Autonomous Monitoring of Organic Crops

Final Report

Wholesome Robotics (Team E):

Aman Agarwal
Hillel Hochsztein
Dung Han Lee
John Macdonald
Aaditya Saraiya

Sponsor: Rivendale Farms
Mentor: George Kantor
Date: 13th December 2019

1 Abstract

This project focuses on developing a robotics solution for autonomous monitoring for organic farming at Rivendale Farms, Bulger PA. A four-wheel skid steered robot platform called the Robotanist [1] was utilized for this project. For autonomous navigation, two distinct approaches were utilized which include LIDAR row-perception and RTK GPS based fusion methods to achieve autonomous row-following and row-switching functionalities in outdoor environments. A CNN based plant health detection based on the UNet [2] architecture was developed to detect the presence of pests and diseases in a variety of plants from the Brassica family. The model achieves the required 80% precision and recall for holes and fungus classification in a select variety of plants. An easy-to-use, interactive GUI has been developed to present geo-tagged image data associated with the classification. The salient features of the GUI include visualization support for multiple rows of data, color-coded graphics to show pest and disease levels, note-taking and allow farmers to change plant health detection results. We present results from various field trials performed at Rivendale farms to validate the robot's functionalities.

Contents

2 Project Description

Organic vegetable farming introduces several challenges for farmers in growing high-quality crops with high crop yield. Most notably, without artificial pesticides and herbicides, it is extremely challenging to control the invasion of pests and weeds into the crop beds. Pests may directly harm the crops by eating them or spreading diseases. The longer an infestation goes unnoticed, the more difficult it becomes to control. Therefore, it is of the utmost importance for organic farmers to monitor their fields for disease and pest pressures in order to prevent more widespread damage.

Robotic monitoring of organic vegetable farms poses a potential solution. Mitigation of diseases and pests is very specific to each threat, and a robot that could handle every threat it encounters would be prohibitively complex. However, a robot that can automatically survey the field and quantify disease and pest pressure, and deliver reports on disease and pest pressure to farmers so that they may respond in a timely and informed manner would deliver significant value to farmers.

3 Use Case

3.1 Narrative

On a sunny day, the user moves the robot to the field, selects the monitoring mode and places the robot at the start of the first row. The robot autonomously collects images from the field by autonomously driving through the rows, being careful to not crush plants while traversing through a row. After collecting data, the robot returns the starting point, from which the user moves the robot back to the barn and connects it to the docking station. The robot processes the data and provides insights to the user about pest and disease pressures in the form of heat maps, location, and trends in pests.

3.2 Graphical Representation

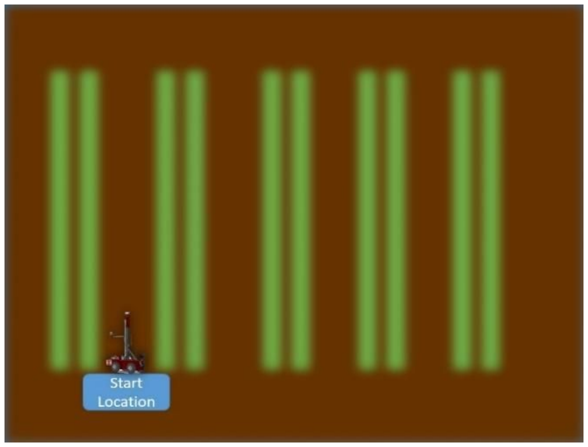


Figure 1 On a sunny day, the user carries the robot to the field, selects the monitoring mode and places it at the start of the first row

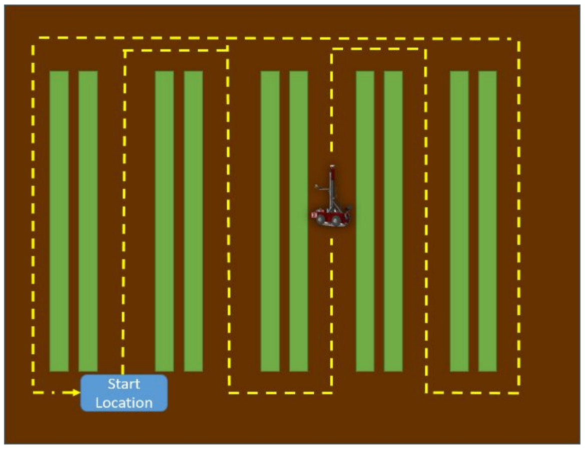


Figure 2 The robot autonomously collects images from the field by navigating through the field and returns back to the starting point

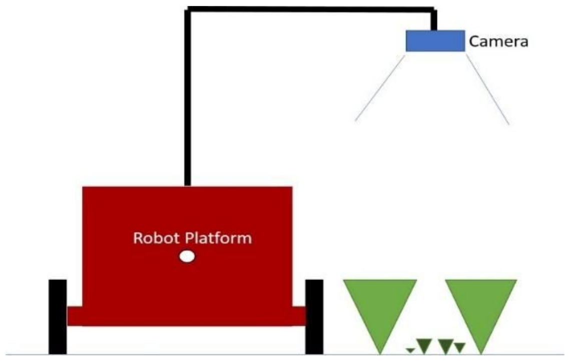


Figure 3 The robot collects visual and location data while moving through the field.

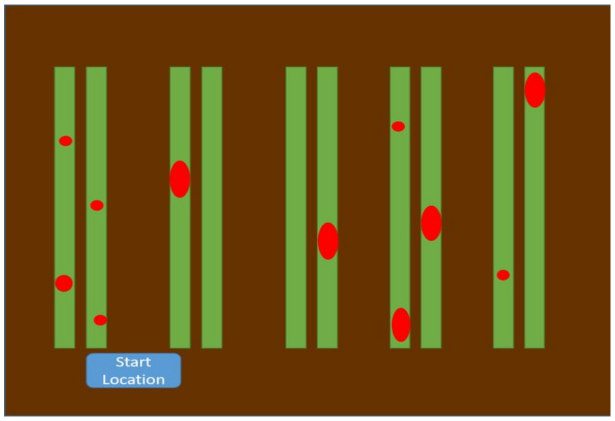


Figure 4 The user then carries the robot back to the barn and connects it to the docking station

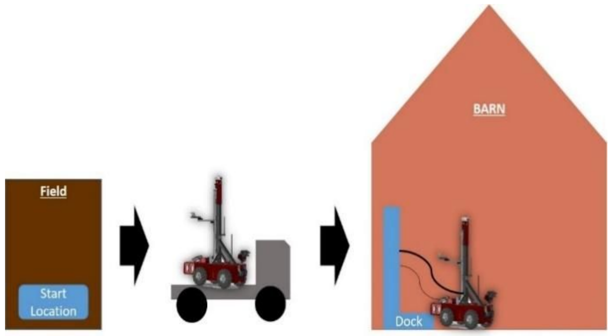


Figure 5 The user carries the robot from the starting point back to the barn and connects it to the docking station

4 System Level Requirements

We have updated our requirements in two notable ways. First, we have descoped weeding, so all requirements associated with weeding have been updated or removed accordingly. We also added requirements for the report generation, since we added an interactive GUI to visualize the results of field monitoring.

4.1 Performance Requirements

The Robot shall:

Table 1 Performance Requirements

Requirement	Subsystem
MR1. Autonomously localize itself	Navigation
MR1.1. In the correct row with 80% accuracy	
MR1.2. Cross track control error < 3 in within the row	
MR2. Autonomously switch between rows of the field with 80% success rate	
MR3. Collect visual data with 75% usable images	Monitoring
MR4. Identify signs of disease on plant with precision and recall > 80%	
MR5. Identify pests and /or signs of pests with precision and recall > 80%	

MR6. Generate meaningful reports within 24hrs of collection	Visualization
MR6.1. Label row with plant name	
MR6.2. Label image with severity level	
MR6.3. Allow user to see and change the severity level	

Note: MR denotes a Mandatory Performance Requirement.

4.2 Non-Functional Requirements

The robot will:

Table 2 Non-Functional Requirements

Requirement	Subsystem
MN1. Fit in the row of width 24in	Platform
MN2. Accommodate various control modes via kill switch and joystick	User Interface
MN3. Have sufficient battery life to complete a run of Rivendale brassica field	Platform
MN4. Not damage plant during navigation	Navigation

Note: MN connotes a Mandatory Requirement, DN connotes a Desirable Requirement.

5 Functional Architecture

The functional architecture has been simplified to only a single monitoring mode. In this operation mode, the robot localizes itself in the field and then uses the location and a pre-built map to navigate throughout the field. The robot collects images which it processes later to identify signs of pests and disease. In the end, reports are generated to communicate to the user information about crop health.

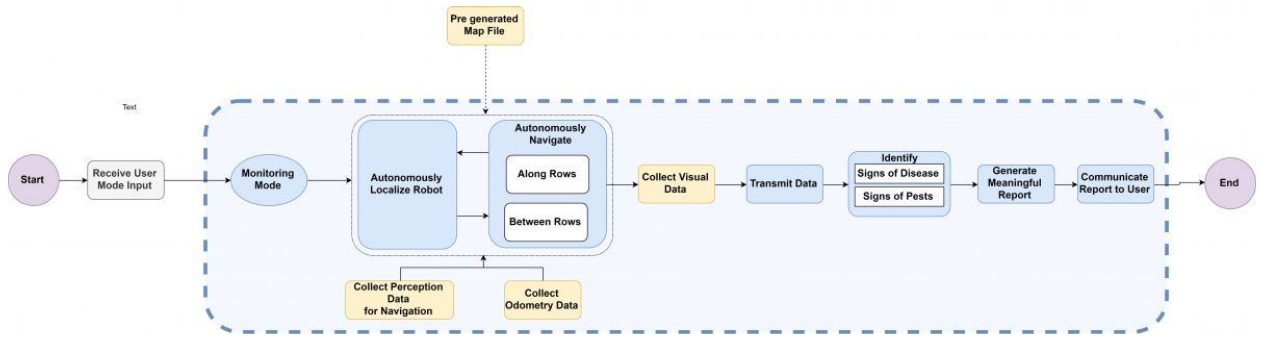


Figure 6 Functional Architecture

6 System Level Trade-Studies

6.1 Robot Platform Trade Study

The robot platform consists of the hardware required for mobility of the system including hardware and software. The platform will be used to mount the sensors and the manipulator and will traverse the field. The Robot Platform trade study aims to find an optimal platform for the system which will adhere to all relevant requirements of the system.

Table 3 Robot Platform Trade Study [1] [2] [3] [4] (Parameters rated out of 5)

Parameter	Weight %	Jackal	Husky	4WD Rover	Flipper Rover	Robotanist
Width of Robot	25	4	2	5	5	3
Speed of Robot	10	4	2	5	3	4
Lateral Stability	20	2	4	1	1	5
Payload Capacity	25	2	5	3	3	4
Run Time	5	5	2	3	3	5
Wheel base	15	5	2	4	4	1
Weighted Sum	100	3.3	2.95	2.93	2.81	**3.44**

The criteria considered were the width of robot (to ensure the robot can fit in row 24" wide), lateral stability (to ensure that the robot does not topple while moving along a row), payload capacity (to ensure that the robot is able to carry the required sensor and manipulator payloads), speed of the robot (to ensure that the robot covers the required area in a given time), battery runtime (to ensure that the robot has enough power to do the required tasks for the entire brassica field on a single charge), and the robot's turning radius (to ensure that the robot can switch rows with ease). Through the trade study, the Robotanist platform [5] was considered suitable for the project.

6.2 Perception Algorithm Trade Studies

6.2.1 Monitoring Model (Spring)

Table 4 Monitoring Perception Trade Study (Parameters rated out of 5)

Type	Weight (%)	Binary Segmentation	Faster RCNN + GAN	YOLO + GAN
Detection Accuracy	40	2.5	5	4
Information Gained	40	3	5	5
Low Complexity	20	5	3	3
Value		3.2	4.6	3.4

The computer vision algorithm is essential for monitoring plant health regarding disease and signs of pests (such as holes). We compared a few aspects of the different options and weighted the values of these benefits as they pertain to our project. For example, the information gained corresponds to the size (measured in area) and prevalence (how many holes there are) of the damage. Generative networks such as Conditional GANs are capable of outputting a segmentation mask conditioned on an image input while object trackers can maintain a reference to absolute numbers of disease/signs of pests. Other metrics included the detection accuracy and implementation complexity. Note that since the computation is done offline, speed is not considered as a criterion. The Faster RCNN + GAN turned out to be the best choice as it has the highest detection accuracy while having the same score as YOLO in other criteria.

6.2.2 Monitoring Model (Fall)

Table 5 Monitoring Perception Trade Study (Parameters rated out of 5)

Type	Weight (%)	Unet	Dilated Resnet	InceptionV2
Small Amount of Training Data	50	5	3	2
Easy to Implement	30	5	4	3
Potential Performance	20	4	4.5	5
Value		4.8	3.6	2.9

The model implemented in the Spring 2019 semester stagnated around 60% precision and recall despite numerous efforts. Therefore, some other architectures were taken into consideration, specifically Unet, Dilated Resnet, and InceptionV2. Since the dataset we created had only roughly 100 - 200 images available per plant, requiring a small amount of training data is critical. Although larger networks usually have pre-trained weights available, they are mostly trained on common household and urban objects instead of agricultural images. In conclusion, Unet ended up being the best choice for this project due to its relatively small number of parameters, therefore requiring fewer data and being easier to implement.

6.3 Sensor Trade Study

Table 6 Perception Sensor Trade Study

Type	Weight (%)	Intel RealSense	Custom made Active Lighting Stereo - Small	Custom made Active Lighting Stereo - Tiny	ZED
Range Accuracy	*10*	5	4.5	4.5	4.5
Generate Useful Data	*40*	3	5	5	3
Low min distance	*10*	4	5	5	3
Low weight	*20*	5	4	3.5	5
Robustness	20	5	4	3	5
Value		4.35	4.6	4.49	4.05

Assuming we are using a neural network for weeding and plant health perception, the quality of RGB-D data would directly affect the outcome of these tasks. It is assumed that weeding and plant health monitoring share the same sensor, and hence the trade study criteria must account for both modalities. We compared a few key aspects of the different sensor options and weighted the values of these benefits as they pertained to our project. For example, generating useful data is key to the success of this task, because there is a relatively limited amount of data available from the field; hence, sensors with active lighting have a great advantage. Other metrics included the range accuracy, which would affect the accuracy of localization in weeding, and the minimum installation distance between target and sensor (usually demanded by stereo cameras). Since the sensor would be cantilevered, the weight of the sensor is crucial, as heavier sensors could affect the stability of the overall robot platform. Although the Intel Realsense is rather lightweight and has high range accuracy, it is subject to disturbances from sunlight; for this reason, the custom-made stereo camera with active lighting is the optimal choice.

Table 7 Mapping and Navigation Sensor Options

Type	***Weight (%)***	**3D LiDAR + IMU**	**2D LiDAR + IMU**	**Structured Light Camera + IMU**	**Time of Flight Camera + IMU**
Robustness	*30*	10	10	4	8
Information Richness	*30*	10	7	6	4
Low post processing	*25*	8	6	8	8
Low cost	*15*	4	6	10	8
Value		8.65	7.5	6.5	6.8

After knowing the generating the map and planned out a global path, the navigation algorithm is used by the robot to follow the path and avoid hitting plants. The quality of sensors supporting the navigation algorithm & mapping is essential to the success of this task. An IMU sensor is a basic sensor present in all options to provide orientation and angular velocity information. On top of that, we compared a few aspects of the different options of range/depth sensors and weighted the values of these benefits as they pertained to our project. For example, we compared the robustness of sensors, which corresponds to the sensors' ability to produce quality data under vibration or glare from sunlight. In general, sensors that emit pulses to the environment and compute distance-based reflected pulses are more robust to sunlight. The range that a sensor is able to work is also captured in this criterion. Other metrics included the information richness that a sensor is able to provide. For example, LiDAR has a 360-degree field of view, and 3D LiDAR has an additional vertical field of view on top of that. Low post-processing requirements are important for the schedule to be met. In general, the more straightforward a sensor is to use, the better. 2D LiDAR, for example, requires stitching over time to generate 3D data. The cost is less of an issue since many of these sensors are available in inventory with a fair amount of stock.

The 3D LiDAR is far more superior to other options in that it provides rich information and is robust to environment changes. Note that in accordance with our project scope, we do not consider performance on rainy days, where LiDAR is known to yield poor performance. Also, there are ROS packages that support 3D LiDAR directly to make implementation easier.

6.4 Localization Algorithm Trade Studies

6.4.1 SLAM Module (Spring)

During the Spring semester, we were initially considered using a full 3D SLAM pipeline to solve the outdoor localization problem for both in-row navigation and row-switching. During the Fall of 2018, direct SLAM methods such as LSD SLAM [4], state-of-the-art feature-based SLAM methods such as ORB SLAM [5] and appearance based methods such as RTAB-Map [7] were considered. However, we decided to move forward with testing the Lidar Odometry and Mapping (LOAM) package [6] in the Spring semester. As LOAM is a pure-LIDAR based approach and has been developed at CMU, it had the best support available for debugging. Hence, these factors played in LOAM being taken up for initial testing. The V-LOAM algorithm was dropped as an option as the package is closed-sourced and it currently being commercialized by Kaarta. Appearance-based and direct approaches such as LSD SLAM and RTAB-Map are unstable for outdoor environments due to drastic changes in lighting conditions over time. Table 6 showcases the SLAM algorithmic considerations which were taken into consideration at the start of the Spring semester.

Table 6 SLAM Trade Study (Parameters rated out of 5)

Parameter	Weight (%)	LOAM	LSD SLAM	ORB SLAM	RTAB- Map
Robustness to outdoor environments	30	4.5	3	4	3
Sensors required and compatibility with pipeline	30	5	5	5	2
GPU requirements	20	5	5	5	5
ROS Package availability and local	20	4.33	3	4	4
Value	100	4.716	4	4.5	3.3

*(*Note: The cost of the 3D LIDAR has not been included because of its availability in FRC and MRSD inventory)*

However, by the mid of March, the team concluded that a full 3D SLAM package was not required. In order to use the predefined structure of the farm environment, a particle filter model was developed. The particle filter utilizes a standard odometry based motion model as described in [8]. The particle filter has required desirable properties of describing multi-modal distributions by using importance sampling based approach to weigh different particles and converge to the best state estimate. A LIDAR row detection based sensor model was developed for the update step of the particle filter. This model was successfully used for row following and switching in indoor environments with fake plants.

6.4.2 Localization Module (Fall)

During the Fall semester, the outdoor navigation in the farm provided a new set of challenges. For outdoor localization, the particle filter based model developed in the first semester did not work out of the box. Hence, the RTK GPS based navigation approach was considered. The RTK GPS based approach had the advantage of ideally providing cm level precision. The robot_localization package could be used out-of-the-box without use of custom algorithm development. The EKF-UKF model is computationally less intensive as compared to the particle filter based sensor mode. However, the RTK GPS based method received low weights for the ease of outdoor testing due to previous difficulties in setting up the RTK GPS.

Both the row detector + particle filter model and EKF + UKF based fusion approach was used from Mid October - Mid November. However, the row detector + particle filter model was given up during Mid November due to the computational complexity of the particle filter and the difficulty in debugging the filter on ROS Bags. A simple weighted cost function based approach based on a simple sensor and motion model was utilized to achieve successful row following. By the end of the project, the row detector + simple localizer as well the EKF + UKF based platform were utilized to demonstrate different functionalities. The EKF + UKF based model was used to perform in-row navigation as well as row switching in the baseball field patch in CMU. Table 7 showcases the localization algorithmic considerations which were taken into consideration at the start as well as last month of the Fall semester.

Table7 Localization algrithms Trade Study (Parameters rated out of 5)

Parameter	Weight (%)	Row Detector + Particle Filter	Row Detector + Simple Localizer	EKF + UKF based fusion
Robustness to outdoor environments	30	4	4	4.5
Inherent accuracy of sensors involved	20	4.7	4.7	4.5
Computational complexity	15	4	5	4.5
Ease of use for outdoor testing	20	5	5	4
Implementation complexity	15	4	4.5	5
Value	100	4.34	4.565	4.475

7 Cyber-physical Architecture

The following figure shows the overall cyber-physical architecture of the robot which has been primarily divided into components which include User Interface, Processing, Sensing and Output. The high-level cyber-physical breakdown of each subsystem has been done below (the further breakdown is discussed as part of each subsystem).

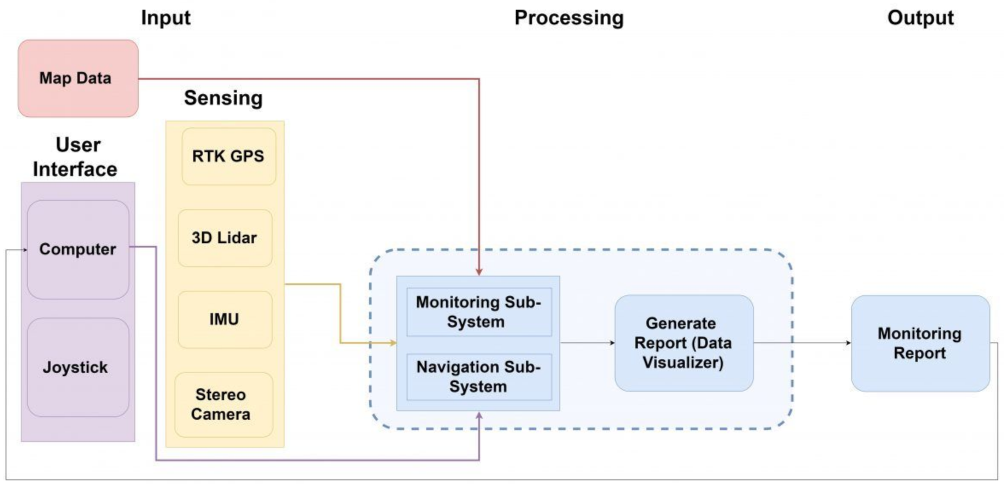


Figure 7 Overview of Cyber-physical Architecture

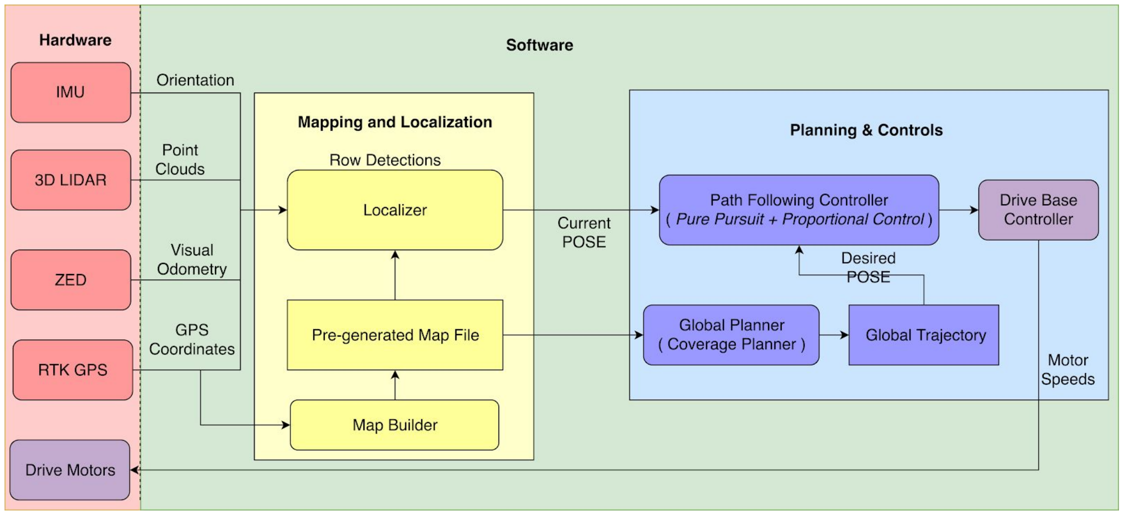


Figure 8 Cyber-physical Architecture of Navigation Subsystem

The navigation subsystem is used for monitoring. We have implemented two separate navigation subsystems in parallel; a lidar-based one and an RTK-based one, to explore all options and compare performance. The lidar-based navigation inputs information from 3D LIDAR (Velodyne VLP16 Puck), an IMU, and a stereo camera (ZED). The IMU and LIDAR are interpreted by a row detector, which is fed into the localizer along with visual odometry from the stereo camera in order to output the robot's pose. For the RTK-based solution, the sensor data goes into the robot_localization node and which fuses sensor inputs from the RTK-GPS and IMU to provide a position estimate output. The map, consisting of GPS points at the beginning and end of each row, is passed through a global planner which outputs a trajectory. The robot's pose is then driven to the trajectory pose using a pure pursuit controller.

The monitoring subsystem takes images from the robot's active lighting stereo camera and passes it through a perception pipeline to identify signs of pests and disease. The pipeline includes segmentation of holes and fungus via U-Net [1] which is fed into a threshold to generate a disease and pest severity classification.

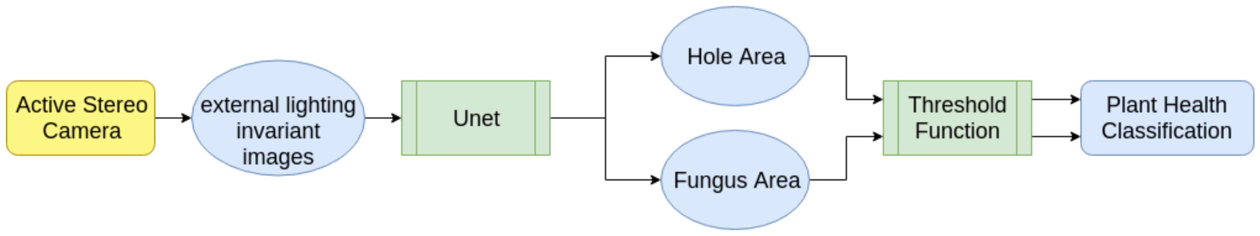


Figure 9 Cyber-physical Architecture of Monitoring Perception Pipeline

8 System description and evaluation

8.1 System/subsystem descriptions/depictions

8.1.1 Autonomous Navigation

Overview

The autonomous navigation sub-system has three major components, localization, planning and controls. The system has a map that is built once every season, after that during autonomous navigation, the system gets sensor data which goes to the localizer. The localizer provides an estimate of the robot's position. The map file generated before is used by the global planner to build a global trajectory of the robot. Finally, the controller uses the trajectory generated and the current pose of the robot to provide the desired velocity of the robot which are finally implemented by a low-level drive-based controller.

We tried two approaches for navigation. The two approaches differed by the way we localized the robot. The first strategy, the Lidar Row Detection based Navigation strategy, used the inherent structure of the farm to simply the problem of localization. In this approach, a row detector computes row lines from point clouds, and this is combined with visual odometry from the row detections. We had good results from this strategy inside the row, but the results were not good outside it. Further, this strategy worked well only when the plants were big and point cloud data from the plants was usable. This led us to try an alternative strategy where we fused RTK GPS and other sensor inputs to localize the robot. This strategy works well irrespective of the growth stage of plants, position of the robot. However, it requires an RTK GPS base station to be present and thus requires additional infrastructure.

Row Detection based Localization

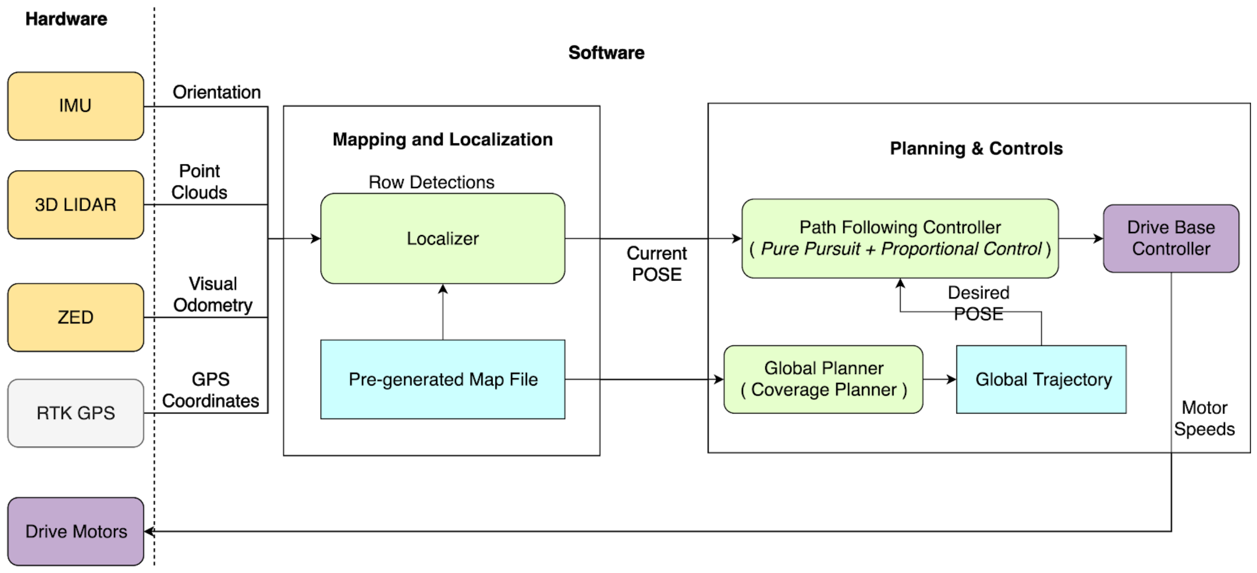


Figure 9 Architecture using Row Detections

The row detector is responsible for translating 3D laser scans into row lines. We chose to use the Hesse normal form of a line, in which a line is represented by its normal *n* and perpendicular distance to the origin *d*. Points *r* which lie on the line are therefore defined by $r \cdot n = d$. The normal *n* can equivalently be represented by $\theta = atan2(n_y, n_x)$. This parameterization was chosen over the standard in order to represent vertical lines correctly and prevent numerical instability.

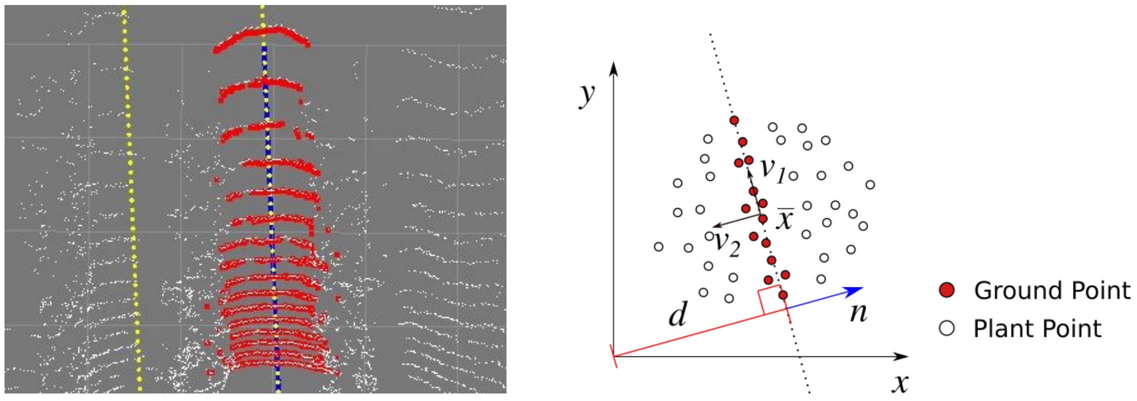


Figure 10: Left: Row perception output with lidar points (white), segmentation of the ground (red), estimated row center line (blue), and global trajectory (yellow); Right: Calculation of row line with eigenvectors $v_1$and v_2 , ground point mean *x*, row distance *d* and row normal *n*

Row detection is accomplished by first using the orientation estimate in roll and pitch from the Xsens IMU to rotate the point cloud into a stabilized ground frame. Next, the ground and plants are segmented by height and intensity of the laser returns. The ground in general should have a lower z-coordinate and lower intensity than the plant points. The position and orientation of the row is computed by projecting the points onto the ground plane (e.g. removing the Z coordinate in the ground-stabilized point cloud), and taking the orientation to be the direction of the major axis of the pointcloud. This vector is computed by computing the SVD of the covariance matrix of the points, and taking the eigenvector corresponding to the largest eigenvalue of the matrix v_1 to be the direction of the row. The eigenvector corresponding to the second-largest eigenvalue v_2 is therefore normal to the direction of the row, and this vector is taken to be the opposite of the normal $v_2 = -n$ in the Hesse normal form of the row line. The distance d is calculated by taking $x \cdot n$ the dot product of the normal n and the mean position x of the points. This process is depicted in Figure 10; with the output shown on the left and the process itself outlined on the right.

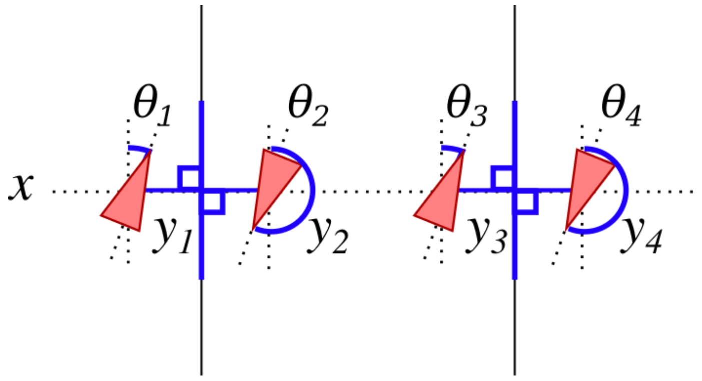


Figure 11: Visualization of the ambiguity in position given a fixed position *x* along the row and two rows. There are two possible poses for each row.

The localizer initializes its position at (0, 0, 0), assuming that the robot was roughly placed at the beginning of the first row, facing down the row. When localizer receives an incremental position update from VO, it applies this incremental update. When it receives a row detection, given the X coordinate of the current estimate, there are four possible positions and orientations of the robot, assuming the row detection in correct, as shown in Figure 11. It picks the closest pose to the current pose to resolve the ambiguity. In this way, the location will “snap” to the known position relative to the row lines in the map when they are visible, and the position will be updated with VO in-between detections.

RTK GPS based Localization

Before we can do autonomous navigation, a map of the field needs to be built, we chose a very simple map representation, i.e. the start and end GPS location of each row. These were collected using the map builder which collected outputs of the RTK GPS according to user input, this is presented visually in the flowchart below.

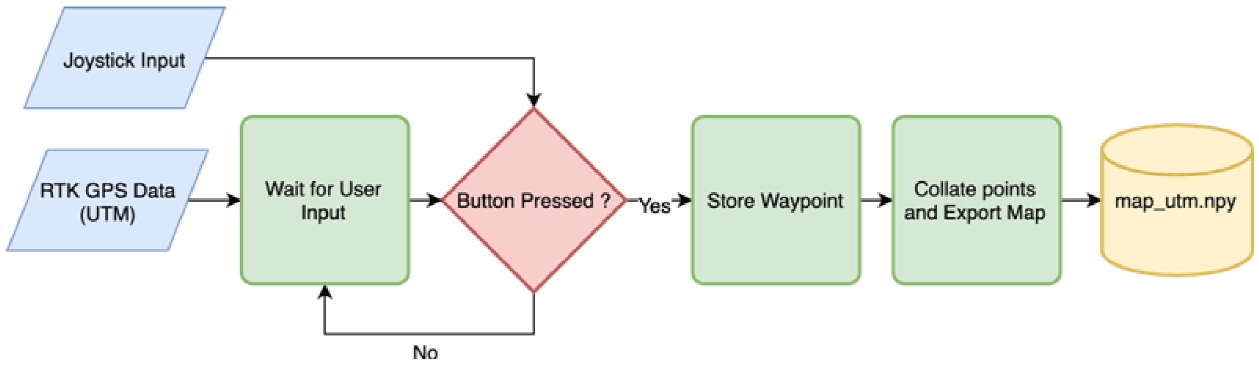


Figure 12 Flowchart showing pipeline for map-builder

Once the map is built, we make use of 'robot_localization' package to fuse the sensor readings from the RTK GPS and IMU using an extended Kalman filter to get an accurate estimate of the position and orientation of the robot. Although we initially tried fusing visual odometry from the ZED, we found that it was not required for our application as our robot moves at relatively low speeds ranging from 0.2 to 1 m/s. Thus, the final implementation works by fusing data only from the RTK-GPS and IMU and provides an estimate of the robots position in the entire six degree of freedom space, the reasons for the same will be covered in section 8.2.

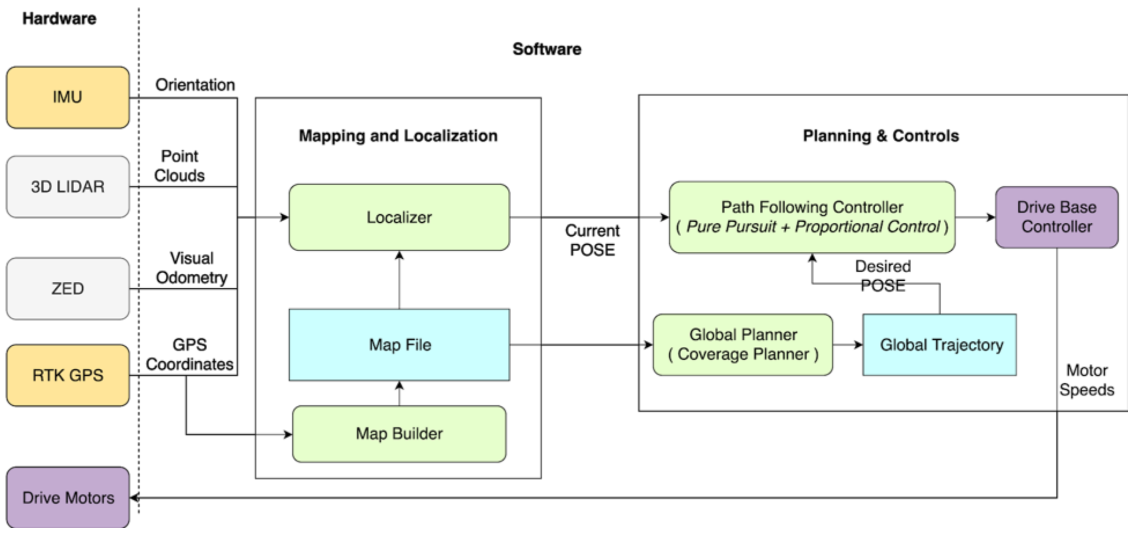


Figure 13 Flowchart showing entire pipeline for navigation using RTK GPS and IMU

The robot_localization package runs three nodes to complete the localization process.

The package runs 2 separate fusion algorithms, one running fusion between IMU and VO and other running fusion between RTK GPS and IMU. These separate nodes are required as GPS does not provide continuous measurement, further robot_localization makes use of three frames. First, is the 'base_link' frame which is the robot's frame, the next is the 'odom' frame which is the frame at the robot's starting position, the position in this frame can be subject to drifts and the 'map' frame which is a world fixed frame, the position in this frame need not be continuous and can have discrete jumps.

Thus, the first fusion between VO and IMU provides transform from odom → base_link whereas the second fusion node provides transform between map → base_link however to ensure consistency in the TF tree, its published as a transform from map → odom frame. The following figure represents the high-level architecture.

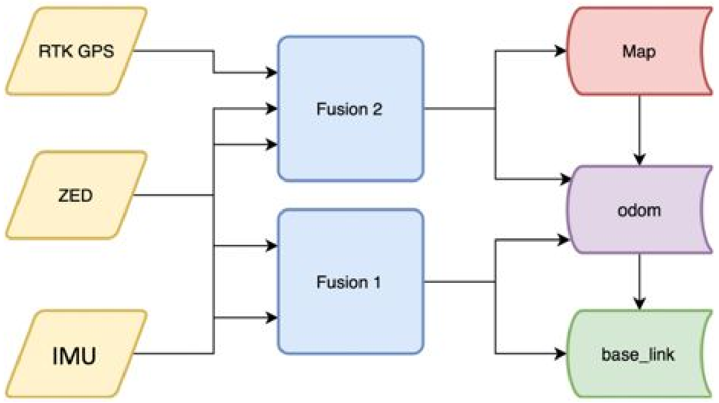


Figure 14 Flowchart showing fusion pipeline of robot_localization

Planning & Controls

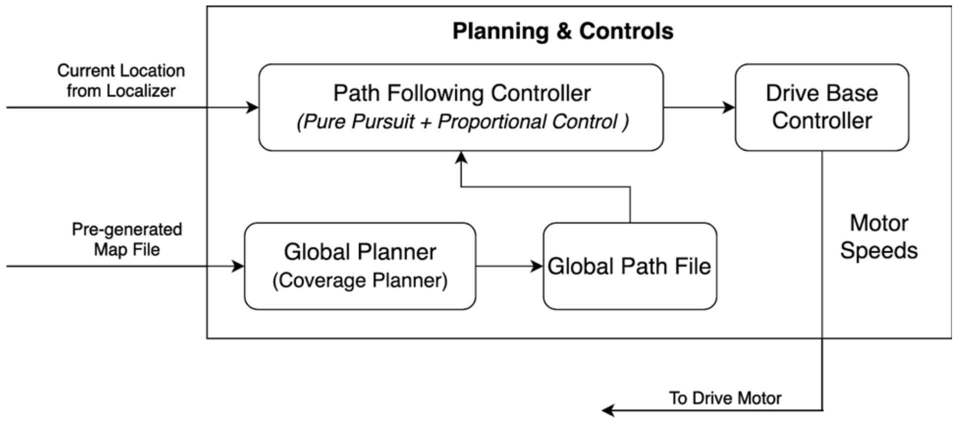

Planning & Controls
Current Location from Localizer
Path Following Controller (Pure Pursuit + Proportional Control)
Drive Base Controller
Pre-generated Map File
Global Planner (Coverage Planner)
Global Path File
Motor Speeds
To Drive Motor

Figure 15 Flowchart showing planning and controls pipeline

Planning sub-system

The robot should start from a designated starting point, enter the first traversable row and then switch rows to go to the next one. It should repeat this process to cover all the rows in the field. A coverage path planner was developed which identifies the order in which the rows in the field should be traversed. The algorithm takes the input as the coordinates of the start and end points of the traversable rows from the RTK GPS. These points are then rotated and translated to the map frame, using the UTM→ MAP transform from the robot_localization TF tree. Next, the order of traversing rows is identified. The planner is subject to the following constraints while planning the global path:

1. Cover Entire Area: The robot should cover the entire field of interest.

2. Traverse Each Row in Both Directions: Since the stereo camera is mounted on only one side of the platform, the robot needs to traverse each row in both directions so that the plants on each side of the row are captured.
3. Account for Turning Radius: The robot utilizes a skid-steer mechanism and has a large wheelbase. Hence it has a large turning radius. The planner should account for this while making the plan.

The robot should be able to switch from the first to the second traversable row. In case the distance between the traversable rows is less than the allowable turning limit of the robot, the robot skips the row and switches to the next one. Also, the robot does not repeat going through a row in the same direction twice. Finally, we get a list specifying the order in which the robot should traverse the various rows.

This order is converted into a path by generating a rectangular path while switching rows and a straight path for motion along a row. This path is then discretized in small steps of approx. 0.1m and is passed to the controller as a `.npy` file.

Controls Sub-system

The control sub-system takes the global path from the planner and current location from the localizer as input and utilizes a control algorithm to track the specified path. Since the robot is a skid steering robot, at any instant the robot can have a velocity along the angular and forward direction only. The sub-system utilizes a proportional controller to calculate the desired linear velocity and utilizes a pure pursuit algorithm to calculate the desired angular velocity of the robot.

Given its current position and orientation in the field frame from the localizer, the controller uses the pre-generated map file and finds the farthest point within a lookahead distance which is in a similar direction as that of the robot. This desired location is transformed into the robot frame. This goal position in robot frame is used to get the forward velocity using a proportional control using Eq. 1 and angular velocity using a Pure Pursuit controller using Eq. 2 with a lookahead distance of ~1m.

$$V_{forward} = K_p * x_{Goal}^{Robot\ Frame}.$$

$$\omega = \frac{(2*V_{forward}*y_{Goal}^{Robot\ Frame})}{l^2}$$

These velocities are then transmitted to the drive-based controller.

8.1.2 Plant Health Monitoring

Updates of the Pipeline

The monitoring model and the inference pipeline have been updated significantly in this semester. Specifically (1) Mask-RCNN is removed from the pipeline and Unet is used instead (2) a simple threshold function is introduced to replace logistic function (3) the 3 level severity system is replaced by the binary classification (4) foliage area estimation is removed from the pipeline.

Unet is a simple encoder-decoder architecture which was first proposed to do binary semantic segmentation of cell boundaries for medical image analysis. Since it has relatively fewer parameters, it is suitable for this project in which the amount of data is limited (roughly a hundred images for each plant). Network architecture aside, with the output of detected hole and fungus areas, a threshold function is used to replace logistic functions. One reason for this change is that there is a more established analysis methodology for threshold functions such as receiver operating curve. To further simplify this problem, the original three levels severity system has been reduced to binary classification: essentially a presence detection system. This is because when the severity has gone to severe it is probably too late for the farmers, so any presence is useful information. As for the leaf foliage estimation, it was originally proposed so that we could divide affected area by total areas, arriving at a dimensionless metric to gauge the severity of pest holes and fungus. This was needed since the previous field setup required us to change the height of the camera during a monitoring run. However, the stereo algorithm used for foliage estimation fails to generalize across plants of different shapes. Thus, the foliage estimation actually hurts the overall accuracy. Additionally, the new field setup does not require us to change the height of the camera, and so the camera height can be set to a fixed value for each plant with a prior known height to get a consistent metric. As it hurts accuracy and is no longer needed for consistent crop monitoring, leaf foliage estimation has been abandoned.

In the updated pipeline, given an RGB input image, the image is fed into the Unet model to detect hole and fungus area, which would be further sent into a threshold function to obtain a binary label (e.g. this plant has signs of pest).

Challenge of the system

TThere are two major challenges for the system, the lack of ground truth and training model with a small dataset. Initially, farmers asked evaluation based on the percentage of the leaf area e.g. 10% of the leave foliage is fungus. However, pixel-wise ground truth is very hard to obtain as many symptoms are ambiguous, lacking clear boundaries, etc. Initially, intersection over union was used i.e. if predicted fungus area intersects with human annotation, then it's considered valid. But again, this metric suffers from the ambiguity of symptoms and the fragmented distribution of annotations. This issue is finally resolved by simplifying the problem to a binary classification problem where humans can confidently tell whether or not a plant has fungus or pest issues.

Training of the actual model was not a trivial task. Initially, Mask-RCNN was used for this project because of its allegedly strong semantic segmentation power, and of its versatile capabilities e.g. showing bounding boxes at the same time which can be used for counting. However, Mask-RCNN is indeed a complex network that has to be pre-trained first, which was usually done on common daily objects such as cars, people, tables, etc. The Mask-RCNN model failed to yield performance that meets the target of this project. To resolve the issue, a much simpler UNet model was proposed. Originally used to biology cell boundary detection, UNet was designed to work on an extremely small dataset of 30 images. Thus, this model with fewer parameters is possible to train from scratch with our small dataset of 100 images per plant. The actual training process still encountered some issues of imbalanced training data: in any given image, there are far more pixels representing background that those representing fungus and pest holes, in fact, the ratio of background to fungus to holes is roughly 100:10:1. As a result, the model learned to predict every pixel as a background. To resolve this, a weighting function was introduced during the training process to account for this imbalance which indeed generates good semantic results.

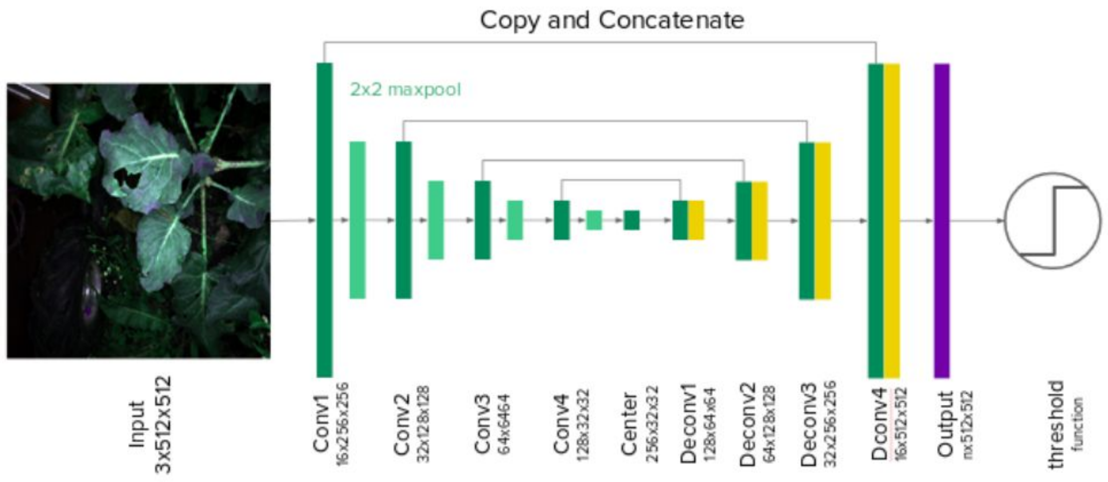


Figure 16 Inference Pipeline

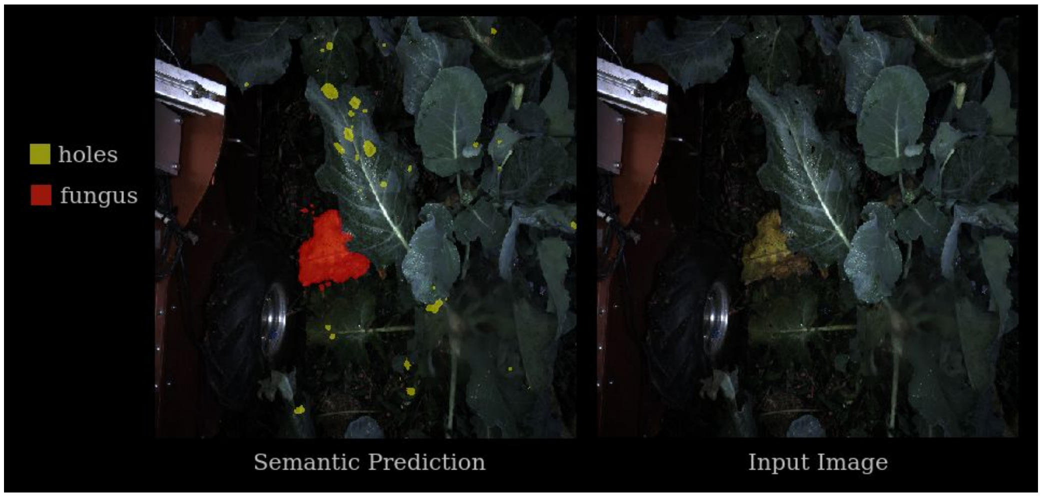


Figure 17 Fungus and pest hole areas are detected

8.1.3 Visualization

Overview

The purpose of the entire project is to give the farmers a tool for understanding what is happening on their farm. With this in mind, it is critical that the farmers be able to see and even interact with the reports generated by the robot. For this reason, we developed an interactive graphical user interface (GUI), incorporating feedback from the farmers, that captures the key functionalities they required to get the most out of our system. At a very high level, the visualizer needs to portray the results of the plant health in a way that corresponds spatially to the locations of the images in the field.

Pipeline

The main goal of the visualization pipeline is to provide geo-tagged, meaningful data to the farmers which allows them to gauge the pest and disease pressure in the field. Figure 18 shows a block diagram that summarizes the flow of information in the visualizer pipeline. The first step in the pipeline to provide ROS Bag data as input. The ROS Bag Parser module extracts the left and right images from the stereo camera and associates each image with its corresponding nearest GPS location. The ROS Bag parser has been designed to handle multiple ROS Bags as input which would represent multiple individual rows of data collected by the robot. There is an optional exposure checking module in the ROS Bag parser which checks if the majority of images are not either over-exposed or under-exposed. The second step in the pipeline involves adding pest and disease classification labels for each image. In order to achieve this, the images are passed through the UNet classification pipeline which has been described in detail in Section 8.1.2. The labeled and geo-tagged image data is passed into the visualizer GUI for visualization. The details of the GUI features have been explained in the next section.

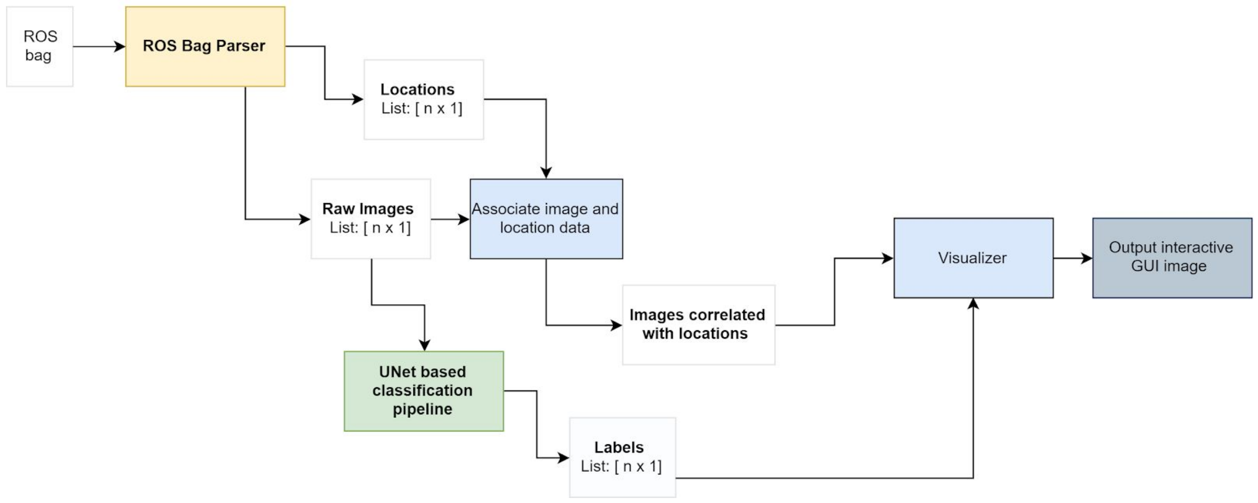

ROS bag
ROS Bag Parser
Locations List: [n x 1]
Raw Images List: [n x 1]
Associate image and location data
Images correlated with locations
UNet based classification pipeline
Labels List: [n x 1]
Visualizer
Output interactive GUI image

Figure 18 Visualizer pipeline at a glance

Field Visualization

The first screen the farmers need to see is one that considers the entire field. The farmers are most interested in a zoomed-out view of their field, such that they can track the spread of a disease or pest and view its implications in the context of their entire farm. The pipeline returns data associated with its longitudinal and latitudinal coordinates, meaning that a simple plotting of each image's value at its stored location yields a fair representation of the farm. However, farmers are not actually interested in individual pictures. There are almost 100 plants per row and an average of more than three pictures per plant. Graphing each image as its own datapoint, while effective, creates an overload of information, making the visualization far less meaningful. The farmers requested, as a way to better contextualize the data, that pictures be clustered into sets, and that a single value be reported for each set. To do this, the visualizer uses a simple polynomial fit algorithm to fit all the data from a single row (each data point is labeled as belonging to one specific row) to a line. Cluster points are then generated at even intervals along the line (the number is a hyperparameter that the user can configure) and each datapoint is associated with the cluster closest to its location. What we get then is a visualization with less datapoints, each of which corresponding to multiple images and their results.

To make the visualization easy to read and meaningful to the farmer, each datapoint is drawn with an icon corresponding to the types of issues detected in that cluster's images. Specifically, if any one of the image in the cluster has fungus, the fungus icon is drawn. The same is true for the pest icon. If both fungus and pests are detected, each in at least one image, both icons are drawn. The icons are also colored based on the overall health of all the images. That is, if every image in the cluster has both disease and pests, the icon is drawn gray, if all are completely healthy it is drawn green. Any other combination is colored by the total number of issues (the number of images with pests + the number of images with disease) divided by two times the number of images (figure 18 displays both the icons and the color spectrum used).

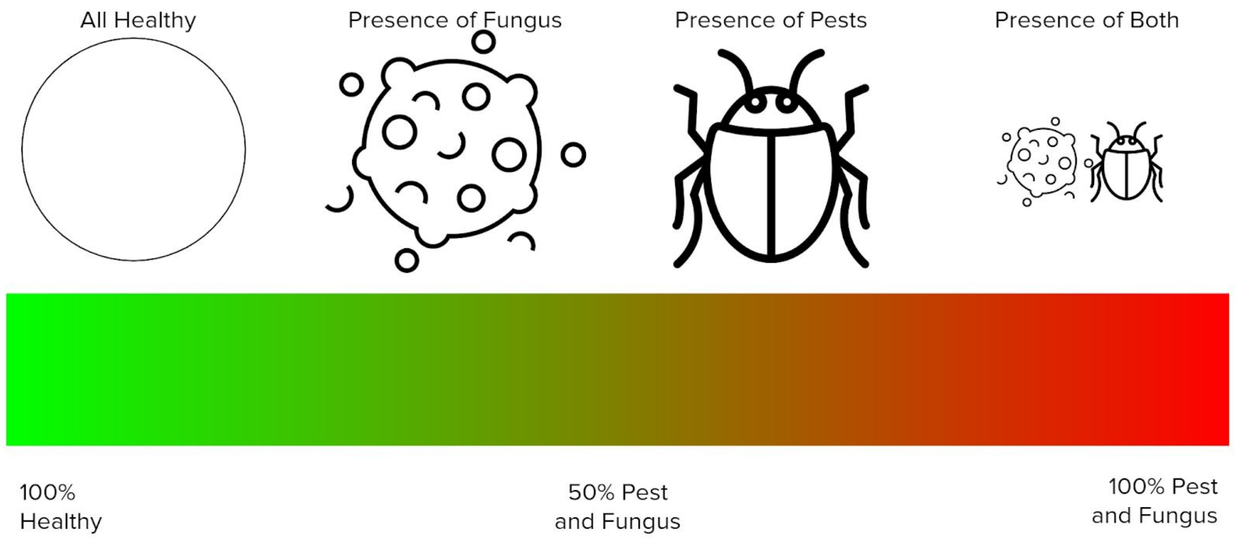


Figure 19 Icons and coloring used for visualization

The resulting graphic has a scale bar, generated using a known conversion equation between longitude and feet, and a compass rose, drawn to aid the farmer in interpreting the plot. An example of the final visualization can be seen in figure 20.

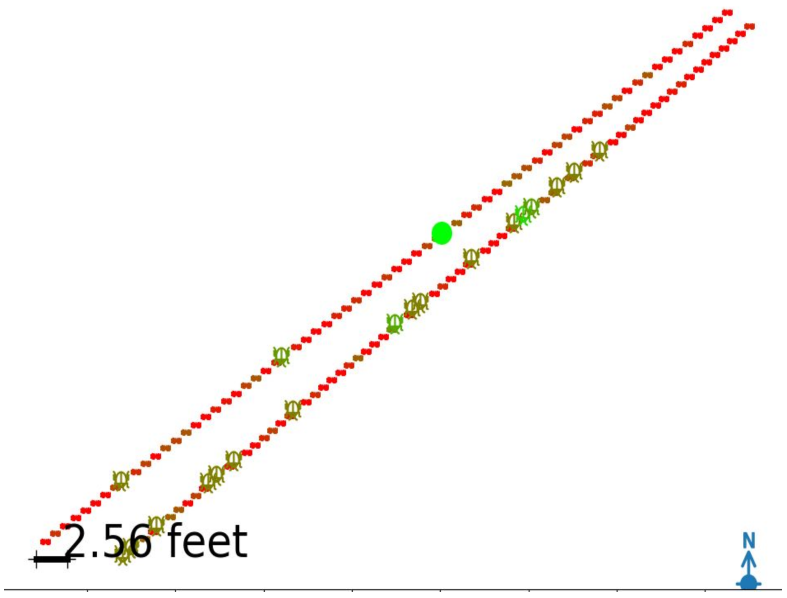


Figure 20 Example of visualization for two rows

Cluster Viewing and Editing

The farmers also need the ability to look at and change the details of a particular cluster of images. Clicking on any icon opens a second window (see Figure 21) where the farmer can inspect the images of that cluster and override the health scores given by the monitoring inferences. The window includes navigation buttons that switch images; these buttons deactivate when there are no more images in that direction. There are also buttons to change the hole or fungus values of the active image or all the images in the cluster. Beneath these buttons there are 2 rows of green/red indicators. The first row indicates whether the corresponding image has holes, and the second row does the same for fungus. For example, in Figure 21 the first three images of the cluster have holes, the fourth has fungus, and the fifth has both. Finally, at the bottom of the page is a text field where the farmer can store notes about the cluster. For example, if the farmer is able to identify a specific strain of fungus or species of pest, they may want to take note of that in the given space. These changes are all saved, such that they can be recalled when the cluster's window is reopened and the visualization of the field is updated to match the saved disease and hole scores.

Closing the visualization of the field saves all of the data, such that it can be recalled the next time the farmer runs the visualization script.

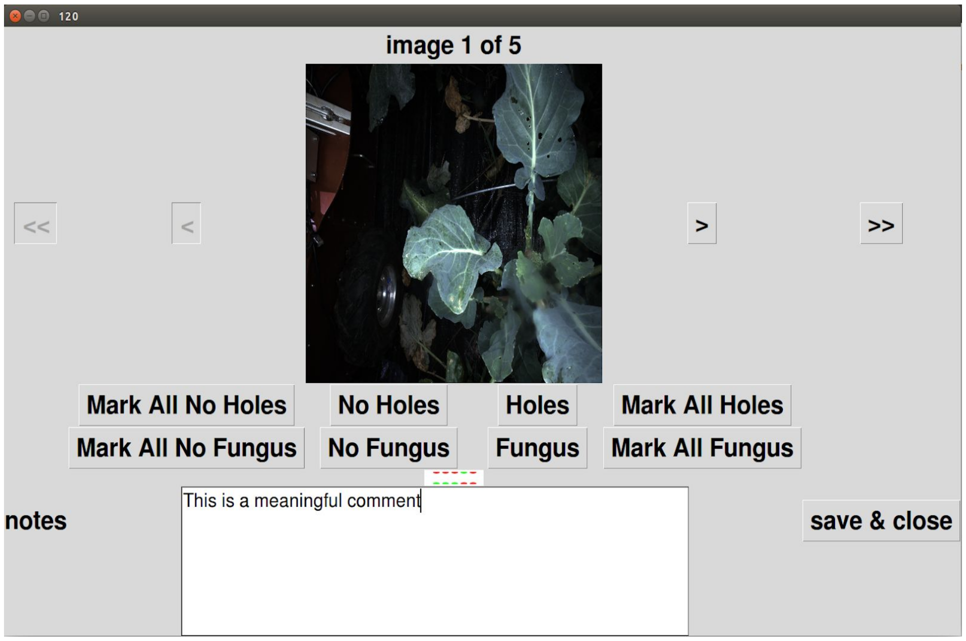


Figure 21 Example of GUI window for a specific cluster

8.2 Modeling, Analysis, and Testing

We first conducted a field test at Rivendale Farms in September to collect training data for our plant health perception algorithms and collect requirements from our sponsor, Rivendale Farms. One concern we had was the length of the sensor boom required to hang the camera over the plants. We, therefore used a spreadsheet to determine the desired center of mass, payload capacity and footprint of the robot in order to meet our requirements without tipping. We converged on using the same mobile base design as an existing robot in the FRC, the Robotanist. This vehicle has a narrow track and a long wheelbase which enables it to fit in between rows of the field.

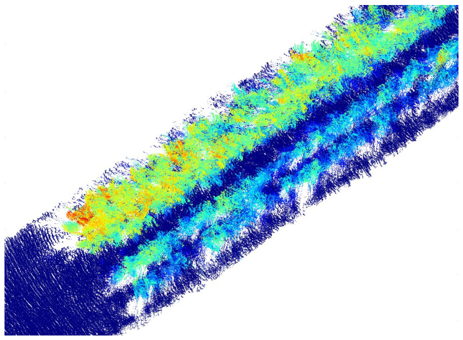


Figure 22 Top-down point cloud view of a row of crops at Rivendale, used for the analysis of possible row perception strategies.

Using the RTK GPS data from the field test, we constructed top-down 3D point cloud views, shown in Figure 22, of the farm which we used to analyze the width of the robot which was permissible as well as the variation in height of plants, both with would constrain our design. We also visualized the LIDAR scans of the crops in order to analyze what solution would best work for localization. We found that a Gaussian Mixture Model could convincingly generate a down-row view, as shown in Figure 23 (e.g. one which collapses the axis facing down the long direction of the row) from centroids and covariances computed from data. This lead to an initial row detection approach based on a particle filter with a sensor model derived from this Gaussian Mixture Model.

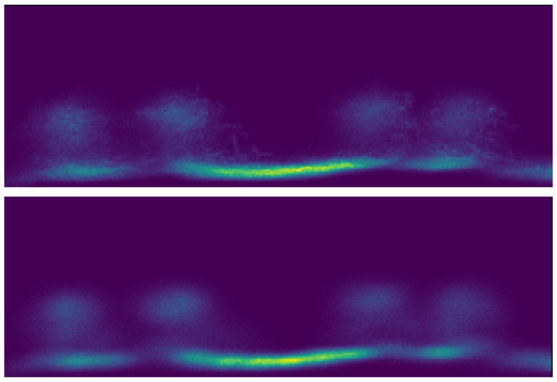


Figure 32 Actual down-row view of point cloud (top) and point cloud generated from Gaussian Mixture Model (bottom)

After the initial field visit and analysis, we converged on a navigation approach and tested it in March on the dead brassica crops. While these crops were not necessarily representative of the crops when they are fully grown, they enabled us to test in a low-risk environment and also have exposure to the effects of bulldozing while row following and turning. We were able to follow a row for several meters with a simplified navigation architecture and used ROS bags collected from that test in order to improve our approach. Included in our ROS bag was RTK GPS data which we used to validate our location estimate during our SVD.

We iterated on our navigation approach by testing our row detector on data gathered in Fall 2018 as well as testing our planning and controls with an artificial test setup in Spring 2019. This approach enabled us to validate our system even though access to the actual test site was challenging to achieve.

Figure 24 Changes to the Rivendale farm: addition of tarp to prevent weed growth and metal hoops for tunnel containment of plants. The metal hoops prevented navigation on all but a few rows of crops.

During the Fall 2019 semester, we decided to switch to RTK GPS based navigation as we we worried about the feasibility of lidar-based navigation due to poor results during our final Spring 2019 field test. We thought that RTK itself could be accurate enough to ground-truth and improve lidar-based navigation. However, initial field test experiments with RTK GPS showed that is was not as accurate as we liked. Even though it was globally accurate, it was often not accurate enough in certain parts of the field to avoid hitting plants. As a result, we decided to parallelly pursue RTK based navigation and lidar-based navigation. Without ground-truth, lidar-based navigation relied heavily on field tests, and we ramped up our field testing efforts to reflect that. RTK based navigation also consumed a large amount of field testing time, as strong results on-campus often did not translate to strong results in row-following on the field. We completed six field visits this semester, with the final weeks leading up to Thanksgiving break often seeing a field test every week.

On the farm, several changes to the test environment narrowed the scope of our project, shown in Figure 24. Firstly, the total number of rows for brassica was significantly reduced, due to issues with the previous year's crops. Many rows added metal hoops for containment tunnels. This limited the rows on which we could reasonably navigate, as it made the rows too narrow for our robot. A weed suppression tarp was added to the brassica crops, which made the ground very dark in lidar-view. We decided to switch our row-detection efforts from detecting and estimating the orientation and relative position of plants, to estimating these quantities of the ground row-center line. We did this after seeing that the intensity of the plants' points versus ground was very discriminative. However, this later caused a problem when the ground became muddy.

EKF-fusion-based localization was used in order to mitigate the risks associated with row-perception based navigation. Since the robot_localization package assumes the robot to be omnidirectional, its process covariance was modified so that its estimated dynamics more closely resembled a skid-steering robot. Since testing in field can be done only a limited number of times a semester. We ensured that we collected sensor data during every field visit and saved them as rosbags. This allowed us to test the pipeline on realistic data before taking to the field. Further, the EKF fusion was tested on campus before testing in the field. Despite all these measures we faced various challenges in actual implementation of the localizer in the field. This was confusing as our RTK-GPS based localizer worked well in rosbags and during testing on campus but not in field. We later identified the issue was that we were not modelling the environment well. We were implementing a 3 DOF fusion and thus were not accounting for variations in the ground. This caused the robot to have a wrong estimate of its position due to its high form factor. This issue was resolved by expanding the fusion to 6 DOF and transforming the location of GPS antenna received to the base of the robot by utilizing the orientation of the robot and the static transform between base and GPS-antenna. The following image explains the challenge and how it was resolved.

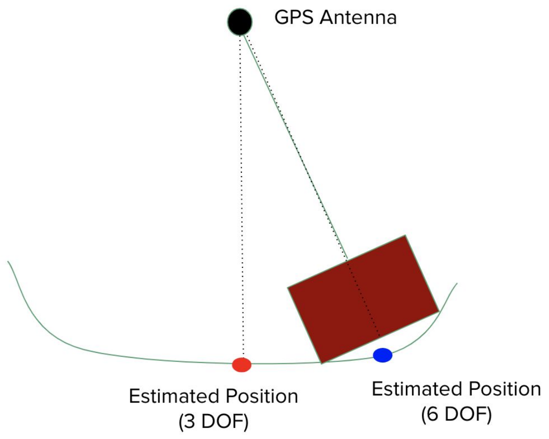


Figure 25: The image shows how the estimate between the 6 DOF fusion and 3 DOF fusion varied because of varying orientation of the robot.

As for monitoring, data is separated into training, development and testing splits. Specifically, training split is mostly used to train the network and the development split is used for parameter tuning to find the best threshold value, and the testing split is used to test how well the model can generalize on an unseen dataset. To preserve the best generalization power, the threshold valued at equal-error-rate (EER) is chosen. EER is the threshold value where the percentage of false acceptance rate equals the false rejection rate. Fundamentally, this assumes that it is equally likely for the system to see a positive or negative symbol.

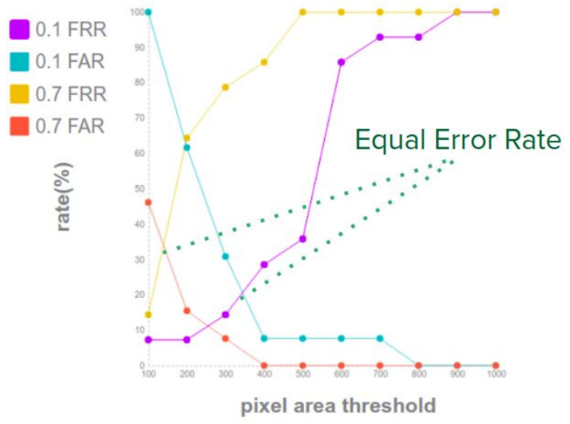


Figure 26 select area threshold with equal-error-rate

8.3 Performance Evaluation

The Fall Validation Demonstration tested the integration of the entire system by evaluating the process of collecting and analyzing data from the field. That said, the success metrics can be broken down into three sets corresponding to the three subsystems. While we were unable to show the complete process from beginning to end, due to availability conflicts during the last week of the semester we did successfully demonstrate each piece of the pipeline working meaningfully and compatible with each other subsystem.

The first set of metrics evaluated during the FVD were those relating to navigation. At the initial demonstration we successfully showed that the robot avoided plants and navigated down the row successfully using our detection based approach, but we were unable to show that the row switch was at least 80% effective. We confirmed this metric during the encore presentation, where we used the updated RTK GPS and 6 DoF IMU Extended Kalman Filter approach.

We also showed that the monitoring pipeline worked according to specifications on the data collected. We showed calculations proving the pipeline could be run for the entire field within 13 hours (well under the required 24 hours) and we also showed, on new data, that the network exhibited greater than 80% precision and recall for holes and fungus on multiple plants (results for fungus detection on curly kale was among the highest performance).

Finally, we demonstrated the functionality of our visualization tool in realtime. We passed our objective metrics for data saving and GUI features. The most important metric is unfortunately subjective, namely, whether the farmers find the visualization meaningful and helpful. We can only speculate on this metric, based on the feedback we received previously for the GUI, which was completely positive.

We completed our FVD encore with all of our metrics successfully met, meaning that the project, at a glance, can be deemed a success. We recognize that we did not show exactly the scenario depicted by the FVD test outline, but by showing the individual parts artificially run as a pipeline (i.e. using the data from one part to feed into the next) we very closely simulated what we had initially planned to show.

Table 8 Performance metrics for Binary Classification of Fungus and Holes

Binary Classification Performance			
Plant Type	**Category**	**Precision**	**Recall**
Broccolini	**Pest Hole**	79%	100%
	Fungus	95%	95%
Curly Kale	**Fungus**	91%	83%

8.4 Strengths and Weaknesses

We were able to deliver a plant health classifier which met our performance requirements for precision and recall of hole and fungus presence. We were also able to integrate this into a pipeline which inputted images collected from autonomous navigation, and displayed the data in a user-friendly top-down field view with inspection capability. However, our autonomous navigation was a weak point. We were able to demonstrate reliable row following on the farm with our lidar-based navigation. We were also able to independently demonstrate row following and switching on an artificial plant setup on-campus using RTK GPS. However, we were not able to demonstrate row-switching on the field. Iteration on in-field row navigation was slow, due to the lead time required for field visits, including planning, packing the van, driving to the field, assembling and setting up the RTK base station, and debugging any hardware issues caused by disassembling and reassembling the hardware. We also faced increased time pressure as the plants slowly died during the semester. Row-switching in particular, depends on regular field tests. While in-row lidar perception can benefit from recorded data, the view of the row seen for row switching depends on the specific trajectory of the robot, which itself depends on the perception output. For this reason, recorded data is less useful for debugging lidar-based row switching. More regular field access, particularly during the summer growing season, could have improved the results of the navigation pipeline. With the lidar navigation in particular, one challenge was that we built an approach assuming that the ground outside the row would have a particular intensity of reflection based on the grass. However, repeated testing turned this area to mud, which confused the ground-plant classifier.

The dataset which we created to train the plant health classifier was also a weakness. We collected and labeled data from brassica plants in the field during Fall 2018, and used this dataset for our development during Spring 2019 and for the results shown at SVD. However, Rivendale faced extreme difficulty with these crops which lead to major changes at the farm or the 2019 growing season. They changed the crops they were growing, which caused us to have to create a new dataset, from scratch, during Fall 2019. Due to the increased time pressure, we decided to out-source the data labeling to Scale.AI. However, Scale.AI was not familiar with agricultural labeling tasks, being more experienced with autonomous driving applications in particular. Frustratingly, all of the labels they generated were erroneous and not suitable for training. About half of the labels were usable for training after refinement work. However, this still left about half of the data that had collected essentially unlabeled, leaving much to be desired. We also encountered weed and other nuisance plant growth during the season that was not present in our training dataset, which was therefore mis-classified as fungus by the network, and lead to a drop in our metrics. Definitely having a more extensive and robust dataset could have improved the metrics on the plant health monitoring side.

One challenge in general for agricultural computer vision applications is getting suitable ground truth labels given the difficulty in finding people with the proper skill set. Susanna, the lead grower at Rivendale, left the company while we were still doing our project and was not replaced as of the end of our project. For this reason, we were not able to get final expert judgement on the efficacy of our monitoring system. We were able to get her expert judgement for the majority of the time we were working on the system, however, and built our approach around her feedback.

9 Project Management

9.1 Schedule

As compared to the previous semester, we added additional focus to track our set milestones and regularly updated the progress in achieving the milestones. The major milestones in the start of the semester included an autonomous navigation MVP, an MVP for the data parsing and visualization pipeline and updated plant health detection models which can achieve the requirements for precision and recall. A minor milestone for the hardware side of the project was to create plant guards and test them out in the field.

For achieving the autonomous navigation MVP, by September mid, we achieved the goal of debugging and setting a stable RTK GPS base station, which helped us mitigate one of our major risks coming into this semester. The RTK GPS based navigation node was integrated with the planning and controls pipeline by the end of September. However, both RTK GPS based navigation and LIDAR based navigation did not work initially in the first week of October. Hence, we needed to implement a sensor-fusion based approach for navigation. The time required for sensor fusion had not been explicitly planned for and caused us to drift from the expected milestones for the navigation pipeline. For the hardware aspect of the project, the plant guards had been modelled, manufactured and tested by the end of October. The work package to design new camera mounts was de-scoped and that completed the required hardware modifications for the robot.

For monitoring plant health detection module, we were able to test the binary classification modules by the end of September, which put us on schedule. However, in order to achieve the required precision and recall for the fungus and hole detection, we could not meet the milestone to achieve that by the start of October. The milestone was achieved in mid October. It required till the mid of November to reiterate and provide final results for the plant health detection pipeline. The exposure checking part of the pipeline was de-scoped as we relaxed those requirements to adjust with the changes in the field made by the farmers in Fall semester. The ROS Bag Parser and visualizer module was on track until mid October. The integration process involved version conflicts and it prevented us from achieving our milestone of a fully integrated pipeline until mid November.

Overall, we were able to hit all our major milestones by the FVD. We had some time to explore some of our stretch goals such as improving LIDAR row detections and trying fusion approaches for autonomous navigation. We were also able to add farmer requested features such as data points clustering and note-taking for the visualizer. However, allocating extra time for integration and testing could have ensured that all the milestones were met on the exact dates we had planned at the beginning of the semester.

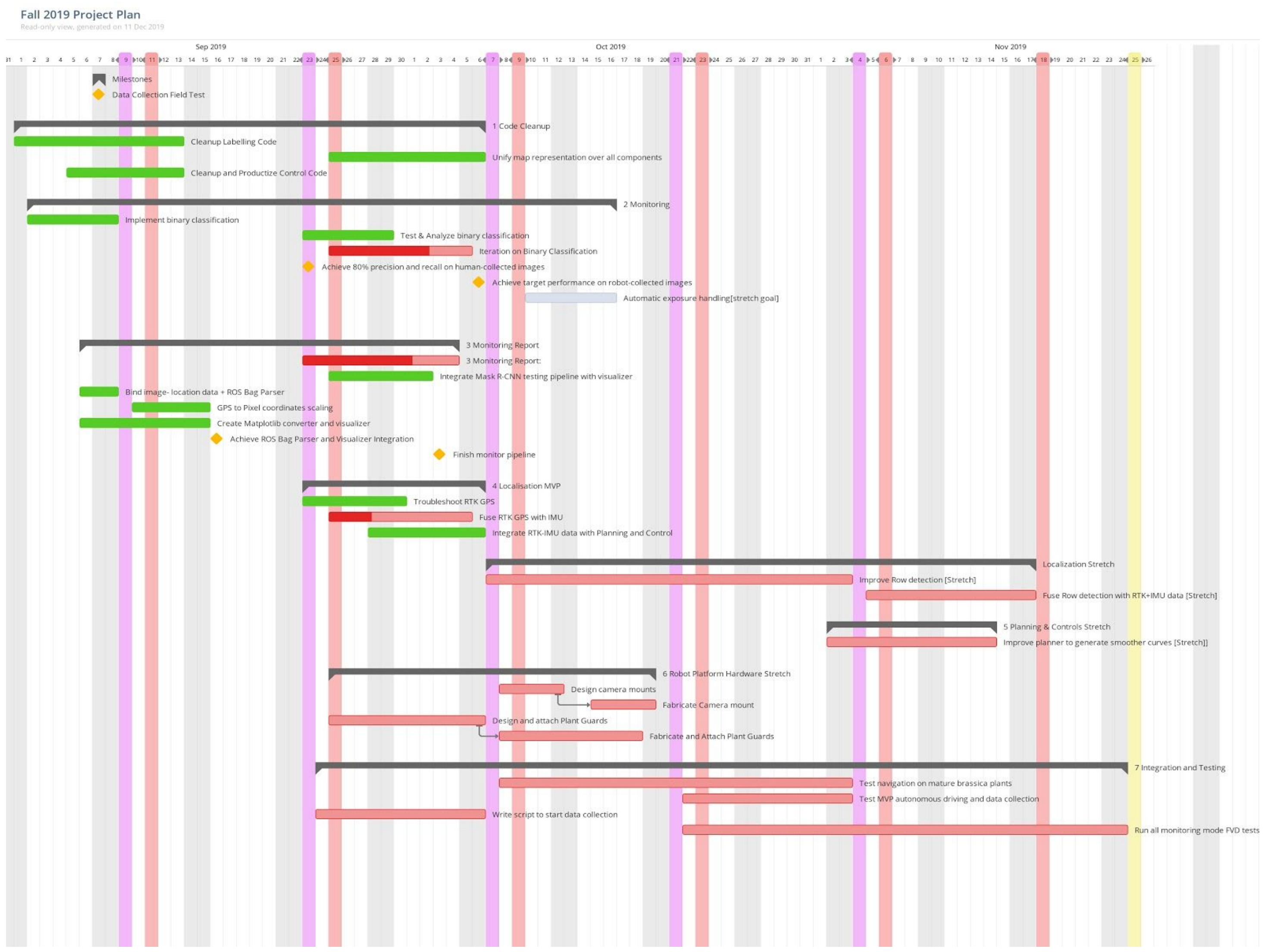


Figure 27 Gantt chart for the fall semester's work plan

9.2 Parts List and Budgeting

We were largely under budget for this project. We have used only 32.5% of our MRSD budget and approximately 20% of our sponsorship budget. This is because we were given a number of key components from our academic sponsor, and we were able to secure a lidar puck from the MRSD inventory. We also cut down drastically on our expected spending by descoping weeding (initially estimated at around $30,000, mostly due the cost of a robot arm).

Table 9 Budget (actually spent)

Part Category	Cost	Budget
Zotac	$1,200	Sponsor
Computing components	$4,00	Sponsor
Robot Platform Hardware	$5,000	Sponsor
IMU	$1,500	Sponsor

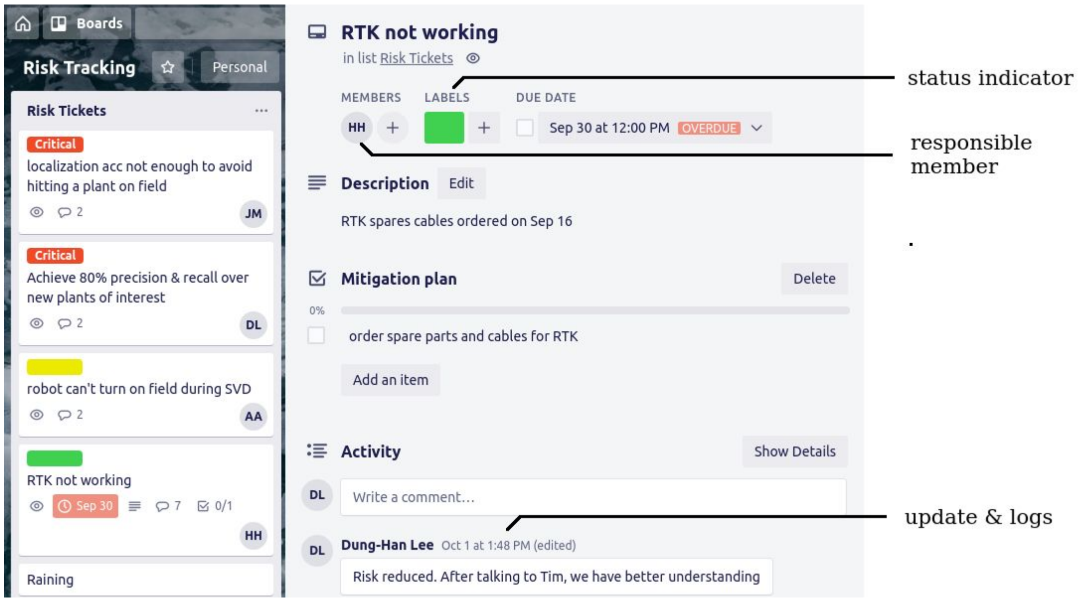
PCB items	$30	MRSD
DIN Rails	$60	MRSD
Fake Plants	$250	MRSD
Robot Replacements and Modifications	$115	MRSD
Weeding End Effector	$440	MRSD
Field Test tools	$435	MRSD
Plant guards	$250	MRSD

9.3 Risk Management

The process of creating the Work Breakdown Structure and Schedule enabled us to identify certain critical risks that could hinder the progress of the report. Each risk has been categorized and assigned labels of low, medium, or high for both their likelihood and their impact on the project.

The most important risks in the spring semester were (1) not able to assemble new platform in time (2) work scope is too big (including both monitoring and weeding) (3) missing out critical subsystem problems due to insufficient communication among team members (4) lack of in field training data. All of these risks had been mitigated or resolved in the next semester by descoping weeding and assembling new platform. Furthermore, team members have been condensed into navigation and monitoring subgroups where cross communication was frequent enough to get rid of ambiguity. Last but not least, enough training data had been collected.

In addition, Trello board was used to keep track of the risks throughout the semester. Red, yellow and green are status indicator where red representing high likelihood and medium to high severity about the potential impact and green suggesting low likelihood. Each of the risk ticket can be assigned a responsible person and due date. Furthermore, each ticket can extend into a board for action items and issue logs.



Risk Tickets

Critical
localization acc not enough to avoid hitting a plant on field
2 JM

Critical
Achieve 80% precision & recall over new plants of interest
2 DL

robot can't turn on field during SVD
2 AA

RTK not working
Sep 30 7 0/1 HH

Raining

RTK not working
in list Risk Tickets

HH Sep 30 at 12:00 PM OVERDUE

Description
RTK spares cables ordered on Sep 16

Mitigation plan
0%
order spare parts and cables for RTK

Activity
DL Write a comment...
DL Dung-Han Lee Oct 1 at 1:48 PM (edited)
Risk reduced. After talking to Tim, we have better understanding

Figure 28 Trello: Risk Management Tool

The following table list all the major risk during fall semester. All of the risks have been mitigated (reduced likelihood or remove completely) until the end of the semester.

Table 10 Risk Management

	Risk	Risk Category	Likelihood	Impact	Solution
1	Robot can't turn on field due to blocking weeds	Logistics/Facilities	High	High	Remove weeds manually
2	Achieve 80% precision & recall over plants of interest	Technical	High	Medium	Use separate model for each type of planet
3	RTK not working	Logistics/Facilities	Medium	High	Talked to experienced technician and document the exact steps for setup
4	Localization accuracy not enough to avoid hitting a plant on field	Technical	High	High	Parallely develop both RTK based navigation and LIDAR based navigation methods

10 Conclusion

With the end of the Fall semester, we were able to test our completed system in the field. We successfully met our FVD requirements, within the realm of the constraints given by the availability of the farm and its plants. We will be sharing our results with our sponsors at Rivendale farms in the near future.

10.1 Key Lessons

The key lessons can be summarised as follows:

1. The importance of risk management and backup plans was observed specifically when the project was going off-schedule. Time spent in burning down risk is valuable, as we significantly de-risked the RTK system by investing time into debugging and understanding the configuration of the system.
2. It is important to define a manageable scope up-front. Our project initially had many components in scope such as weeding which were later de-scoped.
3. Collecting agricultural datasets is challenging due to the diversity of nuisance plants which can present themselves and confuse the classifier.
4. Labeling agriculturing datasets is challenging due to difficulting in finding people which can generate suitable quality labels.
5. Components dependent on field testing with progress more slowly, and efforts should be made to create ground-truth so that these components can be evaluated offline
6. Agriculture growth cycles are not in sync with the MRSD project timeline. This hindered our progress as testing for certain key functionalities was delayed due to a lack of plants in the field.

10.2 Future Work

The goals for any future work are as follows:

1. Implementation of the weeding system that we had previously descoped
2. Improving the robustness of the navigation subsystem
    a. Reducing false-positive and false-negative lidar row detections when navigating outside of the row, which should lead to reliable lidar-based row-switching
    b. Fusing row detections with VO in an Kalman filter, rather than a simple fusion solution as we have currently, which would reduce the impact of false-positive row detections
    c. Evaluating the effectiveness of fusing row detections, VO, IMU, and the 6DOF RTK estimate in one filter
3. Improving the robustness of the plant health monitoring classification system
    a. Improving the quality of the plant health monitoring dataset by collecting more images with the presence of additional weeds and other nuisance plants with currently cause false positives
    b. Experimenting with more complex models for the semantic segmentation of the plant images, closer to state-of-the-art, such as Dilated ResNets or the fully convolutional DeepLabv3

Of particular note, two team members, John and Dung-Han, plan on pursuing the goals associated with lidar navigation next semester as part as an independent study. They are focused in particular on applying deep-learning based semantic segmentation techniques to the lidar row segmentation problem. By converting the lidar point cloud into an image-like spherical projection view, models such as the U-Net model used for plant health monitoring currently can be used for lidar segmentation and ultimately row detection and classification.

10 References

[1] T. Mueller-Sim, M. Jenkins, J. Abel and G. Kantor, "The Robotanist: A ground-based agricultural robot for high-throughput crop phenotyping," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, 2017, pp. 3634-3639.
doi: 10.1109/ICRA.2017.7989418

[2] O. Ronneberger P. Fischer T. Brox "U-net: Convolutional networks for biomedical image segmentation" Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent. (MICCAI) pp. 234-241 Nov. 2015.

[3] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, 2017, pp. 2980-2988.
doi: 10.1109/ICCV.2017.322

[4] D.Caruso, J. Engel and D. Cremers, "Large-scale direct SLAM for omnidirectional cameras," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, 2015, pp. 141-148

[5] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," in *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147-1163, Oct. 2015. doi: 10.1109/TRO.2015.2463671

[6] J. Zhang S. Singh "LOAM: Lidar odometry and mapping in real-time" Robotics: Science and Systems Conference (RSS) July 2014.

[7] M. Labbé and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based SLAM," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, 2014, pp. 2661-2666.

[8] S. Thrun, W. Burgard and D. Fox, *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2004, pp. 109- 111.