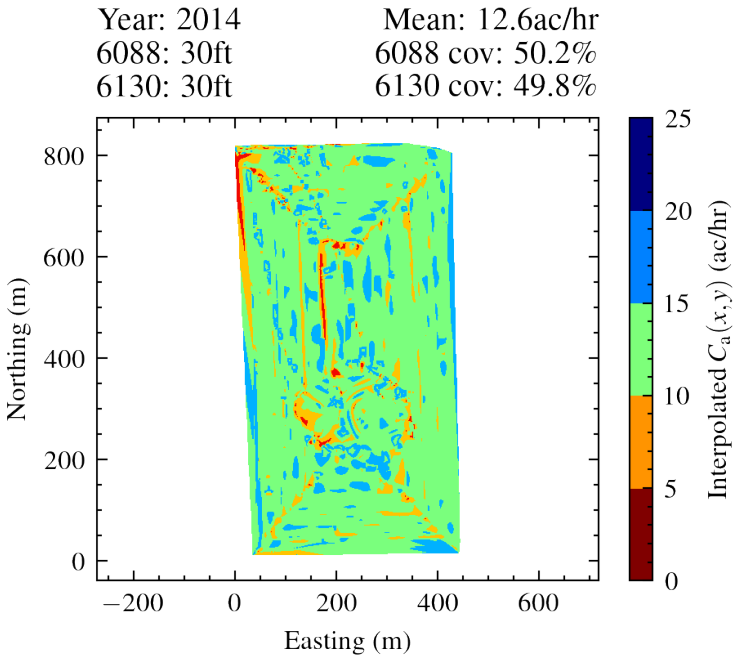


(a) Interpolated area capacity map for 2016.

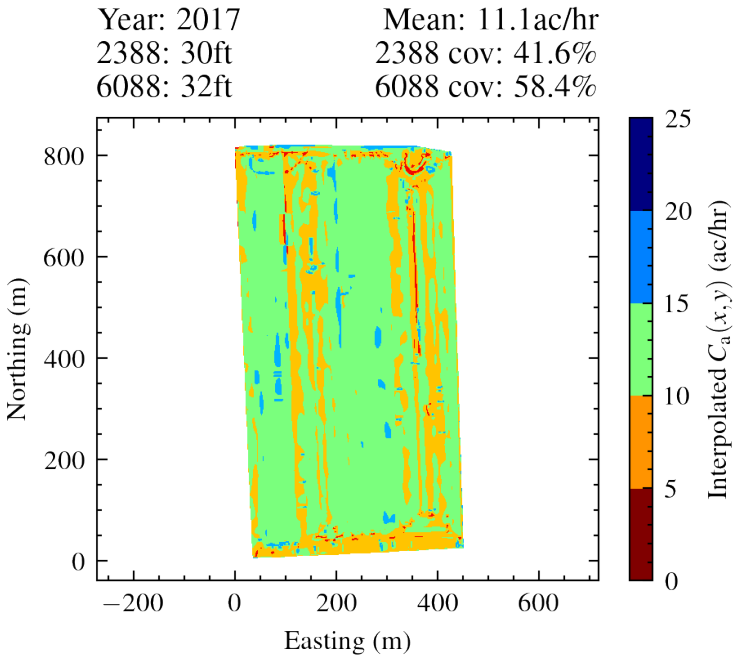


(b) Interpolated area capacity map for 2019.

Figure A.15. Interpolated area capacity map for field SL.

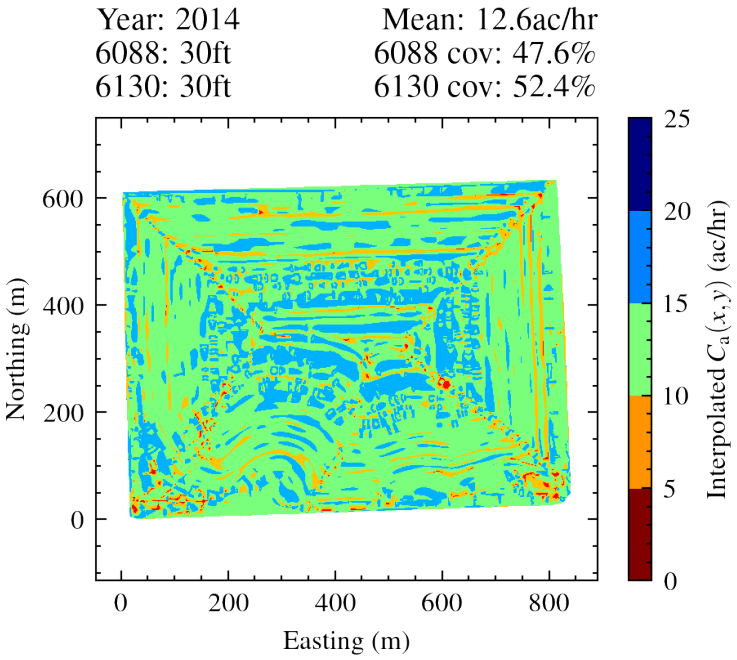


(a) Interpolated area capacity map for 2014.

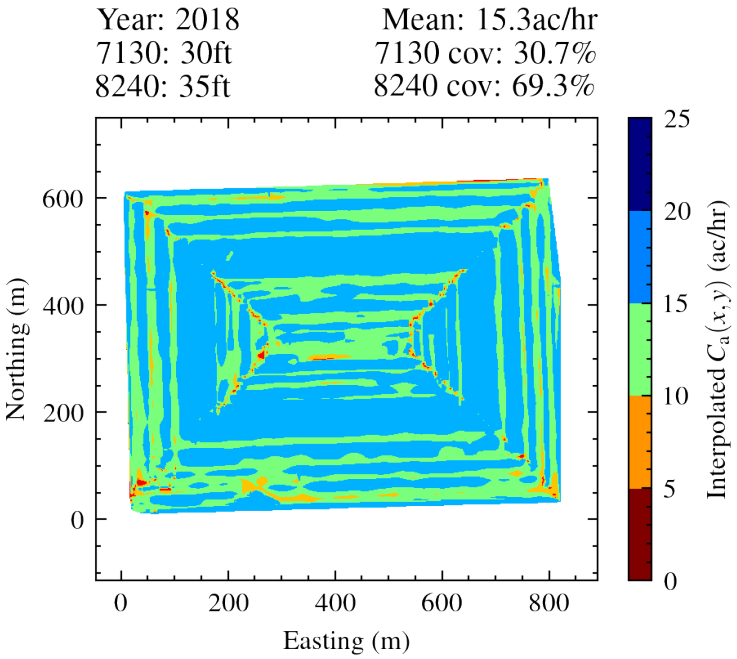


(b) Interpolated area capacity map for 2017.

Figure A.16. Interpolated area capacity map for field BR.

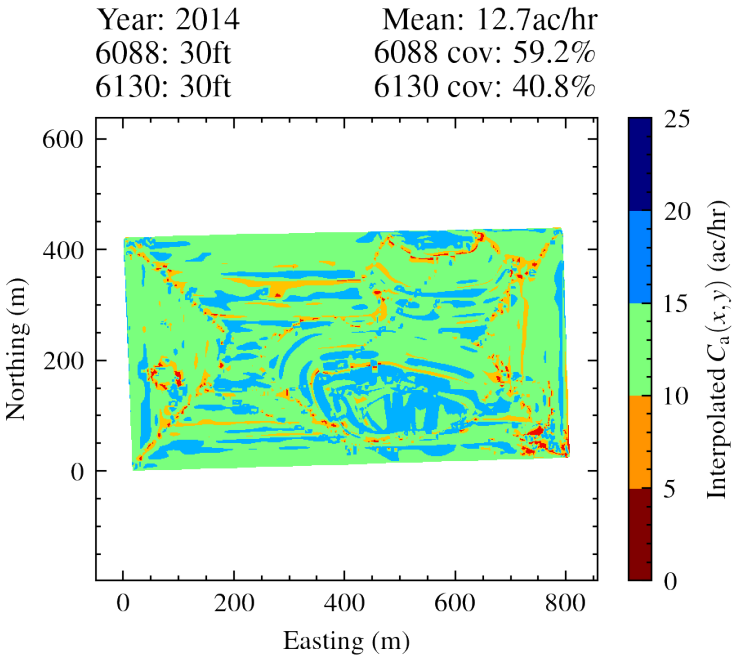


(a) Interpolated area capacity map for 2014.

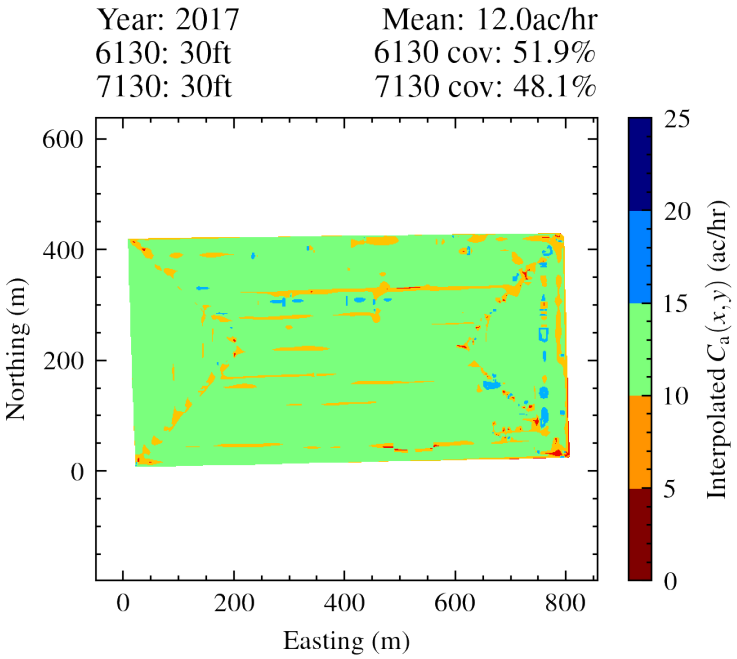


(b) Interpolated area capacity map for 2018.

Figure A.17. Interpolated area capacity map for field CH.

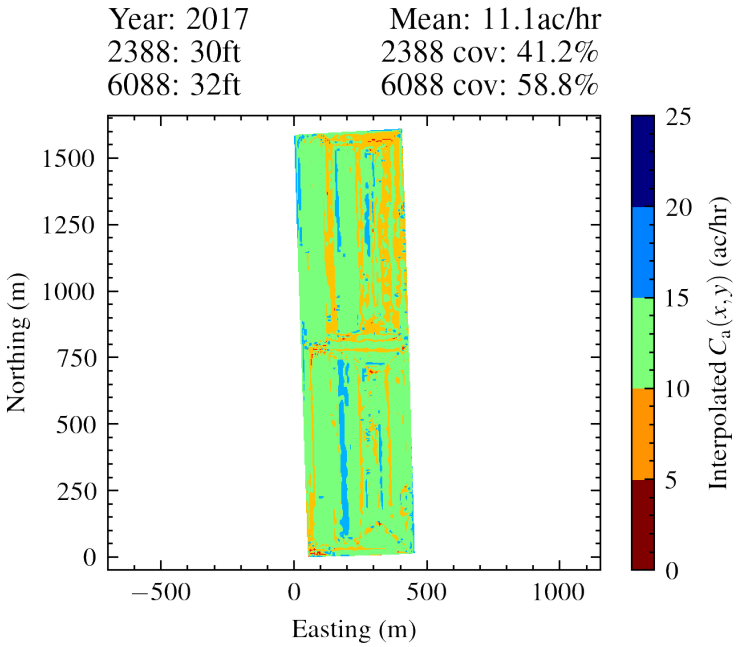


(a) Interpolated area capacity map for 2014.

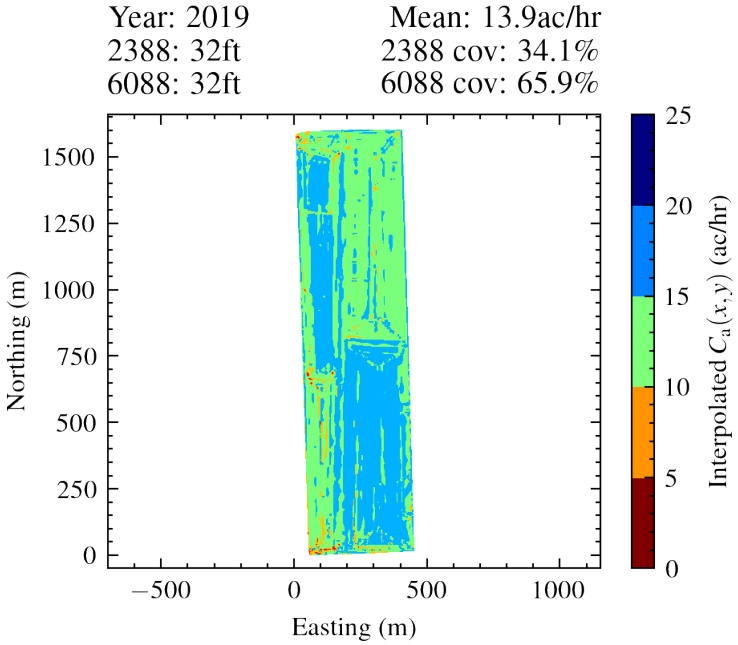


(b) Interpolated area capacity map for 2017.

Figure A.18. Interpolated area capacity map for field PL.

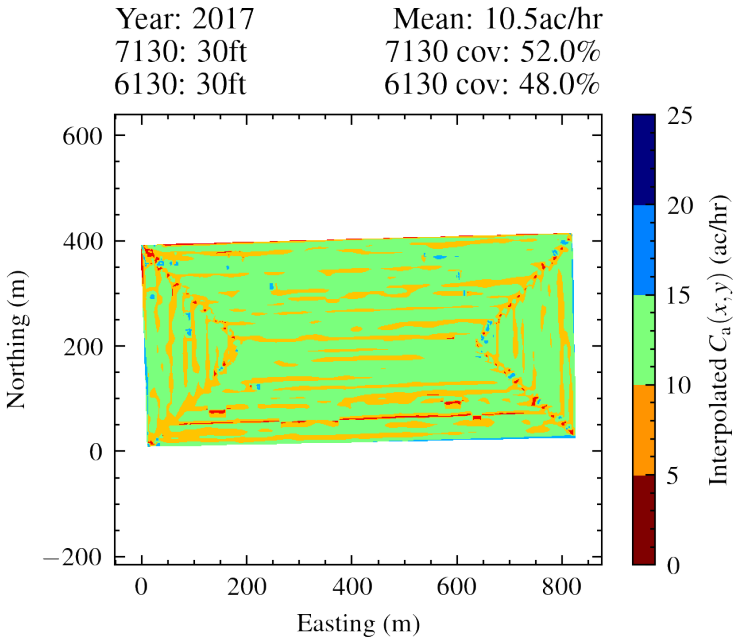


(a) Interpolated area capacity map for 2017.

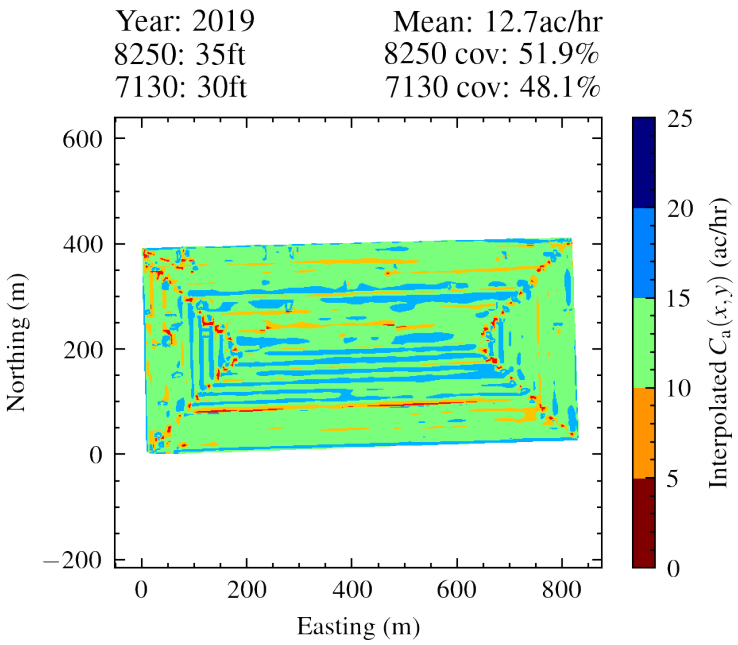


(b) Interpolated area capacity map for 2019.

Figure A.19. Interpolated area capacity map for field LL.

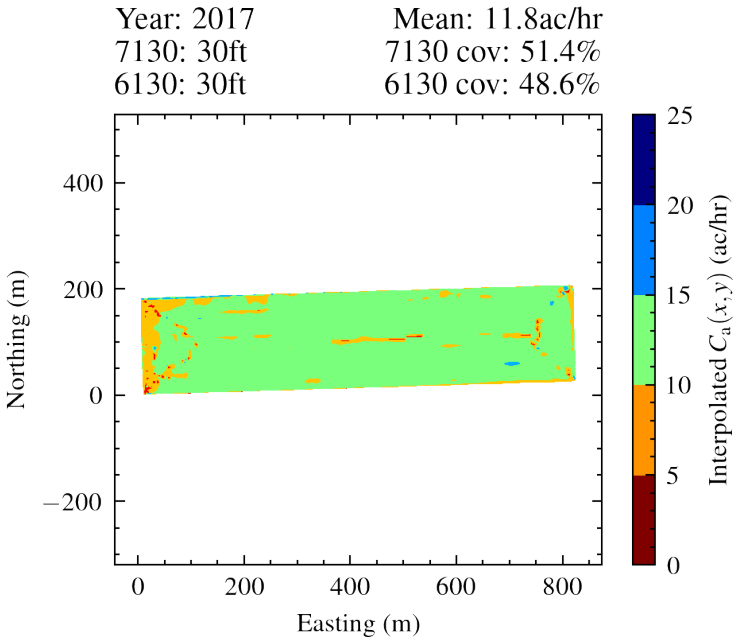


(a) Interpolated area capacity map for 2017.

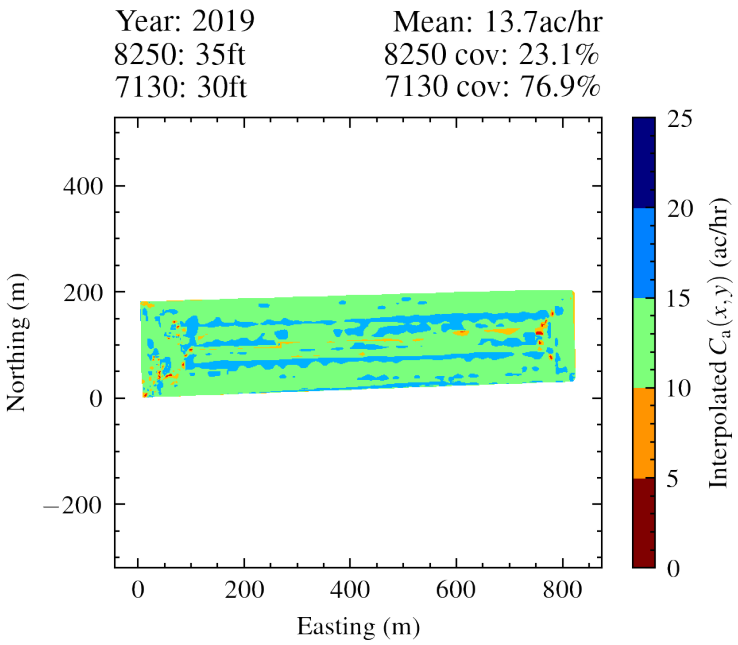


(b) Interpolated area capacity map for 2019.

Figure A.20. Interpolated area capacity map for field RU.



(a) Interpolated area capacity map for 2017.



(b) Interpolated area capacity map for 2019.

Figure A.21. Interpolated area capacity map for field `MU`.

A.4 Emperical Cumulative Density Functions Plots for Chapter 5

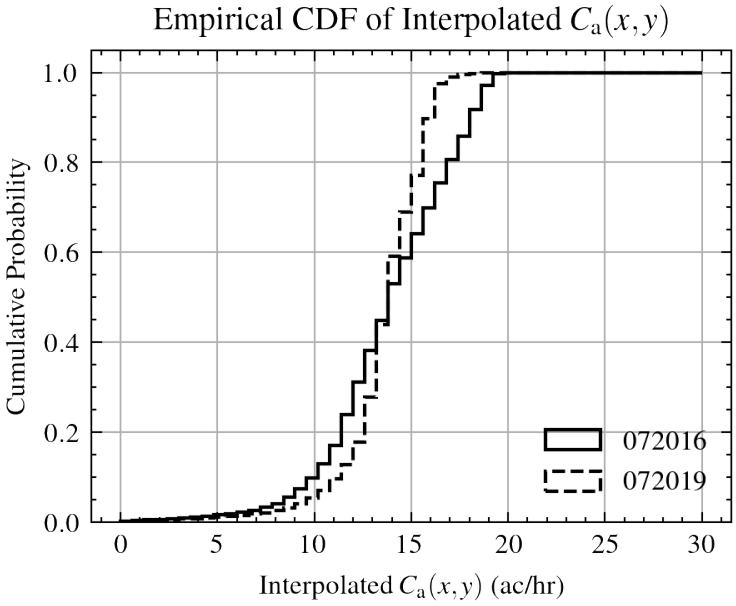


Figure A.22. Empirical CDFs of interpolated area capacity for field `SL`.

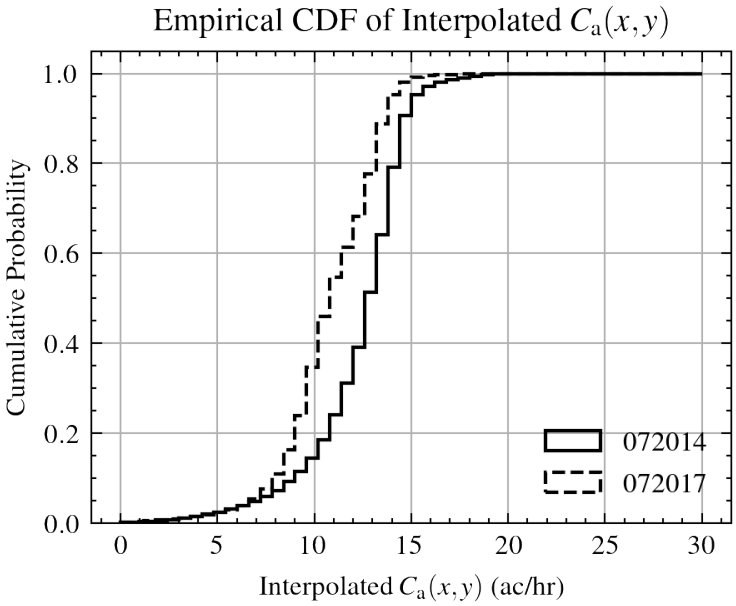


Figure A.23. Empirical CDFs of interpolated area capacity for field `BR`.

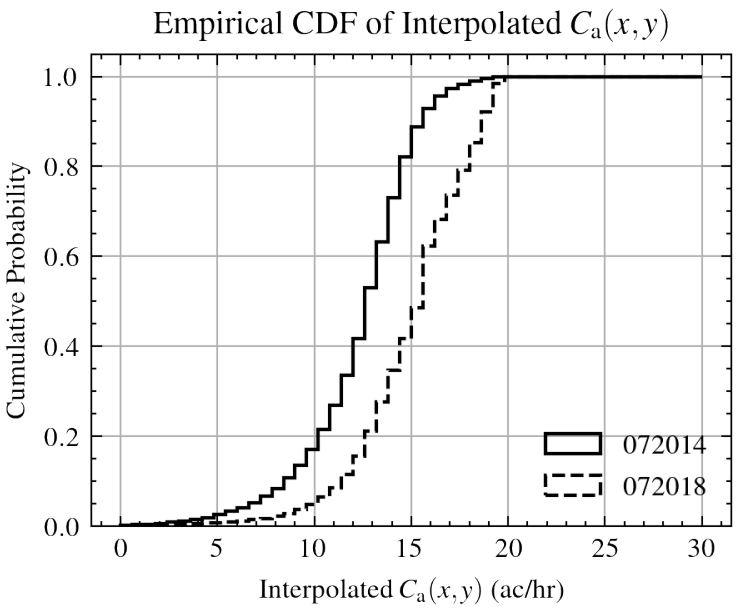


Figure A.24. Empirical CDFs of interpolated area capacity for field `CH`.

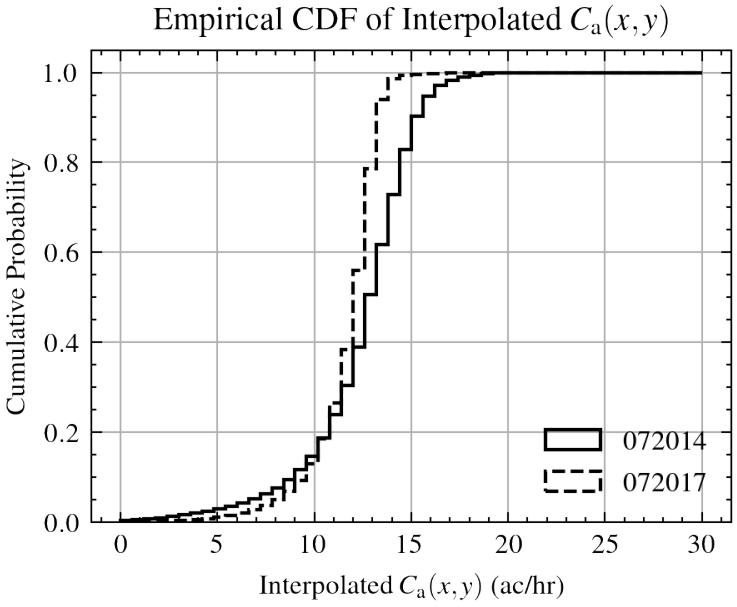


Figure A.25. Empirical CDFs of interpolated area capacity for field `PL`.

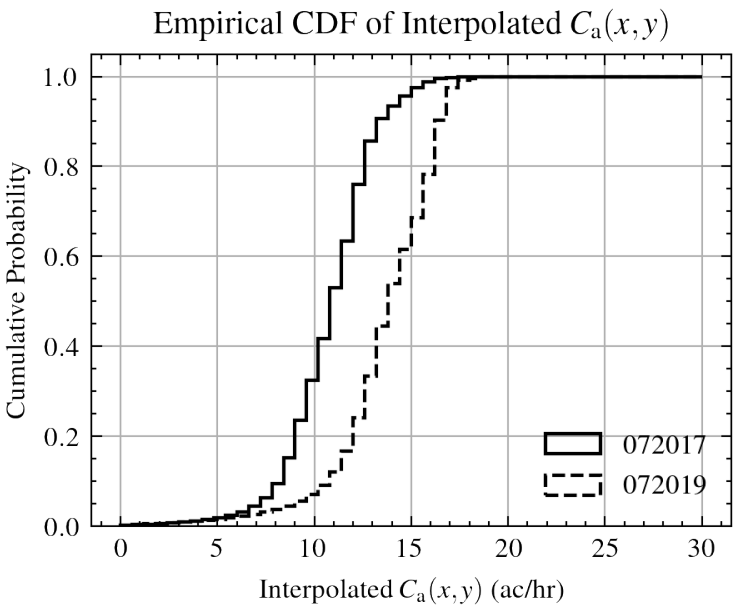


Figure A.26. Empirical CDFs of interpolated area capacity for field LL.

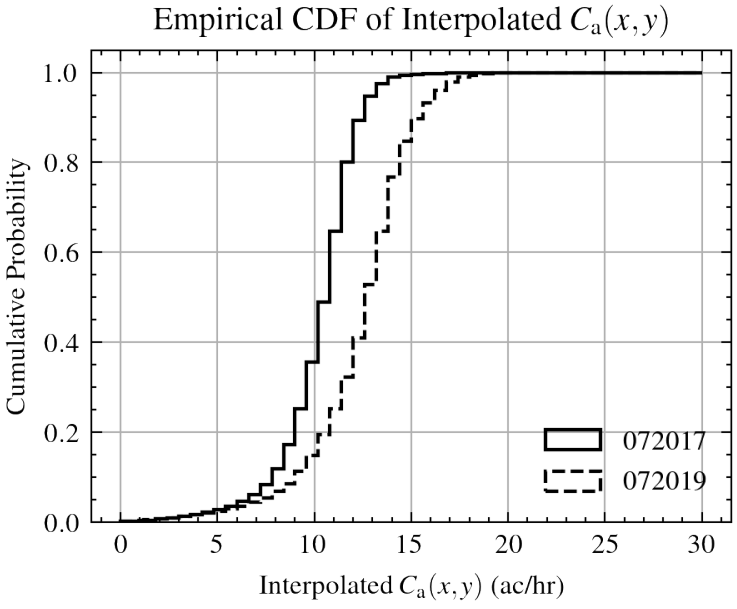


Figure A.27. Empirical CDFs of interpolated area capacity for field RU.

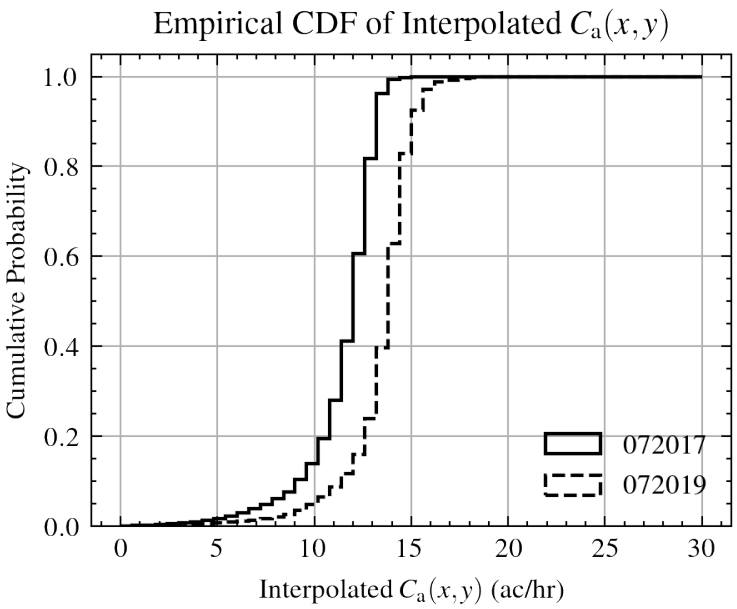


Figure A.28. Empirical CDFs of interpolated area capacity for field `MU`.

A.5 Area Capacity Difference Contour Maps for Chapter 5

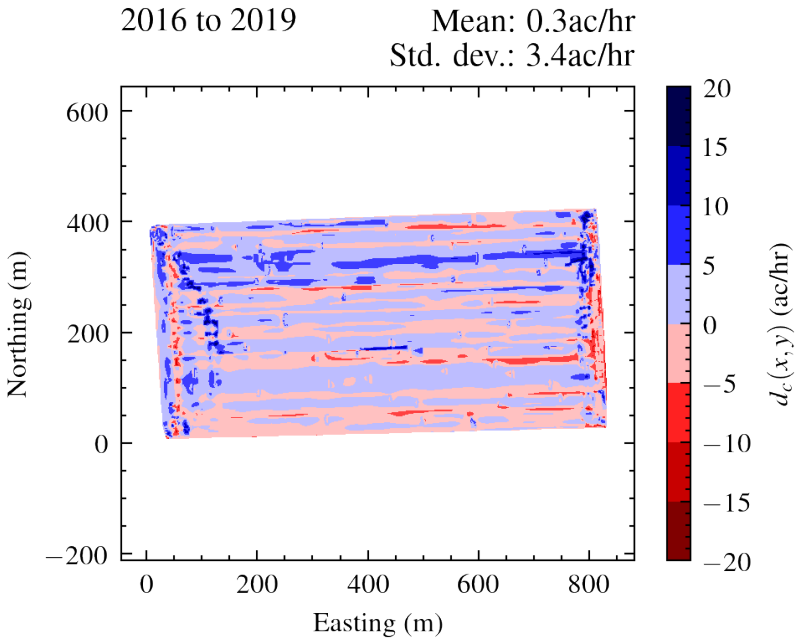


Figure A.29. d_c contour map for field `SL`.

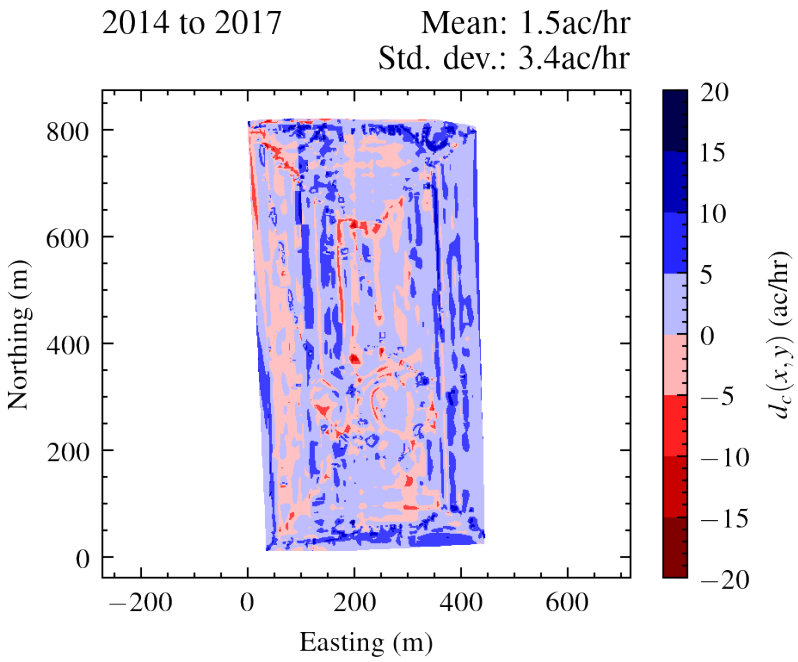


Figure A.30. d_c contour map for field `BR`.

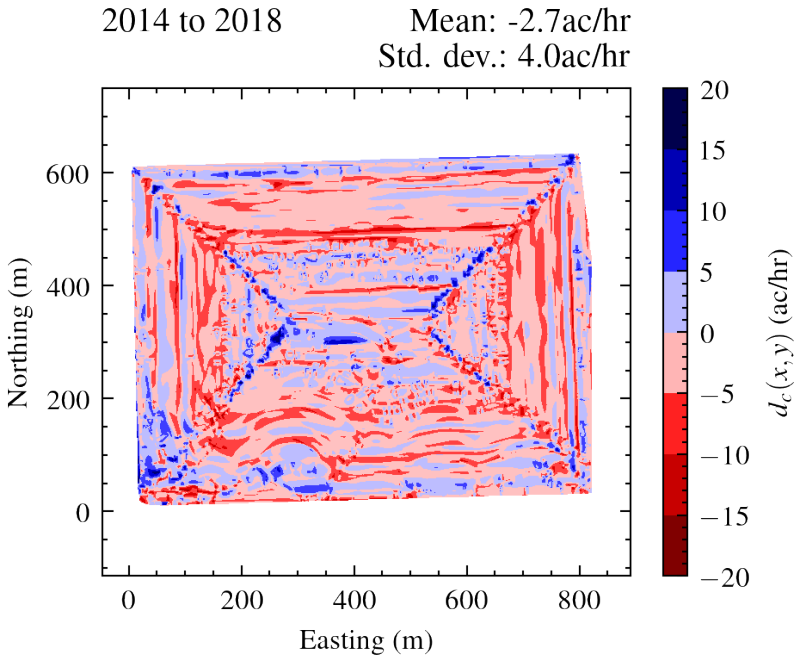


Figure A.31. d_c contour map for field `CH`.

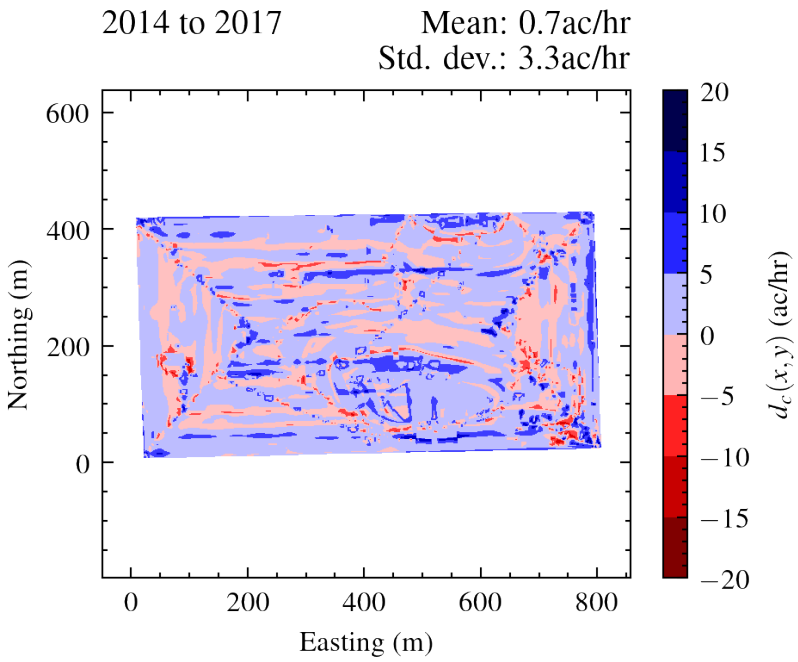


Figure A.32. d_c contour map for field PL.

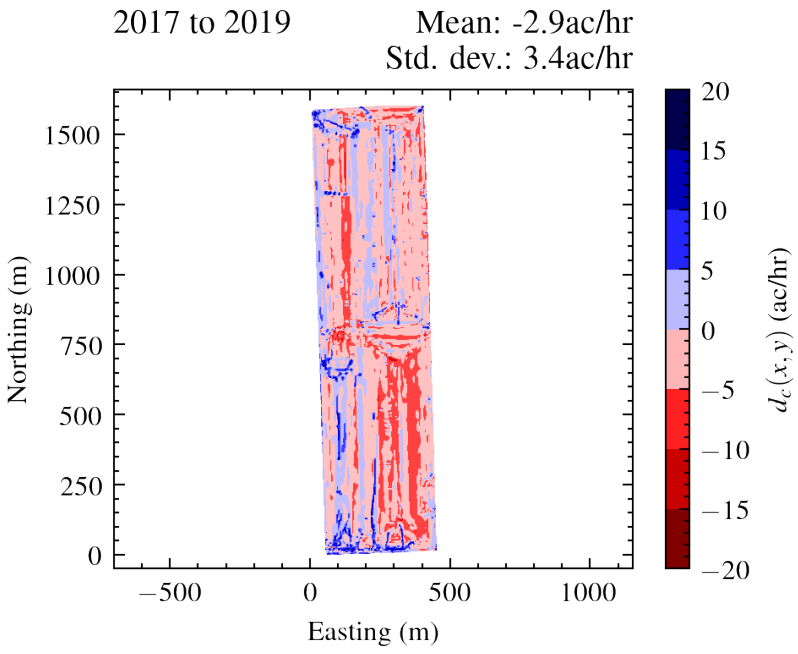


Figure A.33. d_c contour map for field LL.

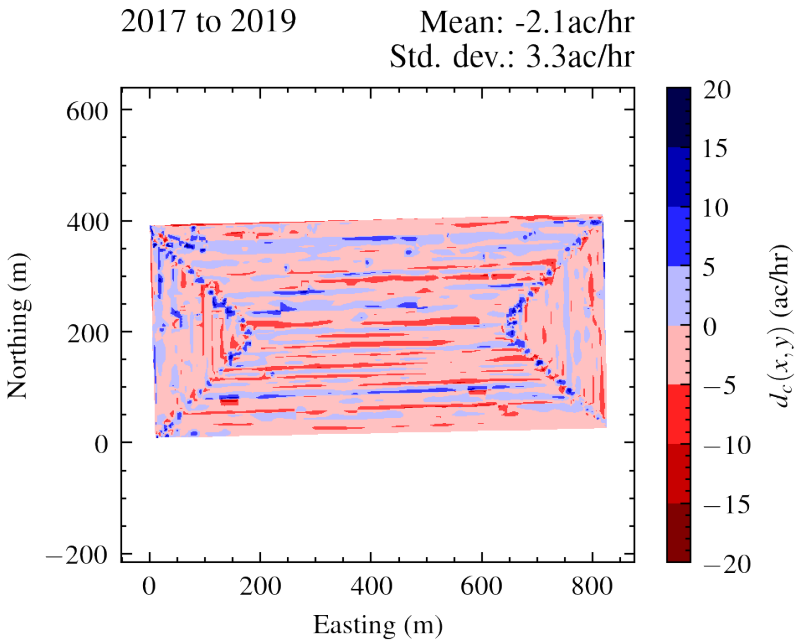


Figure A.34. d_c contour map for field `RU`.

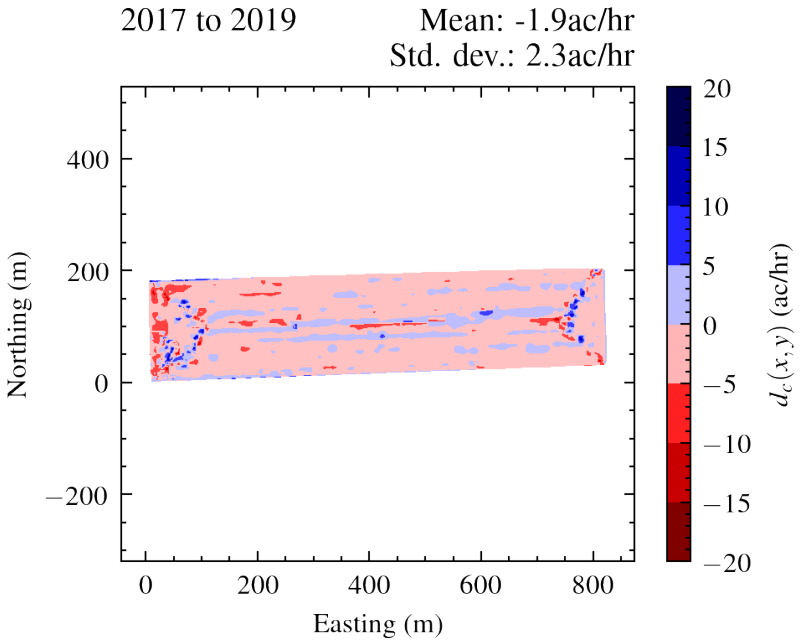


Figure A.35. d_c contour map for field `MU`.

B. TABLES

Table B.1. This work focuses on 7 fields of GPS data. Each field contains data in two separate harvest years. Each year includes GPS data of two combines. Total number of samples and header configurations for each combine are listed. Areas are determined from manually created field boundary polygons.

Field Name	Area (ac)	Year	Model	Header width (ft)	Num. of samples
SL	81	2016	2388	30	13927
			6088	32	13644
		2019	2388	32	12540
			6088	32	39315
BR	85	2014	6088	30	14620
			6130	30	32695
		2017	2388	30	18897
			6088	32	16902
CT	124	2014	6088	30	22352
			6130	30	20719
		2018	7130	30	12384
			8240	35	16494
PL	84	2014	6088	30	21570
			6130	30	16782
		2017	6130	30	15211
			7130	30	15322
LL	161	2017	2388	30	34817
			6088	32	35909
		2019	2388	32	25760
			6088	32	39315
RU	79	2017	6130	30	14652
			7130	30	14234
		2019	7130	30	12255
			8240	35	12341
MU	38	2017	6130	30	6522
			7130	30	6617
		2019	7130	30	10154
			8240	35	2552

Table B.2. Computed metrics for overall efficiency comparisons.

Field Name	Year	Model	Header width (ft)	$s_{\mathrm{h,avg}}$ (mph)	e_{t} (%)	A_{act} (ac)	E_{t} (%)	E_{g} (%)	C_{a} (ac/hr)
SL	2016	2388	30	3.3	85.3	73	83.0	65.9	11.8
		6088	32	4.4	80.6				
	2019	2388	32	3.6	90.6	74	90.2	73.9	12.1
		6088	32	3.9	89.8				
BR	2014	6088	30	3.8	75.5	67	52.7	45.6	11.4
		6130	30	3.9	40.2				
	2017	2388	30	2.6	80.7	78	85.7	66.6	9.3
		6088	32	3.3	91.3				
CH	2014	6088	30	3.9	81.9	112	84.1	63.7	11.3
		6130	30	4.2	86.4				
	2018	7130	30	3.4	79.5	94	84.8	75.8	14.4
		8240	35	4.3	88.7				
PL	2014	6088	30	3.6	73.0	73	70.0	49.6	10.2
		6130	30	4.1	66.1				
	2017	6130	30	3.5	82.6	72	79.0	71.0	11.5
		7130	30	3.4	75.5				
LL	2017	2388	30	2.7	79.4	148	81.9	61.7	9.4
		6088	32	3.4	84.3				
	2019	2388	32	3.3	71.6	148	76.5	56.5	11.1
		6088	32	3.8	79.6				
RU	2017	6130	30	3.3	88.7	71	90.5	73.2	9.9
		7130	30	3.4	92.3				
	2019	7130	30	3.9	89.0	70	87.0	63.4	11.9
		8240	35	3.6	85.1				
MU	2017	6130	30	3.2	80.9	34	85.3	78.8	11.0
		7130	30	3.4	89.7				
	2019	7130	30	3.9	76.2	32	77.7	67.8	12.0
		8240	35	4.4	78.1				

VITA

YANG WANG

wang701@purdue.edu ⋄ waterkingwatergoat.com
465 Northwestern Avenue ⋄ West Lafayette, IN 47907

EDUCATION

Purdue University, West Lafayette — Aug. 2015 – Aug. 2021
Ph.D. in Electrical Engineering

Purdue University, West Lafayette — Aug. 2010 – Dec. 2014
B.S. in Electrical Engineering

EXPERIENCE

Purdue University — Aug. 2015 – May. 2021
Graduate Research Assistant — *West Lafayette, IN*

Spensa Technologies Inc. — Jan. 2015 – May. 2015
Embedded Systems Engineer — *West Lafayette, IN*

Spensa Technologies Inc. — May. 2014 – Dec. 2014
Embedded Engineer Internship — *West Lafayette, IN*

Keithley Instruments, LLC. — Aug. 2012 – Dec. 2012
Test Engineer Internship — *Cleveland, OH*

PUBLICATIONS

1. **Y. Wang** et al., "CANdroid: Freeing ISOBUS Data and Enabling Machine Data Analytics," in 2016 ASABE Annual International Meeting, 2016, p. 1, doi: 10.13031/aim.20162459827.
2. A. W. Layton, **Y. Wang**, J. V. Krogmier, and D. Buckmaster, "Robust Estimation of Field Management Zones Using Multi-Year Yield Data and a Hidden Markov Random Field," in 2016 ASABE Annual International Meeting, 2016, p. 1, doi: 10.13031/aim.20162461641.
3. **Y. Wang** et al., "An Open-Source Infrastructure for Real-Time Automatic Agricultural Machine Data Processing," in 2017 ASABE Annual International Meeting, 2017, p. 1, doi: 10.13031/aim.201701022.
4. **Y. Wang**, A. D. Balmos, J. V. Krogmeier, and D. R. Buckmaster, "An Optimized Adaptive Kalman Filtering Algorithm for Agricultural GPS Data Processing," in 2018 ASABE Annual International Meeting, 2018, p. 1, doi: 10.13031/aim.201800081.
5. A. Lindsay, **Y. Wang**, S. Noel, Y. Zhang, J. V. Krogmeier, and B. Dennis, "CAN-Based Forage Yield Mapping," in 2018 ASABE Annual International Meeting, 2018, p. 1, doi: 10.13031/aim.201801016.
6. **Y. Wang**, A. D. Balmos, D. R. Buckmaster, and J. V. Krogmeier, "Data-Driven Agricultural Machinery Activity Anomaly Detection and Classification," in 14th International Conference on Precision Agriculture, 2018, p. 1, Accessed: Aug. 17, 2020. [Online].
7. S. Noel, A. Ault, **Y. Wang**, A. W. Layton, A. D. Balmos, J. V. Krogmeier, D. Buckmaster, "Real-time on-farm yield trials powered by open-source: connecting ISOBlue, OADA, and the Trials Tracker App," in 2018 ASABE Annual International Meeting, 2018, p. 1, doi: 10.13031/aim.201801597.
8. A. W. Layton, **Y. Wang**, J. V. Krogmeier, D. Buckmaster, "The Trellis Framework for Automatic Food Safety Data Transfer," in 2018 ASABE Annual International Meeting, 2018, p. 1, doi: 10.13031/aim.201801248.
9. **Y. Wang**, Y. Zhang, J. V. Krogmeier, and D. R. Buckmaster, "Combine Harvester Unloading Event Inference Using GPS Data," in 2019 ASABE Annual International Meeting, 2019, p. 1, doi: 10.13031/aim.201901286.
10. **Y. Wang**, H. Liu, J. Krogmeier, A. Reibman, and D. Buckmaster, "ISOBlue HD: An Open-Source Platform for Collecting Context-Rich Agricultural Machinery Datasets," Sensors, vol. 20, no. 20, p. 5768, Oct. 2020, doi: 10.3390/s20205768.