GPS-Guided Autonomous Robot with Obstacle Avoidance and Path Optimization

by

Ryan Moeller

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in the Department of Mechanical Engineering

Idaho State University

Spring 2020

Table of Contents

GPS-Guided Autonomous Robot with Obstacle Avoidance and Path Optimization

Thesis Abstract – Idaho State University (2020)

This thesis discusses the design of an autonomous agricultural robot, including its mechanical components, autonomous navigation system, and waypoint path planning algorithm. This autonomous robot is a prototype and a test platform that can navigate to potato plants previously identified as infected with Potato Virus Y (PVY). This prototype is the first step leading up to a larger, full scale, field-ready robot that will be equipped with an advanced robotic arm for plant removal. This prototype is in-line with emerging technologies in the field of precision agriculture. The navigation system is based on a Pixhawk™ microcontroller and a Real-Time Kinematic (RTK) GPS module. This thesis also covers in detail the mechanical and electrical subsystems of the prototype. Due to potato field size and PVY prevalence, an adjustment to popular optimization algorithms is explored to find the shortest route between each identified sick plant in a center-pivot-irrigated field.

Keywords

Navigation, Obstacle Avoidance, Agricultural Robot, Autonomous Ground Vehicle, RTK GPS, Pixhawk, Traveling Salesman Problem, Precision Agriculture

1 Introduction

1.1 Agricultural Autonomous Ground Vehicle (AGV)

Potato Virus Y (PVY) can devastate crops with losses up to 80%, depending on the specific type of potato [1]. While there are many methods currently being used to generate PVY resistant cultivars, the virus has quickly evolved in the last ten years and continues to have a negative impact on farmers [2]. Potato seed crops are rigorously tested to avoid planting already-infected plants, but PVY identification efforts are not perfect. Even when potato seeds are certified PVY free, testing for susceptibility to PVY vectors is not normally performed adequately [3]. Mechanical removal of infected plants remains a full-proof, albeit currently inefficient, method to prevent the spread of PVY once it is already present in a field.

Current processes for mechanical removal consist of trained field crews identifying the infected plants and then manually removing them, a process known as crop roguing. The major drawback to this process is that identifying PVY correctly by sight is difficult. An experienced roguer may be accurate but is slow and the plants must be mature [4]. Because of this, remote sensing techniques are often employed [5]. In this study, we have partnered with the Geosciences department at Idaho State University (ISU), where researchers have been successful using hyperspectral cameras that can sense ranges of the electromagnetic spectrum that human eyes cannot [6]. In hyperspectral remote sensing, the electromagnetic spectrum is split into hundreds of narrow “bands” of light, making hyperspectral imaging able to detect infections in plants even before physical symptoms are manifested [7][8]. Hyperspectral cameras can be mounted on Unmanned Air Vehicles (UAV), making remote sensing a powerful technique at a leaf scale resolution. This identification strategy combined with Autonomous Ground Vehicles (AGV)

allows entire crops of potatoes to be scanned and remedial measures to be implemented quickly. See Figure 1 for how these two types of vehicles could work in tandem to achieve this.

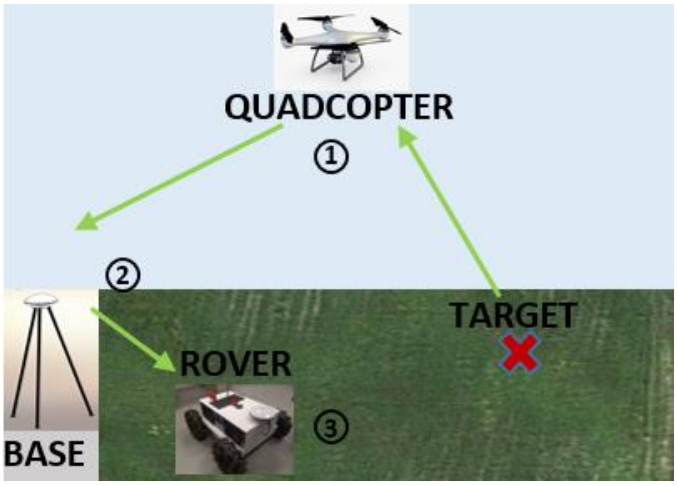

Fig 1. UAV and AGV working in tandem. This figure illustrates the acquisition and delivery of a sick plant's GPS coordinates, labeled "target", using a UAV with hyperspectral imaging capability. The UAV then sends the coordinate to an AGV that can remove the plant.

This UAV-AGV system would streamline both the identification effort and the physical removal. This thesis/research deals with designing a Global Positioning System (GPS) guided AGV that would serve as a prototype for a final platform that would carry a robotic arm to each identified plant and mechanically remove it.

Robots have been increasingly researched for agricultural applications, whether it be for fertilization, pesticide application, or inspection [9]. Robots can be superior to large machinery as they can be more selective where they spray, conserving resources and reducing crop inputs that may have an environmental impact on soil and water. In addition, in some applications it is more cost efficient to send a robot than a manual operator [10].

In recent years, GPS has been a popular solution for autonomous robot navigation [11]. Being that the environment at hand is agricultural, the GPS antenna will benefit from having

minimal obstructions blocking the sky such as buildings and trees. Because the bulk of the growing season for potatoes is during summer months, clouds and storms that can cause GPS systems to have problems will be relatively minimal.

The GPS network is just one of many in the Global Navigation Satellite System (GNSS). GNSS receivers work by measuring the time needed for a signal to reach the receiver from a satellite. Transferred signals traversing through the atmosphere may be slowed down or interfered with, causing error. The average error of commercially implemented GPS is 2-4 meters. In the field, the distance between two potato plants is around 0.3 m (12 inches). A 2-meter resolution would fall well short of the resolution required for an individual plant in the field. Therefore, to improve the accuracy of the navigation system for this AGV, a Real-Time Kinematic (RTK) GNSS system with an error less than 10 centimeters (3.93 inches) was utilized. An RTK system contains two main units: a base station unit and a rover unit. Both receive signals from satellites, but the base station unit is stationary while the rover unit is mounted on the AGV. The base station transmits position information in real-time to the rover, which helps the rover make corrections to its location, thereby achieving higher accuracy.

1.2 Optimizing Path Distance Considering Irrigation Obstacle

In Idaho, the leading producer of potatoes in the USA [12], many potato fields span across a circular area of 0.5 km² (124 acres). For reference, 0.5 km² is equal to 78 Fédération Internationale de Football Association (FIFA) regulation soccer fields. In a potato field of this size, there may be hundreds of sick plants spread across it. Choosing the shortest path possible to visit each sick plant is necessary to conserve battery power and therefore cost, and will also reduce wear on mechanical parts. To find the shortest route between sick potato plants, it is proposed to treat sick potato plants as cities in a Traveling Salesman Problem (TSP). The TSP

deals with finding the shortest possible route between all cities in a given set of cities, visiting each one only once and returning to the starting city. The TSP has been thoroughly studied over the last century and many algorithms can effectively find near-optimum solutions [13][14].

To add complexity, these fields are irrigated with a center pivot system rotating about the field's center. As seen in Figure 2, these irrigation lines can block robot movement underneath them due to low ground clearance. In addition, immediately after the irrigation line passes over a section of ground, that ground becomes impassable to any machine due to mud. Many research articles discuss obstacle avoidance but all deal with small obstacles that take up only a small percentage of the robot's working area [15][16]. A center-pivot irrigation line is the length of the crop circle's radius. For a half a square kilometer field, the radius is 400 meters. Navigating around this irrigation/mud obstacle would cause an optimization algorithm's proposed solution to become extremely inefficient.

Fig 2. Example of a low clearance irrigation line. Irrigation lines of these types are common in potato fields. Source: https://wikifarmer.com/potato-water-requirements-and-irrigation-systems/

The complexity increases even further when considering that AGV movement across rows will cause some plant damage. The AGV could be restricted to only moving along a row but this would be very slow and inefficient. Instead, it is proposed that the AGV be used in the early growing season when plant size is minimal but hyperspectral remote sensing techniques are still effective. The beginning of June is optimal – this gives the plants enough time to grow leaves for the hyperspectral cameras but leaves the rows less crowded. Even as early as July, rows become impassable due to heavy vine growth, see Figure 2. By avoiding the vine growth, the AGV will do less damage. It is then of interest to use an algorithm not only to minimize distance while avoiding the irrigation/mud obstacle, but to also minimize the number of times the AGV crosses rows in the field.

A solution is proposed to address the irrigation/mud obstacle and row crossing problem. The proposed solution is to adjust a normal TSP's cost function to reflect a new path around the irrigation line and let the optimization routine converge to the optimum route based on this new information. In addition, another cost function will be added to account for number of rows that need to be crossed. To address this novel problem, this thesis presents how to use the popular Genetic Algorithm (GA) and Multiple Integer Linear Programming (MILP) optimization routines [17][18].

2 Agricultural AGV

2.1 Methodology

2.1.1 Components

A four-wheel drive system was chosen for the robot, and to increase durability a bearing was mounted to the frame of the robot for the motor shaft to run through. This increased the total payload the robot could carry. SuperDroid Robots™ was the source of most robot hardware (see Table 1). A robot chassis was designed to fit these mechanical components and manufactured at ISU labs.

TABLE 1. LIST OF MECHANICAL COMPONENTS

Item	Description	Quantity	USD/item
IG42 24VDC 078 RPM Gear Motor	24 Volt motor with planetary gears	4	$52.95
Wheel and Shaft Set Pair 8mm bore - 10-inch Traction Lug	Wheel and shaft with connecting hardware	2	$72.58
Electric Power Hookup Kit	Wires, fuses, and switches	1	$17.95
Misc. mounting hardware	Motor mount plate, fastener kit, electrical hookup kit	4	n/a

Calculations were done to ensure the motors would accomplish the desired propulsion over rough terrain. The rated speed of the selected motor/gearbox is 78 RPM. For the 10-inch diameter wheels used that is equivalent to 3.7 linear km/hour (2.3 mph). While this is slow for a wheeled robot, the higher gear ratio provides more torque, a significant advantage over rough terrain, so this rated speed was deemed sufficient. The rated torque of the motor was 0.05814 N·m (570 g·cm). With a 1:84 gearbox ratio that is equal to 4.88 N·m. To determine if this was sufficient, equation 1 was derived from a simple free body diagram. It solves for t, the time it takes to reach angular velocity V from a stopped position.

$$t = V\left(\frac{I_{wheel} \cdot I_{motor}}{T_{motor} - Wrsin(\theta)}\right) \quad (1)$$

The moment of inertia of the motor was assumed to be 0. The moment of inertia of the wheel was calculated to be $\frac{1}{2}mr^2 = 0.018\ kg \cdot m^2$. V was set to the rated speed of 5.18 rad/s (3.7 km/hr) and r was the radius of the wheel. Lastly, W was ¼ the total weight of the robot. For θ = 0, t = 0.02 seconds. See Figure 3 for the results when the equation is solved for all values of θ (0° to 90°). From Figure 3, an asymptote occurs at about 43.5°, indicating that is the maximum theoretical incline possible. It is unlikely this prototype will face terrain for testing that includes inclines greater than 43.5°.

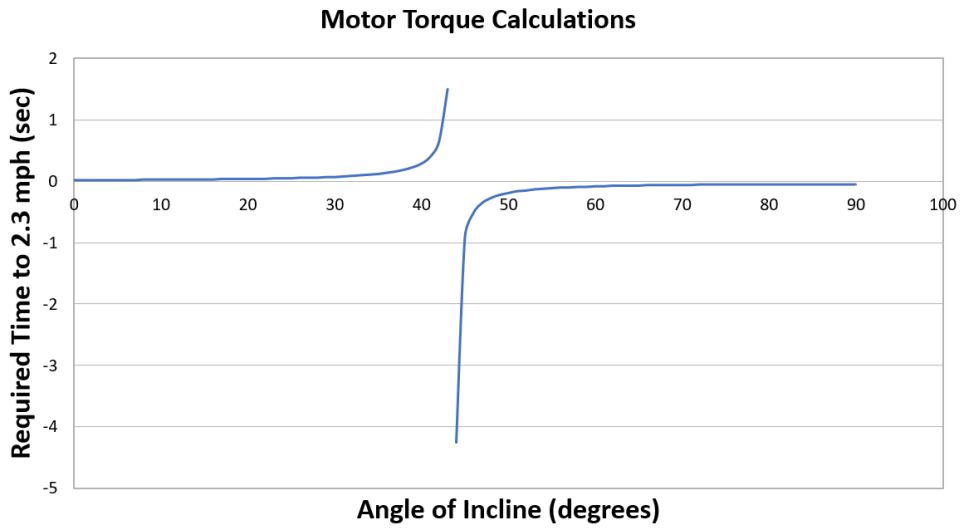

Fig 3. Graph of motor torque calculation showing an asymptote at 43.5°, the theoretical limit of the selected motor.

2.1.2 Enclosure and Bottom Plate

It was important to design an enclosure that protected the electrical components from mud, water, and the elements given its operating environment. Because this robot was a prototype, easy access to the components was also essential. It was designed first in SolidWorks™ and then acrylic sheets were laser cut and glued with Weldon #4 acrylic glue (Dichloromethane). Acrylic was used because it can be easily laser cut, allowing the enclosure to

fit snugly onto the robot thereby increasing protection from the elements. Acrylic is also readily available at hardware stores if repairs were needed.

The enclosure's joints were reinforced with high temperature hot glue. When the acrylic sheets were cut out, holes were included in optimum positions for cables, antennas, etc. Once assembled, these holes were tested and then the enclosure was spray painted white to lower heat transfer from the sun. See Figure 4 for the final result.

Fig 4. Ensembled and painted enclosure, with electrical components mounted and placed on top of robot chassis.

The enclosure is held down by four bolts, and can be lifted off with external electrical components still attached. The components are connected/disconnected through the easy access holes in the enclosure.

In addition to an enclosure, a bottom acrylic plate was laser cut for mounting the electrical components that did not transmit signals and were not waterproof. This could be removed easily and independently from the enclosure for debugging. For calibrating the

magnetometers in the equipment, this bottom plate can stay bolted to the enclosure but lifted off the chassis.

2.2 Results

TABLE 2. ROBOT SPECIFICATIONS

Specification	**Description**
Weight	25 kg (55 lbs.)
Size	43 x 54 x 46 cm (17 x 21 x 18 in)
Cost (Parts/Machining)	$ 855.00 (USD)
Theoretical Speed	3.7 km/hr (2.3 mph)

The final specifications of the robot can be seen in Table 2. Because this prototype robot is not full size, it was not driven in an actual potato field. Potato fields have roughly 30 centimeter (1 foot) deep ruts left by the irrigation equipment. A large vehicle with wheels the same diameter or larger than the ruts would be required to drive in the field.

2.3 Conclusion

This robot can handle inclines with ease and will not be slowed down by lack of pavement or flat areas. It is an effective test bed for systems that will be used on a larger, full scale robot. For the final AGV platform, the wheel diameter would need to be at least 30 cm (1 ft) and the size of the chassis would be need to be determined by other future work such as the robotic arm.

3 Autonomous Navigation

3.1 Methodology

For this robot, an RTK GPS system was combined with magnetometers to provide the robot with its position and heading. A Pixhawk was used for the microprocessor, which calculates the error between current position/heading and desired position/heading and then adjusts wheel speed to correct itself. The Pixhawk also contains onboard magnetometers. The Piksi Multi Evaluation Kit from Swift Navigation was chosen for the RTK GPS. The Piksi Multi has powerful Xilinx Zynq 7020 dual-core ARM Cortex processors and a 666 MHz clock speed. This allows it to perform RTK calculations quickly, obtaining an RTK fix in seconds. It can use several different satellite networks for RTK measurements, including GPS L1/L2, GLONASS G1/G2, BeiDou B1/B2, and Galileo E1/E5. It is also relatively inexpensive as compared to other RTK GPS solutions. The base station and rover (AGV) units each contain four major parts, an evaluation board, lithium ion battery, Free Wave™ radio modem, and a survey grade GNSS antenna. See Figure 5 for wiring. Table 3 has a comparison of similar systems and their most recent prices.

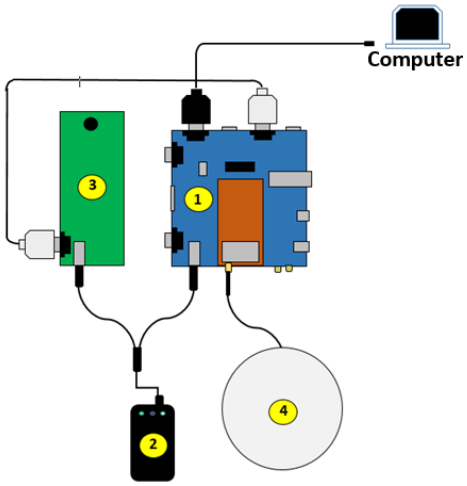

Fig 5. Piksi Multi Evaluation Kit Main Parts – Both the base station and rover contain these four parts. Item 1: Evaluation Board. Item 2: lithium ion battery. Item 3: , Free Wave™ radio modem. Item 4: survey grade GNSS antenna

TABLE 3. COMPARISON OF RTK GPS SYSTEMS

Name	**Cost**
Piksi Multi Evaluation Kit	$ 2,295.00
Trimble R8s	$ 10,200.00
Spectra Precision SP80	$ 12,495.00
Geomax Zenith35 Pro	$ 9,995.00

3.1.1 Evaluation of RTK GPS Components

Before being used on the actual AGV, the components were tested using the radios that Swift Navigation provided. To set up the Piksi Multi in RTK mode, the base station and rover radios had to be programmed to communicate with each other. Then, all components in Figure 5 were wired to each other as shown. The survey grade GNSS antenna and Free Wave radio modem were connected to the evaluation board. The radio modem and evaluation board were powered by a lithium ion battery. Finally, the evaluation board was connected with a USB cable to a computer. Swift Navigation provides a free software called Swift Console™ to configure and interface with the Piksi Multi. 900 MHz or 2.4 GHz radios can be used during evaluation of the GNSS receivers. The evaluation was successful and the next step was integrating the Piksi Multi with the Pixhawk and a Ground Control Station.

3.1.2 Integration with Pixhawk and a Ground Control Station

The radios provided by Swift Navigation were only used during evaluation. The telemetry radios for the Pixhawk were used on the AGV so that RTK corrections could be sent over the same radio link that the Ground Control Station (GCS) shared with the Pixhawk. The GCS had to retrieve the corrections from the Swift Console, a process outlined in Swift Navigation's documentation. For a brief summary of this process see Figure 6.

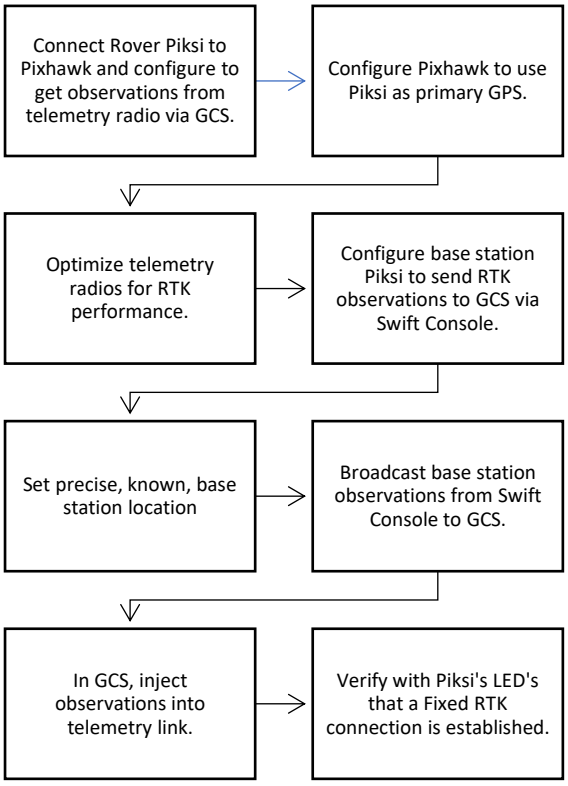

Fig 6. Flowchart of Piksi & GCS integration. The evaluation kit should be used before hand to ensure that all components are working correctly.

Once the observations are sent to the Pixhawk, the Pixhawk can theoretically navigate autonomously with centimeter level accuracy. Without an RTK GPS fix, the Pixhawk performs to the accuracy level of a normal GPS unit (2-4 meters).

The Pixhawk can be set to multiple modes. For this project only three were used: “manual”, “smart RTL”, and “auto.” Manual mode allows an RC transmitter to control the AGV with user input. More on how that was set up is covered in section 3.1.3. When in auto mode, navigation is done by the firmware loaded onto the Pixhawk. In smart RTL mode the robot will trace its path back to its starting location.

Waypoints are chosen in the GCS, usually installed on a laptop, and sent to the Pixhawk via telemetry radio link. When auto mode is turned on, the Pixhawk begins navigating from its current position to the first waypoint in a straight line. Once it gets within a programmable radius from the waypoint, it moves on to the next one. When finished, it returns to its “home” location which is programmed with the GCS.

In Figure 7, the process for setting waypoints is overview. “H” is for home location. Each waypoint has a number next to it to determine the order of navigation. Notice, there is no waypoint number 2 on the map but there is one in the list. This is because a waypoint can be changed from a simple navigational goal, to a trigger event via a drop-down menu. In this case the event named DO_GRIPPER was chosen just as an example. The event triggered could be a camera shutter, grasping mechanism, etc. The event is only triggered when the AGV is within the programmed radius of the trigger’s associated GPS coordinate. In this case DO_GRIPPER lies halfway between waypoints 1 and 3.

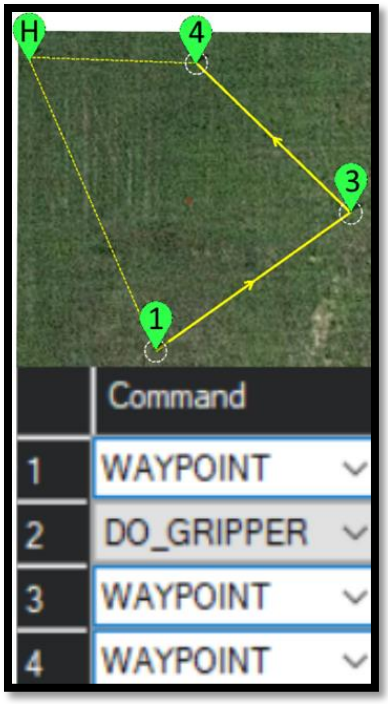

Fig 7. Screenshots of the chosen GCS's navigation planning feature.

3.1.3 Electrical Components

The electrical components all support the Pixhawk in its central role of autonomous navigation. The Pixhawk was paired with a GCS called Ardupilot Mission Planner™ (or Mission Planner, for short) for affordable, customizable, and trusted performance. Along with the Pixhawk a suite of electrical and mechanical hardware was used to have manual as well as autonomous navigation (see Table 4 and Figure 8). Although autonomous navigation is the goal, being able to control the robot manually provides convenience for positioning the robot while testing and also acts as a safety mechanism in case the autonomous navigation malfunctions and the robot deviates from its course or poses a danger to bystanders.

TABLE 4. LIST OF ELECTRICAL CONNECTIONS

#	**Component Name/Model**	**Pixhawk Port Used**	**Other Connections**
1	3S Lithium Polymer (LiPo) battery	Power	Piksi Multi Power
2	Pixhawk	n/a	n/a
3	mRobotics™ SiK Telemetry Radio V2 915Mhz	Telem 1	n/a
4	X4RSB FrSky receiver	Servo rail: ground, power, and signal of SBUS pin	n/a
5	WASP-200 LRF	Telem 2	n/a
6	Sabertooth™ 2x12 Motor Controller	Servo rail: Signal and ground of PWM output pins 1 and 3	Batteries to B- and B+.
7	(x2) 12 Volt 8 Amp hour Sealed Lead Acid Battery	n/a	Motor Controller
8	Buzzer and Safety Switch	Buzzer and Switch	
9	LiPo battery voltage alarm	n/a	3S LiPo battery
10	ReadytoSky™ GPS/Compass	GPS and I2C	
11	Piksi Multi from Swift Navigation	Serial 4/5 port	Battery (via voltage regulator), GNSS survey grade antenna
12	Survey grade GNSS antenna	n/a	Piksi Multi
13	FrSky Taranis Q X7 transmitter (not pictured)	n/a	Radio linked to X4RSB receiver

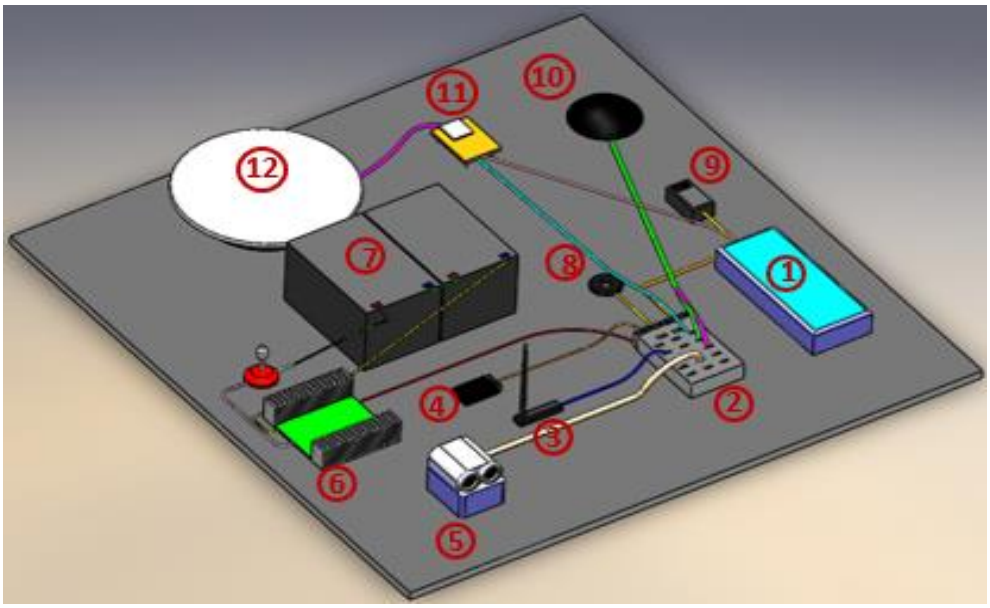

Fig 8. Electrical connections displayed in a 3D CAD model. All components except for the transmitter are shown here.

A LiPo battery was chosen because of its high current output and compatibility with Pixhawk's provided power module. Everything on the AGV, apart from the motors, was powered with this battery.

An onboard voltage alarm is an affordable and easy way to ensure that the LiPo battery does not go below its allowable voltage, preventing damage. The alarm can be programmed to trigger at different voltages, alerting the user when a return to base is needed.

The ReadytoSky GPS/Compass was used as a backup GPS so that the AGV could find its way home if the Piksi failed. It uses a NEO-8N GPS module from u-blox™, and has an advertised maximum accuracy of 0.6 meters (24 inches).

Communication between the Pixhawk and Mission Planner was done through the SiK Telemetry Radios. These radios are designed specifically for Pixhawk/Mission Planner and have a range of 300 m.

A X4RSB FrSky receiver, and FrSky Taranis Q X7 transmitter were used to send commands from the user to the robot. The X4RSB receiver has SBUS capabilities which was necessary to send multiple channels over a single wire that connects into the Pixhawk RCIN pin. The Q X7 is an inexpensive but a highly capable transmitter. It is reliable and has 16 channels, two control sticks, and several switches all programmable in Mission Planner to perform different tasks. One of these switches was programmed to switch the Pixhawk between manual, auto, or Smart RTL mode. The channels for the two sticks on the transmitter were Aileron, Elevator, Throttle, and Rudder. Each of these corresponds to one of the two sticks in either vertical or horizontal movement. In the mixing menu, they were put in an AETR channel configuration. That and the switch used can be seen in Figure 9.

Fig 9. Configuration for channel mixing on the Taranis transmitter. The switch used channel 6.

In Mission Planner the parameters for the Pixhawk were also configured accordingly, as seen in Figure 10.

Fig 10. Mission Planner configuration. Roll = Aieleron. Pitch = Elevator. Yaw = Rudder.

Only roll and throttle were used by the Pixhawk and sent to the motor controller. Roll controlled the tank style steering (left, right) and throttle controlled the speed of the wheels.

The Sabertooth motor controller is robust, and is able to supply two separate channels of 12 continuous amps each, has overcurrent and overheat shut down prevention, several input modes, synchronous regenerative technology (batteries are charged when going in reverse), and has a 5 V, 1A power output for peripheral devices if needed.

For the LIDAR, a WASP 200-LRF was chosen because it is compatible with Pixhawk. It has a range of 0.2-300 meters and an accuracy of < 10 cm. Most importantly it uses a DF13 connector which is what makes it compatible with the Pixhawk.

With the exception of the Piksi multi, all these components required very little programming. Parameters in the Pixhawk had to be programmed to function in the necessary way, but otherwise these components were a plug and play solution. This ease of use is a major reason the Pixhawk was chosen.

3.1.4 Obstacle Avoidance

Besides irrigation equipment and the potato plants themselves, there are not many vertically protruding obstacles in a potato field. Possible obstacles include sudden breaks in normal terrain features, rocks, trees, tractors, humans and animals. Obstacles such as mud and ruts can be partially handled by the design of the vehicle's wheels and motor torque (see section 2.2), and the path planning algorithm. After being watered the field is extremely muddy, and not even full-sized tractor equipment enters these areas. Chapter 4 will touch on how to avoid these areas. For the smaller obstacles that can be navigated around quickly, a simple range finding LIDAR will suffice.

The firmware on board the Pixhawk can handle obstacle avoidance when a range finding LIDAR is connected. Figure 11 shows the parameters that configure the obstacle avoidance protocol known as "dodge". Simply put, when the LIDAR detects an object that is too close, as defined by $RNGFND\_TRIGGER\_CM$, it stops, rotates $RNGFND\_TURN\_ANGL$ degrees, and drives straight for $RNGFND\_TURN\_TIME$ seconds and then resumes a direct course for the next waypoint.

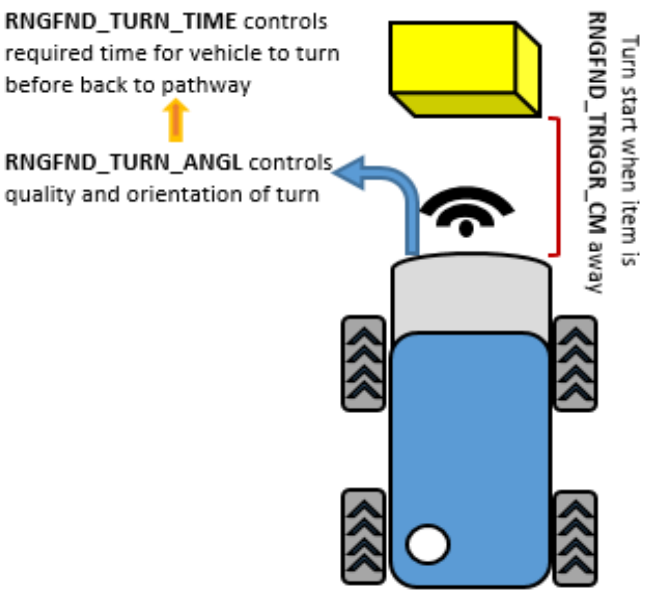

Fig 11. Graphic explaining the parameters involved in setting up "dodge" obstacle avoidance on the Pixhawk using a range finding LIDAR.

3.2 Results

3.2.1 Piksi Multi Accuracy

The RTK GPS was true to its advertised accuracy (less than or equal to ten centimeters). Utilizing a National Geodetic Marker site, the Piksi Multi was accurate to 5.6 centimeters using our setup.

There was significant difficulty finding a location with a GPS coordinate known accurately enough to verify the Piksi's coordinates. A couple methods were considered. The easiest but most expensive is to survey a spot on the ground. This requires special equipment and training, but can be done practically anywhere outside. The more inconvenient method is to find a geodetic marker, a previously surveyed position marked with a permanent landmark. The online National Geodetic Survey Data Explorer gives access to all such publicly recorded markers. However, many are located on privately owned land and are difficult to access. Figure 12 is an example of a marker.

A marker was found at the Pocatello Regional Airport. According to the marker's data sheet, it has an 95% confidence interval accuracy of 1.29 cm. The Permanent Identifier (PID) was AA3682 and the datasheet can be found at the National Geodetic Survey's website [19]. Using the evaluation boards from Swift Navigation, the base station Piksi was placed on this geodetic marker and the rover Piksi on another geodetic marker several hundred meters away (This marker's accuracy was 1.41 cm, PID: AB8163). Fifty RTK GPS fixes (including latitude, longitude, and ellipsoidal height) were collected and converted into the UTM 12N projected coordinate system (for ease of calculation). Pythagorean's theorem was then used to find the difference between the rover's average estimated GPS location and the coordinates listed on the geodetic marker's datasheet. Because the calculated distance was relatively short and the terrain was flat this is an acceptable assumption [22]. The average error was 5.6 centimeters (see Table 5). The airport could only accommodate testing for a short period of time and so only this test was performed.

TABLE 5. PIKSI ACCURACY TESTING

Accuracy (cm)	**Geodetic Marker Location**	**PID**	**Geodetic Marker Accuracy (cm)**
5.6	42° 54' 47.03909" N, 112° 35' 26.36463" W	AB8163	1.41

Fig 12. A photograph of a geodetic marker, taken by the author.

3.2.2 Autonomous Navigation

The performance of the autonomous navigation was satisfactory. The AGV visited the waypoints as commanded in Mission Planner, and all three modes functioned satisfactorily. At times the AGV had problems finding its heading, and would spin around until finding it again. This was a minor issue, though, and higher quality magnetometers or more careful calibration may fix this.

There does exist a limitation related to the waypoint radius parameter described earlier. During testing it was discovered that the lowest value that can be stored in the parameter is 1 meter, so as soon as the AGV is within 1 meter of the next waypoint it would move on to the next waypoint. This radius is not small enough to navigate the AGV next to the sick potato plant. This could possibly be solved many ways. First, because mission planner is open source, code could be written to change the resolution to centimeters instead of meters. Second, a trigger event could be added so that a third-party computer handled the precise navigating within 1 meter of the infected potato plant.

3.2.3 Small Obstacle Avoidance

The WASP 200-LRF LIDAR listed in Table 4 was also tested. In this test, the SiK telemetry radios were bypassed with a usb cable and an object was placed at a measured distance away from the LIDAR. The LIDAR readout in Mission Planner was the same as the measured distance, with a couple centimeters of error on average. This amount of error for dodge avoidance is acceptable and is within the 10-centimeter accuracy described in the manufacturer's specifications.

Obstacle avoidance was not successfully tested with the telemetry connection due to undiscovered issues. Future testing will need to be conducted to troubleshoot the WASP when a telemetry connection is being used.

3.3 Conclusion

The AGV designed and built in this project has an accurate RTK GPS, and combined with the Pixhawk is a suitable testing platform for an autonomous agricultural robot. The robot and components were relatively inexpensive, but robust enough to handle typical terrains the robot will be operational in. Since it uses the Pixhawk it is extremely customizable as well. Mission Planner is a capable GCS and allows for easy waypoint-guided navigation. These features make it useful not just for this project but for many agricultural robot applications. Because it is open source, a solution to its limited waypoint radius should be obtainable whether by programming or triggering a third-party computer. Obstacle avoidance when using telemetry connection will need to be troubleshooted. It is recommended that communication with the manufacturer be established to receive input on the problem.

4 Path Optimization

4.1 Methodology

To find the shortest path between a given number of sick potato plants in a field, first the problem needs to be simplified. Only fields that form an entire circle are considered, as some smaller fields can be semi-circles. In addition, it is assumed there is one center-pivot irrigation line rotating about the crop circle's center. Some irrigation lines can extend up to a public road, as shown in Figure 13. To avoid cars, fences, etc. the option for the robot to go around the outside of the irrigation line is removed. This removes risk of collision and also has the benefit of making it simple to apply the TSP and solve it accordingly.

Fig 13. Screenshot from Google Earth ™ of a potato field in Idaho. An irrigation line is clearly visible extending to (and even watering) a road.

The problem statement is as follows: given n sick potato plants, what is the shortest path to visit each plant once and return to the starting position? A path that visits each plant, or point, exactly once and returns to its starting point is called a Hamiltonian circuit. As the number of sick plants, n, becomes large the number of Hamiltonian circuits increases with $\frac{(n-1)!}{2}$. For $n = 20$, there are 6.08×10^{16} possible Hamiltonian circuits. Nevertheless, the aforementioned

algorithms, GA [20] and MILP [21], can effectively handle the TSP even with $n > 200$. These algorithms use a cost function to find the fitness of a solution. When the set of n points are given in UTM coordinates (or any x-y coordinate system) the cost function is simply the distance formula [22]. See Equation 2, where E represents Easting and N represents Northing, analogous to longitude and latitude, respectively.

$$d = \sqrt{(E_2 - E_1)^2 + (N_2 - N_1)^2} \tag{2}$$

This cost function will only work for paths between plants that are not obstructed by the irrigation line. For pairs of points, whose direct line of sight is obstructed by the irrigation line, the shortest path between the two must intersect with the middle of the crop circle to go around said obstacle (see Figure 14). If the cost is increased for this pair of points it will be less likely to be chosen by the algorithm as part of the final path. In other words, the distance formula isn't changed, rather it is used twice (on obstructed paths only) to find this new distance.

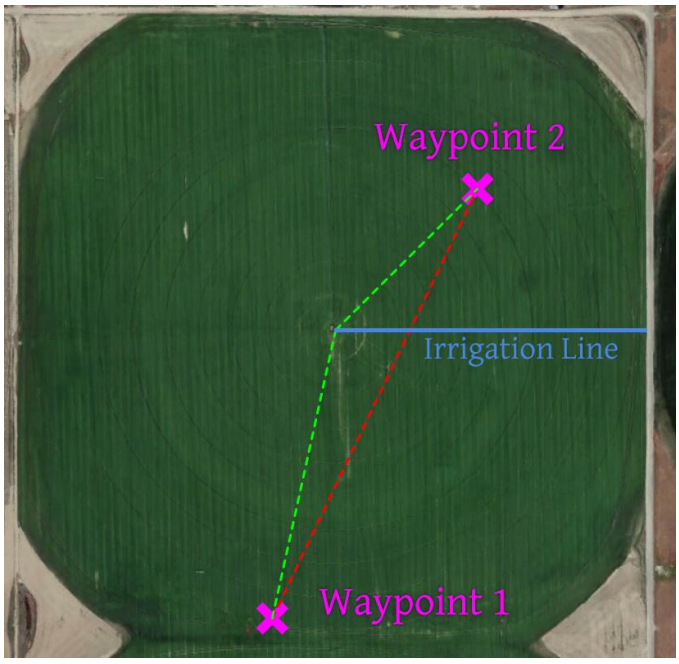

Fig 14. Image showing an example of two waypoints, blocked by an irrigation line (red path), and the next shortest path between them (green).

In order to change the cost associated with pairs of points whose line of sight is obstructed by the irrigation line, first each pair of points under this condition needs to be found. See Table 6.

TABLE 6. LISTING ALL POSSIBLE QUADRANTS FOR TWO WAYPOINTS

Point 1's Quadrant	**Point 2's Quadrant**	**Obstructed**
1	1	NEVER
1	2	NEVER
1	3	SOMETIMES
1	4	ALWAYS
2	2	NEVER
2	3	NEVER
2	4	SOMETIMES
3	3	NEVER
3	4	NEVER
4	4	NEVER

To accomplish this, the field was split into four quadrants. The irrigation line was positioned in the 0° (or 3 o'clock) position. A rotation matrix was applied to the points when the irrigation line was at any other angle. From Table 6 the only case which needs inspection is for points who lie in quadrants I and III, or II and IV. In these two cases, depending on where in each quadrant the points are, the direct path between them may or may not be obstructed. A mathematical test was employed to determine if the points were obstructed or not (see Figure 15).

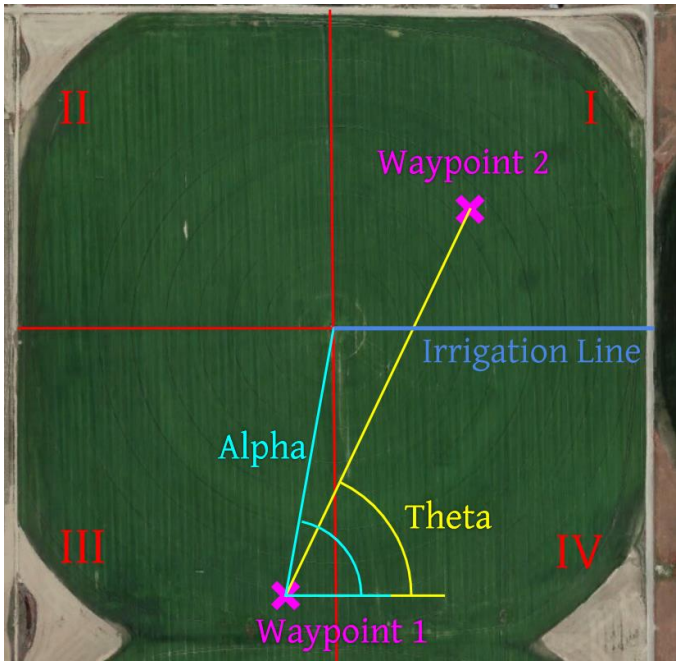

Fig 15. Image illustrating the angles and quadrants used in the mathematical test employed to discover if a pair of waypoints need to be rerouted or not.

This test uses two imaginary lines. One is a direct line in between the two points. The angle between this line and waypoint 1's horizontal is called $Theta$. The second line is drawn between point 1 and the center of the circle. The angle between this line and point 1's horizontal is called $Alpha$. If $Alpha < Theta$, the points have a direct path available between them. If $Alpha > Theta$, then the points cannot be connected by a direct line and the cost of this specific path needs to be increased to reflect the distance between point 1, the center of the circle, and point 2. This same test can be adapted for the case where the points are in quadrants II and IV, respectively. As with the Piksi accuracy testing, the distance formula is used with x and y components and the height (z) is ignored. Agricultural fields are flat and distances are relatively short compared to the curvature of the earth and so this is a good assumption in order to keep calculations simple.

The cost function takes the form of an $n\ x\ n$ matrix. The value of each element in the matrix corresponds to a distance between two points. For example, the element in the 4th row and 8th column is the distance between the fourth and eighth waypoint . The diagonal is all zeros. MatLab™ code was written to use the $Alpha/Theta$ test to identify which elements in this matrix needed to change and then those elements were updated with the new distance. Care was taken to avoid for loops and use instructions that act on vectors instead. This was to ensure the above mathematical test did not add a significant amount of run-time to the algorithm.

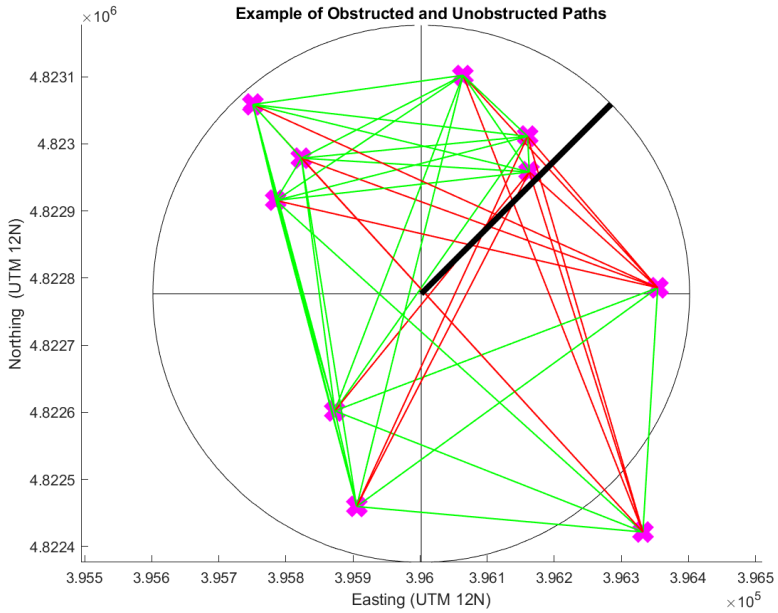

Fig 16. Illustrated in red are the paths whose distances in the cost matrix needed to be increased. The thick black line represents the irrigation line. In this instance, because the irrigation line was not at 0 degrees, a rotation matrix had to be used. Notice some obstructed points will have much larger rerouted paths than other pairs of points.

All of the paths that needed to be routed (red paths in figure 16) were updated in the cost matrix before sending the list of points and their cost matrix to the GA and MILP algorithms. Then a solution of the optimal order was found and could be sent to the Pixhawk for navigating. Sending the optimal order of waypoints from MatLab to Mission Planner was not automated. All

that is necessary is to export from MatLab to a .txt file and import from the Mission Planner interface.

Up to this point it has been assumed that the irrigation line is not actively watering and therefore not moving. If the AGV was used while the irrigation line was watering then a simple solution was explored:

1. Find the shortest distance as if the irrigation line wasn't watering and use the robot's known linear speed to calculate total trip time
2. Use the irrigation line's known angular velocity to calculate how large of a segment will be watered during the amount of time calculated in step 1.
3. Temporarily eliminate any waypoints in this section
4. Have a technician or farm worker identify the point *behind* the irrigation line's starting point at which the muddy terrain becomes passable. Use this point to identify another section in which to eliminate all points
5. Rerun the algorithm for the remaining points.
6. Visit these points, and when done rerun the algorithm for the eliminated sections. Once the robot is done with the first waypoints it can visit the newly watered ones after they have dried sufficiently.

Step 4 includes manual labor and is inefficient. However, with time, a farmer could keep record of how quickly each field dries based on how much water it receives and can come up with accurate tables for this without the need of a technician in the field.

In order to add a cost function for minimizing row crossing, it was assumed that rows ran vertically, or North/South. The row width was set to 75 cm (2.5 ft). The distance formula was

again used to form an $n\ x\ n$ matrix as described before, but only the x-component of distance was considered. When the x-component distance was divided by the row width the number of rows necessary to cross was obtained. This matrix, called $rowsCost$ was added to the original matrix, called $distanceCost$. This total cost matrix was then used in the GA. The MILP algorithm was not used as it uses a cost matrix in the same way the GA does and would not provide insight into the proposed method that the GA could not.

4.2 Results and Discussion

The proposed adjustment to the cost matrix was effective for both algorithms. The MILP consistently found shorter routes but had an inconsistent runtime (higher standard deviation) when compared to the GA. The specific GA used required a deeper understanding of how it functioned in order to avoid irregularities. This is referring to the fact that sometimes the GA found shorter routes with the adjusted cost function than with the original cost function. When combining cost functions for distance and row crossings, the GA algorithm was, again, successful.

The proposed method of finding which routes were blocked by the irrigation line did not add a significant amount of calculation time to the algorithms (see Table 7). Test D did add a significant amount of time but 10,000 instances of PVY is unlikely.

TABLE 7. CALCULATION TIME TO FIND REROUTED PATHS

Test	Number of points	Avg Time (seconds)	Standard Deviation
A	100	1.2417e-04	6.8009e-04
B	500	0.0028	0.0154
C	1000	0.0103	0.0566
D	10000	24.3	3.4

Before examining the effectiveness of the proposed change to the distance cost function, testing was done to find near-optimal parameters for the GA. Test A, B, and C all consisted of running the GA 30 times, each using a randomly generated set of points. The average distance and time were calculated. The number of points used was 150 for all tests. For test A, the population size started at 152 and iterations was 10,000. The new distance cost matrix was used. As both parameters increased, route distances decreased and run time increased. See Table 8.

TABLE 8. DIFFERENT PARAMETERS' EFFECTS ON GA RUN TIME/AVERAGE DISTANCE

Test	AVG Distance (m)	Time (s)	Pop. Size	Iterations
Test A	7557	8	152	10,000
Test B	7298	16	152	20,000
Test C	7219	77	752	20,000

Using the parameters from Test C, the GA and MILP were both compared in terms of how well they chose paths that navigated around the irrigation line. Both resulted in paths that navigated around the irrigation line effectively. In cases where the number of waypoints was greater than 50, the algorithms often didn't select any paths which corresponded to ones whose distance had been altered in the distance cost function. See Figure 17.

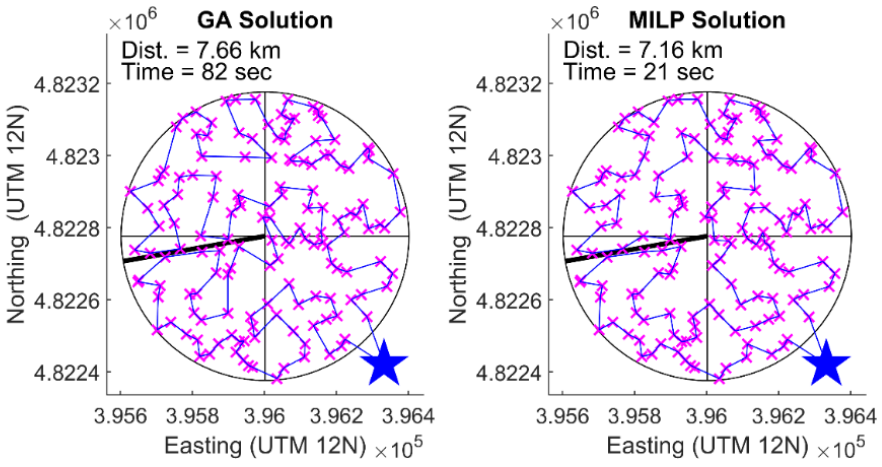

Fig 17. While calculation time and path distance were different, both optimizations converged on solutions that avoided rerouted paths.

Comparing the algorithms with and without the adjusted distance matrix for a smaller number of points more readily shows the effectiveness of said technique. See Figure 18.

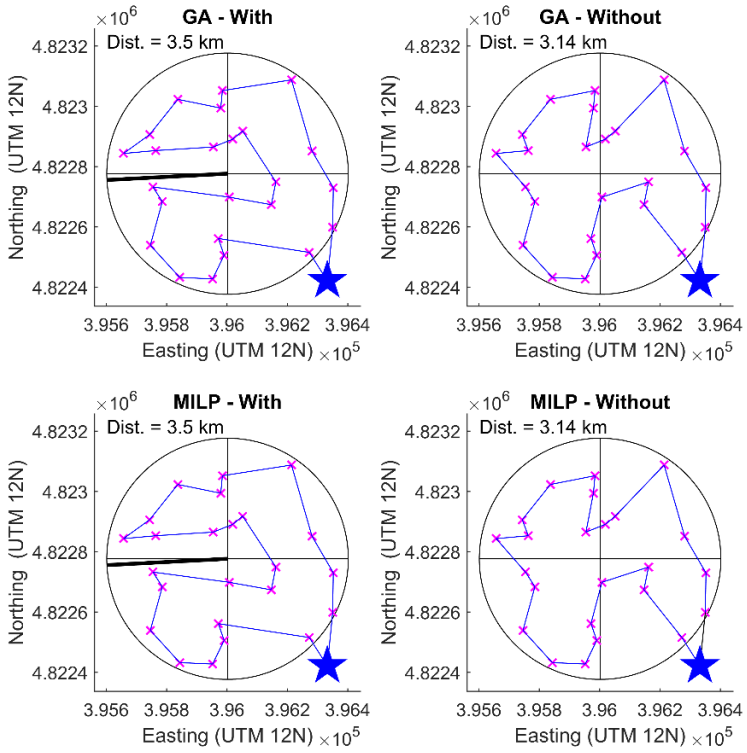

Fig 18. The top row is the GA algorithm with and without (left to right) the adjusted distance matrix to account for the irrigation line. The bottom row shows the same, but for the MILP algorithm. Notice that for so few points the GA and MILP converged to the same solutions.

For a moving irrigation line, the proposed set of steps also performed well, see Figure 19.

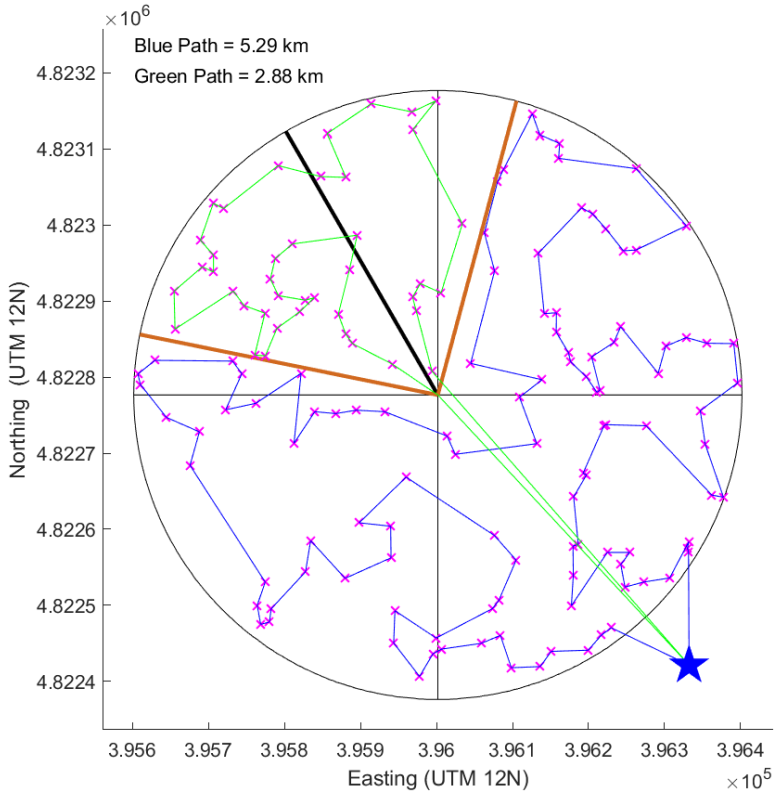

Fig 19. The black line is the starting position of the irrigation line, and it is moving counter clockwise. The area in between the brown lines is where the field would be too muddy for the AGV to operate. The brown line behind the irrigation line's movement was arbitrarily set to be 45° behind the irrigation line's original position. The blue line is the route first path taken and the green line is the second path taken.

The next set of simulations, outlined in Table 9, compared the GA to the MILP in terms of distance and calculation time. 30 test runs were conducted using 150 points. The same set of points was used for the GA and MILP for each test. For the GA the population size was 752 with 20,000 iterations. A t-test for paired means was performed and the p-value for both optimum distance and time the algorithm needed to run was found. The MILP found 4% shorter routes, an average distance of 318 meters. With regards to the standard deviations in run time, the MILP had much more variation.

TABLE 9. COMPARING GA TO MILP

	Distance (m)		Time (s)		P-value	P-value
	AVG	**STD**	**AVG**	**STD**	**Dist.**	**Time**
GA	7219	171	77	0	0	0.253
MILP	6901	136	65	54		

Throughout each test an interesting phenomenon was noticed: the GA algorithm sometimes found a shorter route when the irrigation line was considered compared to when the irrigation line was not considered. This goes against intuition - an obstacle of this size would seemingly result in a longer route, not a shorter one. Why would the GA be unreliable in this regard? The GA requires optimization of its input parameters such as population size, iterations, mutation rate, and crossover strategy [23]. Depending on how future generations are chosen, the GA can have different characteristics. The truncation method, used in Joseph Kirk's GA and for the testing in this paper, can lead to premature convergence [24]. Therefore, to converge to an optimum route, there must be a certain number of original population members that start out with characteristics of the optimum solution. To test this (see Table 10), 3 sets of 30 tests each was done. Test A showed a high number of instances where the irrigation-line-solution was shorter. For test B the number of points was decreased to 25. The iterations were decreased only to help with computation time and did not affect convergence. With a decreased number of points there is a decreased number of mutations necessary to exhibit the required variation to find the optimum route, and so all irrigation-line-solutions were longer than none-irrigation-line solutions. However, in test C, the number of points was kept the same as in Test B but the population size was reduced. Even with the same low number of points, there were 3 tests where the irrigation-line-solution resulted in a shorter route. With so few population members, i.e. such little possibility for variation, changing the cost matrix happened to introduce beneficial variation 3 out of 30 times.

TABLE 10. TESTING GA PARAMETERS ON VARIABILITY AND EFFECTIVENESS

	Number of irrigation line shorter solutions	**# Points**	**Iterations**	**Population size**
Test A	10	150	20000	752
Test B	0	25	1000	752
Test C	3	25	1000	25

Using the average distance found by these algorithms, the total time to visit every plant is easily estimated. The robot's speed is 3.7 km/hr (2.3 mph). For an average TSP solution of 6.9 km, that is equal to $time = \frac{6.9\ km}{3.7\frac{km}{hr}}$, or 1.9 hours. This time will be longer when the irrigation line is watering because of the time it would take to travel from the last waypoint in the first, dry, section, to the first waypoint of the recently watered section. It also doesn't account for how long the robot will remain at each plant. Work on the robotic arm would need to be finalized before this is known accurately, but it could be estimated at 30 seconds (enough time to completely remove the plant and all its roots) For 150 points the total time becomes $1.9\ hrs + 150\ plants * 0.00833 \frac{\text{hrs}}{\text{plant}} = 3.15\ hrs$, assuming the field wasn't actively being irrigated. Seeing this number calls attention to the fact that the AGV needs to either have a large battery capacity or some method to quickly recharge.

Finally, the described method for minimizing row crossings was tested (see Figure 20). The GA was run with just the original distance cost function, and then with just the rows cost function in order to see a baseline of how each cost function affects the algorithm. Then, the two cost functions were added together. Using the rows cost function only was undesirable because the algorithm would pick points which were closest to each other in the x-direction but were sometimes on opposite sides of the field from each other in the y-direction. When distance and rows crossed were weighted equally, the distance was much less than when the rows cost

function was used alone, but there were still fewer rows crossed than without the rows cost function. If it was desired to cross even fewer rows, the rows cost function could be weighted more than the distance cost function as seen in the bottom-right plot of Figure 20.

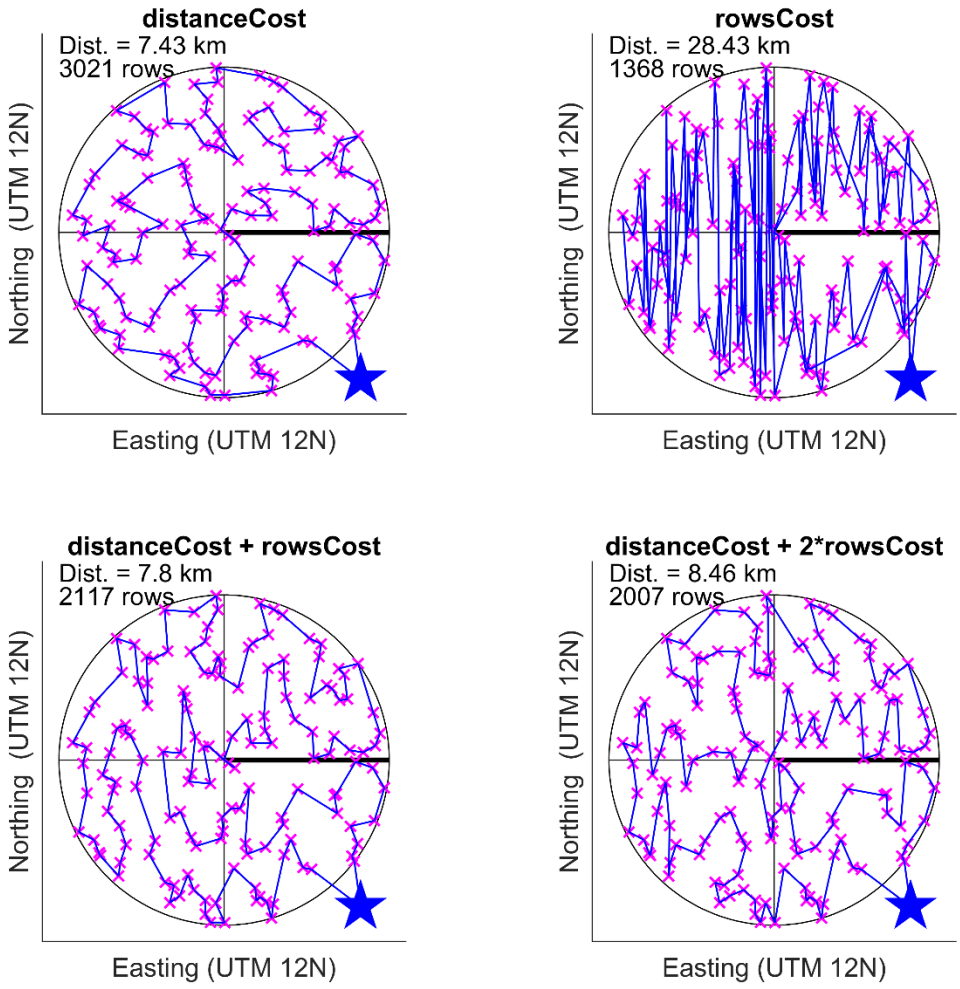

Fig 20. For the testing shown in this figure, the parameters were 150 points, population size of 752, and 20,000 iterations. Top left: Using just the distance cost function resulted in the lowest distance but highest number of rows crossed. Top right: Using just the rows cost function, the number of rows crossed was minimized but the overall distance was too long to be practical. Bottom left: combining both cost functions produced a mix of the first two results. Bottom right: if one parameter is desired over the other (rows crossed vs overall distance), they can be weighted as in this plot, with $rowsCost$ being multiplied by 2.

4.3 Conclusion

Both GA and MILP algorithms only needed a small adjustment to the distance cost function to successfully account for the irrigation line. This simple change was effective and efficient and will be suitable for any algorithm that uses a cost matrix. Between the GA and MILP algorithms, the MILP would be the wisest choice for the robot because it found shorter routes with roughly similar calculation times. If calculation speed was the priority, the GA could be used with fewer iterations/population size to reduce calculation time but still obtain a solution within a few hundred meters of the MILP average solution.

A new cost function was introduced successfully to also minimize the number of rows crossed. Future work could include optimizing motor size and battery life of the final robot. This would be good work because it would be undesirable for the AGV to have to return to its starting point to recharge in the middle of a run. Perhaps multiple charging stations around the field could be explored, as well as solar panels mounted on the AGV.

5 Future Work

The AGV presented in this thesis, its navigation system, and path planning process performed its intended functions. As a prototype it is ready for further testing of features that will eventually be incorporated into the final platform. However, there were some shortcomings, one of them being the LIDAR functionality. Future work is needed to get the LIDAR to function properly when the Pixhawk is connected via radio to Mission Planner, not just when connected via cable. Another point of emphasis is the inability of the Pixhawk to get closer than 1 meter to a waypoint. This particular issue seems easy to solve: a trigger event could be used to turn on a third-party computer to handle navigation when within 1 meter of the waypoint. However, this begs the question that if a third-party computer must be used for this portion of the path, why not just use this computer for the entire navigation? A significantly large portion of background research would need to be done if the Pixhawk was to be replaced entirely. Alternatively, the open source code of the PixHawk could potentially be altered in order to reflect higher resolution than 1 meter.

Another large topic of future work is optimizing robot speed and battery capacity for the final AGV platform. Research would need to be done to identify how many sick plants is expected, on average, in a typical potato field. This number will affect robot runtime which determines what size battery is needed. The AGV's speed capability also effects the runtime: the faster the AGV the less time the batteries are running. However, more speed means more power consumption and higher required capacity. These constrains would need to be optimized in order to minimize the number of times the AGV needs to recharge, or potentially optimized so that it should never need recharging until an entire field is finished. Other considerations for battery

charging could include researching the cost effectiveness of solar panels and/or multiple charging stations around a field.

Besides battery and motor parameters, there are a few other features that need to be researched in order to produce the final AGV. These features include the necessary robot arm technology, vision systems, and communication with a hyperspectral-equipped UAV. A robot arm needs to be developed that is capable of removing the entire plant without leaving any part of it in the field. Otherwise, remnants of removed plants would infect other plants around it. A vision system to identify where on the plant the robot arm needs to grab also needs to be developed. Also, a way to autonomously communicate between the UAV and the AGV needs to be explored.

Lastly, this UAV-AGV system relies on RTK GPS, which in turn relies on precise, surveyed, locations existing in the field. An affordable way of providing these surveyed positions to farmers needs to be proposed.

Despite the essential work that lies ahead, this thesis provides a strong, necessary foundation for future research. The aforementioned features were outside the scope of this Thesis.

Works Cited

[1] *Descriptions of Plant Viruses.* Kew(Surrey) : Commonwealth Agricultural Bureaux and the Association of Applied Biologists, 1970.

[2] Nolte, Phillip, et al. "Effect of Seedborne Potato Virus Y on Performance of Russet Burbank, Russet Norkotah, and Shepody Potato." *Plant Disease*, vol. 88, no. 3, 2004, pp. 248–252., doi:10.1094/pdis.2004.88.3.248.–

[3] Karasev, Alexander V., and Stewart M. Gray. "Continuous and Emerging Challenges of Potato Virus Y in Potato." *Annual Review of Phytopathology*, vol. 51, no. 1, Apr. 2013, pp. 571–586., doi:10.1146/annurev-phyto-082712-102332.

[4] Polder, Gerrit, et al. "Potato Virus Y Detection in Seed Potatoes Using Deep Learning on Hyperspectral Images." *Frontiers in Plant Science*, vol. 10, Jan. 2019, doi:10.3389/fpls.2019.00209.

[5] Krezhova, Dora, et al. "The Effect Of Plant Diseases On Hyperspectral Leaf Reflectance And Biophysical Parameters." *RAD Conference Proceedings*, 2017, doi:10.21175/radproc.2017.55.

[6] Griffel, L.m., et al. "Using Support Vector Machines Classification to Differentiate Spectral Signatures of Potato Plants Infected with Potato Virus Y." *Computers and Electronics in Agriculture*, vol. 153, 2018, pp. 318–324., doi:10.1016/j.compag.2018.08.027.

[7] Krezhova, Dora, et al. "Hyperspectral Remote Sensing Applications for Monitoring and Stress Detection in Cultural Plants: Viral Infections in Tobacco Plants." *Remote Sensing for Agriculture, Ecosystems, and Hydrology XIV*, 2012, doi:10.1117/12.974722.

[8] Ni, Zhuoya, et al. "Early Water Stress Detection Using Leaf-Level Measurements of Chlorophyll Fluorescence and Temperature Data." *Remote Sensing*, vol. 7, no. 3, 2015, pp. 3232–3249., doi:10.3390/rs70303232.

[9] Bengochea-Guevara, José, et al. "Merge Fuzzy Visual Servoing and GPS-Based Planning to Obtain a Proper Navigation Behavior for a Small Crop-Inspection Robot." *Sensors*, vol. 16, no. 3, 2016, p. 276., doi:10.3390/s16030276.

[10] Pedersen, S. M., et al. "Agricultural Robots—System Analysis and Economic Feasibility." *Precision Agriculture*, vol. 7, no. 4, 2006, pp. 295–308., doi:10.1007/s11119-006-9014-9.

[11] Gao, Xinyu, et al. "Review of Wheeled Mobile Robots' Navigation Problems and Application Prospects in Agriculture." *IEEE Access*, vol. 6, 2018, pp. 49248–49268., doi:10.1109/access.2018.2868848.

[12] Omondi, Sharon. “The Top 10 Potato Producing States In The US.” WorldAtlas, *WorldAtlas*, 1 Oct. 2019, www.worldatlas.com/articles/the-top-10-potato-producing-states-in-the-us.html.

[13] Scholz, Jan. “Genetic Algorithms And The Traveling Salesman Problem A Historical Review.” 2018, doi:10.13140/RG.2.2.22632.78088/1.

[14] Htun, Thi Thi. “A Survey Review on Solving Algorithms for Travelling Salesman Problem (TSP).” *International Journal of Scientific and Research Publications (IJSRP)*, vol. 8, no. 12, 2018, doi:10.29322/ijsrp.8.12.2018.p8481.

[15] Liu, Changqing, et al. “Research on Static Path Planning Method of Small Obstacles for Automatic Navigation of Agricultural Machinery.” *IFAC-PapersOnLine*, vol. 51, no. 17, 2018, pp. 673–677., doi:10.1016/j.ifacol.2018.08.119.

[16] Pingzeng, Liu, et al. “Obstacle Avoidance System for Agricultural Robots Based on Multi-Sensor Information Fusion.” *Proceedings of 2011 International Conference on Computer Science and Network Technology*, 2011, doi:10.1109/iccsnt.2011.6182170.

[17] Kirk, Joseph. “Traveling Salesman Problem - Genetic Algorithm - File Exchange - MATLAB Central.” *MatLab Central File Exchange*, 6 May 2014, www.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm.

[18] “Traveling Salesman Problem: Solver-Based.” *MathWorks*, www.mathworks.com/help/optim/ug/travelling-salesman-problem.html.

[19] US Department of Commerce, and NOAA. “National Spatial Reference System, Geodetic Control Map.” *NGS Data Explorer*, National Geodetic Survey, www.ngs.noaa.gov/NGSDataExplorer/.

[20] Raman, Vikas, and Nasib Singh Gill. “Review of Different Heuristic Algorithms for Solving Travelling Salesman Problem.” *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, 2017, pp. 423–425.

[21] Cook, William. In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation. Princeton University Press, 2014.

[22] Langley, R B. “The UTM Grid System.” *GPS World*, Feb. 1998, pp. 46–50.

[23] Zhang, Yu-An, et al. “Effects of Population Size and Mutation Rate on Results of Genetic Algorithm.” *2008 Fourth International Conference on Natural Computation*, 2008, doi:10.1109/icnc.2008.345.

[24] Prasad, G. “Genetic Algorithm Performance Assessment by Varying Population Size and Mutation Rate in Case of String Reconstruction.” *Journal of Basic and Applied Engineering Research*, vol. 4, no. 2, 2017, pp. 157–161.