GRAIN HARVEST LOGISTICS TRACKING TOOLS

by

Logan Heusinger

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Agricultural and Biological Engineering

School of Agricultural and Biological Engineering

West Lafayette, Indiana

December 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. John Evans IV, Chair

School of Agricultural and Biological Engineering

Dr. John Lumkes

School of Agricultural and Biological Engineering

Dr. Dennis Buckmaster

School of Agricultural and Biological Engineering

Approved by:

Dr. Nathan S. Mosier

Dedicated to my wife

ACKNOWLEDGMENTS

To my peers and mentors, a large thank you for your constant support and help. Covid-19 has not made graduate school easy however, all of you have been a blessing. To Eric Kong thank you for being a good peer. Dr. Evans I am supremely thankful for your patience and support through the long process that has been my graduate school studies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

ACRE	Agricultural Center for Research and Education
CR	Coding Rate
FEC	Forward Error Correction
FMIC	Farm Management Information System
IoT	Internet of Things
LoRa	Long Range Radio
LPWA	Low Power Wide Area
M2M	Machine to Machine
NRCS	National Resources Conservation Services
OTAA	Over the Air Activation
RFID	Radio Frequency Identification
RSSI	Received Signal Strength Indicator
SMS	Short Messaging Service
WSN	Wireless Sensor Network

1. STATE ALGORITHM FOR GPS EVALUATION

1.1 Abstract

Farmers run complex operations to fully plant, manage, grow, and harvest crops through the seasons. Careful consideration must be taken when making decisions about machinery usage and the available labor on hand. To help alleviate the tough decision-making process, tools have been created to inform farmers about their machinery and field status. These tools provide useful feedback and large value to farmers looking to plant and harvest. GPS localization and machine state identification provides useful information back to the manager. The tool that was created successfully utilizes GPS data taken from loggers on tractors, combines, and grain trucks to successfully identify the states of all the machines in the field, including, idle, active, on the go, and stationary unload. Initial results of the algorithm provide a 96% success rate in determining the state of the combine during harvest. Additionally, the algorithm was accurate at determining the state of grain carts and grain trucks at the boundaries of the field 94% of the time.

1.2 Introduction

In a world with an ever-growing population and an increase in year over year growth rates, it is imperative that basic human necessities be met for everyone. Chief among these necessities is food and water. Large geographical areas such as the Great Plains of the Midwest provide ample land and area that is suitable for crop growth and the production of foodstuffs. However, in order to create and produce the quantities of harvest that is required, operations and farmers must operate at peak efficiency for the yearly cycle of the land. Commodity crop operations yearly cycles are defined by the planting and harvesting times of their respective crops. It is essential that farmers plant crops at an appropriate time or else the yield will suffer. Roekel and Coulter (2011) showed in Minnesota that a planting delay of two weeks, till May 30, did not significantly impact the yield come harvest time, however when planting was delayed by four weeks, the harvest yield was reduced by an average of 15%. Similarly, it is imperative that harvest operations are performed in a timely manner. With poor weather conditions and large land areas, it is of utmost importance that all machines and workers be operating at maximum efficiency during harvest.

In addition to large land masses which require attention and care in the harvest season, farmers are placed under time pressure due to weather. Coupled with this time pressure, farm managers have seen a steep decline in available workers to man the machinery on the farm. Bronars showed that from 2002 to 2013 “The number of full-time equivalent field and crop workers has dropped by at least 146,000 people, or by more than 20%” (Bronars 2015). Farm managers are faced with the tough decision of how to best manage the full gamut of machinery available to them, in addition, they must also make tough choices for the application of available manpower. With large fields sizes, it is a challenging task to harvest in a timely manner. Managers can be forced to decide between operating additional grain carts or rather utilizing the available manpower for the use of transporting harvest to storage sights.

On farm decision making should be an informed process which utilizes all the available data at hand; in addition, it should account for variables such as weather, yield, and machine throughput to name a few. Machine labor allocation decisions can be complex with a high number of variables to consider in order to achieve the optimal output of the farm system. These systems can have high variability in machine allocation to account for and no decision can be made lightly or without thought. Farming operations will often utilize high cost, high throughput machines such as large combines, grain carts, tractors, and grain carts. This combination of machinery can perform at high rates to fulfill the needs of the operation; however, each machine has a large cost associated with obtaining the machinery and running the equipment in the field. In addition, the problem of in the field operation becomes more convoluted when considering the teamwork and cooperation needed to align all the appropriate parts of the machinery.

One common harvest machinery set is two combines, two grain carts, and four trucks, to remove the grain from the field and transport it to the storage bins and elevators. To achieve maximum output from the machinery selected, the combines field capacity and throughput must be carefully selected and matched with the other equipment in the field to properly utilize the other large machines. Buckmaster and Hilton (2005) showed how important it was to properly evaluate and model each individual system to achieve peak efficiency and utilization. For example, a system capacity could drop by 20% when travelling long distances to unload grain from the grain truck and not enough grain trucks were available to ensure the proper utilization of the harvester in the field. To compound the issue, the combine and grain cart can be used for different purposes come harvest time. Combines will regularly operate both in corn and in soybeans each calendar year. As

such, the output of the combine will drastically differ for both crop types as the bushels per acre for corn and soybeans are largely different. Also, the combine head can vary in width for both corn and soybeans. It is challenging to optimally manage and sync all the machinery that is used in the field. Farmers and managers are always looking for more tools to be able to inform their decision making so that they can properly utilize their fleet of machines. These same tools can also be useful when it comes time to purchase new equipment. A large portion of farmers seek to mitigate the risk that is associated with not being able to fully harvest the crop by purchasing the largest harvester that is available to them in the budget, however, optimization tools could provide the knowledge and insight to make the choice to downsize the harvester or grain cart thus saving money for the farm in the form of upfront cost.

With farmers seeking optimization tools, it becomes imperative to supply and create the methods necessary to make informed decisions about the utilization of all the implements in the field. This can be seen in the various works that are already in the field of agricultural engineering and harvest optimization. Busato and team sought to improve and optimize the logistics for silage operations with the use of a linear programming model. (Busato et al. 2019) This model was used to determine the proper number of units in the field for silage operations, combines and carts, and had an error rate of 3.15% when comparing the simulation model with the field data collected. In addition, the model was capable of prioritizing optimal silage harvest with time constraints imposed on it.

Amiama focused on the creation of a simulation of silage harvest to provided help in optimal decision making. (Amiama et al. 2015) The model used state machines to inform the simulation as to proper event ordering in the field. The tool was useful in determining bottlenecks in the system due to waiting on trucks to return from silos or other hold ups. The model could efficiently inform farm users as to the appropriate number of self-propelled harvesters necessary or the effects that other variables such as the number of packers could have on the time savings of the harvest operation.

Heizinger generated an algorithm which would rate the transportation efficiency of grain transport from field to silo. (Heizinger and Bernhardt n.d.) The system was effective at determining the rate and optimization of the transporters using set zones in the field which were created by the model makers. Transporting units' efficiency and overall uptime could be evaluated by tracking

the GPS points relative to the infield boundaries which were created and the relations between the machines.

A model was built by Layton which would determine which field locations were harvested by which combine when multiple harvesters were present in the field. (Layton et al. 2017) This model utilized GPS logs to create a history of points which were then assigned to a grid the size of the header of the harvester. Using this grid, the model could successfully determine which points of the field were harvested by which combine.

Tools for modeling the transportation from the edge of the field to the storage bins were also considered. Turner made a discrete event simulation model to determine the transportation of grain to the storage facility.(Turner 2018). The system could then be used to evaluate the performance trucks and their overall efficiency when matching truck size and speed to grain harvesting capabilities of the combine.

Zhang created a neural network that could identify and apply the appropriate activity state to data collected for combines (Zhang et al. 2017). In addition to the creation of a neural network, a rule-based algorithm was applied to identify unloading, loading, and harvest states to combines and grain carts. The rule-based algorithm required heavy inputs from expert knowledge to inform states while the neural network could be easily added onto and expanded.

Bochtis showed how vehicle routing problems could be applied to in field operations of combines, and grain carts (Bochtis and Sørensen 2009). Thus, the paths of combines could be solved using previously developed methods for problems such as the travelling salesman problem. This allowed for better path planning in field for combines and carts.

Haffar instrumented a model to inform farming operations the proper selection of machinery needed to optimize the operation. (Haffar and Khoury 1992) The model can consider prior owned machinery and selects the least annual cost solution that is available to meet the harvest requirements. Additionally, the model can output schedules for fields with associated machinery required for each field.

Work was done by Zhang to identify transfers between implements and storage bins using traceability trees. (Zhang et al. 2020) This work built on tree data structures for the GPS logs that were taken in the field. The traceability trees are utilized in tracing harvest from the field, through the implements used in harvesting, and into the storage bins or final destination of the grain.

From the prior work, it becomes clear that there are many tools or concepts of tools to help farmers make informed decisions about their fleet, however, a large portion of these models are either predictive or difficult to set up and fully utilize. There is space for a model that uses simple inputs such as GPS that can inform managers how all the machines are performing on a daily basis. Creating a model that uses simple GPS points would be easy to implement as GPS modules are most likely already installed on machines in the field. Such a tool would also be able to look at historical GPS data to calculate state information and inform users of potential time savings from alternative decisions. It would also have the capacity to identify idle times and track unloading events both in the form of "on the go unloads" and stationary unloads. Such a tool would provide useful information to farmers to make informed decisions about the overall efficiency of their fleet in the fields.

1.3 Objectives

The goal of this project was to determine how accurately harvest states could be determined from time-series location data. To achieve this, the following objectives were identified:

1) Benchmark Data Collection: Collect data for the use of accurately identifying and labeling spatial temporal machinery state data for all equipment in the harvest operation, including, an idle, active, on the go, and stationary unload state for the combine, and a waiting, transporting, on the go, stationary unload, and cart unload for the grain cart
2) Algorithm Development: Generate an algorithm to determine machinery states using time-series location data and heuristic rules
3) Algorithm Optimization: Tune the developed algorithm parameters to reduce estimation error

1.4 Methods and Materials

1.4.1 Data Collection

Benchmarking data was collected from grain harvest machinery used at Purdue's 1600-acre Agronomy Center for Research and Education (ACRE) farm in north central Indiana. The data was collected during the 2021 grain harvest and was comprised of meta-data in the form of daily fields notes, machinery J1939 Controller Area Network (CAN) bus data, and GPS location

data from all machines used in the harvest operation. Machinery included two grain trucks (~35m^3), two John Deere combines (S660 and S670), and one John Deere tractor (8R370) and grain cart (Brandt 1020XR) combination.

Data collection hardware consisted of Kvaser Memorator Pro 2xHS (Kvaser Inc., Sweden) loggers for machines with a CAN bus and Columbus V-990 Mark II GPS loggers (Columbus, Hokkaido, Japan) were used to receive and store time-series location data for machines without CAN bus technology. The Kvaser loggers were powered directly from the J1939 CAN bus port and automatically recorded data when bus traffic was detected. The Kvaser logger stored all the messages published on the CAN bus in a proprietary .kfm file on a SD card. After the data was collected a database (.dbc) file was loaded into the Kvaser Memorator Config Tool program a used to decode the raw data and store in a CSV file.

Figure 1.1: Columbus V990 Logger and Kvaser Memorator Pro 2x HS GPS Logger

The Columbus logger stored GPS data directly to a CSV file. The loggers were powered via micro-USB and featured a battery backup. The loggers featured a motion detection mode that started logging when motion was detected and stopped the log when machines went idle. This allowed for robust data collection that did not require operators to activate the loggers and drastically cut down on the points that were logged This was especially useful for machines such as grain trucks which could be sitting waiting in the field for extended periods of time. The Columbus could receive and capture GPS streams from multiple satellite systems. These systems included the Russian Glonass system, the Japanese Quasi-Zenith Satellite System (QZSS), and the American GPS system. Because of this redundancy, the unit was fit to capture location to a more

accurate degree. In total, the unit had approximately 4m accuracy when viewing an average number of satellites (~8).

Data was collected from four production fields (see table) ranging from 12.4 to 75.9 hectares. In two of the fields (58 and 200) both harvesters were used simultaneously, while in the two others (57 and 70) only the S660 was used. For corn harvest the S660 was equipped with an eight-row head (6.096 m). In soybeans the S660 and S670 were equipped with a 9.14m and 6.10m grain heads, respectively. Both combines had 10.53m^3 capacity grain tanks and augers capable of unloading at 0.134 m^3/sec. The grain cart had a capacity of 35.24m^3 and an unload rate of 0.29m^3

Table 1.1 Field Identification, Size, Crop, and Harvester Used

Field ID	Size (Hectares)	Crop	Harvester
57	18.9	Soybeans	S660
58	12.4	Soybeans	S660 & S670
70	20.2	Corn	S660
200	75.9	Soybeans	S660 & S670

Harvest in all four fields was supported by the grain cart and two trucks. All five machines were equipped with logging equipment. As can be seen in Table 1.1 the S660 was responsible for harvesting in all four fields while the S670 was used in only two of the fields. The harvesters and grain cart tractor were equipped with the Kvaser loggers. For the combine harvesters this allowed for messages relevant to the combine operational status to be collected such as engine rpm, wheel speed, auger status, and highly accurate GPS data. For the grain cart messages were collected for the GPS location, speed, PTO rpm, and heading. The harvesters and grain cart tractor also featured real time kinematic (RTK) differentially corrected GPS receivers that were capable of centimeter level accuracy. The grain trucks did not have a CAN bus and were instead equipped with the Columbus V990 Mark II GPS loggers.

1.4.2 Data Processing

Before the data was used to benchmark the model, the individual machinery logs needed to be processed into files for each crop field harvested, containing the time synchronized machine

state data for all the machines. This required creating a data processing pipeline to convert the Kvaser and Columbus logger files to a standard format, merge, and then spilt by harvested field.

Kvaser Data Sub-Pipeline

The Kvaser loggers used on the harvesters and grain cart tractor stored data in a proprietary binary format. The data was filtered and decoded to the desired signals using the Kvaser Memorator Config Tool Program and a J1939 database file. The resulting CSV files contained the time-series georeferenced machinery state data desired but still in separate files for each machine in a format that was not conducive to merging. The CSV files were processed with a created Python function, found in the appendix, that reformatted the data, specifically, merging the individual time and date component fields into a Datetime data class recognized by Python. The resulting dataset consisted of georeferenced machine information at a 1 Hz interval with a Python readable datetime stamp.

Columbus Logger Data Sub-Pipeline

The data logs for all machines were first exported into comma separated value sheets so that other programs such as Python and Excel could read and write to the files. The data was imported into Excel for minor corrective errors such as number formatting and date formatting. Data types that would not be useful to the algorithm were stripped from the logs to reduce the overall size of the data logs. Data that was corrected included GPS based speed readings for Columbus loggers, and other extraneous information. Included with the removal of data, minor corrections were made such that the data was in a more easily readable format. GPS latitude and longitude points were properly formatted into long float point convention. East and West was removed from the data points and proper sign convention was applied. In addition to the GPS point formatting, time formatting was required. Time logs were formatted to the same convention, yyyy-mm-dd hh:mm:ss , for all of the data files. This change was implemented both for the Columbus loggers and for the Kvaser CANbus loggers. Because of this time formatting, data points were aligned to the correct time across the various machines in the field. In addition to time formatting, the logs were split into day-by-day logs such that each file contained all the points for each harvest event. These logs were then time shifted into the correct time zone being eastern central time.

Some harvest events were split across two days as the workers went on into the night past midnight. Manual checking was performed across the full range of dates to ensure that no data was lost to a cutoff around midnight. A visual representation of the pipeline can be seen in Figure 1.2

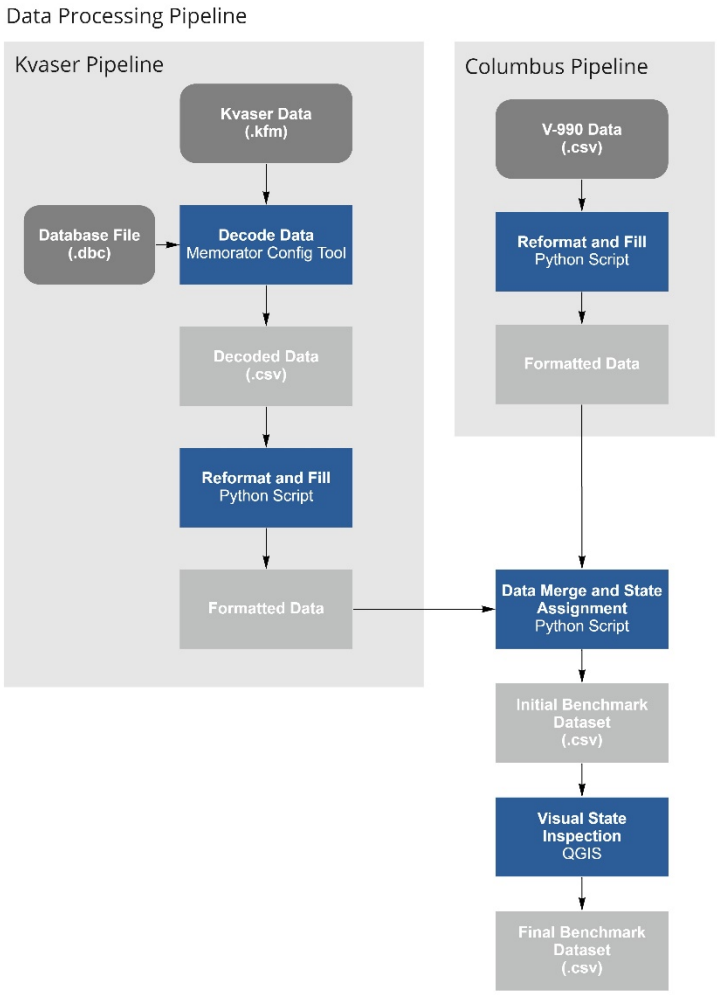

Figure 1.2: GPS data pipeline of transformations

Data Merge and State Assignment

All data points were migrated into QGIS, a program for the visualization of locational data, to verify the validity of the now-cleaned data points. Visual inspection, utilizing QGIS, was done of the points to ensure that all GPS points fell within appropriate field boundaries and no errors were missed when removing unwanted data. Once in QGIS, the data was separated into the four various field boundaries which were identified earlier. Extra data was corrected to further simplify

the data set. This extra data came in the form of transportation data outside the scope of working time in the fields. When powered on, these machines loggers were activated and tracking data, however the combines or grain carts sometimes were not yet in the fields harvesting. Data points of the machines moving from barn to field were corrected. Thus, extraneous transportation data of the day staring up was corrected in the data set and did not inform the model. This practice was also used regarding the end of the day, where excess data points of the returning grain carts and combines was corrected to ensure a clean and accurate data set.

The remaining dataset could then be imported into Python for the analysis and creation of the state algorithm. The goal was to create an algorithm which could identify the states of each machine as it moved throughout the workday. In order to accommodate this algorithm creation, a few more data transformation measures needed to be applied. GPS points were collected in decimal degrees, a useful information from all the implements was the transformation from decimal degrees into a speed using the clock cycle of the data points. Equation 1 was used to transform each subsequent pair of GPS points into a useable speed for the machine.

Equation 1

$$\text{Speed (kmh}^{-1}) = 6371.37 * 3600 * 2 * \arcsin\sqrt{\sin\left(\frac{lat2 - lat1}{2}\right) * 2 + \cos(lat2) * \cos(lat1) * sin\left(\frac{lon2 - lon1}{2}\right) * 2}$$

The speeds for all implements were calculated and expressed as kmh^{-1}. In addition to the speed calculation, a moving average was utilized to ensure the data was resistant to position errors or missing data points. A five-point windowed moving average was used thus ensuring smooth speed transitions and no erroneous jumps in speed.

Truth Data State Assignment

With a useful speed, work was started on creating a state identification algorithm for all the implements in the field. Each machine only has a set number of states that it could be in at any given time thus identifying the total states of all the machines was the first step in creating the algorithm. The harvester was selected as the first machine to have states created. As all the other

machines were dependent on the workings of the harvester, its states were of the upmost importance. The following states seen in Table 1.2 were identified for the combines in the field.

Table 1.2: Combine Operating States

State:	Description
Working	Moving in the field or harvesting
Idle	No movement in the field and not unloading
On the go	Unloading at speed into the grain cart while harvesting
Stationary	Unloading into grain cart while remaining stationary
NA	Catch all for NA values in data set

This encapsulated all the states that the combine could be in at any point in time while in the field. Of second importance, the grain cart was crucial in connecting the combine and the trucks.

Table 1.3: Grain Cart Operating States

State:	Description
Waiting	No movement and no unloading loading from combine
Transporting	Transporting grain to field edge or moving to unload combine
On the go	Receiving grain from combine while moving in the field
Stationary combine unload	Receiving grain from combine while stationary
Stationary unload to truck	Unloading grain into truck while at field edge
NA	Catch all for NA values in data set

These states, shown in Table 1.3, fully captured all the available operating states the grain cart and tractor have in the field. Lastly, the states of the grain trucks were identified and quantified and can be seen in Table 1.4.

Table 1.4: Grain Truck Operating States

State:	Description
Waiting	No movement or unloading from grain cart
Transporting	Movement along field boundaries, transporting grain to bins
Stationary loading	Receiving grain from grain cart while stationary
NA	Catch all for NA values in data set

Not included in the grain truck state list was unloading from the truck to a grain bin or commercial elevator. The focus of this work was to understand the in-field machinery states and how the machines in the system effected each other's states. For this, the value of the truck data was to know if the grain cart or combine was idle because the truck had not returned, not the states of the truck after leaving the field.

To compare the collected data with the results of the state algorithm, a truth data set needed. This "truth data" was created using the other information the Kvaser loggers had collected mainly the PTO rpm, speed, and auger location. With the information that was collected, the states were verified by comparing the true data with the state tables that were created by the algorithm. The truth data identified states based on the following requirements seen in Table 1.5.

Table 1.5: Combine Truth Rules

Combine Truth States		
Idle	Speed < .048 kmh^{-1}	Auger Off
Working	Speed > .048 kmh^{-1}	Auger Off
On the go	Speed > .048 kmh^{-1}	Auger On
Stationary unload	Speed < .048 kmh^{-1}	Auger On

Additionally, requirements were placed on the grain cart to identify the true states which can be seen in Table 1.6.

Table 1.6: Grain Cart Rules

Grain Cart Truth States		
Waiting	Speed < .048 kmh^{-1}	Auger Off
Transporting	Speed > .048 kmh^{-1}	Auger Off
On the go	Speed > .048 kmh^{-1}	Combine state = on the go
Stationary combine unload	Speed < .048 kmh^{-1}	Combine state = stationary
Stationary unload to truck	Speed < .048 kmh^{-1}	Auger On

Truth data could not be created for the grain trucks as the GPS logging that was created was based on the mark II logger which did not have access to CANbus data that the combine and grain cart had. As such optimization of the algorithm centered on the grain cart data and the combine data. In addition, the grain truck data was robust to state changes as the majority of the time was spent either in transit to grain bins or stationary at the field edge.

1.4.3 Algorithm Creation

Using the predefined states in Table 1.3 and Table 1.4 work was begun on creating an algorithm which would identify the states of all the machines in the field based on the time-series location data and a set of heuristics rules.

Rule Generation for State Estimation

Rules were generated for each of the states that were available for the machines in the field. To accomplish this, the rules shown in Table 1.7 were created for the combine states. Also, the grain cart states were made with the rules shown in Table 1.8. Lastly, the states for the grain trucks were created and are seen in Table 1.9.

Table 1.7: Combine State Rules

State:	Rule
Working:	Measured speed > Idle Threshold
Idle:	Measured speed < Idle Threshold and distance to cart > Coordination Threshold
On the go:	Measured speed > Idle Threshold and distance to cart < Coordination Threshold and relative speed < Relative speed Threshold
Stationary	Measured speed < Idle Threshold and distance to cart < Coordination Threshold and relative speed < Relative speed Threshold
NA:	Catch all for NA values in data set

Table 1.8: Grain Cart State Rules

State:	Rule
Waiting	Measured speed < Idle Threshold and distance to combine > Coordination Threshold and distance to truck > GPS cart range
Transporting	Measured speed > Idle Threshold and distance to combine > Coordination Threshold and distance to truck > Cart coordination threshold
On the go	Measured speed > Idle Threshold and distance to combine < Coordination Threshold and relative speed < Relative speed Threshold
Stationary combine unload	Measured speed < Idle Threshold and distance to combine < Coordination Threshold and relative speed < Relative speed Threshold
Stationary unload to truck	Measured speed < Idle Threshold and distance to truck < Coordination Threshold cart range and relative speed < Relative speed Threshold

Table 1.9: Grain Truck State Rules

State:	Description
Waiting	Measured speed < Idle Threshold distance to cart > Cart coordination threshold
Transporting	Measured speed > Idle Threshold distance to cart > Cart coordination threshold
Stationary loading	Measured speed < Idle Threshold distance to cart < Cart coordination threshold
NA	Catch all for NA values in data set

Rule Application to Data

The algorithm would iterate through the combined data set of all the GPS points for the machines in the field labeling each point with the appropriate state. From these initial results, a table of states was created for all the machines as shown in Table 1.10

Table 1.10: Example Data Log

	S660 Combine	Grain Cart	Truck
State: time 10:01:02	Working	Transporting	Waiting
State: time 10:01:03	Working	Transporting	Waiting
State: time 10:01:04	Idle	Transporting	Waiting
State: time 10:01:05	Working	Transporting	Waiting
State: time 10:01:06	On the go	On the go	Waiting
State: time 10:01:07	On the go	On the go	Waiting
State: time 10:01:08	On the go	On the go	Waiting

This initial run would only look at the current GPS location of all the field implements and the speeds of the machines. However, from this, slight errors would come about due to the unstable nature of the speed and GPS accuracy. This would induce faults in states as can be seen by the erroneous “idle” in the S660 combine states. While this point should have been a harvest point, the GPS location may have wavered enough to reduce the speed below the idle thresh hold thus

inducing an idle point. To correct for these errors the state tables were modified to highlight errors in the data.

The state table was changed from a list of states and converted into a list of concurrent states. This would provide the model with a list of how long the machine had been in that specific state. From this table of concurrent states, a history of how long each state was active was created. This was helpful for the removal of incorrect states.

Table 1.11: Concurrent State Count Log

S660 Combine		Grain Cart		Truck	
State	Time in State (s)	State	Time in State (s)	State	Time in State (s)
Working	629	Transporting	200	Waiting	700
Idle	2	Waiting	329	Transporting	200
Working	122	Transporting	66	Waiting	203
On the go	95	On the go	95	Transporting	504
Working	273	Transporting	105	Waiting	192

Table 1.11 highlighted that the idle count for the S660 does not provide much valuable information to the model. The event most likely came about by a reduction in speed for something such as a cornering event. Thus, the idle count can be corrected via a "history check". The goal of the check was to remove any state that lasted less than a certain timeframe and did not provide useful information. The combine or trucks and carts should be in a single state for longer than a set timeframe for it to be considered an event such as unloading on the go. Thus, any state counts that are small enough can be corrected in the state table and the unnecessary values can be either back filled or forward filled to correct for the values.

Rule Parameter Optimization

The rules that determined the various states were tuned on a selected field to "train" the model to have better results. Field 70 was selected to tune the algorithm. Field 70 harvest data featured little overlap or unnecessary movement made by the combine. In addition, each state was represented well in the data for the field. Stationary unloads were performed at the edge of the field, also, the combine unloaded on the go multiple times into the grain cart.

The rules for each state were tuned around a set of variables that had a large impact on the output of the system. As such, the following variables in Table 1.12 were selected to tune the algorithm around.

Table 1.12: Variables for Tuning of State Algorithm

Criteria	Description
Idle Threshold	Value for determining idle vs movement
Relative Speed Threshold	Value for determining relative speed between machines
Coordination Threshold	Value for determining if combines and grain carts were “near” or “far”
History	Value for how many consecutive states required for it to be a true state
Cart Coordination Threshold	Value for determining if grain carts and trucks were “near” or “far”

Each variable was tuned from a list of selected values that can be seen in Table 1.13.

Table 1.13: Tuning Variable Range

Criteria	Candidate Values
Idle Threshold (kmh^{-1})	.40, .80, 1.21, 1.61, 2.01, 2.41
Relative speed Threshold (kmh^{-1})	.20, .40, .80
Coordination Threshold (Decimal degrees)	.00014, .00015, .00016, .00017, .00018, .00019
History	0, 1, 2, 3, 4, 5, 6, 7
Cart coordination threshold (Decimal degrees)	.00016, .00017, .00018, .00019

The algorithm was then run through each possible combination of the variables in Table 1.13. The output of the algorithm was then checked against the truth data that was compiled. The checking was done in an event-by-event manner. The truth data for each event was checked against the matching time of the algorithm output. Events that matched across the truth data and algorithm

output were labeled as correct. The best set of variables that had the most matching states between the algorithm output and the truth data was selected for use and testing the other three fields.

Further exploration of the tuning variables was performed to identify the effect that certain variables had on the output of the algorithm. The overall output of the tuning algorithm can be seen in Figure 1.3.

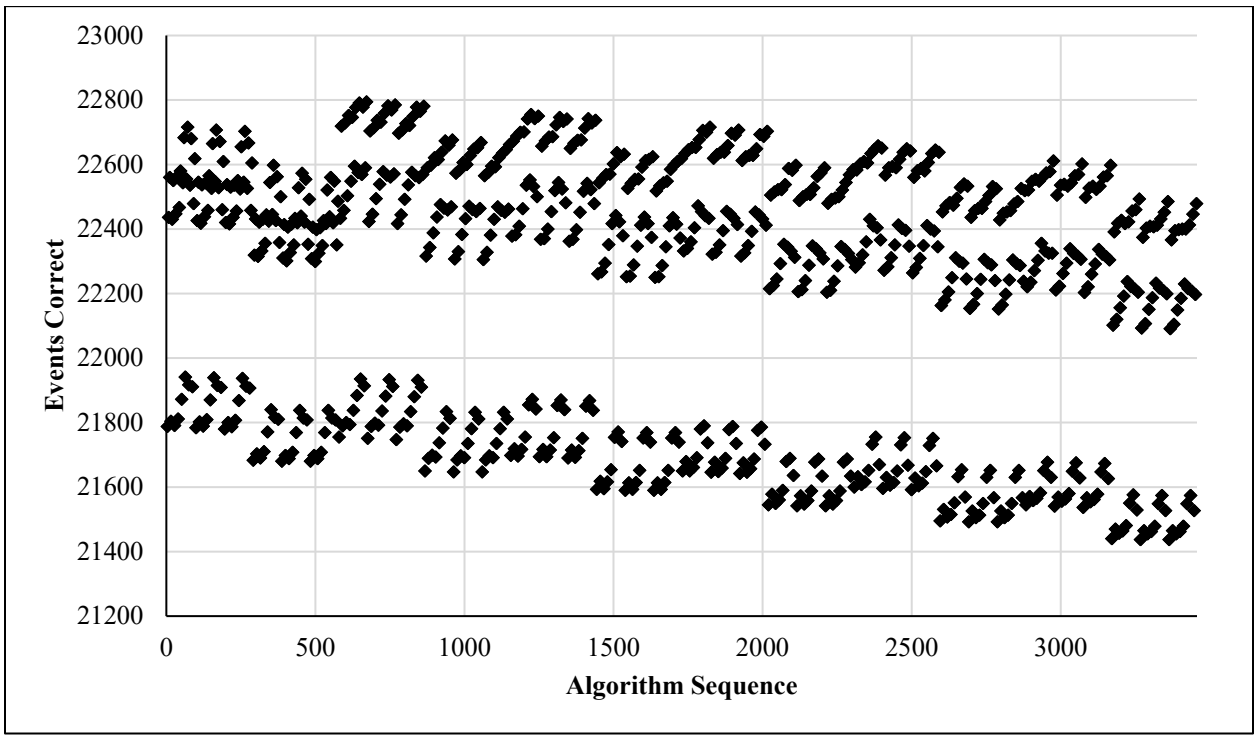

Figure 1.3: Maximum output of tuning conditions for all variables of state algorithm

The visualization highlights some anomalies in the tuning parameters. The large gap that is in the data occurred due to the "Relative Speed Threshold" parameter. At 0.2 kmh^{-1} the algorithm was not able to identify the coordinating states between the grain cart and the combine due to such a tight restriction in speed. This bias caused the large gap that can be seen separating the maximums and minimums in Figure 1.3.

In addition to the relative speed threshold, the idle speed threshold also played large part in the total maximum of the algorithm. The idle speed threshold was critical in determining if the combine and grain cart were experiencing movement or not. The parameter had large impacts on

the overall output of the algorithm. The idle threshold was isolated and the effect it had on the total maximum of the algorithm can be seen in Figure 1.4.

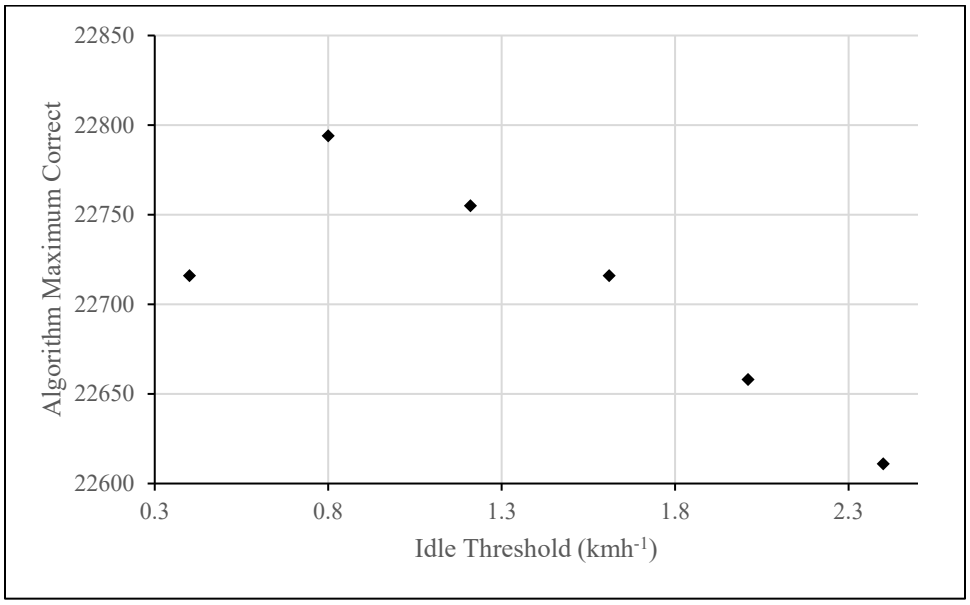

Figure 1.4: Maximum Output vs Idle Threshold

It was evident that the speed had a sharp decline at 0.8 kmh^{-1} with a local maximum at the point. The effects that the speed had can also be visualized and seen in Figure 1.3 where the overall slope of the figure is trending downwards across the sequence. Also of note, the coordination threshold for the combine and grain cart played a large role in tuning for the maximum output.

The coordination threshold was critical in determining if the combine was interacting with another machine in the field. The distance between the two machines played a large role in determining which state the machines were in.

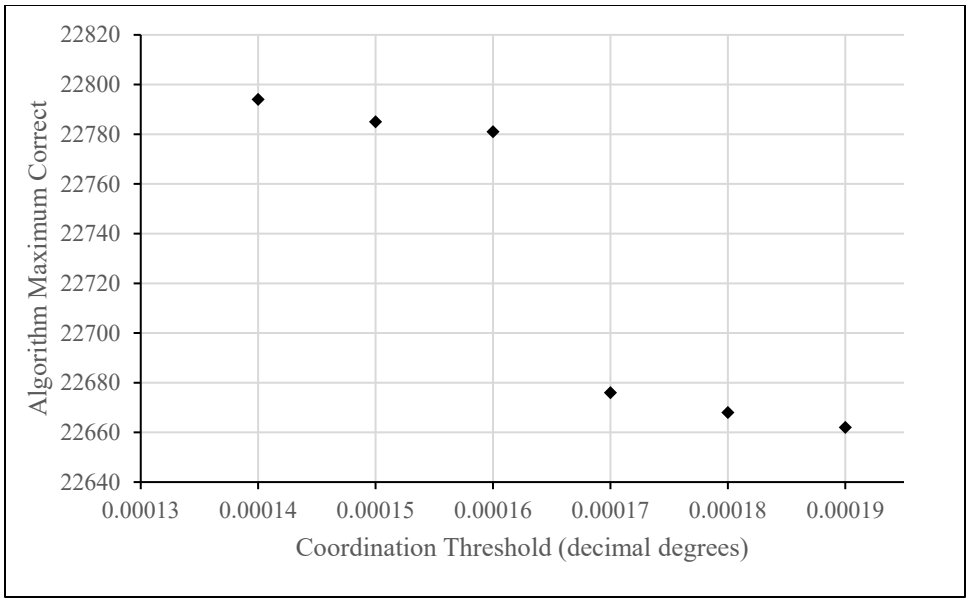

Figure 1.5: Maximum Output vs Coordination Threshold

The effect that the coordination threshold had can be seen in Figure 1.5. A sharp drop off can be seen at .00016 decimal degrees. At .00016 decimal degrees the machines are approximately 18 m away from each other. The cutoff at that distance points to a common distance that the machines are when interacting with each other. Because of this, it can be strongly inferred that the machines interact with each other within an 18m radius and a coordination threshold larger than 18m would introduce errors where the machines are close but not interacting.

1.5 Results

The algorithm was run on all four fields that were selected for analysis. Each field that was analyzed had distinct differences in either the size, crop, or machinery used. The fields were harvested between September 29th and October 2nd, 2021.

1.5.1 Parameter Tuning Results

The first field that was analyzed was field 70. The field had an appropriate number of unload events and, it had simple track lines from end to end. As seen in Figure 1.6 there was very little wasted movement which made this a good candidate for the tuning of the algorithm.

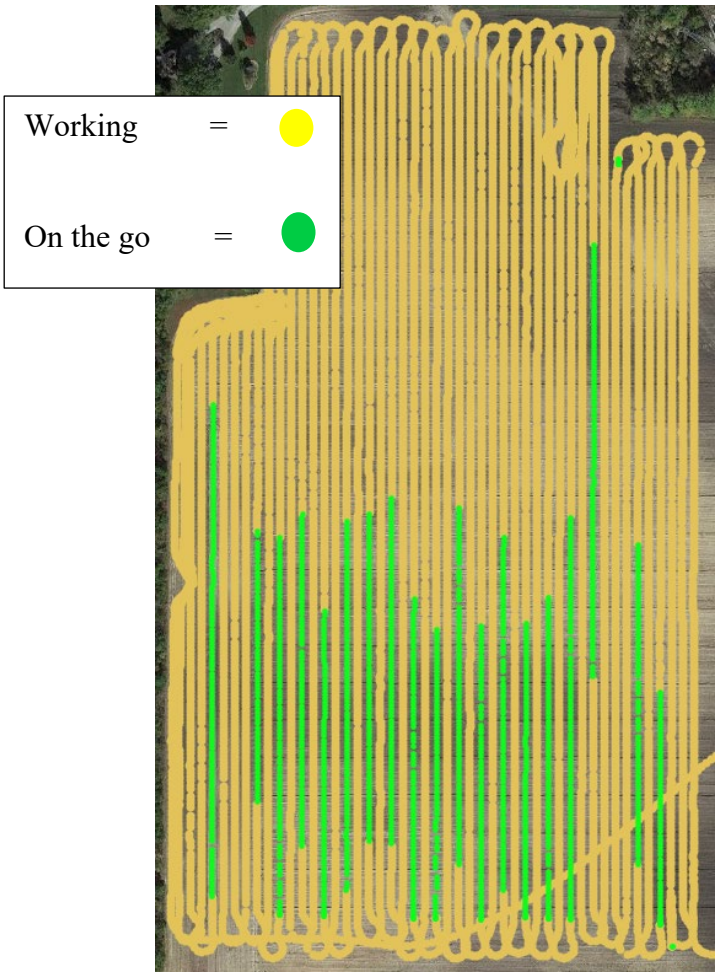

Figure 1.6: Field 70 combine working state and on the go in field paths

The optimization from field 70, and the configuration results can be seen in Table 1.14: Tuning-Selected Parameters

Table 1.14: Tuning-Selected Parameters from Field 70 Harvest Data

Idle Threshold	.8 kmh^{-1}
Relative speed Threshold	.8 kmh^{-1}
Coordination Threshold	.00014 (~16m)
History	2 values
Cart coordination threshold	.00016 (~18m)

Utilizing the selected parameters, the script was run for the data set and the states were assigned to the GPS points for the whole day. The results of the tuning run for the combine can be seen in Table 1.15: Field 70

Table 1.15: Field 70 Tuning Results for Combine with Truth Data Comparison

State	State Algorithm Predicted	State Algorithm Correct	False Positive	False Negative	Percent Error
Working	10175	9969	206	152	1.50%
Idle	172	146	26	34	18.90%
On the go	1672	1545	127	218	12.37%
Stationary Unload	279	217	62	17	7.26%

This field provided a good benchmark of the state algorithm's ability to identify the working state of the machines. The truth data had an adequate amount of all four main states. In total from the verification against the truth data, the state algorithm was over 96% accurate when determining the status of the combine. Overall, only 421 points were missed out of 12,000. The largest percent error was in the idle state, however, in total only 34 points were missed for the idle state. The ide error was accounted for in the abnormal unload behavior. The combine would unload on the go till the end of the field at which point it would unload in a stationary manner. After that, the combine would sometimes sit idle for a few seconds next to the grain cart, thus the state algorithm logged the behavior as stationary unloading rather than idle. The unload on the go to stationary behavior can be seen in Figure 1.7: Field 70 unload

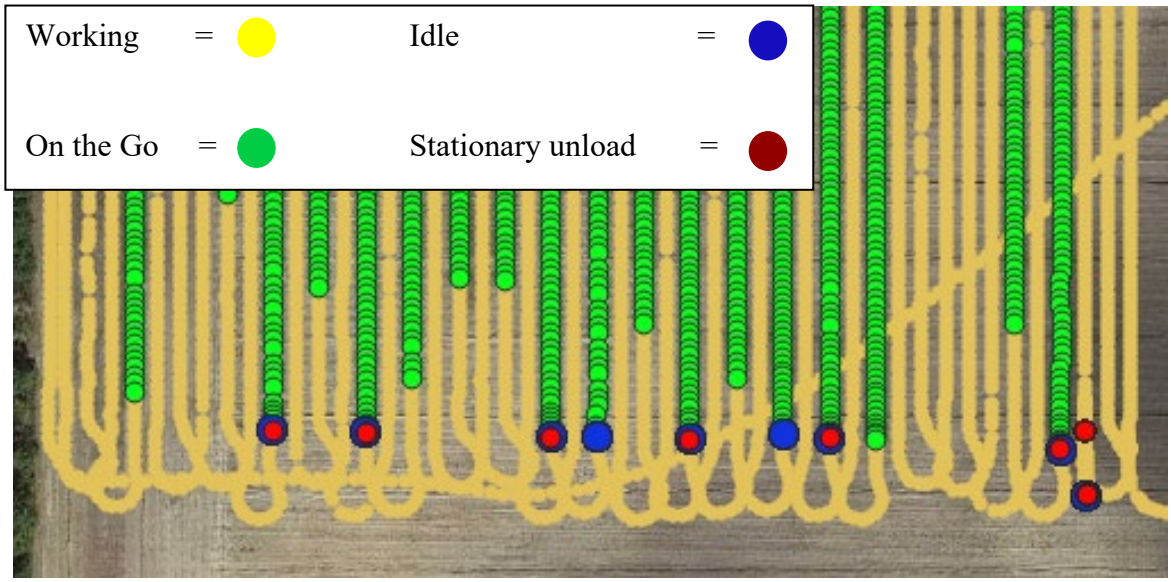

Figure 1.7: Field 70 unload on the go to stationary behavior

In addition to the on the go to stationary error, the state algorithm had small issues identifying on the go states. A good example of the error can be seen in Figure 1.8.

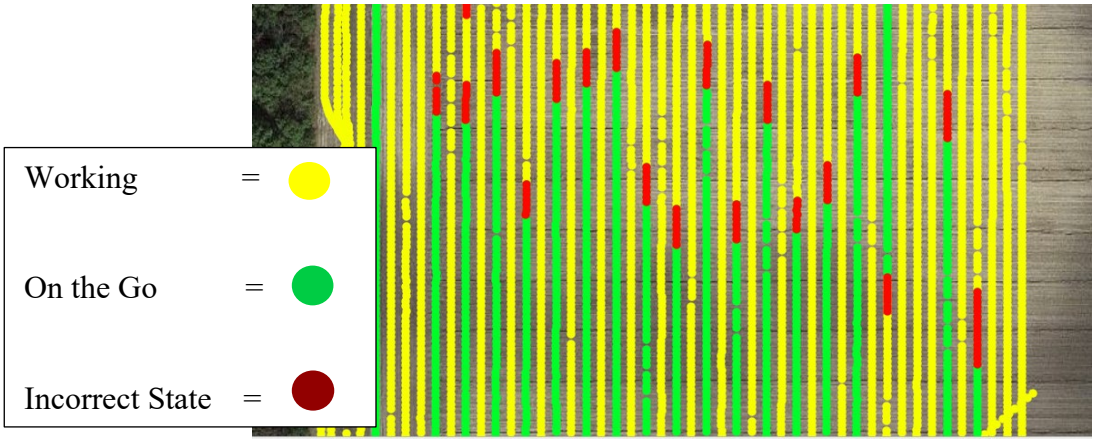

Figure 1.8: Map showing leading and lagging errors in field 70

The leading and lagging error was common in the analysis of the field. This error occurred during the alignment of the grain cart and the combine. The grain cart driver could arrive at the combine and be ready to receive the grain long before the button to unload was pushed. As such, the model struggled to identify at what time the combine began to truly unload into the grain cart. Similarly, after the unload process was complete the drive of the grain cart could wait for a few seconds

before pulling away. When observing from GPS points, there is no distinguishing moment to identify the shift in state. The variance in alignment time between grain cart and combine caused the state algorithm to incorrectly label certain points.

The grain cart for the field was also tracked using the state algorithm and the logs can be seen in Table 1.16

Table 1.16: Field 70 Grain Cart State Analysis

State	Truth Data Values	State Algorithm Predicted	State Algorithm Correct	False Positive	False Negative	Percent Success
Transport	4543	4483	4259	224	284	93.75%
Waiting	4385	3912	3742	170	643	85.34%
Grain Cart Unload	1373	1949	1311	638	62	95.49%
On the go	1763	1662	1556	106	207	88.25%
Stationary unload	234	292	224	68	10	95.73%

The states for the grain cart had a fairly large success rate at over 90% accuracy. The largest percent errors occurred at the waiting state rather than one of the relatively smaller sized states. This behavior was due to the grain cart resting near the trucks when not in motion. This placement of the two implements caused the state algorithm to fail by reducing the waiting state and increasing the grain cart unload state.

1.5.2 Algorithm Validation Results

Field 200

The state algorithm was optimized using field 70. To verify the results of the optimization, the state algorithm was run on the three remaining fields to identify the accuracy of the model.

Field 200 was a large field that had all the potential states included. The field was harvested by two different combines and thus was split across two logs. The paths of the combines can be seen in Figure 1.9 and Figure 1.10

Figure 1.9: Field200 S660 harvester tracks

Figure 1.10: Field 200 S670 harvester path

The output of the machine state for the S660 can be seen in Table 1.17 Field 200 S660 State and the S670 in Table 1.18.

Table 1.17 Field 200 S660 State Analysis

State	Truth Data Values	State Algorithm Predicted	State Algorithm Correct	False Positive	False Negative	Percent Correct
Working	32650	32298	32090	208	560	98.28%
Idle	2875	2569	2211	358	664	76.90%
On the go	1171	1005	871	134	300	74.38%
Stationary Unload	1086	1910	1050	860	36	96.69%

Table 1.18: Field 200 S670 State Analysis

State	Truth Data Values	State Algorithm Predicted	State Algorithm Correct	False Positive	False Negative	Percent Correct
Working	33344	33081	32905	176	439	98.68%
Idle	3274	3239	2795	444	479	85.37%
On the go	1542	1535	1320	215	222	85.60%
Stationary Unload	565	870	363	507	202	64.25%

Overall, the state algorithm was very accurate for determining the states in the field. Overall, the accuracy of the model was just under 96%. The working state of the S670 had an exceptional 98.68% success rate. The model most likely worked hard to reduce the error count of the working state because of the point difference in the states. The working state accounts for almost 10 times the points as the others and thus the model will most likely work to reduce the error in the working state. In addition to the combine output, the grain cart outputs can be seen in Table 1.19 and Table 1.20 respectively.

Table 1.19: Field 200 S660 Grain Cart States Analysis

State	Truth Data Values	State Algorithm Predicted	State Algorithm Correct	False Positive	False Negative	Percent Correct
Transport	15420	15049	14793	256	627	95.93%
Waiting	16880	16699	15309	1390	1571	90.69%
Grain Cart Unload	3225	3118	2188	930	1037	67.84%
On the go	1171	997	886	111	285	75.76%
Stationary unload	1086	1919	1058	861	28	97.42%

Table 1.20 Field 200 S670 Grain Cart States Analysis

State	Truth Data Values	State Algorithm Predicted	State Algorithm Correct	False Positive	False Negative	Percent Correct
Transport	15489	15000	14738	262	751	95.15%
Waiting	17898	18431	16756	1675	1142	93.62%
Grain Cart Unload	3231	2894	2183	711	1048	67.56%
On the go	1542	1533	1344	189	198	87.16%
Stationary Unload	565	867	366	501	199	64.78%

The model also experienced error when the combine unloaded directly into a grain truck at the edge of the field. The state algorithm has no state for unloading into the truck for the combine, as such, the truth data classified the unload as a stationary unload, however the state algorithm had classified the time period as ide. This accounted for a large portion of extra idle state algorithm labels and accounted for a large portion of the missed stationary unload points, approximately 200. The state algorithm also failed when the combine was stopped and idle directly next to the grain cart. The behavior can be seen in Figure 1.11. Because the two machines were located near each other while idle, the state algorithm continued to mark both machines in the stationary unload point.

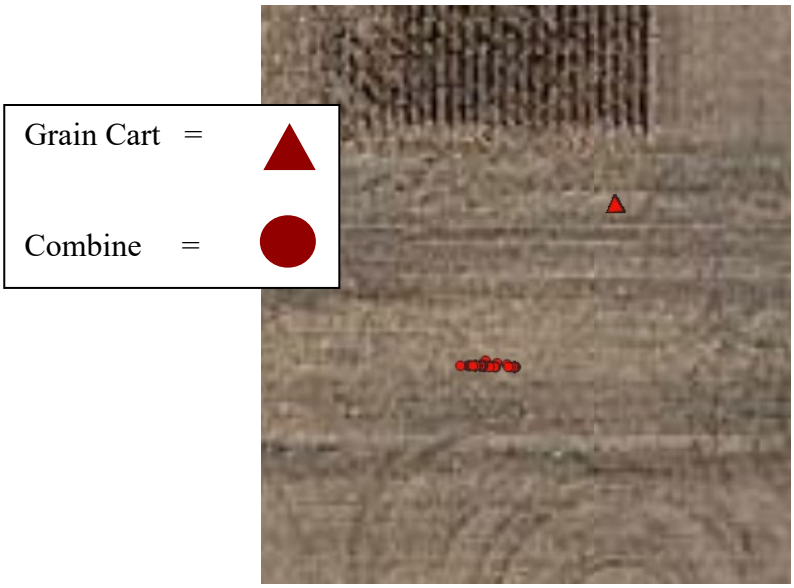

Figure 1.11: Grain cart idle near combine

This behavior accounted for approximately 450 miss-logged idle points in the S670 state table. The points were logged as stationary unload when they should have been labeled as idle. To fix this issue, a state limit could be implemented. Such a limit would kick the state algorithm out of the stationary unload state after a set amount of time and move it into another state. This would limit the number of missed points when the combine and grain cart idle next to one another. Another solution that could help reduce the missed logs for near idling would be the implementation of same heading checks. A same heading check could reduce the errors by not allowing the stationary unload if the two machines do not have the same heading, however implementing that state could cause more errors when the combine does unload in a stationary manner.

Field 58

Field 58 was also harvested by two separate combines. The paths that the two combines took for field 58 can be seen in Figure 1.12.

Figure 1.12: Field 58 S660 and S670 combine tracks

The results of the state algorithm for the S660 and S670 combines can be seen in Table 1.21 and Table 1.22 respectively.

Table 1.21: Field 58 S660 States Analysis

State	Truth Data Values	State Algorithm Predicted	State Algorithm Correct	False Positive	False Negative	Percent Correct
Working	7196	7086	7056	30	140	98.05%
Idle	264	281	208	73	56	78.88%
On the go	331	327	263	64	68	79.46%
Stationary Unload	43	140	43	97	0	100.00%

Table 1.22: Field 58 S670 States Analysis

State	Truth Data Values	State Algorithm Predicted	State Algorithm Correct	False Positive	False Negative	Percent Correct
Working	6834	6719	6697	22	137	98.00%
Idle	802	899	784	115	18	97.76%
On the go	187	192	164	28	23	87.70%
Stationary unload	87	100	80	20	7	91.95%

The field had overall good results for the outcome of the model. With an overall success rate of 96.6% in the S660 field, the model showed good success. In the S670 portion of the field, the state algorithm was accurate at predicting 6697 points out of 7725 only missing 185 points over the whole day. While the combine was in the field for over two hours, the model was inaccurate for only three minutes of log time. The largest percent error for the S670 was the on the go state. In total 23 points were improperly predicted. Of the 23 points, 100% of the misses were due to leading and lagging errors in transition from one state to another. Likewise, in the S660 field 24 of the misses in the on the go state were due to leading and lagging errors. The S660 state algorithm had an error due to the evaluation of the truth data. The threshold for movement was set at .048 kmh^{-1} to evaluate if the combine was moving or not. Because of this, the combine was in the movement state during an unload event, even though it would be classified as a stationary unload by an operator. This behavior can be seen Figure . This on the go unload would have been

better classified as a stationary unload in the truth data, and it accounts for the remainder of the error in the on the go state for the S660.

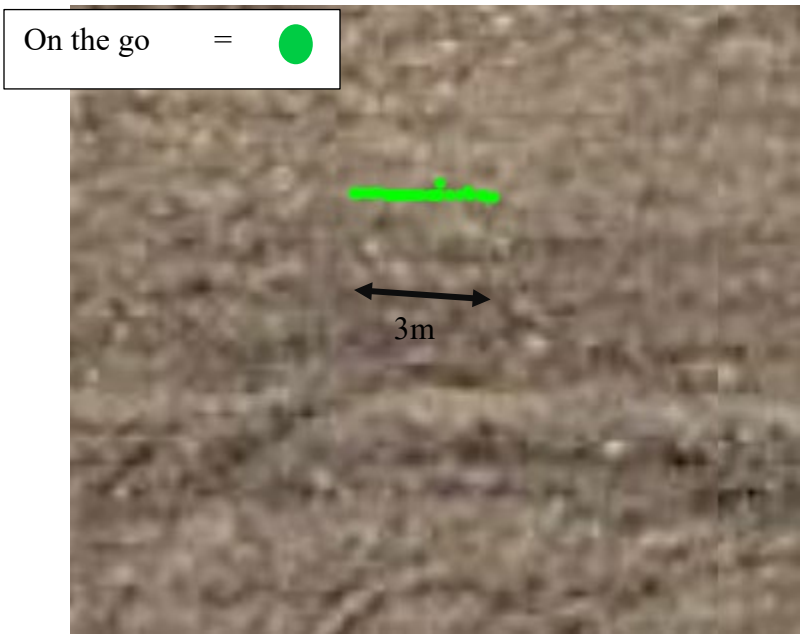

Figure 1.13 Stationary unload points classified as on the go

Table 1.23: Field 58 S660 Grain Cart States Analysis

State	Truth Data Values	State Algorithm Predicted	State Algorithm Correct	False Positive	False Negative	Percent Correct
Transport	2465	2665	2294	371	171	93.07%
Waiting	4565	4933	4502	431	63	98.62%
Grain Cart Unload	460	98	98	0	362	22.80%
On the go	331	328	265	63	66	80.06%
Stationary Unload	43	140	43	97	0	100.00%

Table 1.24: Field 58 S670 Grain Cart States Analysis

State	Truth Data Values	State Algorithm Predicted	State Algorithm Correct	False Positive	False Negative	Percent Correct
Transport	2631	2556	2514	42	117	95.55%
Waiting	4577	4968	4560	408	17	99.63%
Grain Cart Unload	428	101	101	0	327	23.60%
On the go	187	188	166	22	21	88.77%
Stationary Unload	87	97	81	16	6	93.10%

In comparison to the combine state predictions, seen in Table 1.23, the grain cart machine performed adequately as seen in Table 1.24. However, it did not do as well as the combines. In total the machines were accurate 92% and 94% of the time respectively. A large portion of the errors are shared by the grain cart unload state. The grain cart unload was responsible for 300 missed states, however, when the state algorithm predicted an unload into the grain cart it performed at 100% accuracy as seen by the 101 correct logs in the S670 state algorithm. The error for the two machines should appear very similar when the grain cart is not interacting with a combine. The log is identical for the S670 and S660, as there was only one grain cart in the field. The difference in the two predictions lies in how the log was compared to the combine. Because of this interaction, the failed grain cart unload event appears twice in the metrics, one for the S670 and one for the S660 because at that time the grain cart does not interact with the combine.

The failed interaction between the grain cart and the grain trucks was not a byproduct of the state algorithm. Upon visual inspection in QGIS, it was revealed that the grain cart unloaded twice into wagons instead of the grain trucks. The state algorithm did not have a state to account for the interaction between the wagons and the grain cart. From inspection in QGIS 327 points were classified as grain cart unload in the truth data when no grain truck was available to receive an unload. The two unloads into wagons accounted for all the error in the grain cart unload state and also accounted for the excess states that were logged as waiting. Once the points were corrected from the dataset, the accuracy of the grain cart unload state was increased to 100%

accuracy for the S670. In addition, the overall percent error of the grain cart model was reduced to 2.12% which was a much better overall result.

Field 57

Field 57 was a relatively simple field with only unloading on the go. The field was a good benchmark for what a fully optimized combine would look like. During the entire field, the combine was kept moving and it did not have to wait for a grain cart or truck to unload into. The tracks the combine took can be seen in Figure 1.14.

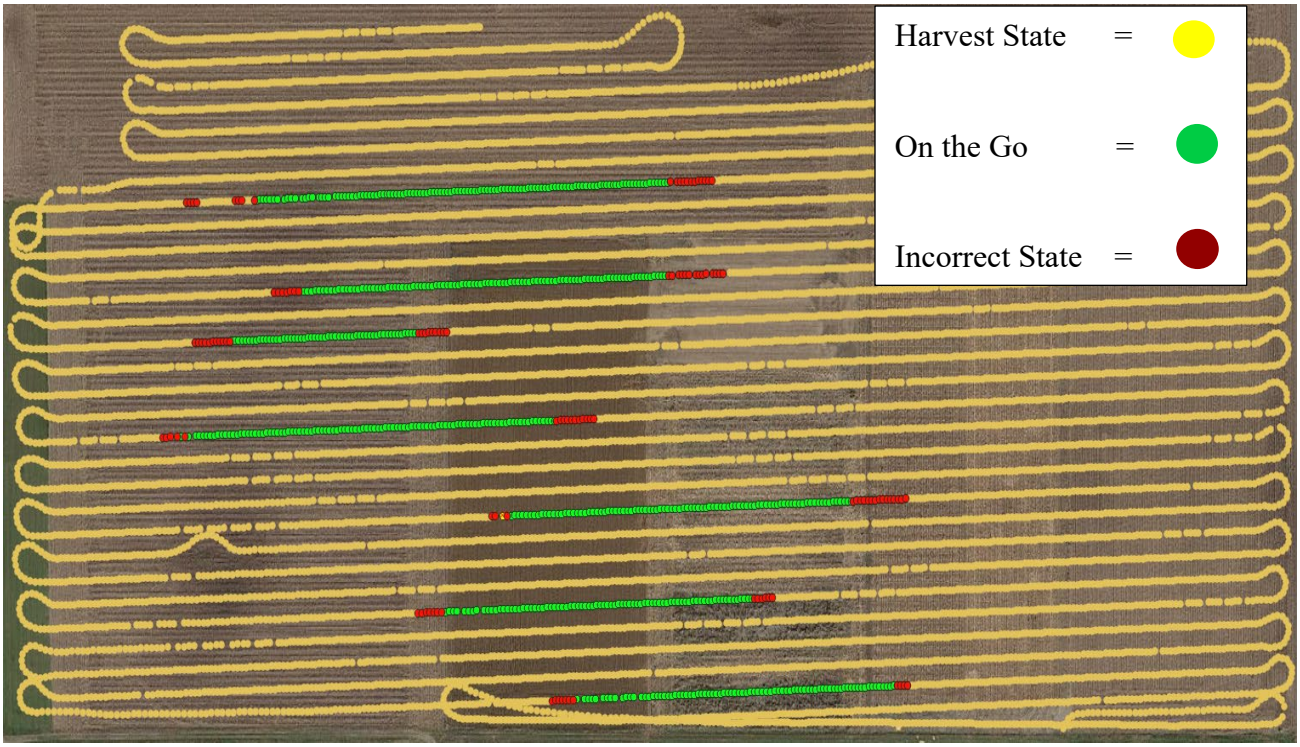

Figure 1.14: Field 57

The results of the state algorithm can be seen in Table 1.25.

Table 1.25: Field 57 Combine States Analysis

State	Truth Data Values	State Algorithm Predicted	State Algorithm Correct	False Positive	False Negative	Percent Correct
Working	9021	8952	8927	25	94	98.96%
Idle	0	0	0	0	0	100.00%
On the go	642	711	617	94	25	96.11%
Stationary Unload	0	0	0	0	0	100.00%

As seen from the results, the combine did not have any idle times during the field. It was fully utilized during its harvest route. The overall success rate of the model was extremely good; it yielded a percent error of only 1.23%. Field 57 specifically highlighted the leading and lagging error that was present for all the fields. A good example of the error can be seen in Figure 1.15

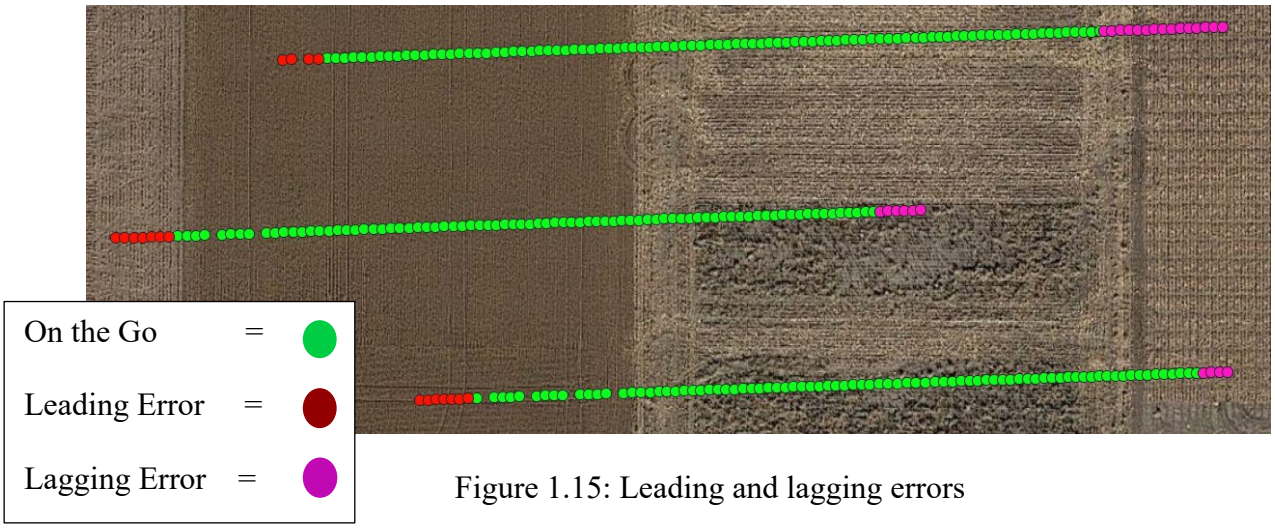

Figure 1.15: Leading and lagging errors

The leading and lagging error was present because of the alignment time of the combine and the grain cart driver. The alignment time between the two machines was never exactly perfect and consistent. Thus, the state algorithm struggled to correctly identify when the combine and grain cart began the unloading sequence. The error did have a unique property, the lag error could

very closely be swapped with the lead error. Most lag error events were offset with the leading error. For every point that was misses at the start due to alignment, an almost identical point was incorrectly added at the end of the unload when the combine and grain cart parted ways. In addition to the combine data, the grain cart results were also collected and can be seen in Table 1.26.

Table 1.26: Field 57 Grain Cart States Analysis

State	Truth Data Values	State Algorithm Predicted	State Algorithm Correct	False Positive	False Negative	Percent Correct
Transport	1713	1609	1527	82	186	89.14%
Waiting	6790	6915	6651	264	139	97.95%
Grain Cart Unload	518	428	409	19	109	78.96%
On the go	642	711	621	90	21	96.73%
Stationary Unload	0	0	0	0	0	100.00%

The grain cart state algorithm performed well overall. The overall success rate of the machine was just under 93%. The greatest error of the state algorithm was in the grain cart unload state. Upon inspection all 109 missed logs of the state occurred at the alignment of the grain cart and the grain truck. The long length of the truck and the placement of the logger in the cab placed the initial unload logs just outside the boundary of the cart coordination threshold. However, the unload event was still captured but, the distance between loggers and the poor GPS quality of the trucks led to the mis-labeling of the initial cart unload states.

1.5.3 Overall Model Results

Overall, the model was very successful in backing out the states from the GPS data alone. The overall model success rate for the combines can be seen in Table 1.27 and was 96.6%.

Table 1.27: Cumulative Fields Multiple Combine States Algorithm Output

State	State Algorithm Correct	Truth Data Values	Percent Correct
Harvest	87675	89045	98.46%
Idle	5998	7215	83.13%
On the go	3235	3873	83.53%
Stationary unload	1536	1781	86.24%

Additionally, the grain cart was also evaluated for total success, sitting at just above 91% correct, the model was very accurate when predicting the state of the grain cart as well as the combine. The overall success rates can be seen in Table 1.28.

Table 1.28: Cumulative Field Grain Cart States Algorithm Output

State	State Algorithm Correct Output	Desired count	Percent Correct
Transport	35866	37718	95.09%
Waiting	47926	50710	94.51%
Grain Cart unload	4979	7208	69.08%
On the go	3282	3873	84.74%
Stationary Unload	1548	1781	86.92%

Utilizing the state algorithm, some interesting data points were derived from the data that would otherwise be unavailable without the CAN bus logs. In total the state algorithm predicted 5998 logs of idle time across the three fields. Converting the time of the logs, the overall idle time of the combine in the field was shown to be just under 100 minutes lost to idling. In addition to the idle times, the combine also unloaded in a stationary manner into the grain cart. In total, the combine sat stationary unloading for 25 minutes, as was predicted by the state algorithm. For the three fields, the combine was not harvesting for over 125 minutes of operational time.

In addition to idle time, the state algorithm was also able to see the impact that unloading on the go had on the harvest operation. The combine unloaded on the go for just under 54 minutes rather than unloading at the edge of the field. Additionally, the total unload time of the harvester was just under 150 minutes for the whole harvest duration. The state algorithm successfully captured 65 out of 65 unload events from the combine into the grain cart. Overall, 20 stationary

unload events were captured and 45 on the go events were successfully reported. As a check, the on the go and stationary unload total time log aligned with the total count of unload events. Each unload event took approximately 70 seconds which aligned with the auger of the combine.

Useful field metrics can also be calculated from the results of the state algorithm. For the three fields, the combine utilization was found to be 94.4% with the idle times and the stationary unloads detracting from the utilization. Additionally, the combine was idle for 7.1% of the time in the field. The combine was unloading, either stationary or on the go for 9.1% of the time in the field.

The state algorithm was largely successful in identifying the states of the machines in the fields, however, the algorithm had higher percent errors for the states outside of the "Working" state. This was most likely due to the tuning of the algorithm as the working state held much more weight due to the number of points that were harvest, as such the tuning most likely selected results which prioritized the working state as it contained an order of magnitude more points than other states. Thus, if desired the model could be changed to optimize for the maximum correct values of idle, on the go, and stationary unload points. As these three states are highly related, most errors of the smaller states were shared with one another. Errors in unload on the go were most likely shared with stationary unload, and stationary unload errors were most likely shared with idle rather than the working state. Thus, tuning could be done to improve the error of the smaller states at a loss to the overall model correctness.

1.6 Conclusion

Farmers and managers are pressed into making hard decisions for the betterment of their machines and equipment. To assist in this endeavor, a tool to help with the optimization of the equipment fleets was created. A data pipeline was successfully created to capture truth data for the 2021 harvest. An algorithm was created and applied to the data to label the states of all the event logs that were captured. To make the tool more widely usable, the algorithm was constrained to only using GPS points that could be captured off the already existent CAN bus or by installing inexpensive GPS loggers. The algorithm parameters were tuned using truth datal collected via CAN bus message and validated on three production fields. In total, the model was able to correctly identify the states of the combine with an accuracy of 96.6%. Additionally, the model predicted the states of the grain carts and grain trucks with an accuracy of 91.75%. This algorithm will enable

future work in both real time and post operational logistical analysis that will empower farmers to identify inefficiencies and bottleneck in their harvest operation.

2. LORA EVALUATION FOR LONG RANGE AND HIGH SPEED

2.1 Abstract

Harvest operations have many constraints to operate at full efficiency. Correct pairing of harvester and transport must be assured to fully utilize implements and workers that are available. Downtime of the harvest operation can occur when the harvester's hopper is filled, and no grain truck is available for unload. During this interaction, the harvester waits until and truck is available for unload. This interaction downtime could be reduced via proper planning and information. If informed that no truck will be available soon, harvesters could choose to perform low efficiency work and not fill the hopper thus incurring downtime. LoRa was identified as a technology that could provide real time information for harvesters thus giving them the tools needed to make informed decisions. LoRa was tested with differing configurations to identify if it would be an appropriate tool for mobile applications at long ranges such as 16 km. From the testing, LoRa's applicable range varied based upon line of site and height of antenna with a maximum applicable range of 9.6 km. Additional testing was done regarding the use of LoRa at speeds ranging from 32 kmh^{-1} to 97 kmh^{-1} to identify the effect that high speed had on signal strength. The speed of the mobile end node had large impacts on the resilience of the LoRa signal. At 16 kmh^{-1} the signal reliability dropped from 77% to 53%.

2.2 Introduction

With an increase in world population across the past decade, it has become imperative that commodity operations increase in both yield and efficiency to meet the needs of an ever-growing world. From 2015 to 2020 the world's population has grown by 400,000,000 people. (Worldometers, 2020) With this exponential growth in population, food scarcity and commodity crops have been a focus of improvement to meet the increasing demand for yield. As such, it is important that farmers and harvest operations have peak efficiency during operating hours to maintain high rates of harvest with as much resource and logistical optimization as possible. Inefficient harvesting operations lengthen harvest duration and increase the risk of field loss due to over mature crops and weather events.

Improper sizing of machinery can create waste both in the form of unneeded excess capital spending, or in the form of harvester downtime due to a lack of trucking capacity. Buckmaster and Hilton (2005) showed that the matching of harvester and transporter directly correlated to theoretical field efficiency, and additionally created a digital model which would allow users the ability to correctly size and utilize the equipment and labor force that was available to them. With a declining trend in labor availability, it has become imperative to use as few implements as possible. From 1950 to 20000 there has been a decline of over 50% of hired farm laborers across America (Anon 2020b). This decline has forced a focus on larger machinery and higher total implement capacity with a reduction in implement count. This higher capacity machinery is wasted if the system cannot keep up and carries a high initial capital cost that carries over in energy costs and total farm costs. Utilizing proper path planning and whole farm modeling, harvest operations could improve the system capacity and throughput without an increase in machinery count or capacity. This improvement to capacity would allow for downsizing of machinery and a reduction in total system cost. It would also have the added benefit of a reduction in worker count for the operation.

To achieve proper path planning, real time system monitoring must be in place for all implements within the field. Harvest operations often perform kilometers from base points or unloading areas. As such, it is imperative that long range technologies be considered when selecting an appropriate technology to create a real time mapping tool for in field operations. Cellular technologies provide simple and elegant solutions for fleet management and other such tracking operations. Its simple success can be seen throughout public use with the implementation of google maps and other connectivity programs. However, cellular connection suffers greatly in areas with low population density and rural areas. Due to the lack of coverage within rural areas, utilizing cellular connections for fleet tracking yields poor results with regards to long-distance, real-time tracking. Coverage of rural areas within America was classified as “Cropland areas where farming occurs still lag far behind in adequate fixed and mobile broadband access.” (Kane and Borghei 2017) With the vast area of rural America lagging behind in LTE and broadband coverage, technologies beyond cellular must be examined for use within the long-range communication sphere as the inadequate coverage of cellular signal would not be a reliable solution.

To circumnavigate the lack of coverage within rural areas, M2M (Machine to Machine) and low power wide area (LPWA) networks were evaluated for their ability to send and receive data across long distances that harvest operations would require. Often LPWA networks sacrifice high data rates to gain high range and noise resistance. LoRa was identified as a potential technology which could fill the gap in long range communication in real time. LoRa is a proprietary radio modulation technology which operates within the 915MHz license free range band for US operators. Radio nodes can be assembled to form a LPWAN called LoRaWAN as seen in Figure 2.1. LoRaWAN is often configured in a star shaped network such that end nodes communicate with gateways which in turn transmit data onto central servers at which point end users can access the data in a variety of ways such as tablet or mobile device.

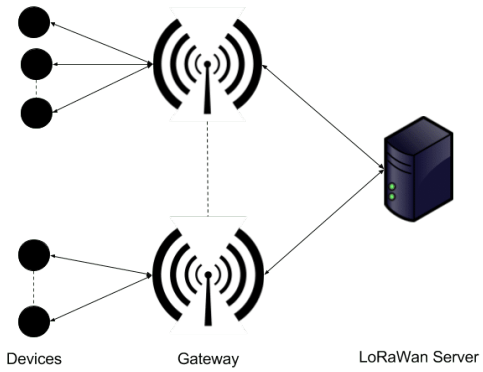

Figure 2.1: LoRa star network configuration

This long range M2M technology makes it ideal for communication within the field of agriculture as it is resistant to noise and can travel long distances. In addition, data rates that are needed for real time localization of in field machines is rather low and can be handled by the physical limitation of LoRa.

2.2.1 Internet of Things Within Agriculture

Lakhwani et al. (2019) focused on the potential uses of the Internet of Things within developing agricultural fields such as in India. Within the review, some key benefits of IoT were identified, including, efficiency of input, cost reduction, profitability, sustainability, food safety, and environmental protection. The paper also focused on some of the challenges of agriculture and

the benefits of having implemented an IoT solution. Issues such as crop growth timing were addressed in addition to watering solutions, and crop profit calculations. Utilizing the IoT, farmers were able to check on plant stages during the growth phase and predict future outcomes for profit and cost analysis. Additionally, farmers had the ability to properly plan irrigation schedules in accordance with other data inputs, such as weather patterns and history. The IoT within the review allowed farmers to check current outputs and in field metrics with historical data within the cloud and perform checks and comparison to previous years. Furthermore, farmers could monitor real time changes within the field and make proper decisions based on data gathered from the sensor clusters within the fields.

Zhao et al (2010) explored the use of IoT within greenhouse production environments. Data was collected within a greenhouse using IoT style sensors and clusters. Temperature and humidity data was collected in a machine to machine (M2M) style of network. The M2M network was then supported with real time data transfer using a mobile terminal via a short messaging service (SMS) gateway. Data was then transferred via a GSM modem which enabled users' access to data via mobile web apps. Additionally, GSM modem alternative uses were explored such as its use in vehicle tracking or home automation. Researchers were able to display stored data from the greenhouse in formats which would provide value to users of the system. Data was stored in online databases and could be queried by users to obtain, 24 hour, weekly, or monthly data. This data could also be used within the real time to provide alarms or alerts when temperatures fell below a set threshold.

Farooq's review paper sought to evaluate the evolution of the IoT within the most recent developmental phase (Farooq et al. 2020). The paper targeted seven main research questions which pertain to all facets of the IoT within agriculture. Within the seven research questions, five were directly related to the development of solutions using IoT for cycle analysis. Mainly, how has the frequency of approaches been changed related to IoT agriculture over time? What approaches are used to address problems related IoT agriculture? What are the main application domains of IoT in agriculture? What were the primary focuses of the selected studies? What type of IoT devices/sensors have been used in agriculture? Which IoT network/communication protocols are used in agriculture? From the data and papers which were reviewed, it was determined that interest in IoT within agriculture has been rising steeply since 2015 with over double the number of papers published from 2015 to 2018. Additionally, the review identified how IoT was being applied to

each separate domain. 70% of the papers reviewed used IoT as a monitoring system for operations such as temperature monitoring, or humidity and soil temperature monitoring. 25% of studies focused on controls while only 5% of the papers reviewed focused on the tracking domain. Additionally, the papers were sorted with respect to the major focus of each study. A breakdown of each focus could be seen in Figure 2.2.

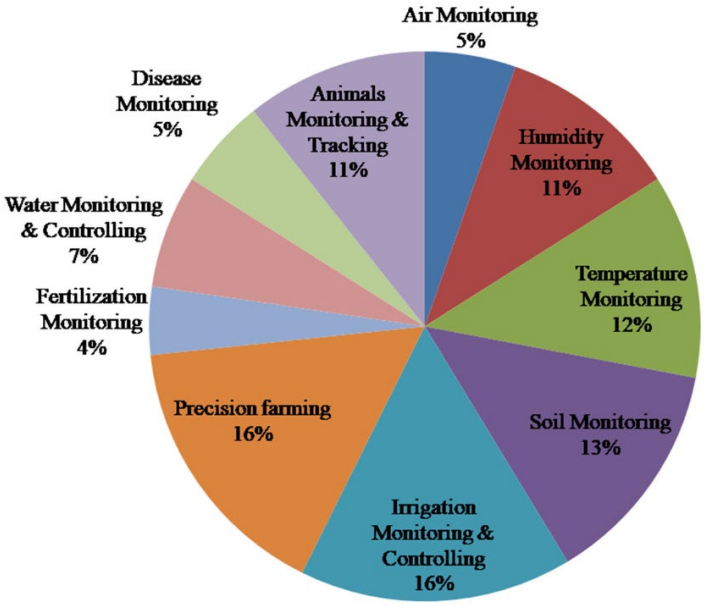

Figure 2.2: IoT application focus breakdown (Farooq et al. 2020)

Moreover, the review identified which communication technologies were being developed for use with IoT. From the papers reviewed, 29% used wireless sensor network (WSN) and 15% utilized WIFI connections. Long range radio (LoRa) was the fifth most popular communication technology within the study with Zigbee and Radio-Frequency Identification (RFID) being more utilized. The review also identified issues that an IoT user could expect to deal with when setting up or using a network. The large issues facing IoT users were cost based, the creation of a large scale IoT solution carried many costs that compounded for small users creating large costs. Each device and sensor accrued project cost. In addition, subscriptions to mobile fees for a large sensor count could create a high cost to the end user. Aside from cost, lack of knowledge affected the implementation of IoT. Rural farmers were unable or unwilling to understand the technology necessary to properly implement and maintain a IoT solution. Lastly, the review determined security issues within IoT.

Inherently, IoT had many security risks within its systems due to the low energy hardware. Complex algorithms and high-level securities could not be run on hyper efficient modules which utilized extremely low amounts of energy. Cloud services could also be targeted with denial-of-service attacks or other database attacks.

Khanna and Kaur performed a thorough review on the uses and growth of the IoT within precision agriculture (Khanna and Kaur 2019). Among the review work done, the amount of communication standards available to IoT was identified. From the technologies identified, LoRa was identified as being a long-range option with ranges from 3000m to 5000m. Other technologies offered greater data rates however, provided much less range than LoRa. The researchers highlighted the varying novel uses of each communication technology and it potential to increase value for farmers. Importantly, the creation of an intuitive communication interface and cost were key challenges to IoT. Additionally, availability for users, and data confidentiality were issues impeding the growth of the IoT sector within agricultural practices.

Another paper strove to evaluate the theoretical limits of LoRa utilizing the standard protocols used under the European standards (Petäjäjärvi et al. 2017). Because of this the team tested LoRa with a spreading factor of 12 rather than the normal spreading factor of 10 in the United States. The increase in spreading factor would contribute to a longer-range transmission over the North American standard of 10. The team determined that between 2 and 5 km the end node successfully transmitted its data at 85% and 88% success rate. In addition, they observed signal deterioration at speeds of 40km/h.

Lavric and Popa sought to determine the effect that multiple end nodes would have on a large-scale system that could be utilized in a variety of civil fields (Lavric and Popa 2018). In large endeavors multiple end nodes would be competing for the same broadcasting time for the gateways in the system. As such, packet collision could occur in the network and packets would be lost. The team determined at high spreading factors; the long transmission time adversely affected the performance of the network as collisions would surely occur. However, utilizing the lowest duty cycle, the gateway could support just shy of 1000 end nodes. For an eight-channel gateway the network could support approximately 8000 end nodes.

2.2.2 Farm Management

This review identified research studies performed withing farm management systems from 2008 to 2017 (Tummers, Kassahun, and Tekinerdogan 2019). From the systematic search, 38 studies were deemed to be of high enough standards. The review classified a farm management information system (FMIS) as "an FMIS supports decision making and helps with keeping track of the current business process to maximize the profit of a farm." From the review, seven research questions were used to evaluate the various studies. Pertinent to this study are the following questions: what are the current FMIS described in the literature, which domains are supported, what are the delivery models, what are the features of existing FMIS, and what are the obstacles to existing FMIS? From the review, 28 of the studies die not name a specific FMIS or rather proposed a new design for the creation of a FMIS. Concurrently, of the studies 16 were performed in the arable farming domain. The livestock domain was the second most common with 5 studies being reviewed on the topic. Continued from the review, a majority of studies utilized an application approach for developing a FMIS for the end user. Applications could be applied locally on any computer. In contrast, only 6 of the studies utilized a platform approach which would allow users to create and input their own plugins to enhance the capabilities of the platform. 81 separate features were identified from the various studies. All 32 studies identified the farmer as a key feature within the FMIS, in contrast, only 4 studies identified machinery tracking as a key feature in a FMIS. Likewise, 4 studies identified harvest management as a key feature, and only 2 studies identified driver assistance as key features within a FMIS. The review also recognized over 53 separate obstacles in the development of a FMIS. Adoption rates of FMIS was a chief obstacle, socio-demographic factors and other contingent factors hindered the implementation. Cost remained an obstacle, not all FMIS provided immediate or observable value to farmers, or simply cost too much for farmers to see profitability from the system. Other issues remained inhibitors to FMIS, for example, understandability was a key factor for farmers using management systems. The systems must be easy to operate and easy to introduce to the farm for value to be seen. Connection to internet was an issue that prohibited adoption; not all farmers had wireless connection to internet in rural areas in the fields, as such, farm management systems that utilized heavy internet traffic could be inefficient or unavailable to many farmers in areas with connectivity.

This work performed by Buckmaster focused on the benefits and potential efficiency gains that could be made via the optimization of machinery selection and cycle timing (Buckmaster and Hilton 2005). An optimization tool was created by analyzing and diagraming cycle times of various farming implements during the harvesting season. The spreadsheet tool that was created was useful in that varying inputs could be selected to manage differing farm sizes or machinery selection. Analysis was performed on a self-propelled forage harvesting system. The harvester had a capacity of 40Mg DMh^{-1} with a field efficiency of 80%. This harvesting system was then analyzed for total system capacity with a range of transport systems. System capacity rose as transport count rose, however, labor utilization for all implements stayed the same for each scenario. Increasing the count of transporters, increased the system capacity, increased the harvester efficiency, and the labor utilization remained similar. In addition, transport distance was also considered in the model. As transport distance rose, system capacity dropped significantly. Beyond 16km, system capacity dropped by over 20% with transporters being more fully utilized in the 10 – 15 km range.

2.2.3 The Case for the Use of IoT Within Cycle Management

As can be seen from the previous exploration into IoT and system management, farmers are looking for a way to properly manage their fleet such that labor utilization can be optimized, profits and yields can be maximized, and overall system downtime can be reduced. IoT provides a unique opportunity for farmers to get real time data about various implements and factors from their fields into their hands. By utilizing the developing technology that is IoT farmers can make informed decisions on the fly and be able to increase the overall system capacity of their overall operation.

From the previous work shown, IoT has its own problems and issues that must be dealt with to create a system which provides real value to farmers in a way that is both economical and efficient. Farooq's work showed that over 70% of developmental work in agricultural IoT was in monitoring specifically about temperature, moisture, and humidity monitoring. While these methods can provide value to farmers, value can be extracted from IoT in other ways, especially with regard to tracking and controlling. There is a large gap within the research area for tracking farm implements and controlling destinations. Farooq's work shows that approximately 5% of papers published within the agricultural IoT domain dealt with tracking and fleet management. This can be in part to the lack of long-range technology that is utilized for IoT work. Khaana

identifies and classifies the various technologies that are available to work with IoT. Among them, LoRa stands out as having long range application that may be suitable for working with a tracking application.

Developing an IoT tracking solution would be able to provide a discernible amount of value to customers and adopters of the technology. As can be seen from Buckmaster's work, total system capacity fell by 20% when transporters were forced to travel more than 16km. The drop in capacity could be offset by increasing the number of total transporters in the system to ensure that the harvester is always operating in the field. However, increasing the count of transporters is not always a viable option for farmers. Farmers are already struggling with finding and maintaining capable workers during the harvesting season and may not be able to accrue enough workers for their fleet. Additionally, extra transporters require more storage space and upfront cost to purchase. Farmers may view the reduced system capacity as an acceptable loss when compared to the upfront and long-term cost of utilizing another transporter. With IoT and LoRa farmers may be able to offset the loss from lack of transport availability. LoRa systems could be created such that harvest operators could remain informed of wait times and choose to operate within low efficiency fields to mitigate the wait time and reduce the overall downtime of the combine. By doing so, the operators could increase the overall system capacity.

When developing IoT solutions for agricultural use, care must be taken to avoid or resolve potential issues and conflicts that would halt the adoption of said technology. From the review papers, multiple common issues or inhibitors arose that must be dealt with. Cost remained a large issue and a common theme among the IoT review studies. Tracking large fleets would require a large count of sensors. Thankfully, LoRa sensors are relatively cheap, at less than $100, and due to the low power nature of them, using milliwatts, require little to no upkeep. LoRa technology also has the large benefit of operating within the unlicensed spectrum of 915 MHz Operating in the 915 MHz range allows for cheaper systems as no monthly subscription upkeep cost is needed such as with LTE technology 5g phone service.

In addition to cost, ease of application and ease of implementation must be considered when developing. Utilizing LoRa a system can be made which allows combine or harvester operators to see in real time the location of the implements in the field and the transporters ferrying goods. This type of a system could be made in a way which is easy to setup via prepackaged code and easy to utilize as it requires very little beyond LoRa modules. In addition, the system can be

made using off the shelf components which are open source such as Arduino or Raspberry PI. This will further reduce the complexity and cost of said systems thus encouraging the adoption of the technology.

Multiple review papers also identified data security as a large issue with regards to IoT Lora signals can be send encrypted or unencrypted, however the security of the system can be drastically reduced with the removal of cloud computing and cloud access. LoRa signals could be sent directly to basepoints in the combine and all computations could be performed locally. This removal of the networking step would allow for increased security for farmers and would remove the upkeep and excess steps of maintaining and utilizing cloud data.

2.3 Objectives

The goal of this work was to evaluate LoRa as a communication technology for tracking grain trucks that are used to transport grain away from the field to storage or market. To complete the evaluation the following objectives were created:

1. Create a LoRa system to be used in testing for long range and high-speed application
2. Create and maintain a data pipeline for retrieval and evaluation of geospatial data
3. Perform long range and high-speed testing of the LoRa system

2.4 Methods and Materials

2.4.1 LoRa System

The LoRa system that was created was based on the Semtech SX1276 chip. This chip was a proprietary chip featuring multichannel spread spectrum communications. Semtech advertised the following capabilities from the chip: 168 dB maximum link budget, 127 dB dynamic range received signal strength indicator (RSSI), 20dBm constant RF output vs. V supply. (Semtech, n.d.) This chip was housed within an Arduino shield which was sourced from Dragino Technology Co. The Arduino shield was directly compatible with an Arduino Uno. This allowed for quick assembly and integration of the SX1276 with an Arduino microcontroller. In addition, the Dragino shield housed a MT3339 style sticker global positioning unit (GPS).

Figure 2.3: Dragino LoRa shield

The Arduino shield also had mounting points for external antennas both for LoRa mounting and for the GPS unit. A 28dB gain 3V SMA style GPS antenna (WGP supply, Unites States) was sourced for the project to always assure proper GPS fixes. This antenna, seen in Figure 2.3, could be mounted in various locations such that it was not influenced by the orientation of the Arduino shield and could be attached to metal surfaces via a magnetic backer. In addition to the GPS antenna, an antenna tuned for the 915MHz frequency was used to boost the LoRa signals being sent and received. Initial testing of the unit was performed using a .9 dBi helical antenna that was received stock with the unit. After initial testing, a 6 dBi HGV-906U omnidirectional antenna (L-com, United States) was obtained to boost signal strength. The antenna, shown in Figure 2.4, was an omnidirectional antenna with a 30-degree vertical beam width and 50-ohm impedance. (L-com n.d.) Also, the antenna could withstand winds of up to 108MPH which made it ideal for high-speed mobile testing. This 23-inch antenna was attached using N to SMA adapters and could be mounted with an angled bracket on a mast.

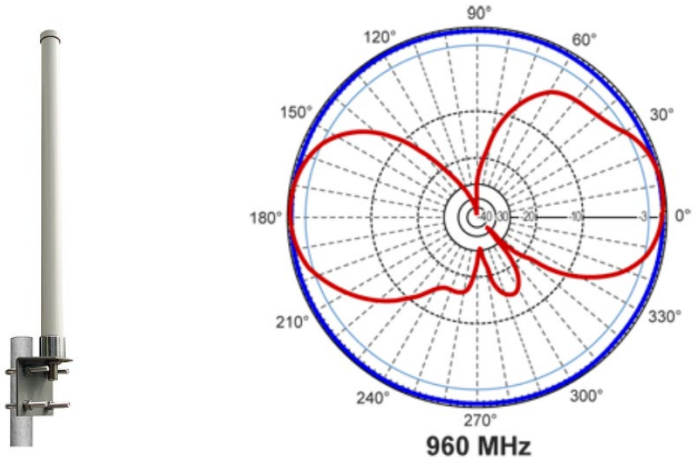

Figure 2.4: HGV-906U Antenna and radiation pattern at 960 MHZ (L-com)

To contrast the Dragino Arduino shield, am OLG01 gateway was used to create a connection point with the end nodes. This gateway was also sourced from Dragino and has built in 3g/4g capabilities. Within the unit, it is controlled via a 400MHz processor running Linux, additionally the unit housed 16 MB of flash memory which could be written or read from. The unit was powered via 12V input and had Wi-Fi access point capabilities for controlling and programming the unit.

To help in the selection of antennas and appropriate hardware the Egli (1957) pathloss shown in Equation 2 was utilized to make approximations about the range of the equipment.

Equation 2:
$$PL = Gt * Gr * \left(\frac{Ht * Hr}{d^2}\right)^2 * \left(\frac{40}{f}\right)^2$$

Where:

- PL = pathloss(dB)
- Gt = Transmitting antenna gain
- Gr = Receiving antenna gain
- Ht = height of transmitting antenna (m)
- Hr = height of receiving antenna (m)
- d = distance between transmitting and receiving antenna (m)
- f = operating frequency (MHz)

2.4.2 Data Processing Pipeline

A pipeline for data retrieval and analysis was necessary for the evaluation of LoRa at long distance and high speed. GPS points were broadcasted using a mobile end node. Simultaneously, the same GPS point was recorded via serial into a text file to be used later in the analysis. The recorded GPS point was then stripped of excess data and transformed into a .csv file with time logs and location logs. The GPS point that was broadcasted at the end node was received by the gateway antenna. The point was then recorded as a LoRa packet and sent into Excel for data cleaning and transformation. The data was then transformed into a .csv file of the same layout as the initially recorded GPS point. The two .csv files were imported into QGIS for future analysis and comparison of the sent data and received data. The pipeline was visually represented in Figure 2.5.

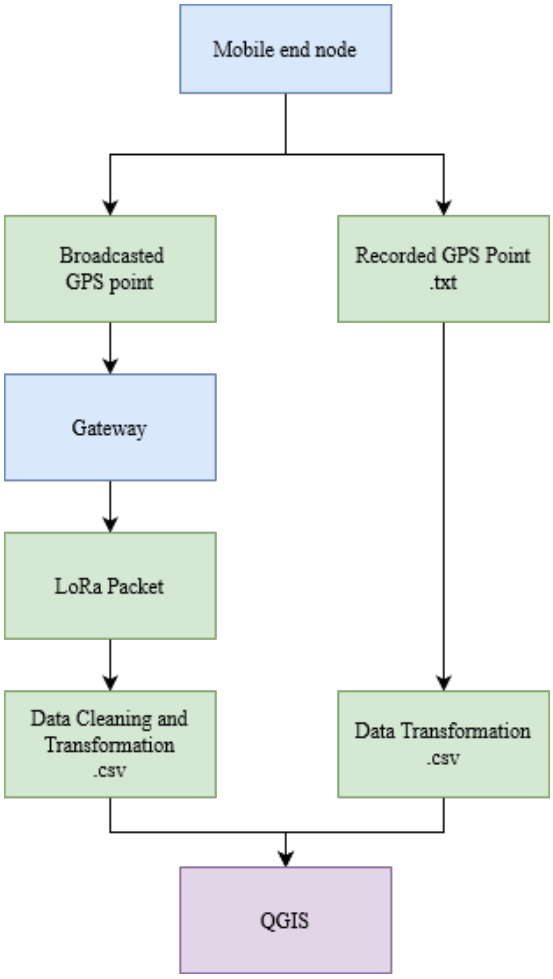

Figure 2.5: Data pipeline of GPS points sent and received at end node and gateway

2.4.3 System Testing

Testing was carried out at the Agricultural Center for Research and Education (ACRE) at Purdue University. Testing was done using a mobile truck and stationary tower antenna located at the center of the farm.

Range Testing

To evaluate the capabilities of the LoRa system, a path was planned through the city of West Lafayette and though the agricultural sector north of ACRE. To create a valid path through the city, Radio Mobile Online was used to create raster maps of potential coverages which the LoRa system could provide. Radio Mobile is an online tool which uses the Irregular Terrain Model developed by Hufford. (Hufford 1982) The online tool considered many different parameters which were useful for testing. The tool could be calibrated for antenna type, antenna height, elevation, line losses, and many other parameters. Radio mobile was calibrated with the same setup used for testing to gather a proper idea of where and how far coverage would be seen with the LoRa network. Coverages could be made for 3 distinct setups which were carried out: gateways at 7m, 10m, and 16m. Coverage maps were then overlayed in QGIS, an open-source GIS software, and path lines for testing were drawn out.

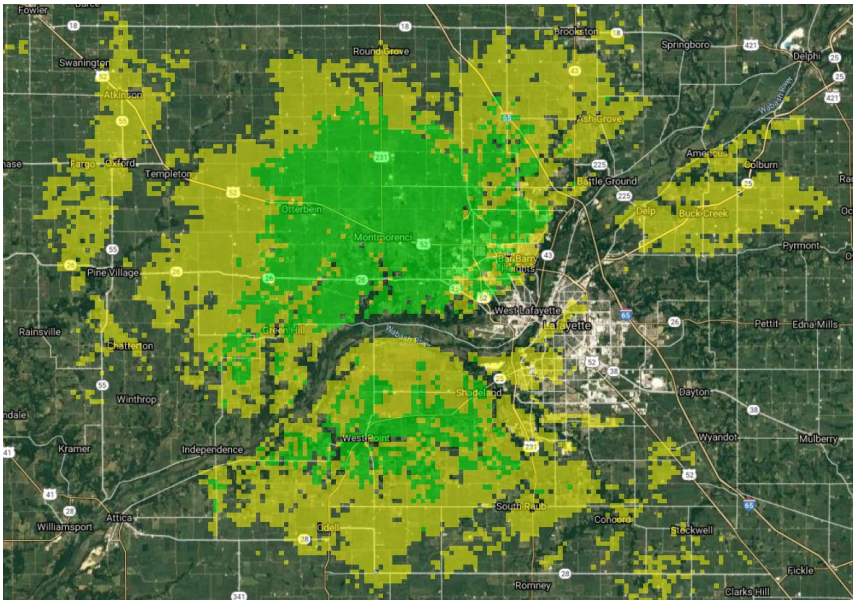

Figure 2.6: 16m High gateway coverage map of West Lafayette

From the coverage maps which were created and shown in Figure 2.6, a grid path was chosen through the high coverage (green) areas. The path chosen in Figure 2.7 traveled 190km both through rural and urban terrain. This path utilized country and county roads which harvest operations would be likely to use.

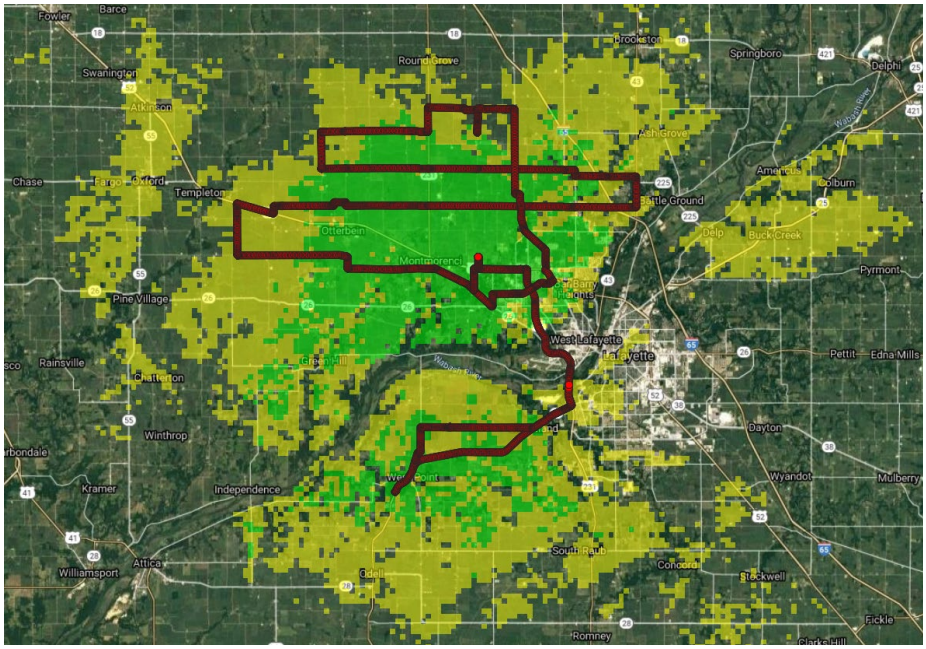

Figure 2.7: Path through coverage map

A stationary gateway was placed at ACRE farms in the center of the pattern and elevated using a telescoping tower with 12V solar power available. The mobile Arduino end node and was connected to the back of a truck via a mast, as seen in Figure 2.8 and powered with on board 12V from laptops.

Figure 2.8: Telescoping tower and truck mount

Data was collected in real time as the truck drove the given path. The trucks velocity was the maximum velocity of each road, with speeds ranging from 50 kmh^{-1} to 110 kmh^{-1}. In contrast to the normal star network which LoRa utilizes, end node data was collected directly from the LG01 gateway. Normal star network utilized the gateway as a bridge between the end nodes and servers, however, with a lack of broadband connectivity and access to the internet in rural areas it was imperative that the system be able to pull data directly from the gateway without accessing the server side as it would in real world applications. As such, utilizing a bash script, RSSI and LoRa messages were stored directly on the gateway and pulled off the gateway in real time mimicking real-world applications.

The route was driven with the telescoping tower in three different configurations, once with the stationary gateway at each height: 7m, 10m, and 16m. In all three configurations, the end node was located at a height of 2.4 m from ground height. The route was driven with both low and high gain antennas to evaluate the impact that each antenna had on signal strength and on signal reliability.

The end node had the following configuration which maximized range of the system and additionally maximized robustness to noise. The node was set to transmit using the PA boost pin available to the chip. As such, the board was transmitting at 20 dBm, which was the maximum hardware limit set by the FCC. With the high gain antenna, the system reached a transmission level

of 26 dBm minus minor cable and connection losses. The chip operated at a spreading factor of 10 and a bandwidth of 125kHz. This configuration produced packets with an on the airtime of 370.7 ms. The mobile end node sent a packet once per 2.4 seconds.

Arduino programming was done utilizing the open-source IDE available online. The initial sketch to transmit data utilized prebuilt libraries to make integration and creation of the LoRa network simpler. The Radiohead library (kenbiba, 2016) was useful in creating sketches to fully utilize the abilities of the Semtech chip and Dragino shield. This library was coupled with the TinyGPS library (Lee, 2019) to create sketches that could read NMEA streams from the external GPS unit. Additionally, for further testing, the LMIC library (Kooijman, 2015) was used to create sketches which utilized LoRaWAN capabilities and OTAA.

Speed Testing

After initial testing was finished, further testing was required to determine the accuracy and use of LoRa under high-speed conditions. Testing was performed in a similar manner as in previous experiments. However, constraints were placed upon speed and careful attention was placed on maintaining steady speeds. Testing was performed with gateway at 3.7m of elevation from ground level. Like the first testing, the mobile end node was placed at the top of a truck for data transmission. Unlike prior work, the route driven was in a previously established high reception area seen in Figure 2.9. As such, a low gain antenna was used for ease of testing.

Figure 2.9: High Speed Testing Location

The route taken was approximately 2 km in length in a tangential direction to the gateway. An initial baseline was taken of the route with 6 stops taken along the way. The truck with the end node waited at each stop for approximately 2 minutes. After the baseline was taken, a slow pass of the route was taken at 16 kmh^{-1}. Thereafter, the route was driven at increasing speeds incrementing by 8 kmh^{-1} to 105 kmh^{-1}. Each lap had a set amount of acceleration length which was not tracked, and high speeds laps over 65 kmh^{-1} were taken twice.

2.5 Results

2.5.1 Range Testing

Initial testing of the network was performed at ACRE farms at Purdue. With an initial route due north of the stationary gateway, the system was expected to have range between 8 and 16 kilometers. Line of sight for the area was high with little to no impedances to visibility over a long range other than the slope of the terrain at various intervals. Tree cover was low to the north and no large buildings or compounds were directly north of the gateway.

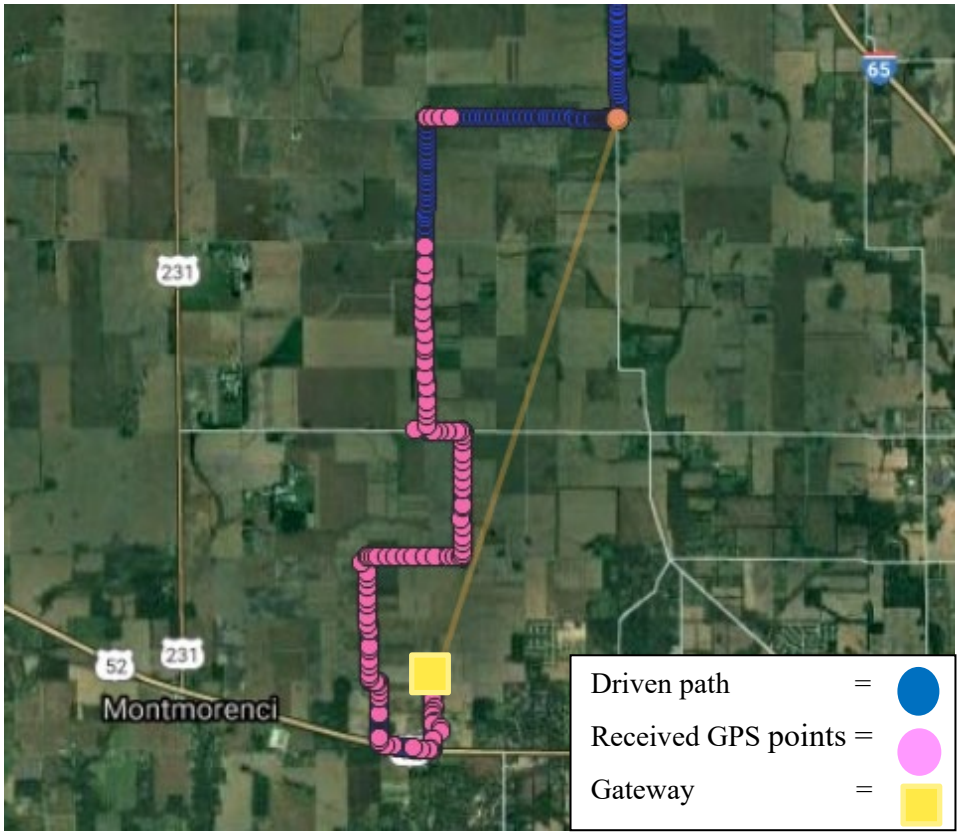

Figure 2.10: 7m gateway north bound testing with a max range of 7600m

From the initial testing shown in Figure 2.10, the maximum distance that the signal could be received was 7.6 km from the gateway to node. However, at that distance, signal strength and reliability were low with only 4 points being recieved. Signal was unexpectedly lost at 5.2 km from the gateway and only returned on two occasions once at 7.4km and once at 7.7km. Of 55 packets that were sent at that range only four were received. Lidar data from National Resources Conservation Service (NRCS) was obtained and utilized in QGIS to identify the geographical landmarks which potentially interfered with the signal. The LiDAR elevation was accurate to .300m. (Jinha 2020a)

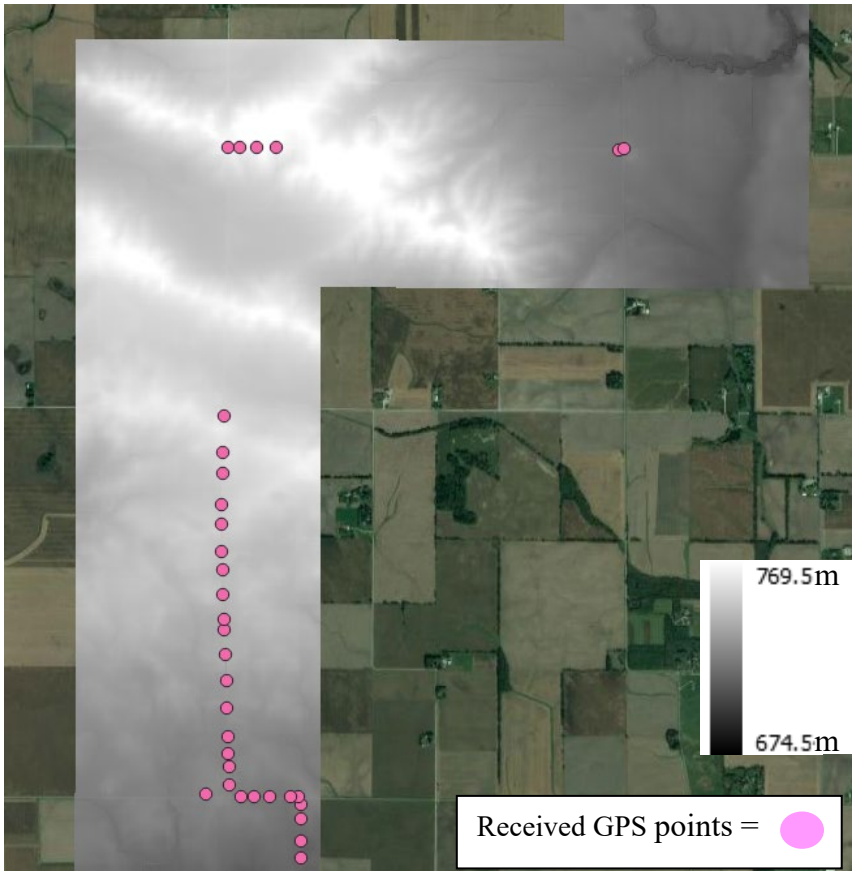

Figure 2.11: 7m gateway path overlayed on elevation raster highlighting missed data points

The elevation data shown in Figure 2.11 proved that there was little terrain change that would impact signal. The Lidar did show that the points in at the top of the map where slightly higher with an elevation of 238m compared to the surrounding landscape at 222m potentially allowing for signal reception and successful packet transfer.

Further testing was also done to compare the signal strength and range of the gateway when changing the ground elevation of the gateway.

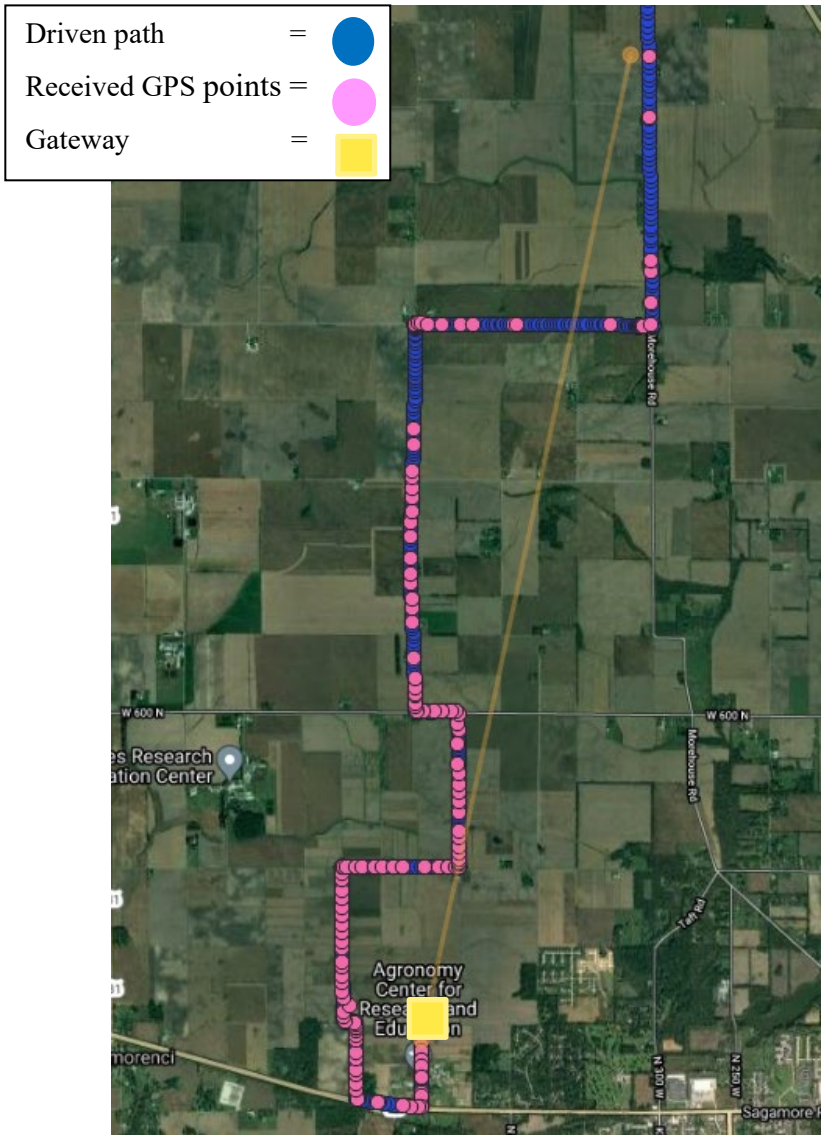

Figure 2.12: 10m gateway received data points with max range of 10500m

A large difference in coverage was expected when comparing results from the 7m gateway, shown in Figure 2.12, with the 10m gateway. The extra 3m of height allowed for better line of site over terrain variance and small obstacles such as trees or bushes. By elevating the gateway by 3 additional meters, the range of the LoRa system increased by over 2.9 km with a total range of 10.5 km. Similar to the 7m gateway the system failed to transmit reliable data as the range increased. Reliability of the system dropped dramatically at 6.3 km which was a marked

improvement over the 7m gateway at 5.2 km. The drop in signal could be due to a grove of trees along the sightline of the signal and thus would inhibit both signals reducing their overall range. However, the signal could be received when the truck moved east and thus cleared the obstacle impeding the signal.

In addition to testing at 7m and 16m heights, the gateway was also placed at a height of 16m off the ground. Like the previous routes, the truck was driven north with an eastward dogleg along the path shown in Figure 2.13.

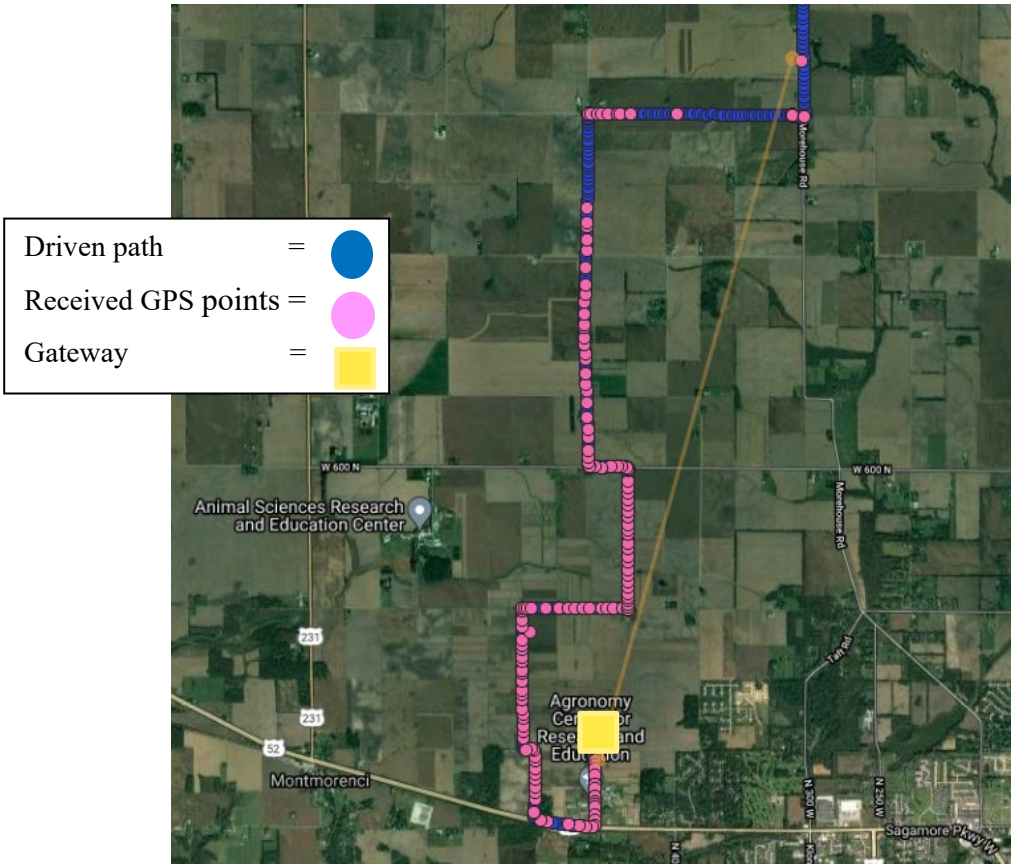

Figure 2.13: 16m gateway received data points with max range of 8300m

Utilizing the Egli (1957) model from Equation 2, it was expected that the 16m gateway would have much better signal capabilities. However, the maximum range of the 16m gateway was 8.3 km, like the 7m gateway. Comparative to the previous tests, the signal was severely limited above a range of 6.3 km from the gateway. Because of the similarities in signal degradation between all three tests, it was assumed that some obstacle was obstructing the line of sight between

the gateway and the end node and thus signal was lost at the same range for all the subsequent tests. The test showed that it was imperative for LoRa operations that line of sight between the end nodes and the gateway was always maintained. Without line of sight, the signal loss became too large for the gateway to obtain a useable symbol and thus corruption or packet loss can occur.

Table 2.1: Packet Reliabilty Within Receiving Range of Gateway

Tower Height	Packets Sent	Packets Received	Success Rate
7m	261	105	40.2%
10m	331	128	38.7%
16m	362	157	43.4%

Table 2.1 shows the direct reliability of the LoRa signal that was experienced at highway speeds of approximately 95 kmh^{-1} while moving with little angular velocity to t

he gateway itself. Only packets sent within the reliable data range were considered and extreme outliers were corrected for packet reliability. From this, a large packet drop was seen across the different configurations, independent of the height of the gateway, when the signal was steady. Upon closer inspection of the packets as viewed in Figure 2.14, at regular intervals the signal being sent from the end node was not received by the gateway due to weak signal.

Figure 2.14: Packet reliability

As can be seen in Figure 2.14 the gateway received approximately one in three sent signals from the end node even when within the range of the LoRa signal.

2.5.2 Secondary Range Testing

Continued testing was done utilizing the capabilities of the telescoping tower to further understand the range of the network, especially with regards to larger areas and more dynamic land coverages. As such, data was collected from areas all around West Lafayette. In addition to the range testing, line of sight was to be evaluated with the use of local forests in the area. Urban areas were also evaluated to ascertain the capabilities of the LoRa signal. However, urban landscapes were unable to have direct line of sight to the gateway without heavy forest initially impeding the signal.

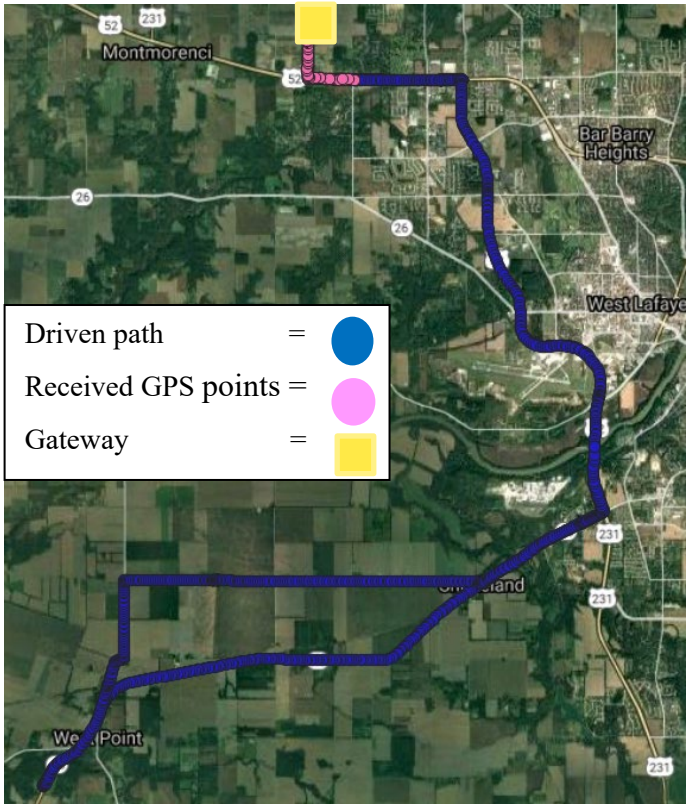

Figure 2.15: Cross valley testing of LoRa signal with a 16m gateway

As can be seen from Figure 2.15 the end node signal did not penetrate the landscape to make it back to the gateway. The heavy forest inhibited the signal to the point where no readable data was received. Closer inspection of the data showed how drastically the wooded area blocked the signal transmission.

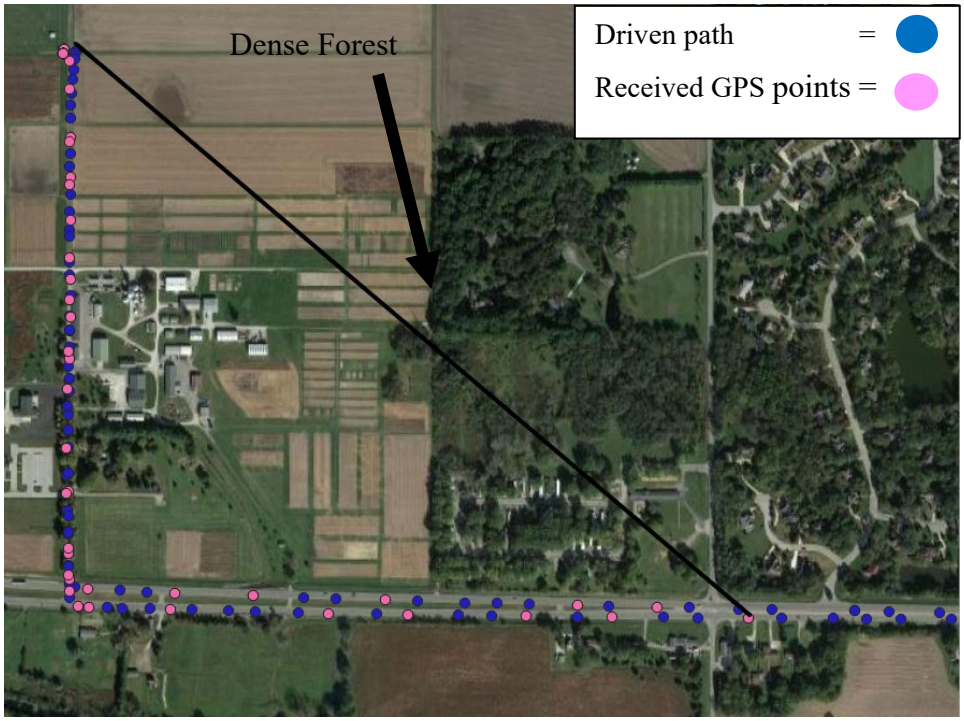

Figure 2.16: Forest impeding line of sight at close range of gateway

Figure 2.16 shows the immediate the effect the forest had on the signal that was received. Previous testing revealed that the signal could be received at over 10 km in distance, however, from this test the signal was only received at 1.2 km. Due to the strong restriction in signal range, it was hypothesized that the signal could not propagate south past the heavily forested area to reach the other side of the valley. Even if the signal had the ability reach that high range and pass over the flat lands and low valley, the forested area would cut the ability of the signal to such a high degree that no packets would make it to or from the gateway due to the heavy vegetation.

More testing was performed on the system to identify the potential range with a higher variety of land covers. It was also critical to identify if LoRa could penetrate through small, wooded areas and still receive signal on the other side. Data was collected utilizing the northern

fields of the ACRE properties which had wide areas with little to no ground cover which could interfere with the radio signal.

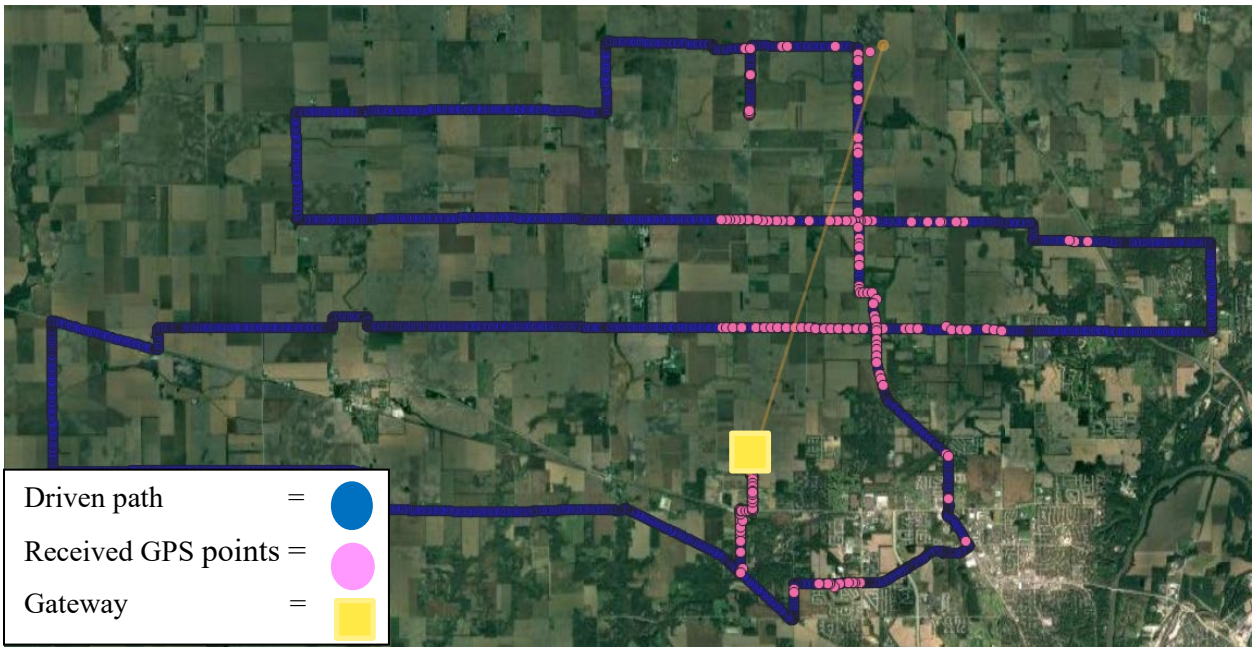

Figure 2.17: 7m gateway path and received packets

Figure 2.17 highlights the path that was driven to fully test the properties of the LoRa signal at ACRE farms. The maximum range that the system was able to achieve with the tested route was 10 km. This testing aligned with previous testing that was done, however, the range of this test in comparison was approximately 2.5 km longer than prior testing as seen in Figure 2.10. Due to the unreliable nature of the radio as established in Table 2.1, the increase in range could be due to mere chance that more packets were received at a greater distance than before. Alternatively, other factors outside the scope of the experiment could be a root cause of the difference in results. These factors could range from weather conditions to relative humidity to speed of the end node varying.

The results from the test highlight some more interesting useful information to identify. The data collection especially highlighted the effect that line of sight had on the network of LoRa signals which can be seen in Figure 2.18.

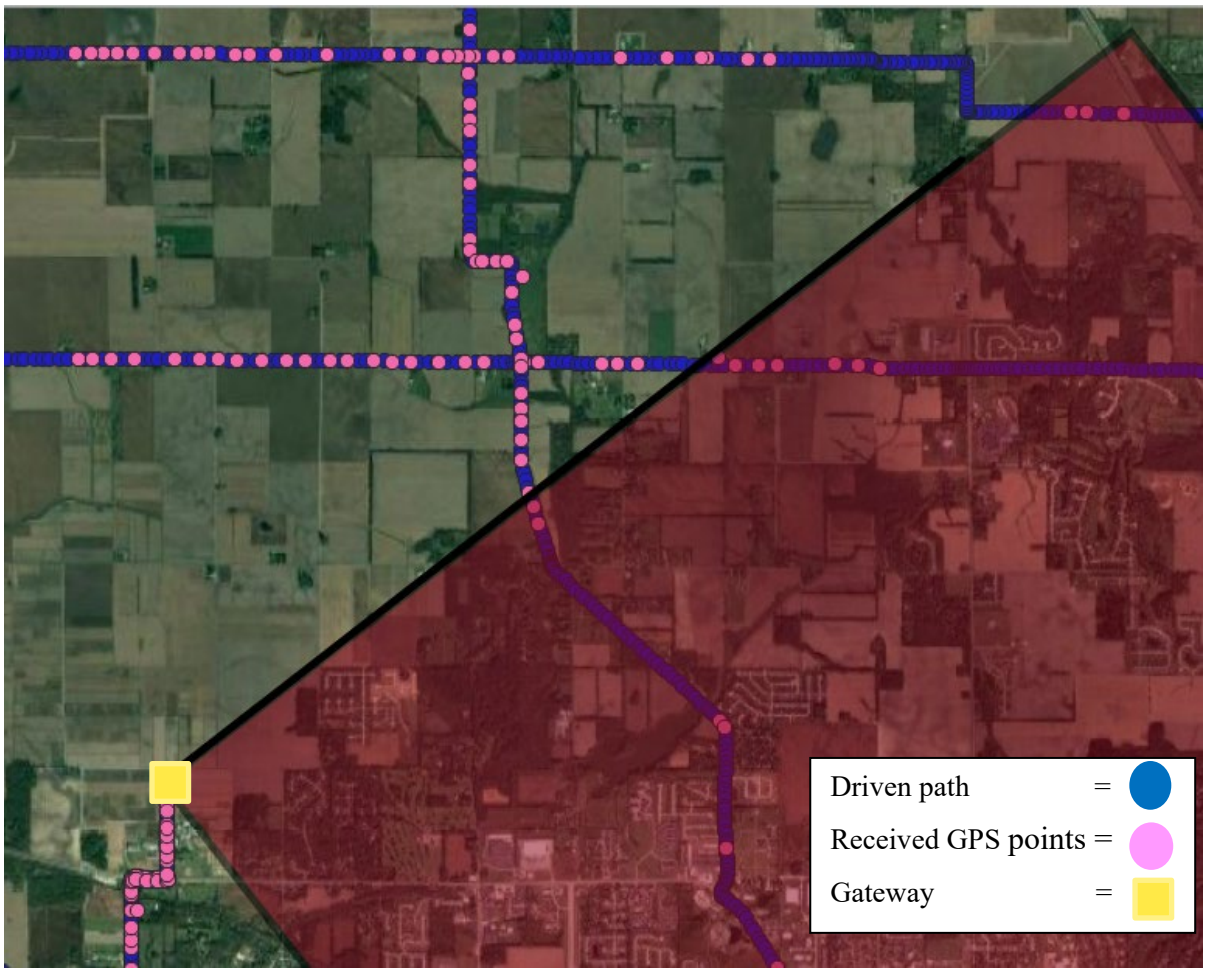

Figure 2.18: Line of sight infringement

Following the tree line just northeast of the gateway tower, a line was drawn beyond which, little to no LoRa signals are received as seen in Figure 2.18. The forest coverage on the ground, seen in red, severely restricts the capabilities of the system to send and receive messages.

This testing was also performed with the gateway at 16m to identify the difference an additional 31’ of height would add or retract from the range of the system. The full data collection of the 16m gateway can be seen in Figure 2.19.

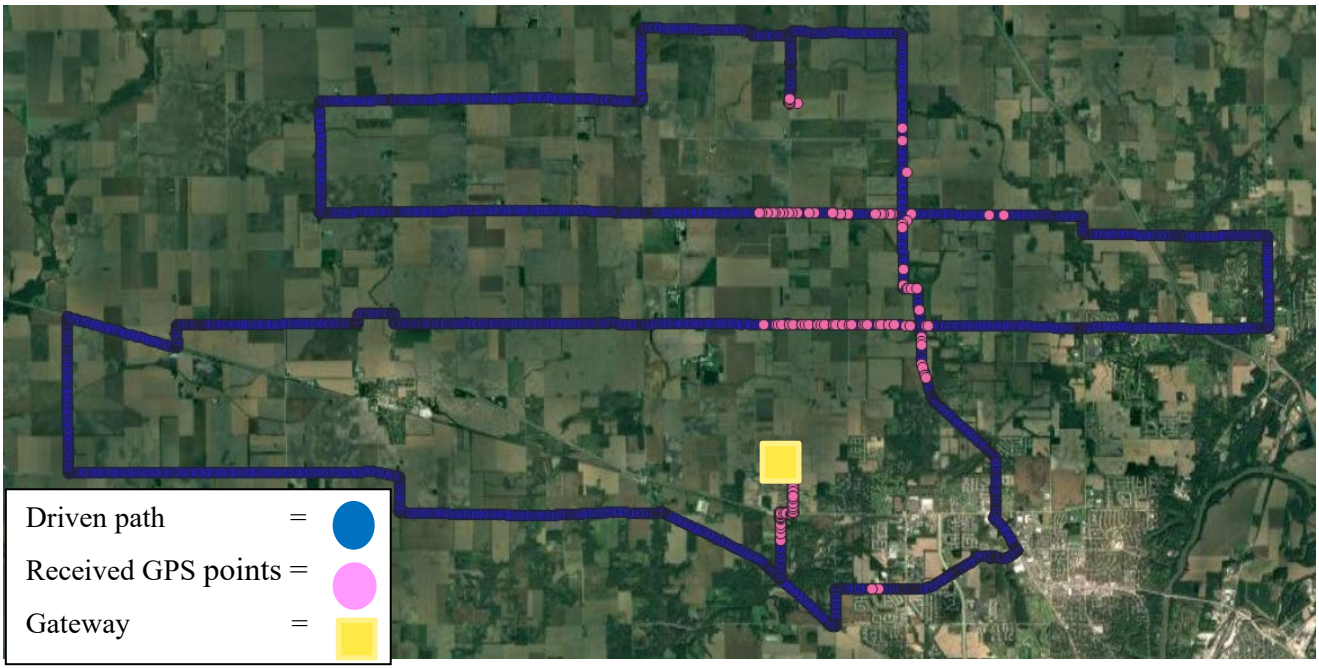

Figure 2.19: 16m gateway testing path with received points

From Figure 2.19 it can be seen in contrast to the 7m gateway in Figure 2.17, the higher elevation gateway performed poorer than the lower gateway. Curiously, it was expected that the higher elevation of the gateway would allow for more terrain clearance and provide a better line of sight for the mobile end node. From the testing however, the 16m gateways maximum range was only 8.16 km. Testing done prior also confirmed this maximum range of the gateway as seen in Figure 2.13. From Equation 2 it was seen that the height of the gateway should increase the range of the signal by a factor of the square of the heights. As such the results from the 16m gateway were unexpected. By increasing the height of the gateway, the overall range of the system was reduced by 1850 meters. It was expected to see an increase of 3.7 km.

This decrease in range was an unexpected result from the testing. Continued testing was also done with the gateway at 10m ground height to determine any large differences in previous testing.

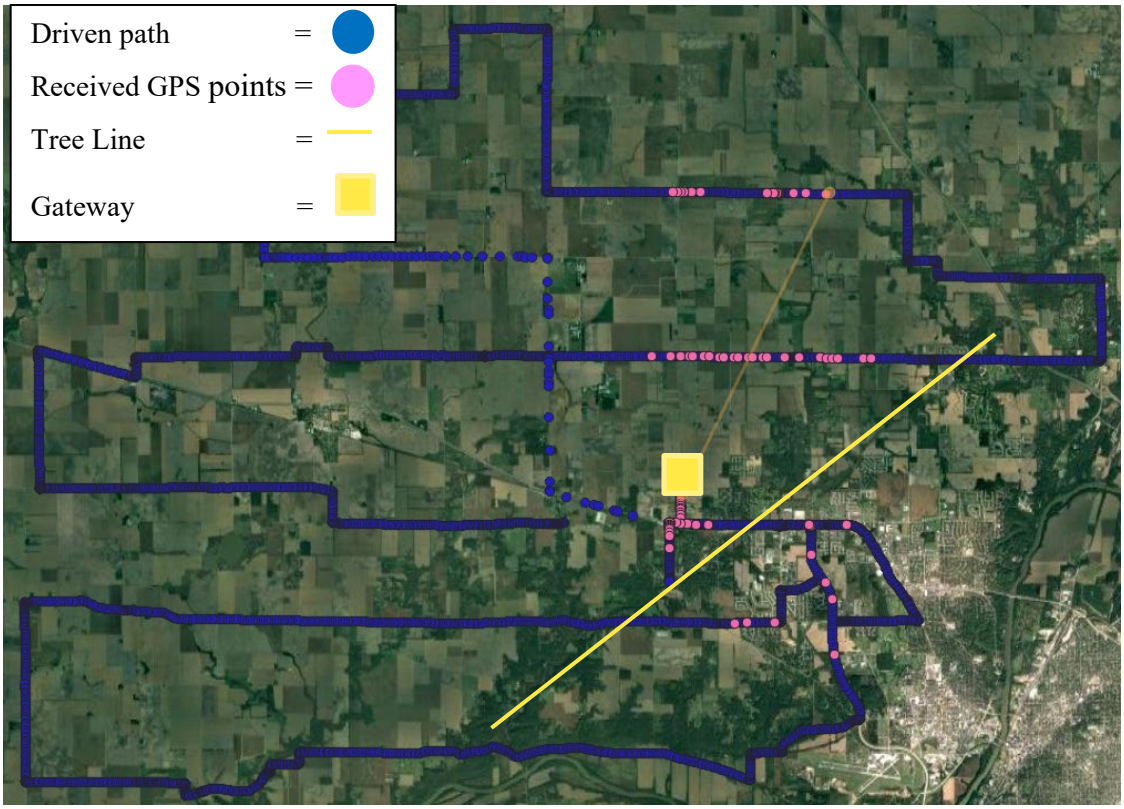

Figure 2.20: 10m gateway testing path with received points highlighting tree coverage

The data from the 10m gateway produced similar results to the 16m gateway. The southern legs of the 10m gateway test highlighted the inability of the LoRa signal when pertaining to piercing through heavy forest and ground cover. As seen in Figure 2.20 the signals were dropped at a rate of near 100% when south of the tree line. The heavy forest did not allow for signal to pass through. The maximum range received from the 10m testing was 8300 meters. This coincided with the reliable data signal range seen in Figure 2.12. However, from the testing of the 10m tower, distinct packet loss was seen along the route, specifically along the most northern leg. Previous testing had shown that area to be within the range of the gateway and yet, a large portion of packets were lost along the route. Like the 16m gateway, the expected outcome of the testing was greater than what was experimentally obtained.

Packet loss across all three gateways heights was measured and compared as seen in Table 2.2. This was collected only in a 10 km distance in areas where signal strength was high and could be counted as moderately reliable, thus ignoring areas with no signal reception of any manner.

Table 2.2: Packet reliability of high signal strength

Tower Height	Packets Sent	Packets Received	Success Rate
7m	411	188	45.7%
10m	259	74	28.5%
16m	411	167	40.6%

In addition, as all three tests were performed in the same area with a data rate of 0.41 Hz, numerical values for sent and received packets within an area can be collected and compared. Directly north of ACRE was an area where high signal was obtained for all three gateways across multiple tests, as such a suitable area to compare signal strength of the three heights in the same area can be selected in the fields north of ACRE. This allows for an accurate comparison of the packet reliability in a broad area free from obstacles.

Table 2.3: Varied heights gateway packet reliability

Tower Height	Packets Sent	Packets Received	Success Rate
7m	978	210	21.5%
10m	703	90	12.8%
16m	978	224	22.9%

Table 2.3 highlights some potential issues that LoRa faces when being utilized in mobile applications. In an area where signal strength has been proven to be strong and where messages are reaching the gateway, the reliability rate of the packet reception is remarkably low. The 10m gateway has extremely poor values in comparison to the 16m and 7m gateway. This potentially could be accounted for based on the position of the gateway upon setup of the telescoping tower. The 10m test varied in that the gateway signals could have been blocked by the material of the tower, while testing of the 7m and 16m gateway took this into account when positioning the gateways.

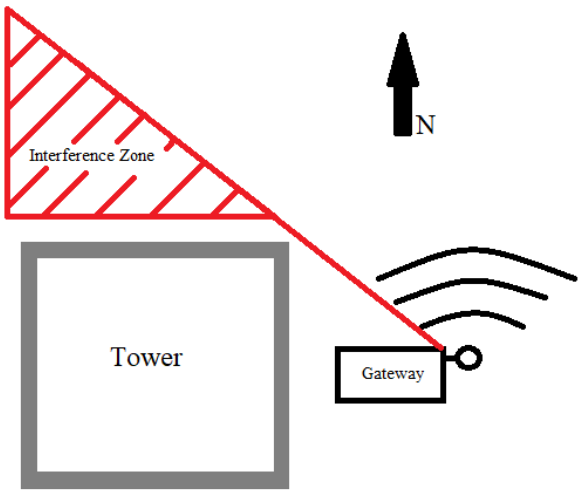

Figure 2.21: Illustration of tower material interference with LoRa signal reception

In Figure 2.21, it can be seen how the material of the tower itself could inhibit the signals from making it to the antenna of the gateway at near and far ranges. As such, signals originating from the northwest of the tower could be obscured from view of the gateway. However, this phenomenon does not explain the poor performance of the 10m gateway with respect to the routes that were due north of its position. In addition, as seen in Table 2.2Table 2.2 the 10m gateway also performed poorly when receiving signals in high reliability areas with a success rate of 28.5%.

Curiously, the 7m gateway outperformed the 16m gateway by 1700m for total range. Table 2.2 and Table 2.3 show similar results when comparing packet loss across the two systems. From Table 2.2 the 16m gateway had a higher packet reliability but a lower overall range. It can be inferred then that the taller gateway had a more reliable signal to closer instances while having a poorer overall range. This was most likely due to the height advantage that the gateway had over the 7m tall tower. However, it was surprising that the 7m tower still outperformed the other gateway by 1700m.

The 7m gateway could have been performing so much better than the other gateway due to the high gain of the mobile antenna. Figure 2.22 showed how radiation pattern of the antenna coupled with the uneven terrain could have created a situation with poor reception for the taller gateway on sloped terrain. Thus, the taller gateway would have better reliability when it was connected due to its height, however it may not have had the range due to the signal radiation missing the height of the gateway and impacting the ground. This could also explain why the 7m

gateway had better results than the 10m gateway as it was most likely in a more calibrated height than the others, thus giving it more range but poorer reliability due to its height.

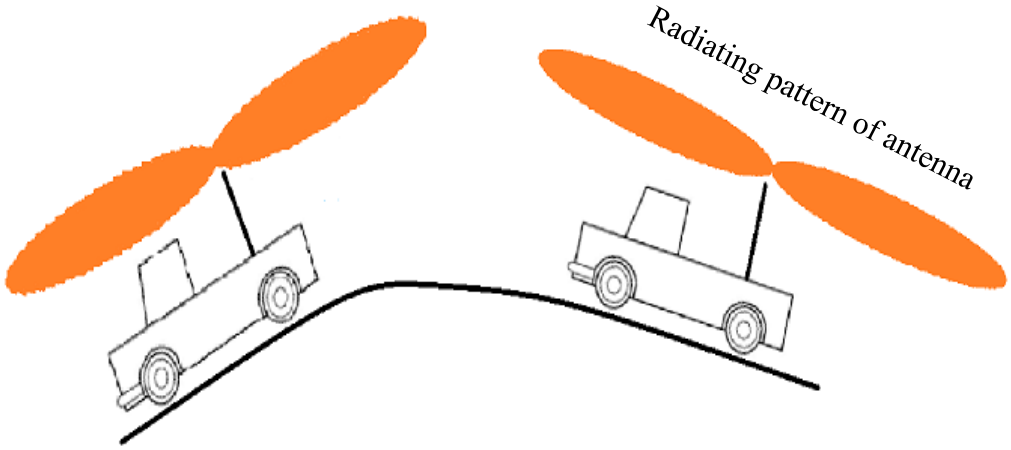

Figure 2.22: High gain antennas irradiation patterns on uneven terrain – not to scale

To further test this, the high gain antenna was removed from the end node to identify what effect it had on the system and if better results could be captured from the higher elevation gateways.

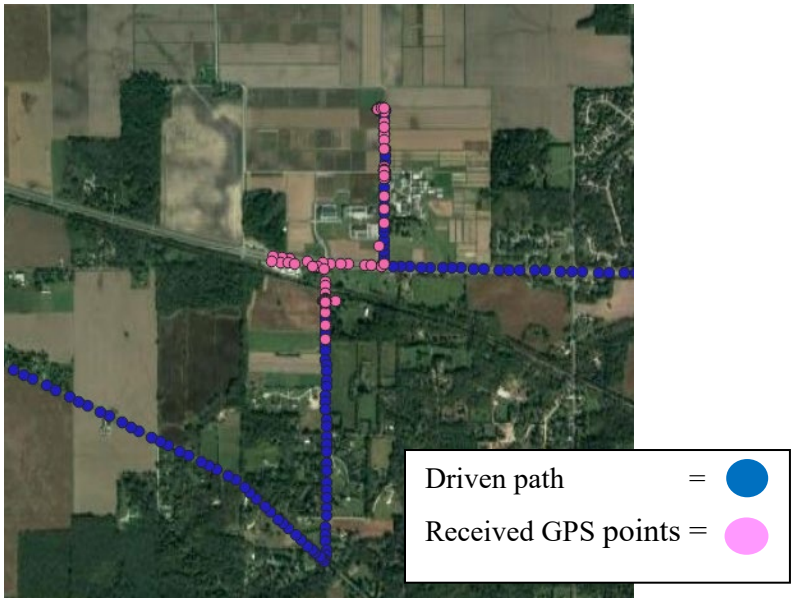

Figure 2.23: Received points with low gain antenna testing

As can be seen in Figure 2.23, the max range of the system without the high gain antenna was 1110 meters from the gateway. This testing solidified the need for a properly calibrated antenna when utilizing LoRa systems as high gain antennas drastically improved the output of the system in comparison to lower gain antennas.

The full set of trial runs and intermediate routes could be compiled to obtain a full coverage map of ACRE farms for all the varying setups of the gateways. From this coverage map, an idea of dead spots on the farm and other high coverage areas can be obtained.

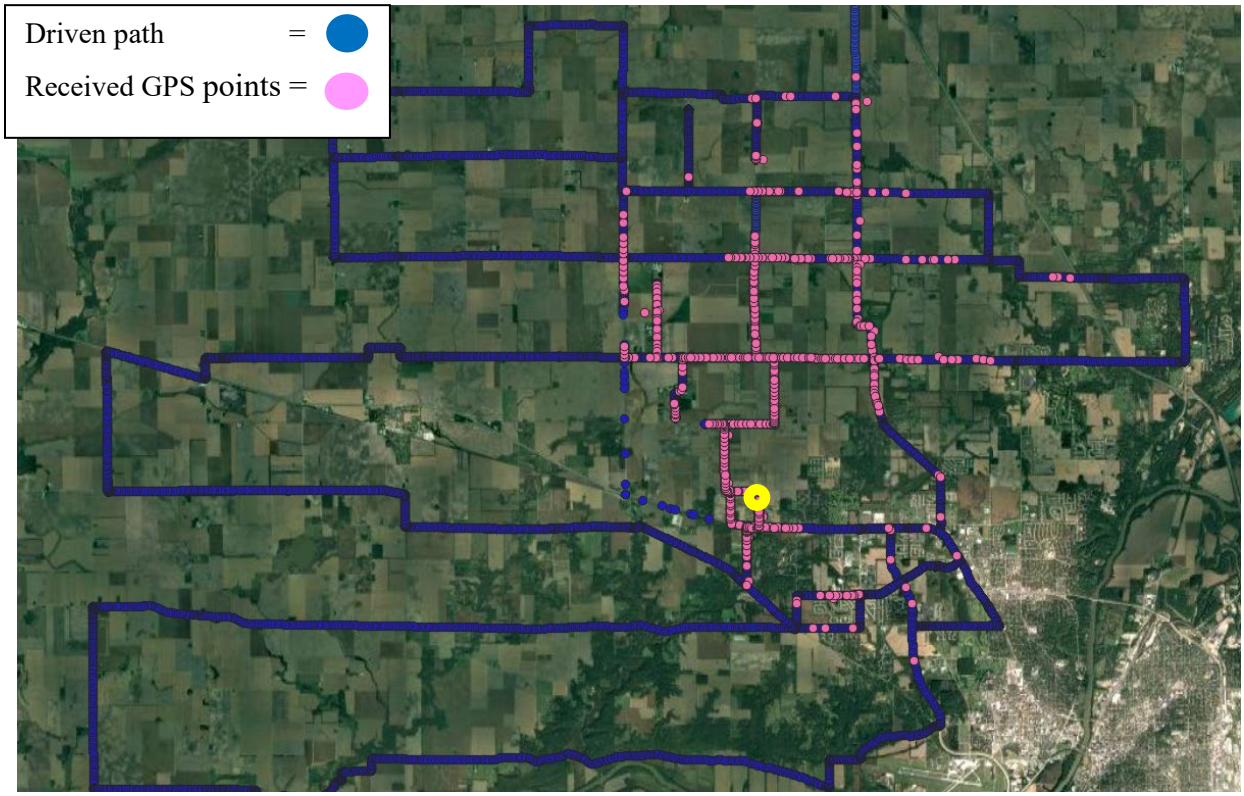

Figure 2.24: ACRE LoRa coverage map from testing runs connecting to 7, 10, and 16m gateway

This map highlights a major key to utilizing mobile LoRa: line of sight is top priority. From Figure 2.24 it can be seen how drastically the tree coverage and landscape shaped the total area of the network signal. South of the gateway no signals were able to penetrate the wooded area which ran along the length of the road. To the northeast, a 45-degree line of coverage can be seen which follows the tree line of the forest. From this, it can be assumed that care must be taken when placing gateways for network creation.

The full coverage map also provides more useful insight. Figure 2.24 shows that there is reliable coverage with a maximum reach of 13 km from end to end. Additionally, the gateways cover over 95 square kilometers of land. However, the reliability of the network was extremely poor. Table 2.2 and Table 2.3 showed a reliability of approximately 40% in areas where signal was regular and only a 23% reliability in areas where signal was known to be strong but signal was irregular. This irregularity can also be seen when comparing the full coverage map to individual tests. The western half of the coverage area was not received during testing as seen in Figure 2.20 but via the coverage map the area tested at that time has full reception and can be received by the gateway at various heights. Due to this, the unreliability of the system is extremely high, when performing optimally, a large area is covered and would be useful for logistical tracking, however, the network does not always receive signals from those areas that had previously been receiving packets. This could be due to the gateway acquiring the signal of the end node as it travels across the landscape. At high speeds, a delay of 1 minute in acquiring the signal can results in kilometers of missed packets.

2.5.3 Speed Testing

To verify this high-speed interaction, testing was done along the same strip with the gateway set at the same height for each pass. The speed of the mobile node was increased to ascertain any differences in signal due to the speed of the end node. It was hypothesized that at higher speeds, the gateway would struggle to obtain signal from the gateway and in addition have a higher packet loss.

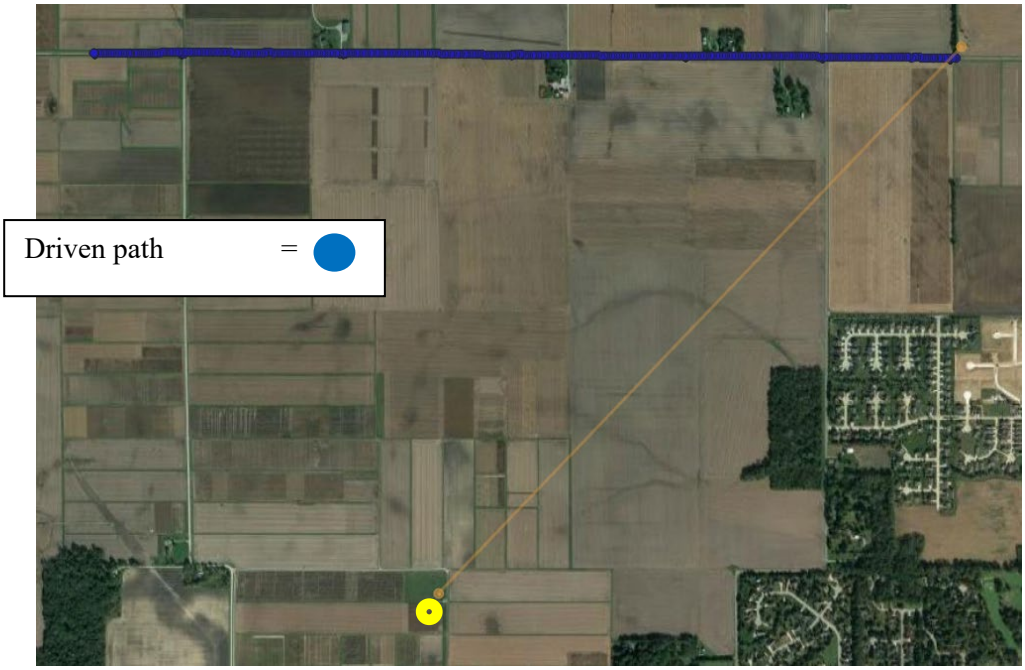

Figure 2.25: Speed testing route

The testing path shown in Figure 2.25 was performed from 8 kmh^{-1} to 96 kmh^{-1}, and the maximum range of the test was 2400 meters, and the data rate was 0.41 Hz. Additionally, the area that testing was performed in was a well know high signal strength area. As such, it was expected that the gateway would be able to see the end node in any stationary configuration.

Table 2.4: High speed packet reliability at 2400m

Speed (kmh^{-1})	Packets Sent	Packets Received	Success Rate
0	123	100	81%
8	292	225	77%
16	230	121	53%
24	163	86	53%
32	117	60	51%
40	102	50	49%
48	76	36	47%
64	61	27	44%
96	35	15	43%

Table 2.4 the high success rate of the baseline stationary testing can be seen. When stable and stationary in the field the LoRa end node transmitted strong steady signals to the gateway with very little packet loss. Diving further into the baseline data, the dropped packets were concentrated at the start of the stationary positions. Thus, the packets that were lost when performing stationary testing, could have been a result of the change in velocity from high speed to 0 kmh^{-1} when the truck came to a stop. The signal would not have had a chance to be connected until after a few seconds had passed at 0 kmh^{-1}, thus the assumed success rate of the baseline can be marginalized as slightly above 81%.

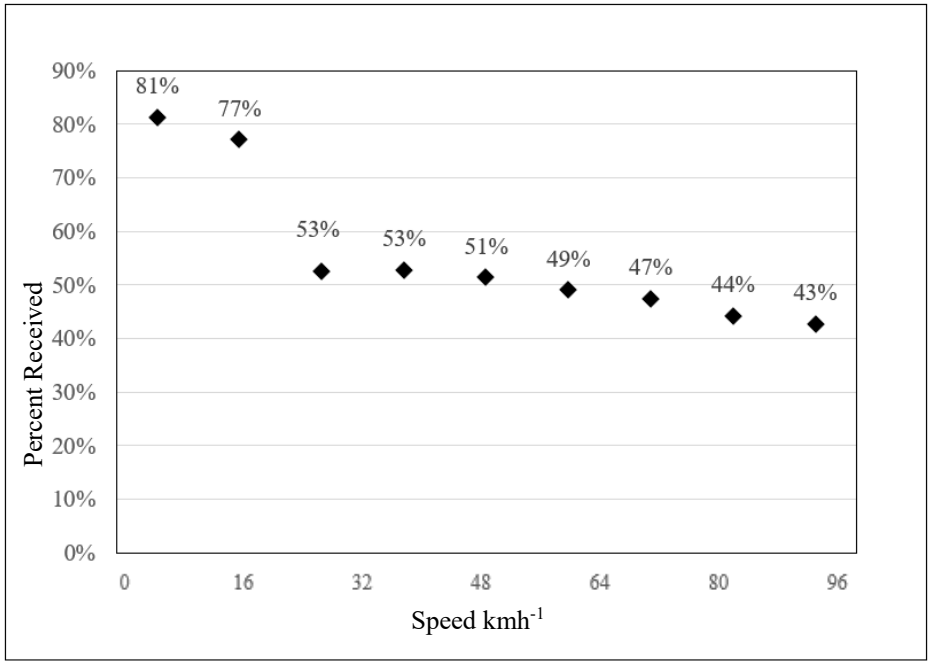

Figure 2.26: Reliability vs travel speed at 0.41 Hz data rate at 2400m

Visualizing the data from Table 2.4 in Figure 2.26, a sharp decline in packet success rate can be seen around the 5mph mark with a steady decline in packet reliability as speed increases. The high-speed testing reveals that the previous unreliability issues with the gateway were most likely due to the speed of the end node as it traveled cross country. The effects of this can be seen dramatically in Figure 2.20 with the 10m gateway. When testing was done on that gateway, the mobile end node was traveling eastward across the northern routes at speeds excess of 96 kmh^{-1}. From the high-speed testing, the packet reliability of that route was at best 43% disregarding other

factors such as line of sight, obstacles, or distance to gateway. When considering the high-speed aspect of the testing, the lack of coverage on the western portion of the map is confirmed as the signal was struggling to be acquired at such high speeds. This would confirm the large difference in packet loss when comparing the total coverage map of ACRE with the coverage test of the 10m gateway. This high-speed packet loss effect can also be seen in the other coverages including the 7m gateway and the 16m gateway.

The high-speed effect also showcased the increase in packet loss around obstacles or intrusions in line of sight. When navigating around obstacles which would impede the LoRa signal, the slower speeds had better reconnecting times with the gateway when compared to the increased speed testing. Additionally, the high speed of the end node increased the effect that the obstacle had on the signal. Intrusions into the line of sight would cause much larger signal loss when compared at high speeds when compared with lower speeds.

Figure 2.27: 8kmh^{-1} and 96 kmh^{-1} packet reception at 2400m

Figure 2.27 highlights the effect that the high speed had on packet loss around line-of-sight obstacles. As seen by the dark blue, the driven path extends equally along the route, however, at two distinct points obstacles stop the LoRa signal as seen by the two gaps in the cyan markets. Signal were received in between the two points, yet the 96 kmh^{-1} testing had no reception through the whole area. The obstacles impeded the signal such that the gateway fully lost reception of the signal and reacquisition took much long at speed. Thus, a large gap was seen where coverage drops and remains unreachable in an area where it is known to have high signal.

2.6 Conclusion

A LoRa network was created in order fully test the capabilities of the technology at ACRE farms in West Lafayette. A mobile end node was used in conjunction with gateways at varying heights to identify and create a coverage map of the area. Further testing was performed on the network to identify the effect that high speed had on the system. From the testing it was found that under best circumstances, the network had a packet reliability of 45.7% while testing using a mobile end node. This reliability decreased further when looking at set areas where signal strength was known to be acceptable. High speed testing revealed that at speeds of 16 kmh^{-1} the reliability of the network drops by 24% in comparison to 5mph. The high-speed aspect of fleet tracking makes LoRa an unsuitable technology for its use in agriculture. Similar tests produced wide varieties of coverage maps using the same techniques. The high speed interfered too much with the signal reliability to be utilized it in a fleet tracking manner for grain trucks. In addition, testing revealed the effect line of sight had on the system. Heavily wooded areas produced dead zones which would not allow signal through it. The terrain of rural America would heavily interfere with signal with regards to dynamic systems such as grain truck pathing. As such, LoRa is not a suitable technology for use in mobile high speed fleet tracking. However, from the analysis of the data, LoRa could be utilized for in field tracking of slower machines such as combines and grain carts, under the condition that line of sight can be made between machine and gateway.

BIBLIOGRAPHY

Amiama, Carlos, José M. Pereira, Angel Castro, and Javier Bueno. 2015. "Modelling Corn Silage Harvest Logistics for a Cost Optimization Approach." *Computers and Electronics in Agriculture* 118:56–65. doi: 10.1016/j.compag.2015.08.024.

Anon. 2020b. "(Infographic) The U.S. Farm Labor Shortage." *AgAmerica*. Retrieved June 16, 2021 (https://agamerica.com/blog/the-impact-of-the-farm-labor-shortage/).

Bochtis, Dionysis, and Claus Sørensen. 2009. "The Vehicle Routing Problem in Field Logistics Part I." *Biosystems Engineering* 104. doi: 10.1016/j.biosystemseng.2009.09.003.

Bronars, Stephen. 2015. "A Vanishing Breed How the Decline in U.S. Farm Laborers Over the Last Decade Has Hurt the U.S. Economy and Slowed Production on American Farms." *Partnership for a New American Economy* 25.

Buckmaster, Dennis R., and James W. Hilton. 2005. "Computerized Cycle Analysis of Harvest, Transport, and Unload Systems." *Computers and Electronics in Agriculture* 47(2):137–47. doi: 10.1016/j.compag.2004.11.015.

Busato, Patrizia, Alessandro Sopegno, Niccolò Pampuro, Luigi Sartori, and Remigio Berruto. 2019. "Optimisation Tool for Logistics Operations in Silage Production." *Biosystems Engineering* 180:146–60. doi: 10.1016/j.biosystemseng.2019.01.008.

Farooq, Muhammad Shoaib, Shamyla Riaz, Adnan Abid, Tariq Umer, and Yousaf Bin Zikria. 2020. "Role of IoT Technology in Agriculture: A Systematic Literature Review." *Electronics* 9(2):319. doi: 10.3390/electronics9020319.

Haffar, Imad, and Ramzi Khoury. 1992. "A Computer Model for Field Machinery Selection under Multiple Cropping." *Computers and Electronics in Agriculture* 7(3):219–29. doi: 10.1016/S0168-1699(05)80021-3.

Heizinger, Valentin, and Heinz Bernhardt. n.d. "Algorithmic Efficiency Analysis of Harvest and Transport of Biomass." 5.

Hufford, George A. 1982. "A Guide to the Use of the ITS Irregular Terrain Model in the Area Prediction Mode."

Ji-chun Zhao, Jun-feng Zhang, Yu Feng, and Jian-xin Guo. 2010. "The Study and Application of the IOT Technology in Agriculture." Pp. 462–65 in *2010 3rd International Conference on Computer Science and Information Technology*. Vol. 2.

J. J. Egli, "Radio Propagation above 40 MC over Irregular Terrain," in *Proceedings of the IRE*, vol. 45, no. 10, pp. 1383-1391, Oct. 1957, doi: 10.1109/JRPROC.1957.278224.

Jung, Jinha. 2020a. “Indiana DEP Lidar Data Products Data Accuracy and Data Access.” *Https://Lidar.Jinha.Org/*. Retrieved July 6, 2021 (https://lidar.jinha.org/docs/Indiana_3DEP_Lidar_Data_Products_Data_Accuracy_and_Data_Access.pdf).

Kane, Tom, and Bernard Borghei. 2017. “WIA_RuralAmerica-2.Pdf.” *WIA*. Retrieved June 23, 2021 (https://wia.org/wp-content/uploads/WIA_RuralAmerica-2.pdf).

Kenbiba. 2016. GitHub Repository, *https://github.com/kenbiba/RH-RF95.*

Khanna, Abhishek, and Sanmeet Kaur. 2019. “Evolution of Internet of Things (IoT) and Its Significant Impact in the Field of Precision Agriculture.” *Computers and Electronics in Agriculture* 157:218–31. doi: 10.1016/j.compag.2018.12.039.

Kooijman, Matthijs. (2015). GitHub Repository *https://github.com/Matthijskooijman/Arduino-Lmic*.

Lakhwani, Kamlesh, Hemant Gianey, Niket Agarwal, and Shashank Gupta. 2019. “Development of IoT for Smart Agriculture a Review.” Pp. 425–32 in *Emerging Trends in Expert Applications and Security*. Vol. 841, *Advances in Intelligent Systems and Computing*, edited by V. S. Rathore, M. Worring, D. K. Mishra, A. Joshi, and S. Maheshwari. Singapore: Springer Singapore.

Lavric, Alexandru, and Valentin Popa. 2018. “Performance Evaluation of LoRaWAN Communication Scalability in Large-Scale Wireless Sensor Networks.” *Wireless Communications and Mobile Computing* 2018:e6730719. doi: 10.1155/2018/6730719.

Layton, Alexander W., Yaguang Zhang, James V. Krogmeier, and Dennis R. Buckmaster. 2017. “Determining Harvesting Efficiency via Multiple Combine GPS Logs.” in *2017 Spokane, Washington July 16 - July 19, 2017*. American Society of Agricultural and Biological Engineers.

Lee, Junhyuk. (2019). GitHub Repository *https://github.com/Neosarchizo/TinyGPS*.

L-com. n.d. “HyperLink Wireless 800/900 MHz 6 DBi High Performance Omnidirectional Antenna.” *L-Com Global Connectivity*. Retrieved June 24, 2021a (https://www.l-com.com/Images/Downloadables/Datasheets/ds_HGV-906U.pdf).

Petäjäjärvi, Juha, Konstantin Mikhaylov, Marko Pettissalo, Janne Janhunen, and Jari Iinatti. 2017. “Performance of a Low-Power Wide-Area Network Based on LoRa Technology: Doppler Robustness, Scalability, and Coverage.” *International Journal of Distributed Sensor Networks* 13(3):1550147717699412. doi: 10.1177/1550147717699412.

Roekel, Ryan J. Van, and Jeffrey A. Coulter. 2011. “Agronomic Responses of Corn to Planting Date and Plant Density.” *Agronomy Journal* 103(5):1414–22. doi: 10.2134/agronj2011.0071.

Semtech. n.d. “SX1276 | 137MHz to 1020MHz Long Range Low Power Transceiver | Semtech.” Retrieved June 24, 2021b (https://www.semtech.com/products/wireless-rf/lora-core/sx1276).

Tummers, J., A. Kassahun, and B. Tekinerdogan. 2019. “Obstacles and Features of Farm Management Information Systems: A Systematic Literature Review.” *Computers and Electronics in Agriculture* 157:189–204. doi: 10.1016/j.compag.2018.12.044.

Turner, Aaron P. 2018. “Development of a Decision Support System for Capacity Planning from Grain Harvest to Storage.” University of Kentucky Libraries. doi.org/10.13023/etd.2018.376

Zhang, Yaguang, Aaron Ault, James V. Krogmeier, and Dennis Buckmaster. 2017. “Activity Recognition for Harvesting via GPS Tracks.” in *2017 Spokane, Washington July 16 - July 19, 2017*. American Society of Agricultural and Biological Engineers.

Zhang, Yaguang, James V. Krogmeier, Aaron Ault, and Dennis Buckmaster. 2020. “APT3: Automated Product Traceability Trees Generated from GPS Tracks.” *Transactions of the ASABE* 63(3):571–82. doi: 10.13031/trans.13384.

APPENDIX

Code

Data Frame Join

```
# -*- coding: utf-8 -*-
"""
Created on Tue Oct 19 10:54:31 2021

@author: loghe
"""

### CREATE SIUNGULAR DATA FRAME from mismatched cels

import pandas as pd

date = "10-2"


graincart =     pd.read_csv(r'C:/Users/loghe/Desktop/ACRE unload/cleaned grain cart data/' + date + '.csv')
combine =       pd.read_csv(r'C:/Users/loghe/Desktop/ACRE unload/cleaned S660 data/' + date + '.csv') # s660
# combine =       pd.read_csv(r'C:/Users/loghe/Desktop/ACRE unload/cleaned S670 data/' + date + '.csv') # s660
freightliner =  pd.read_csv(r'C:/Users/loghe/Desktop/ACRE unload/cleaned freight data/' + date + '.csv')
international = pd.read_csv(r'C:/Users/loghe/Desktop/ACRE unload/cleaned int data/' + date + '.csv')
```

```
graincart = graincart.set_index(graincart['TIME'])
# graincart = graincart.dropna()
# graincart.index = pd.to_datetime(graincart.index)
# graincart.index = graincart.index + pd.Timedelta(hours = -5)

combine = combine.set_index(combine['TIME'])
# combine.index = pd.to_datetime(combine.index)
# combine.index = combine.index + pd.Timedelta(hours = -5)



freightliner =  freightliner.set_index(freightliner['TIME'])
# freightliner.index = pd.to_datetime(freightliner.index)
# freightliner.index = freightliner.index + pd.Timedelta(hours = -5)

international = international.set_index(international['TIME'])
# international.index = pd.to_datetime(international.index)
# international.index = international.index + pd.Timedelta(hours = -5)

df= combine.join(graincart,lsuffix = " S660" ,rsuffix =' Grain Cart')
df = df.join(international, rsuffix = ' international')
df = df.join(freightliner, rsuffix = ' freightliner')

df = df.drop(labels = ['DAY S660','Hours S660','Minutes S660','Seconds S660', 'Year S660' ,
                       'Month S660','DAY Grain Cart','Hours Grain Cart','Minutes Grain Cart',
                       'Month Grain Cart','Seconds Grain Cart','Year Grain Cart','TIME S660',
                       'TIME','TIME Grain Cart','D1','D5','D0','ABSTime S660',
                       'TIME freightliner','ABSTime Grain Cart',
                       'Heading Grain Cart'
                        ],axis = 1)  #'Unnamed: 0 S660','Unnamed: 0 Grain Cart',
df['Lat Grain Cart'].fillna(method = 'ffill',inplace = True)
df['Long Grain Cart'].fillna(method = 'ffill',inplace = True)
```

```
df['Lat int'].fillna(method = 'ffill',inplace = True)
df['Long int'].fillna(method = 'ffill',inplace = True)

df['Lat freight'].fillna(method = 'ffill',inplace = True)
df['Long freight'].fillna(method = 'ffill',inplace = True)

df.to_csv(r'C:/Users/loghe/Desktop/ACRE unload/combined data/'+ date + ' combined.csv')
```

State algorithm

```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 21 22:25:18 2021

@author: loghe
"""
## SCRIPT FOR COMBINED DATA

import pandas as pd
import math
from math import radians, cos, sin, asin, sqrt
import numpy as np


date2 = "10-2"
```

```
# perfected = pd.read_csv(r'C:/Users/loghe/Desktop/ACRE unload/full data/' + date2 + '/combined data.csv')
perfected = pd.read_csv(r'C:/Users/loghe/Desktop/ACRE unload/cleaned combined data points/' + date2 + '.csv')
perfect_S660 = perfected['state'].tolist()
perfect_S660_index = perfected.TIME
perfect_grain_cart = perfected['Cart State'].tolist()

# perfect_Grain_Cart = perfected['Cart state'].tolist()


def script(speed , gps_offset , list_length, closing_speed, truck_gps_offset):


    tic = 1

    for zz in range(tic):
        # df = pd.read_csv(r'C:/Users/loghe/Desktop/ACRE unload/full data/' + date2 + '/combined data.csv')
        df = pd.read_csv(r'C:/Users/loghe/Desktop/ACRE unload/cleaned combined data points/' + date2 + '.csv')
        df = df[ int(len(df)/(tic)*zz) : int(len(df)/(tic)*(1 + zz))]

        S660LAT = df['Lat S660'].tolist() #combine
        S660LONG = df['Long S660'].tolist()
        # S660SPEED = df['SPEEDS660'].tolist()

        Grain_CartLAT = df['Lat Grain'].tolist() #cart
        Grain_CartLONG = df['Long Grain'].tolist()
        # Grain_CartSPEED = df['SPEEDGrain_Cart'].tolist()
```

```
internationalLAT = df['Lat int'].tolist() #truck
internationalLONG = df['Long int'].tolist()
# internationalSPEED = df['SPEEDinternational'].tolist()

# ih19LAT = df['LATih19'].tolist() #truckpd
# ih19LONG = df['LONGih19'].tolist()
# # ih19SPEED = df['SPEEDih19'].tolist()

freightlinerLAT = df['Lat freight'].tolist() #truck
freightlinerLONG = df['Long freight'].tolist()
# freightlinerSPEED = df['SPEEDfreightliner'].tolist()

# df = df.set_index(df['DATE'])


dftime = df.index.tolist()



f = open(r'C:/Users/loghe/Desktop/unloads/STATEMACHINE/' + date2 +'interim.txt',"w")
f.close
f = open(r'C:/Users/loghe/Desktop/unloads/STATEMACHINE/' + date2 + 'interim.txt',"a")


state = ''

speedS660 =[]
speedGrain_Cart = []
speedinternational = []
# speedih19 = []
speedfreightliner = []
```

```
statelist = []

def haversine(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 3959 # Radius of earth in mi. Use 3959 for miles
    return c * r * 3600

for j in range(len(dftime)):
        speedS660.append(haversine(S660LONG[j-1], S660LAT[j-1], S660LONG[j], S660LAT[j]))
        speedGrain_Cart.append(haversine(Grain_CartLONG[j-1], Grain_CartLAT[j-1], Grain_CartLONG[j], Grain_CartLAT[j]))
        speedinternational.append(haversine(internationalLONG[j-1], internationalLAT[j-1], internationalLONG[j], internationalLAT[j]))
        #  speedih19.append(haversine(ih19LONG[j-1], ih19LAT[j-1], ih19LONG[j], ih19LAT[j]))
        speedfreightliner.append(haversine(freightlinerLONG[j-1], freightlinerLAT[j-1], freightlinerLONG[j], freightlinerLAT[j]))
```

```
    speedS660[0] = 0
    speedinternational[0] = 0
    speedGrain_Cart[0] = 0
    # speedih19[0] = 0
    speedfreightliner[0] = 0


    #calculate moving average of the speed the length of the window and place in a holder
    window = 5
    intermediate_avspeedS660 = np.convolve(speedS660, np.ones(window)/window, mode='valid')
    intermediate_avspeedGrain_Cart = np.convolve(speedGrain_Cart, np.ones(window)/window, mode='valid')
    intermediate_avspeedinternational = np.convolve(speedinternational, np.ones(window)/window, mode='valid')
    # intermediate_avspeedih19 = np.convolve(speedih19, np.ones(window)/window, mode='valid')
    intermediate_avspeedfreightliner = np.convolve(speedfreightliner, np.ones(window)/window, mode='valid')

    avspeedS660 = []
    avspeedGrain_Cart = []
    avspeedinternational = []
    avspeedfreightliner = []
    # avspeedih19 = []

    # fixes the first portion of the window due to no moving average there and place into final average speed list for each
    for i in range(window-1):
        avspeedS660.append(speedS660[i])
```

```
for i in range(len(intermediate_avspeedS660)):
    avspeedS660.append(intermediate_avspeedS660[i])


for i in range(window-1):
    avspeedGrain_Cart.append(speedGrain_Cart[i])
for i in range(len(intermediate_avspeedGrain_Cart)):
    avspeedGrain_Cart.append(intermediate_avspeedGrain_Cart[i])


for i in range(window-1):
    avspeedinternational.append(speedinternational[i])
for i in range(len(intermediate_avspeedinternational)):
    avspeedinternational.append(intermediate_avspeedinternational[i])


# for i in range(window-1):
#     avspeedih19.append(speedih19[i])
# for i in range(len(intermediate_avspeedih19)):
#     avspeedih19.append(intermediate_avspeedih19[i])


for i in range(window-1):
    avspeedfreightliner.append(speedfreightliner[i])
for i in range(len(intermediate_avspeedfreightliner)):
    avspeedfreightliner.append(intermediate_avspeedfreightliner[i])


"""USES MPH FROM HAVERSINE"""
```

```
    """FIRST STATE MACHINE FOR COMBINE
    #
    #
    #
    """

    # gps_offset = .00017


    for j in range(len(dftime)):

     # """setup state  machine and state swaps"""
        #harvest moving and not near grain cart

      if (avspeedS660[j] >= speed) and (abs(S660LAT[j]-Grain_CartLAT[j]) >= gps_offset or
abs(S660LONG[j]-Grain_CartLONG[j]) >= gps_offset):
            state = "harvest"
            # statelist.append('1')
            statelist.append('harvest')

      #unload on go - moving and near cart - and similar speed as prior
      elif (avspeedS660[j] >= speed)  and (abs(S660LAT[j]-Grain_CartLAT[j]) <= gps_offset
and abs(
            S660LONG[j]-Grain_CartLONG[j]) <= gps_offset) and abs(avspeedS660[j] -
avspeedGrain_Cart[j]) < closing_speed :
            state = "onthego"
            # statelist.append('2')
            statelist.append('on the go')

      #harvest - moving and near cart but not same speed
```

```
        elif (avspeedS660[j] >= speed)  and (abs(S660LAT[j]-Grain_CartLAT[j]) <= gps_offset
and abs(
                S660LONG[j]-Grain_CartLONG[j]) <= gps_offset) and abs(avspeedS660[j] -
avspeedGrain_Cart[j]) >= closing_speed :
            state = "harvest"
            # statelist.append('2')
            statelist.append('harvest')

        #stationary unload - not moving and near cart  and similar speed
        elif (avspeedS660[j] < speed) and (abs(S660LAT[j]-Grain_CartLAT[j]) <= gps_offset and
abs(
                S660LONG[j]-Grain_CartLONG[j]) <= gps_offset) and abs(avspeedS660[j] -
avspeedGrain_Cart[j]) < closing_speed:
            state = "stationary"
            # statelist.append('3')
            statelist.append('stationary unload')

        #idle - not moving and near cart  and not similar speed
        elif (avspeedS660[j] < speed) and (abs(S660LAT[j]-Grain_CartLAT[j]) <= gps_offset and
abs(
                S660LONG[j]-Grain_CartLONG[j]) <= gps_offset) and abs(avspeedS660[j] -
avspeedGrain_Cart[j]) >= closing_speed:
            state = "idle"
            # statelist.append('3')
            statelist.append('idle')

        #idle - not moving and not near cart
        elif (avspeedS660[j] < speed) and (abs(S660LAT[j]-Grain_CartLAT[j]) > gps_offset or
abs(
                S660LONG[j]-Grain_CartLONG[j]) > gps_offset):
            state = "idle"
```

```
            # statelist.append('3')
            statelist.append('idle')



            #check for na
        elif math.isnan(Grain_CartLAT[j]) or math.isnan(Grain_CartLONG[j]) or
math.isnan(S660LAT[j]) or math.isnan(S660LONG[j]):
                statelist.append('5')

            #catch all for states
        else :
                statelist.append('6')




        #  """ start of states and state work"""
    S660_state = statelist
    updatedstate = statelist



    df_S660 = pd.DataFrame(S660_state)
    df_S660.columns = ['STATE']
    df_S660['SHIFT'] = df_S660.STATE.shift(periods = -1, fill_value = 1)



    shift = df_S660['SHIFT'].to_list()
    step = [0] * len(shift)
    count = [0] * len(shift)
    p = 0
```

```
#sets the step based on the differences in the offset we use this step later to fix the "true state" also creates the count of in a row counter
for i  in range(len(shift)):
    if S660_state[i] != shift[i]:
        step[i] = S660_state[i]

for i in range(len(shift)):
    if step[i] != 0 :
        count[i] = 0
        p = 0
    elif step[i] == 0 :
        p = p + 1
        count[i] = p



df_S660['STEP'] = step
df_S660['COUNT'] = count
df_S660['COUNT'] = df_S660.COUNT.shift(periods = 1, fill_value = 1)

df_S660_reduced = df_S660.loc[(df_S660['STEP'] != 0 ) & (df_S660['COUNT'] > list_length)]



S660_time = df_S660_reduced.index.tolist()
time =[0]* len(df_S660_reduced)



for i in range(len(S660_time)):
    time[i] = dftime[S660_time[i]]
```

```
df_S660_reduced['TIME'] = time
del df_S660_reduced['SHIFT']
del df_S660_reduced['STEP']

index = list(df_S660_reduced.index.values)
statelist = df_S660_reduced['STATE'].tolist()
countlist = df_S660_reduced['COUNT'].tolist()
updatedstate_S660 = [0] * len(S660LAT)

#changes statelist with the updated states. this will apply backwards for unkown areas where it flips bewtween states
for i in range(len(index)-1):
    for j in range(index[i],index[i+1]):
        updatedstate_S660[j] = df_S660_reduced.STATE.iloc[i+1]
for i in range(index[0]):
    updatedstate_S660[i] = statelist[0]
for i in range(index[-1],len(updatedstate_S660)):
    updatedstate_S660[i] = statelist[-1]



"""added work for grain cart to truck unloads


    ###
    #
    #
    #
    #
    """
```

```
cart_state = [0]*len(updatedstate)
#these are repeated values from above which can be changed for the grain cart state machine
# gps_offset = .00017


for j in range(len(cart_state)):

    ##on the go
    if (avspeedGrain_Cart[j] >= speed)  and (abs(S660LAT[j]-Grain_CartLAT[j]) <= gps_offset and abs(
        S660LONG[j]-Grain_CartLONG[j]) <= gps_offset) and abs(avspeedS660[j] - avspeedGrain_Cart[j]) < closing_speed:
        cart_state[j] = "on the go"



    ##transporting - moving and near cart but not same speed
    elif (avspeedGrain_Cart[j] >= speed)  and (abs(S660LAT[j]-Grain_CartLAT[j]) <= gps_offset and abs(
        S660LONG[j]-Grain_CartLONG[j]) <= gps_offset) and abs(avspeedS660[j] - avspeedGrain_Cart[j]) >= closing_speed:
        cart_state[j] = "transporting"


    #stationary unload
    elif (avspeedGrain_Cart[j] < speed)  and (abs(S660LAT[j]-Grain_CartLAT[j]) <= gps_offset and abs(
        S660LONG[j]-Grain_CartLONG[j]) <= gps_offset) and abs(avspeedS660[j] - avspeedGrain_Cart[j]) < closing_speed:
        cart_state[j] = "stationary"
```

```
        #waiting - near cart but not same speed
        elif (avspeedGrain_Cart[j] < speed)  and (abs(S660LAT[j]-Grain_CartLAT[j]) <= gps_offset and abs(
                S660LONG[j]-Grain_CartLONG[j]) <= gps_offset) and abs(avspeedS660[j] - avspeedGrain_Cart[j]) >= closing_speed:
            cart_state[j] = "waiting"

        #stationary unload to international
        elif (avspeedGrain_Cart[j] < speed) and (abs(internationalLAT[j]-Grain_CartLAT[j]) <= truck_gps_offset and abs(internationalLONG[j]-Grain_CartLONG[j]) <= truck_gps_offset) and abs(avspeedinternational[j] - avspeedGrain_Cart[j]) < closing_speed:
            cart_state[j]  = "cart unload"


        #stationary unload to freightliner
        elif (avspeedGrain_Cart[j] < speed) and (abs(freightlinerLAT[j]-Grain_CartLAT[j]) <= truck_gps_offset and abs(freightlinerLONG[j]-Grain_CartLONG[j]) <= truck_gps_offset) and abs(avspeedfreightliner[j] - avspeedGrain_Cart[j]) < closing_speed:
            cart_state[j]  = "cart unload"


        # #stationary unload to freightliner
        #  elif (avspeedGrain_Cart[j] < speed) and (abs(ih19LAT[j]-Grain_CartLAT[j]) <= truck_gps_offset and abs(ih19LONG[j]-Grain_CartLONG[j]) <= truck_gps_offset) and abs(avspeedih19[j] - avspeedGrain_Cart[j]) < closing_speed:
        #     cart_state[j]  = "ih19 unload"


        ##last three without similar speeds
```

```
        #stationary unload to international
        elif (avspeedGrain_Cart[j] < speed) and (abs(internationalLAT[j]-Grain_CartLAT[j]) <=
truck_gps_offset and abs(internationalLONG[j]-Grain_CartLONG[j]) <= truck_gps_offset) and
abs(avspeedinternational[j] - avspeedGrain_Cart[j]) >= closing_speed:
            cart_state[j]  = "waiting"


        #stationary unload to freightliner
        elif (avspeedGrain_Cart[j] < speed) and (abs(freightlinerLAT[j]-Grain_CartLAT[j]) <=
truck_gps_offset and abs(freightlinerLONG[j]-Grain_CartLONG[j]) <= truck_gps_offset) and
abs(avspeedfreightliner[j] - avspeedGrain_Cart[j]) >= closing_speed:
            cart_state[j]  = "waiting"


        # #stationary unload to freightliner
        # elif (avspeedGrain_Cart[j] < speed) and (abs(ih19LAT[j]-Grain_CartLAT[j]) <=
truck_gps_offset and abs(ih19LONG[j]-Grain_CartLONG[j]) <= truck_gps_offset) and
abs(avspeedih19[j] - avspeedGrain_Cart[j]) >= closing_speed:
        #     cart_state[j]  = "waiting"


        #waiting
        elif (avspeedGrain_Cart[j] < speed):
            cart_state[j] = "waiting"

        #idle
        elif(avspeedGrain_Cart[j] >= speed):
            cart_state[j] = "transporting"

        #NA
        else:
```

```
        cart_state[j] = "NA"



    df_cart = pd.DataFrame(cart_state)
    df_cart.columns = ['STATE']
    df_cart['SHIFT'] = df_cart.STATE.shift(periods = -1, fill_value = 1)




    shift = df_cart['SHIFT'].to_list()
    step = [0] * len(shift)
    count = [0] * len(shift)
    p = 0

    #sets the step based on the differences in the offset we use this step later to fix the "true state"
also creates the count of in a row counter
    for i  in range(len(shift)):
       if cart_state[i] != shift[i]:
          step[i] = cart_state[i]

    for i in range(len(shift)):
       if step[i] != 0 :
          count[i] = 0
          p = 0
       elif step[i] == 0 :
          p = p + 1
          count[i] = p



    df_cart['STEP'] = step
    df_cart['COUNT'] = count
```

```
df_cart['COUNT'] = df_cart.COUNT.shift(periods = 1, fill_value = 1)

df_cart_reduced = df_cart.loc[(df_cart['STEP'] != 0 ) & (df_cart['COUNT'] > list_length)]



cart_time = df_cart_reduced.index.tolist()
time =[0]* len(df_cart_reduced)



for i in range(len(cart_time)):
    time[i] = dftime[cart_time[i]]



df_cart_reduced['TIME'] = time
del df_cart_reduced['SHIFT']
del df_cart_reduced['STEP']

index = list(df_cart_reduced.index.values)
statelist = df_cart_reduced['STATE'].tolist()
countlist = df_cart_reduced['COUNT'].tolist()
updatedstate_cart = [0] * len(S660LAT)

#changes statelist with the updated states. this will apply backwards for unkown areas where
it flips bewtween states
for i in range(len(index)-1):
        for j in range(index[i],index[i+1]):
            updatedstate_cart[j] = df_cart_reduced.STATE.iloc[i+1]
for i in range(index[0]):
    updatedstate_cart[i] = statelist[0]
for i in range(index[-1],len(updatedstate_cart)):
    updatedstate_cart[i] = statelist[-1]
```

```
""" TRUCK STATES"""
#international

international_state = [0]*len(updatedstate)
# speed = 1
# gps_offset = .00015


for j in range(len(international_state)):

    ##moving
    if (avspeedinternational[j] >= speed):
       international_state[j] = "transporting"


    #stationary unload from cart
    elif (avspeedinternational[j] < speed)  and (abs(internationalLAT[j]-Grain_CartLAT[j])
<= gps_offset and abs(internationalLONG[j]-Grain_CartLONG[j]) <= truck_gps_offset):
       international_state[j] = "stationary unload from cart"


    #stationary unload to international
    elif (avspeedGrain_Cart[j] < speed) and (abs(internationalLAT[j]-Grain_CartLAT[j]) >
truck_gps_offset or abs(internationalLONG[j]-Grain_CartLONG[j]) > truck_gps_offset):
       international_state[j]  = "waiting"

    #NA
```

```
            else:
                international_state[j] = "NA"

        df_international = pd.DataFrame(international_state)
        df_international.columns = ['STATE']
        df_international['SHIFT'] = df_international.STATE.shift(periods = -1, fill_value = 1)

        shift = df_international['SHIFT'].to_list()
        step = [0] * len(shift)
        count = [0] * len(shift)
        p = 0

        #sets the step based on the differences in the offset we use this step later to fix the "true state"
also creates the count of in a row counter
        for i  in range(len(shift)):
            if international_state[i] != shift[i]:
                step[i] = international_state[i]

        for i in range(len(shift)):
            if step[i] != 0 :
                count[i] = 0
                p = 0
            elif step[i] == 0 :
                p = p + 1
                count[i] = p


        df_international['STEP'] = step
        df_international['COUNT'] = count
        df_international['COUNT'] = df_international.COUNT.shift(periods = 1, fill_value = 1)
```

```
df_international_reduced = df_international.loc[(df_international['STEP'] != 0 ) & (df_international['COUNT'] > list_length)]


international_time = df_international_reduced.index.tolist()
time =[0]* len(df_international_reduced)


for i in range(len(international_time)):
   time[i] = dftime[international_time[i]]


df_international_reduced['TIME'] = time
del df_international_reduced['SHIFT']
del df_international_reduced['STEP']

index = list(df_international_reduced.index.values)
statelist = df_international_reduced['STATE'].tolist()
countlist = df_international_reduced['COUNT'].tolist()
updatedstate_international = [0] * len(S660LAT)

#changes statelist with the updated states. this will apply backwards for unkown areas where it flips bewtween states
for i in range(len(index)-1):
      for j in range(index[i],index[i+1]):
         updatedstate_international[j] = df_international_reduced.STATE.iloc[i+1]
for i in range(index[0]):
   updatedstate_international[i] = statelist[0]
for i in range(index[-1],len(updatedstate_international)):
   updatedstate_international[i] = statelist[-1]
```

```
# """ IH19"""

# #ih19

# ih19_state = [0]*len(updatedstate)
# # speed = 1
# # gps_offset = .00015


# for j in range(len(ih19_state)):

#         ##moving
#         if (avspeedih19[j] >= speed):
#             ih19_state[j] = "transporting"


#         #stationary unload from cart
#         elif (avspeedih19[j] < speed)  and (abs(ih19LAT[j]-Grain_CartLAT[j]) <= gps_offset
and abs(ih19LONG[j]-Grain_CartLONG[j]) <= gps_offset):
#             ih19_state[j] = "stationary unload from cart"


#         #stationary unload to ih19
#             elif (avspeedGrain_Cart[j] < speed) and (abs(ih19LAT[j]-Grain_CartLAT[j]) >
gps_offset or abs(ih19LONG[j]-Grain_CartLONG[j]) > gps_offset):
```

```
#             ih19_state[j]  = "waiting"

#         #NA
#         else:
#             ih19_state[j] = "NA"

# df_ih19 = pd.DataFrame(ih19_state)
# df_ih19.columns = ['STATE']
# df_ih19['SHIFT'] = df_ih19.STATE.shift(periods = -1, fill_value = 1)

# shift = df_ih19['SHIFT'].to_list()
# step = [0] * len(shift)
# count = [0] * len(shift)
# p = 0

# #sets the step based on the differences in the offset we use this step later to fix the "true state" also creates the count of in a row counter
# for i  in range(len(shift)):
#     if ih19_state[i] != shift[i]:
#         step[i] = ih19_state[i]

# for i in range(len(shift)):
#     if step[i] != 0 :
#         count[i] = 0
#         p = 0
#     elif step[i] == 0 :
#         p = p + 1
#         count[i] = p


# df_ih19['STEP'] = step
```

```
# df_ih19['COUNT'] = count
# df_ih19['COUNT'] = df_ih19.COUNT.shift(periods = 1, fill_value = 1)

# df_ih19_reduced = df_ih19.loc[(df_ih19['STEP'] != 0 ) & (df_ih19['COUNT'] > list_length)]


# ih19_time = df_ih19_reduced.index.tolist()
# time =[0]* len(df_ih19_reduced)


# for i in range(len(ih19_time)):
#     time[i] = dftime[ih19_time[i]]


# df_ih19_reduced['TIME'] = time
# del df_ih19_reduced['SHIFT']
# del df_ih19_reduced['STEP']

# index = list(df_ih19_reduced.index.values)
# statelist = df_ih19_reduced['STATE'].tolist()
# countlist = df_ih19_reduced['COUNT'].tolist()
# updatedstate_ih19 = [0] * len(S660LAT)

# #changes statelist with the updated states. this will apply backwards for unkown areas where it flips bewtween states
# for i in range(len(index)-1):
#         for j in range(index[i],index[i+1]):
#             updatedstate_ih19[j] = df_ih19_reduced.STATE.iloc[i+1]
# for i in range(index[0]):
#     updatedstate_ih19[i] = statelist[0]
```

```
    # for i in range(index[-1],len(updatedstate_ih19)):
    #     updatedstate_ih19[i] = statelist[-1]









    """ freightliner"""

    freightliner_state = [0]*len(updatedstate)
    # speed = 1
    # gps_offset = .00015


    for j in range(len(freightliner_state)):

        ##moving
        if (avspeedfreightliner[j] >= speed):
            freightliner_state[j] = "transporting"


        #stationary unload from cart
        elif (avspeedfreightliner[j] < speed)  and (abs(freightlinerLAT[j]-Grain_CartLAT[j]) <=
truck_gps_offset and abs(freightlinerLONG[j]-Grain_CartLONG[j]) <= truck_gps_offset):
            freightliner_state[j] = "stationary unload from cart"


        #stationary unload to freightliner
```

```
        elif (avspeedGrain_Cart[j] < speed) and (abs(freightlinerLAT[j]-Grain_CartLAT[j]) >
truck_gps_offset or abs(freightlinerLONG[j]-Grain_CartLONG[j]) > truck_gps_offset):
            freightliner_state[j]  = "waiting"

        #NA
        else:
            freightliner_state[j] = "NA"

    df_freightliner = pd.DataFrame(freightliner_state)
    df_freightliner.columns = ['STATE']
    df_freightliner['SHIFT'] = df_freightliner.STATE.shift(periods = -1, fill_value = 1)

    shift = df_freightliner['SHIFT'].to_list()
    step = [0] * len(shift)
    count = [0] * len(shift)
    p = 0

    #sets the step based on the differences in the offset we use this step later to fix the "true state"
also creates the count of in a row counter
    for i  in range(len(shift)):
        if freightliner_state[i] != shift[i]:
            step[i] = freightliner_state[i]

    for i in range(len(shift)):
        if step[i] != 0 :
            count[i] = 0
            p = 0
        elif step[i] == 0 :
            p = p + 1
            count[i] = p
```

```
df_freightliner['STEP'] = step
df_freightliner['COUNT'] = count
df_freightliner['COUNT'] = df_freightliner.COUNT.shift(periods = 1, fill_value = 1)

df_freightliner_reduced = df_freightliner.loc[(df_freightliner['STEP'] != 0 ) & (df_freightliner['COUNT'] > list_length)]


freightliner_time = df_freightliner_reduced.index.tolist()
time =[0]* len(df_freightliner_reduced)


for i in range(len(freightliner_time)):
   time[i] = dftime[freightliner_time[i]]


df_freightliner_reduced['TIME'] = time
del df_freightliner_reduced['SHIFT']
del df_freightliner_reduced['STEP']

index = list(df_freightliner_reduced.index.values)
statelist = df_freightliner_reduced['STATE'].tolist()
countlist = df_freightliner_reduced['COUNT'].tolist()
updatedstate_freightliner = [0] * len(S660LAT)

#changes statelist with the updated states. this will apply backwards for unkown areas where it flips bewtween states
for i in range(len(index)-1):
      for j in range(index[i],index[i+1]):
         updatedstate_freightliner[j] = df_freightliner_reduced.STATE.iloc[i+1]
```

```
    for i in range(index[0]):
        updatedstate_freightliner[i] = statelist[0]
    for i in range(index[-1],len(updatedstate_freightliner)):
        updatedstate_freightliner[i] = statelist[-1]









    """FINAL  LAST TOUCH UPS TO DF AND FINAL PRODUCT //  PLOT CREATION
        ###
        #
        #
        #
        #
        """
#commmented out for speed
    for i in range(len(updatedstate)):
        if updatedstate[i] == 1:
            updatedstate[i] = "harvest"
        elif updatedstate[i] == 2:
            updatedstate[i] = "on the go"
        elif updatedstate[i] == 3:
            updatedstate[i] = "stationary"
        elif updatedstate[i] == 4:
            updatedstate[i] = "idle"
        else:
            updatedstate[i] = updatedstate[i]

    # df_final = pd.DataFrame(updatedstate_S660)
```

```
# df_final.columns = ['Combine State']
# df_final['Cart state'] = updatedstate_cart



# df_final['international state'] = international_state
# df_final['freightliner state'] = freightliner_state
# df_final['ih19 state'] = ih19_state

# df_final['index'] =  list(range(len(df)))
# df_final['time'] = dftime

# df_final['S660 lat'] = S660LAT
# df_final['S660 Long'] = S660LONG

# df_final['Grain_Cart Lat'] = Grain_CartLAT
# df_final['Grain_Cart Long'] = Grain_CartLONG

# df_final['international Lat'] = internationalLAT
# df_final['international Long'] = internationalLONG

# df_final['ih19 Lat'] = ih19LAT
# df_final['ih19 Long'] = ih19LONG

# df_final['freightliner Lat'] = freightlinerLAT
# df_final['freightliner Long'] = freightlinerLONG
# df_final['S660 speed'] = avspeedS660
# df_final['Grain_Cart speed'] = avspeedGrain_Cart



# df_S660_reduced['TIME'] =    df_S660_reduced['TIME'].shift(periods = 1)
```

```
    # df_cart_reduced['TIME'] =    df_cart_reduced['TIME'].shift(periods = 1)
    # df_international_reduced['TIME'] =    df_international_reduced['TIME'].shift(periods = 1)
    # df_ih19_reduced['TIME'] =    df_ih19_reduced['TIME'].shift(periods = 1)
    # df_freightliner_reduced['TIME'] =    df_freightliner_reduced['TIME'].shift(periods = 1)


    # df_reduced = pd.concat([df_S660_reduced, df_cart_reduced, df_international_reduced, df_ih19_reduced, df_freightliner_reduced ], axis = 1)
    # df_reduced.columns = ['S660 state', 'S660count','time1', 'Grain_Cart state', 'Grain_Cart count','time2' , 'international state', 'international count','time3',
    #                       'ih19 state', 'ih19 count', 'time4', 'freightliner state', 'freightliner count','time5']
    # df_reduced['time'] = df_reduced.time1.combine_first(df_reduced.time2)
    # df_reduced['time'] = df_reduced.time.combine_first(df_reduced.time3)
    # df_reduced['time'] = df_reduced.time.combine_first(df_reduced.time4)
    # df_reduced['time'] = df_reduced.time.combine_first(df_reduced.time5)
    # del df_reduced['time1']
    # del df_reduced['time2']
    # del df_reduced['time3']
    # del df_reduced['time4']
    # del df_reduced['time5']
    # df_reduced[['S660 state','Grain_Cart state','international state', 'freightliner state' , 'ih19 state']] = df_reduced[['S660 state','Grain_Cart state','international state', 'freightliner state' , 'ih19 state']].fillna(method = "bfill")

    df['UPDATEDSTATE'] = updatedstate
    df['CARTSTATE'] = cart_state
    # df['index'] = list(df2.index.values)

    # df_final.to_csv(r'C:/Users/loghe/Desktop/unloads/STATES/updated' + date2 + " 10_tic " + str(zz) +'.csv' )
```

```
        # harvest = ['harvest'] * 34428

        S660_same = sum(x == y for x, y in zip(perfect_S660,updatedstate))
        Grain_Cart_same = sum(x == y for x, y in zip(perfect_grain_cart ,updatedstate_cart))
        return(S660_same,Grain_Cart_same, updatedstate , updatedstate_cart, df)



record =[]

gps_list = [.00014,.00015,.00016,.00017,.00018,.00019]
speed_list =[.125,.25,.5,.75,1,1.25,1.5]
closing_speed = [ .125,.25,.5]
a = 1
y = 1
for i in speed_list: #cut off speed
    for j in gps_list: # gps set
        for k in range(0,6): # list length
            for l in closing_speed: # list length
                for m in gps_list: #truck offset


                    output = script(i,j,k,l,m)
                    record.append('S660 = ' + str(output[0]) + ' Grain_Cart = ' + str(output[1]) +  ' sum =
'
                            + str(output[0] + output[1]) + " values " + str(i) + ", " + str(j) + ", " + str(k) +
"," + str(l) + ", " + str(m))
                    y = y + 1
                    print(y)
                    if ((output[0] + output[1]) > a):
                        a = output[0] + output[1]
                        b = (i,j,k,l,m)
```

```
print(a,b)
c = script(b[0],b[1],b[2],b[3],b[4])
df = c[4]
df['state machine S660'] = c[2]
df['state machine grain cart'] = c[3]
df.to_csv(r'C:/Users/loghe/Desktop/ACRE unload/state machine output/'+ date2 + ' state machined.csv')
```

LoRa End Node Code

```
#include <SPI.h>
#include <RH_RF95.h>
#include <SoftwareSerial.h>
#include <TinyGPS.h>
TinyGPS gps;
SoftwareSerial ss(3, 4);
#include <LoRa.h>


int count = 0;
int device_id = 10009; // ID of this End node

void setup() {
  Serial.begin(9600);
  ss.begin(9600);
  //while (!Serial);

  Serial.println("LoRa Sender");

  if (!LoRa.begin(915E6)) {
```

```
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  LoRa.setSyncWord(0x34);
  LoRa.setTxPower(20);
  LoRa.setSpreadingFactor(10);
}

void loop() {
    bool newData = false;
    for (unsigned long start = millis(); millis() - start < 1000;)
  {
    while (ss.available())
    {
      char c = ss.read();
      // Serial.write(c); // uncomment this line if you want to see the GPS data flowing
      if (gps.encode(c)) // Did a new valid sentence come in?
        newData = true;
    }
  }
  float flat, flon;
  gps.f_get_position(&flat, &flon);
  long lat = flat * 1000000;
  long lng = flon * 1000000;
  Serial.print("Sending packet: ");
  Serial.println(count);
  // compose and send packet
  LoRa.beginPacket();
  LoRa.print("<");
  LoRa.print(device_id);
  LoRa.print(">");
```

```
  LoRa.print(lat);
  Serial.print(flat , 6);
 // LoRa.print("&field2=");
  LoRa.print(lng);
  Serial.print(flon , 6);
  // LoRa.print(counter);
  LoRa.endPacket();
  count++;
  delay(1000);
}
```