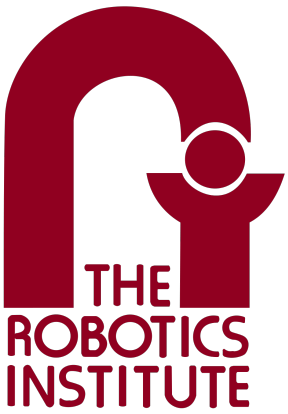
Carnegie Mellon University
The Robotics Institute, School of Computer Science

Conceptual Design Review

Team G
Aditya Gaddipati | Bruce KwangKyun Kim | Pallavi Madhukar |
Steven Brills | Vaidehi Patil

Sponsor: Dr. Oliver Kroemer

May 7, 2022



Abstract
The past few years have seen an increase in the number of indoor vertical farms bolstered by advances in lighting, hydroponics, control and automation. However, many core activities such as harvesting and pollination are still labor intensive because traditional machines like tractors cannot be used in vertical farming. In this report, Bob the Farmer system is presented as a robust alternative to tackle multiple farm activities like pollination and harvesting. The system is based on the Stretch RE1 platform and can navigate through greenhouse environment, perceive tomato flowers, fruits and pollinate, harvest them respectively using custom tools. Each of the subsystems are evaluated and results of unit and integrated testing are presented.

Table of Contents

1 Project Description

The past few years have seen an increase in the number of indoor vertical farms bolstered by advances in LED lighting, hydroponics, control, and automation. However, many of the core activities such as harvesting, and pollination still require intensive labor as traditional machines like tractors cannot be used in vertical farming. In fact, on average vertical farmers spend around $1,200 per acre per day on just manual picking. Alternatives such as bumblebee pollinators need time and care for the bees to acclimatize to the environment. Combined with the recent labor shortage, there is a need for a robust automation solution that can handle multiple vertical farm activities.

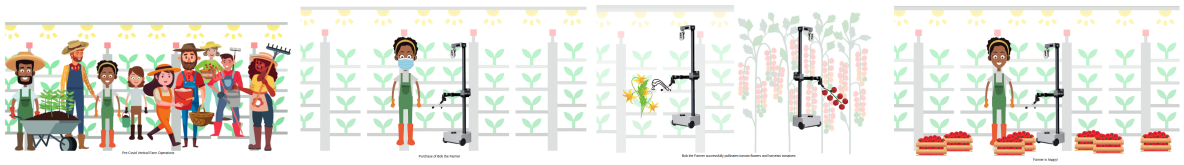
Our goal is to develop a mobile manipulation robotic system based on the Hello Robot Stretch RE1 platform that can pollinate and harvest tomatoes on indoor farms. Our robotic system will make use of custom-designed tools, autonomously navigate around the controlled setup environment that mimics a vertical farm, identify the interaction sites where the task must be performed, and perform the desired action. These tasks will be performed with minimal human interaction except for the option of a manual override to circumvent unexpected risks.

2 Use Case

Tina owns an indoor vertical farm, where she grows cherry tomatoes. A lot of work is involved for seeding, pruning, pollinating, harvesting and maintenance- all year round as seen in Fig. 1(a). As the pandemic hit, Tina's labor force was cut down drastically. This was when Tina decided to buy BTFv7, an autonomous robot specially designed for indoor farming, as seen in Fig. 1(b).

The BTFv7 was deployed and extensively took over the harvesting and pollinating process, as seen in Fig. 1(c). For the pollination process, the robot started at a fixed position, scanned the row, and identified the tomato flowers. It then localized points on the flower stem, navigated towards the point, and manipulated the end effector (vibrator) towards it. The vibrator contacted the flower stem and successfully dispersed pollen. After successful pollination, tomatoes started growing on the farm.

During harvest season, the robot was fitted with the harvest tool. It started at the given fixed position, scanned the row, identified clusters of tomatoes, and grouped them in columns. Points generated from these columns were used for navigation. The robot then localized contact points on tomato stems and harvesting was done at these points. Harvesting action included both cutting and gripping the cut bunch of tomatoes. The harvested bunch of tomatoes were dropped below (in the cart) by the robot. No tomatoes were harmed in the process. Tina is happy with the year's harvest and realizes that BTFv7 was a great investment, as seen in Fig. 1(d).



(a) (b) (c) (d)

Figure 1: Depiction of Use Case

3 System Requirements

The high-level objectives of our system are given in Fig. 2. The objectives were formulated based on:

1. The problem statement and the end goal
2. Conversations with the project sponsor and the program advisors
3. Inter-team discussions
4. Considerations of the robot specifications
5. Considerations of the time and budget constraints

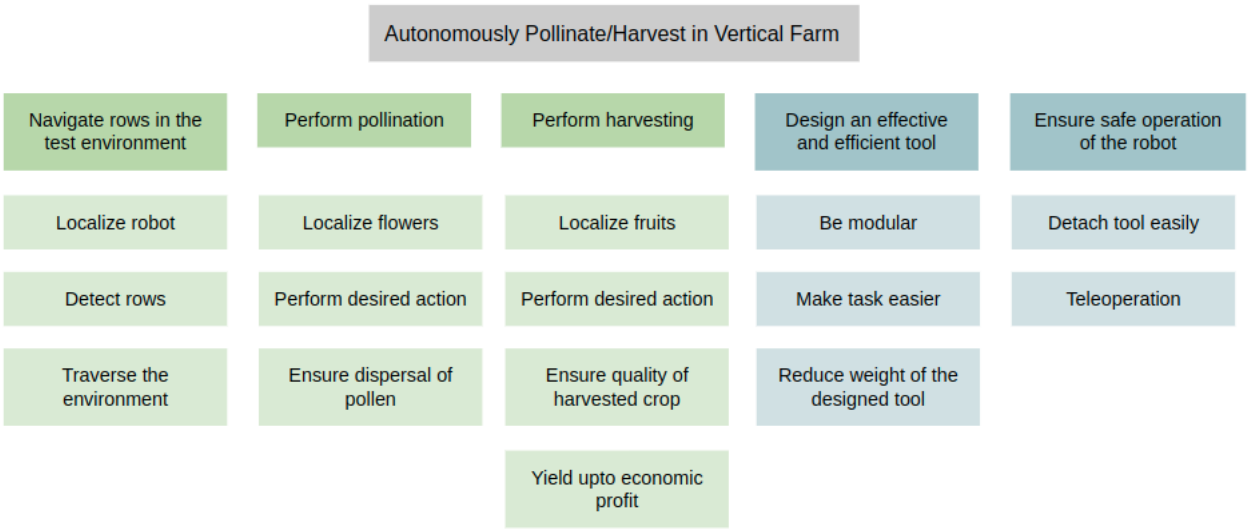


Figure 2: Objective Tree

The system requirements were formulated based on these objectives. The functional requirements(Appendix A.) were developed from the first three objectives. When developing the requirements, the process flow was considered along with the subsystems. The parameters for navigation-based performance requirements and manipulation-based performance requirements were decided based on the environment setup and robot specifications. The parameters for perception-based performance requirements were decided based on current research and industry standards [1, 2]. The non-functional requirements that qualitatively evaluate the system revolve around the last two objectives. The focus was on tool design and the safe operation of the robot. Other non-functional requirements include considerations of the robot's specifications.

3.1 Mandatory System Level Requirements

3.1.1 Mandatory Performance Requirements

The system will

M.P.1. Perform collision avoidance by maintaining a minimum 10 cm from rows
M.P.2. Have a tomato flower detection model with 50% mAP
M.P.3. Identify tomato flower clusters with 70% success rate
M.P.4. Localize flower clusters to within 5 cm (Euclidean) distance
M.P.5. Have a tomato fruit detection model with 50% mAP
M.P.6. Identify tomato fruit clusters with 70% success rate

M.P.7. Localize fruit clusters to within 5 cm (Euclidean) distance
M.P.8. Actuate bot such that tool is within 5 cm (Euclidean distance) of localization point
M.P.9. Cut tomato bunches in less than 2 passes

3.1.2 Mandatory Non-functional Requirements

The system will
M.N.1. Have a maximum payload capacity of 0.5kg
M.N.2. Ensure safe operation by enabling tool to be detached mechanically
M.N.3. Ensure safe operation by enabling manual override via teleoperation

3.2 Desirable System Level Requirements

3.2.1 Desirable Performance Requirements

The system will
D.P.1. The robot must successfully change rows when at the end of one row
D.P.2. Identify tomato flower clusters with less than 30% false positives
D.P.3. Preserve flowers 60% of the time
D.P.4. Identify tomato fruit clusters with less than 30% false positives
D.P.5. Preserve the quality of tomatoes with a maximum damage of 10%

3.2.2 Desirable Non-functional Requirements

The system will
D.N.1. Be protected from debris like leaves, pollen, and dust
D.N.2. Be simple to operate
D.N.3. Have a modularized tool subsystem such that the different tools can be fitted easily and in the same way
D.N.4. Have sufficient battery life of 30 minutes at maximum payload
D.N.5. Work in a partial occlusion setting

M.P.2. and M.P.5. were added, and M.P.3. and M.P.6. were modified to differentiate between identification of individual flowers/fruits (i.e. model performance) and the identification of clusters of the same (i.e. overall detection performance).
The difference between these performance indicators is evident when we consider the following cases:
1. When a flower in a cluster is not detected but other flowers have been detected, hence the cluster is detected. In this case, the model performance is affected but the detection performance is not.
2. When all flowers in two clusters have been detected, but they are identified as a single cluster. In this case, the model performance is not affected but the detection performance is.
The model performance is evaluated by mAP. The detection performance is evaluated by a success rate which indicates how many clusters were identified of the total number of existing clusters.

Initially, M.P.7. was thought to be sufficient as a criterion towards both perception and manipulation subsystems. However, this was not the case, especially since the actuation of the bot has been faulty. As a result, a separation was made - M.P.7 is the localization of the clusters by the perception subsystem and M.P.8. is the actuation of the bot by the manipulation subsystem to reach this localization point.

D.P.2. and D.P.4. were added as an additional criterion apart from M.P.3. and M.P.6. During experimental runs, it was observed that though the success rate of identification of clusters was above the requirement, false positives existed. These are not considered in the calculation of the success rate but contribute to the success of the entire system. It is detrimental to the system if it decides to stop at a location that is a false positive and attempts the action (pollination/harvesting) there. Hence it was added as a requirement.

4 Functional Architecture

The Functional architecture as seen in Fig 3. consists of 4 main subsystems-

1. Tomato Detection and Localization subsystem's function is to detect and locate the tomato clusters in the environment and also to detect the exact point on the stem where the robot needs to cut.
2. Row Navigation subsystem's function is to make the robot navigate in between the rows of the farm without colliding, stop at the desired locations and switch rows at the end of rows.
3. Manipulation subsystem's function is to move the arm to the detected cluster location and actuate the tool at the detected cutting point on the stem.
4. Behaviour Planner's function is to switch between the navigation and manipulation tasks.

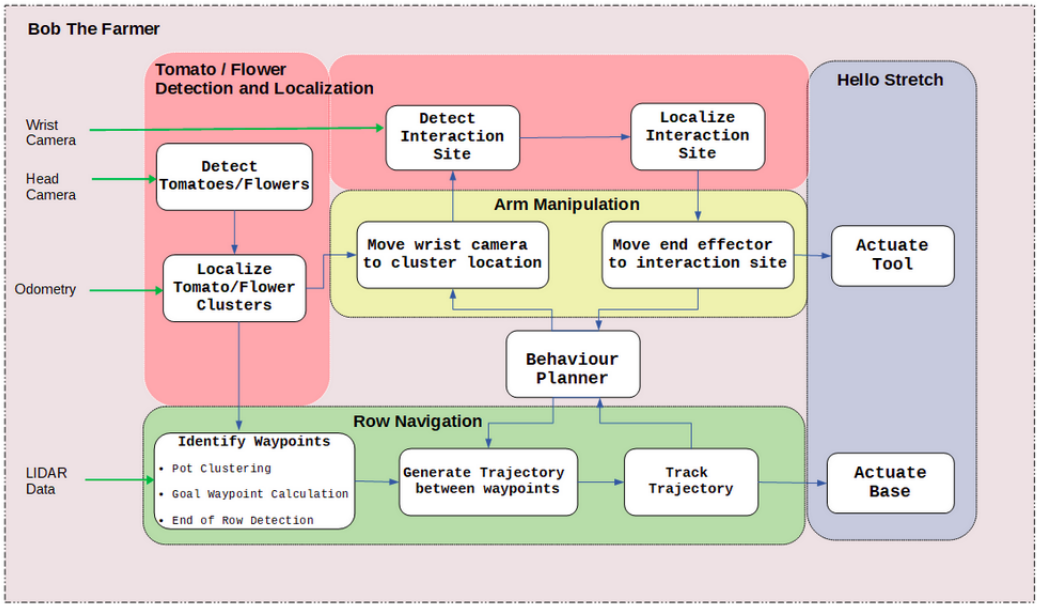


Figure 3: Functional Architecture

5 Cyberphysical Architecture

The Cyberphysical architecture is depicted in Fig 4. The Perception system detects the tomato clusters using the robot's head camera. It inputs the RGBD images and outputs the cluster locations. The navigation block takes the laser data as input, detects pots in the rows, fits a line through them, tracks waypoints parallel to the row and switches rows when end of row is detected. The navigation system subscribes to the detected cluster locations and stops the robot at each cluster. The manipulation system calculates the exact joint values to position the wrist camera in front of the detected cluster. Then using the wrist camera the exact location on the stem where the robot needs to harvest is detected. The manipulation system calculates the joint values to move the end-effector to the cutting location and actuates the tool.

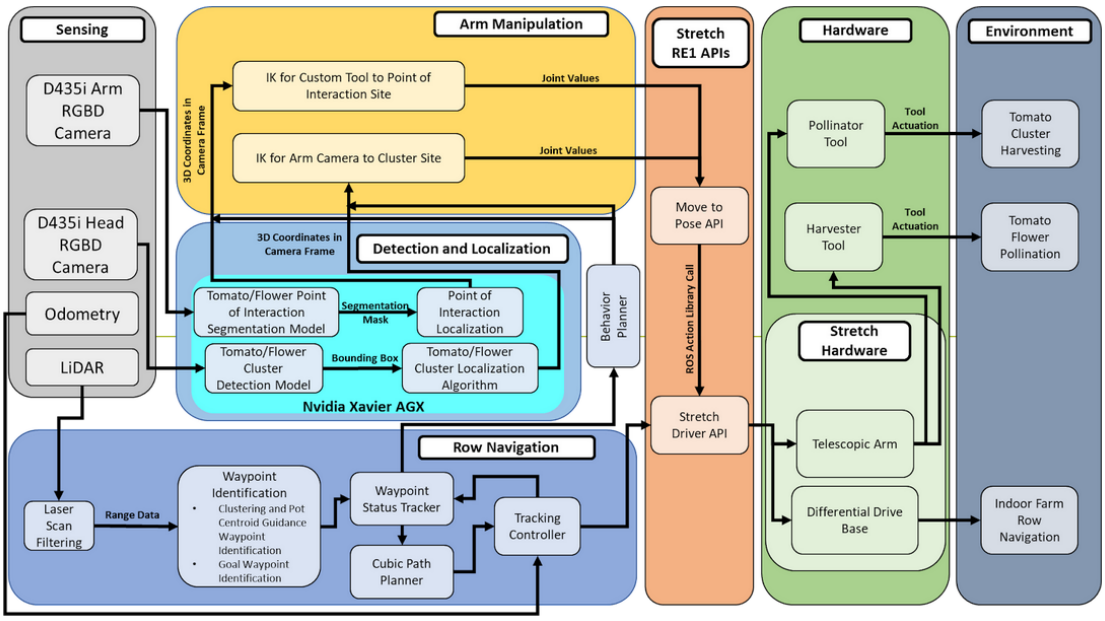


Figure 4: Cyberphysical Architecture

6 Current System Status

6.1 Spring Semester Targeted System Requirements

The system requirements and corresponding subsystems and system elements (Subsystem: Elements) were identified:

6.1.1 Mandatory Performance Requirements

The system will
M.P.1. (Navigation: LiDar pot detection, Static rows)
Perform collision avoidance by maintaining a minimum 15 cm from rows
M.P.3. (Perception: Detection model, Clustering, Dynamic environment)
Identify tomato flower clusters with 60% success rate
M.P.4. (Perception: Detection model, Clustering, Localization, Dynamic environment)
Localize flower clusters to within 10 cm (Euclidean) distance
M.P.6. (Perception: Detection model, Clustering, Dynamic environment)
Identify tomato fruit clusters with 60% success rate
M.P.7. (Perception: Detection model, Clustering, Localization, Dynamic environment)
Localize fruit clusters to within 10 cm (Euclidean) distance
M.P.9. (Customized tool: Harvester tool actuation)
Cut tomato bunches in less than 2 passes

6.1.2 Mandatory Non-functional Requirements

The system will
M.N.1. (Customized tool: Harvester tool, Pollinator tool)
Have a maximum payload capacity of 0.5kg
M.N.2. (Customized tool: Harvester tool, Pollinator tool)
Ensure safe operation by enabling tool to be detached mechanically
M.N.3. (Integration: Behavior planner, Teleoperation)
Ensure safe operation by enabling manual override via teleoperation

6.1.3 Desirable Performance Requirements

The system will
D.P.2. (Integration: Localization, Actuation, Pollinator tool actuation)
Preserve flowers 60% of the time
D.P.4. (Integration: Localization, Actuation, Pollinator tool actuation)
Preserve the quality of tomatoes with a maximum damage of 10%

6.1.4 Desirable Non-functional Requirements

The system will
D.N.3. (Customized tool: Harvester tool, Pollinator tool)
Have a modularized tool subsystem such that the different tools can be fitted easily and in the same way

6.2 Overall System Depiction

The robotic platform used is The Hello Robot Stretch RE1. The various subsystems - perception, manipulation, navigation, and customized tool - work on different parts of the robot as seen in Fig. 5. The perception subsystem makes use of the cameras on the head and wrist. The manipulation subsystem makes use of the lift and arm. The customized tool subsystem makes use of the wrist and tools. The robot comes with a default Stretch Compliant Gripper. This gripper was replaced with customized tools that can perform the desired action i.e., pollinating or harvesting. The navigation subsystem makes use of the LiDAR and base.

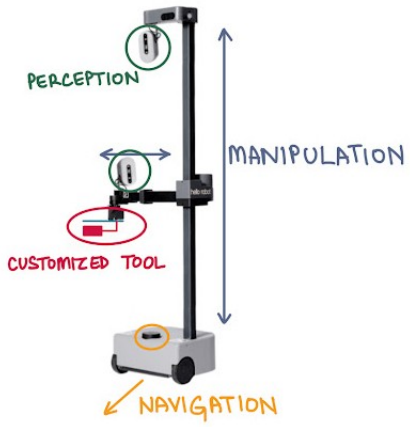
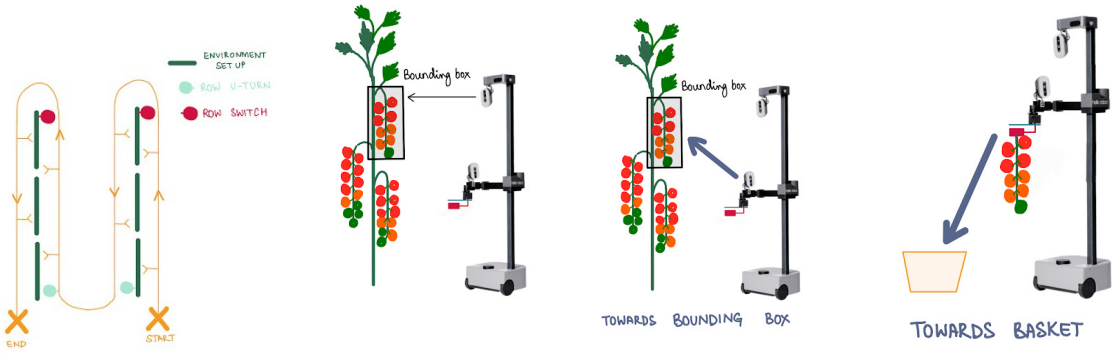


Figure 5: Robot System

The navigation subsystem enables the robot to traverse the rows of the environment in a path as seen in Fig. 6(a). While the robot is traversing the environment, the perception subsystem is detecting and localizing tomato fruit/flower clusters as seen in Fig. 6(b). The robot stops at a cluster location and actuates its lift and arm towards it such that the tool is in its vicinity as seen in Fig. 6(c). The tool is then actuated at this location - the harvester grips and cuts the tomato bunch ; the pollinator vibrates near the flowers. If the task is harvesting, the robot then drops the bunch near the ground as seen in Fig. 6(d). If the task is pollination, the robot retracts its arm.



(a) (b) (c) (d)

Figure 6: System Depiction

6.3 Subsystem Descriptions

6.3.1 Navigation

The navigation subsystem consists of four major elements that work hand-in-hand to ensure successful navigation around the farm environment. The details of each system are explained below:

1. **LiDAR based Pot Detection System:**

   Fig. 7 shows the overall flowchart of the LiDAR-based pot detection system. The basic idea is to estimate the location of the pots in the environment so that the robot can track the row easily. The fundamental nature of the laser scan that is being exploited here to be able to detect the pots is that the points in the laser scan form arcs around the pots where the laser reflects off the surface. Armed with this knowledge, the laser scan is converted into an iterable point cloud type after which the points are iterated over. A "cluster" is formed by iterating over the points and grouping the points that fall within a certain minimum euclidean threshold between each other. Once this threshold is violated, the cluster formed so far is passed into the circle fitting function which returns a fitted circle described by a centroid location and a radius. A sanity check is performed to ensure that the radius falls within the expected radius of the pot in the environment. If the radius is too large or too small, the point is discarded as a spurious detection and the next cluster is formed. This is done until all the points in the scan have been iterated over.

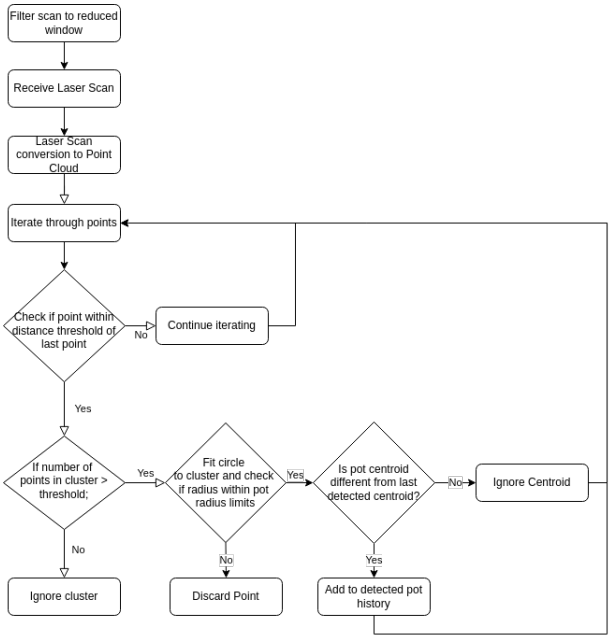


Figure 7: LiDAR Based Pot Detection Algorithm

Since there is an accumulating drift error in the odometry, the absolute location of the detected pots cannot be used for reliable tracking. Instead, it is far more reliable to use the instantaneous detection of the pots to track the parallel line to the rows. This is done by using the pot centroids that are immediately in front of the robot to fit a straight line. A line parallel to this straight line and offset by a safety distance is then created and published to the fourth navigation sub-system the Waypoint Publishing and Tracking to use as a guide while publishing way-points.

2. **End of Row Detection and Waypoint Generation:**

For the spring demonstration, it was decided that we use Aruco markers to identify the end of the row. This decision was made in order to endure accurate end of row detections without errors as it was a critical requirement in order to facilitate the other functions of the robot. Moreover, the environment inside a real greenhouse can be easily modified to accommodate such markers based on our conversation with greenhouse owners. There are two unique motions that need to be executed for row switching based on the orientation of the robot. The arm of the Hello Stretch RE1 faces towards the right of the base when the base is facing forwards. This implies that when traversing a row with plants on either side, the robot can only manipulate plants on the right and not on the left. In order to interact with plants on the left, the robot needs to reach the end of the given row and perform a u-turn within the same row. After the second pass of the robot within that row, now with its arm on the side of the row that it previously couldn't interact with, it can perform a row switch into the next row. These two cases, the u-turn and the actual row-switch are illustrated in more detail in Fig. 8.

Currently, the way the detection system works is by generating four waypoints around the end of the row based on the location of the detected ArUco marker. There are two different types of ArUco markers (two different ids) that indicate whether the robot is expected to preform a row switch or an in-row u-turn. Since the ArUco locations are much more accurate when up close to the marker, the generated waypoints are continuously updated as the robot approaches the ArUco markers until the robot gets sufficiently close after which the waypoints are frozen and executed by the robot.

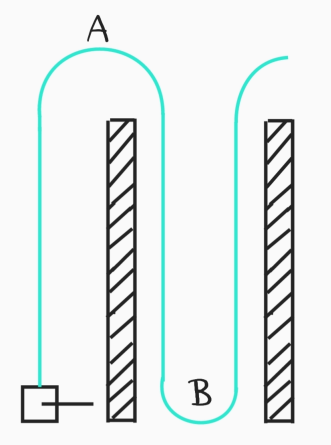


Figure 8: Row Turn Types: A. Row-Switching vs B. In-Row U-turn

Since the LiDAR needs to be able to "see" pots in order to track them, the fourth turn way-point is a deliberate blind forward motion that brings the robot sufficiently into the row to be able to "see" the pots. So far, such a blind motion has not shown an issue during operation despite operating on especially rough concrete floors.
Fig. 9 shows an RVIZ vizualization of the robot performing a turn around the end of the row with the way-points visible.

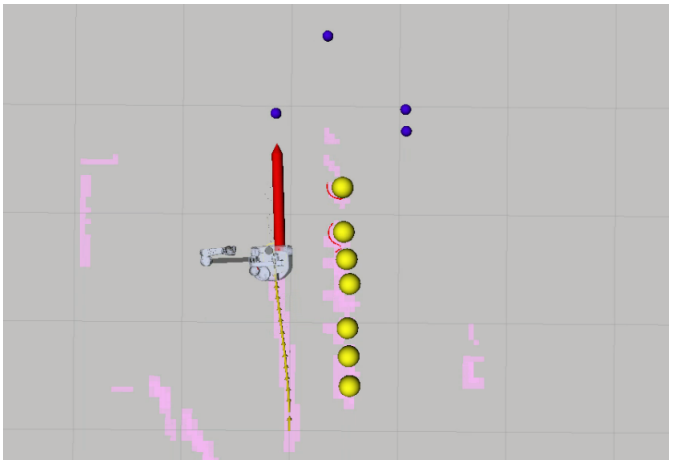


Figure 9: A Row Switching Operation as Seen on RVIZ

3. **Target Goal Sorting and Waypoint Generation:**

   Once a detection is received from perception, the detected goal is published to the Target Goal Sorting and Waypoint Generation system. Here, the detected cluster is projected onto the line parallel to the pots. Since at any given time the published detections repeats the complete history of detections, this system first checks whether the published detection has already been tracked. If a new detection is observed, this detection is projected onto the line parallel to the pots and the new projection is stored as a target goal position. The system also sorts the goals based on the x-value of the detection in the base-link frame of the robot. This is to ensure that a tomato cluster detected later but is in fact the closest to the robot is appropriately targeted first instead of in the order of the received detection. The target goals are then converted into the odometry frame which is fixed and subsequently stored in the form of a list.

This list is used to navigate to the target cluster locations when the robot is in the exploit mode.

4. **Way-point Publishing and Tracking:**

   The Waypoint Publishing and Tracking system executes commands based on the assigned mode which can be “explore” or “exploit”. In the “explore” mode, the robot receives the guide vector parallel to the pots but it does not listen for detections. The robot incrementally moves along the rows through what are called “exploratory” goals. Once the detection is received, the behavior planner shifts the navigation module into “exploit” mode where it tracks a particular goal.
   Since detections are updated in the background based on what is published by the perception sub-system, the Waypoint Publishing and Tracking system publish the goal closest to the robot first. Once this is done, the goal is removed from the list of detections to be tracked and added to the history of detections where it is used to remove duplicates.

The overall high-level system diagram can be seen in Fig. 10. Some subsystems of navigation have been deliberately merged together for simplicity.

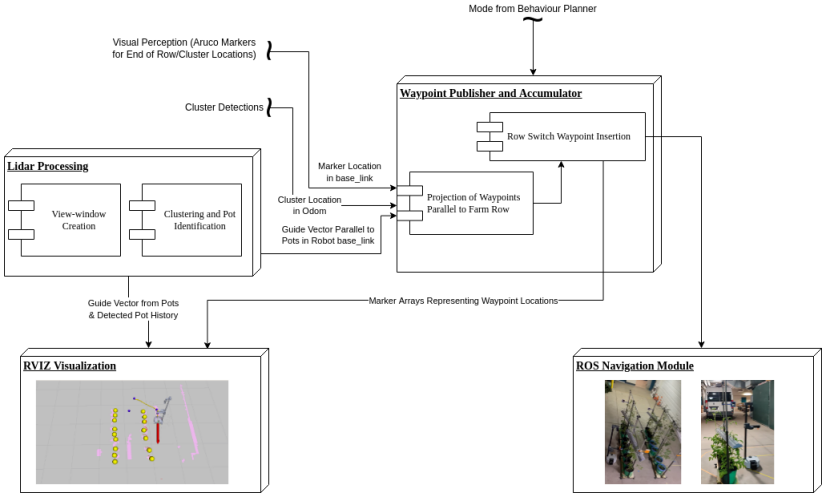


Figure 10: High-level Operation of Navigation Module

6.3.2 Perception

In the spring semester, development in the perception subsystem was focused on the following:

1. **Object Detection for Detecting Tomatoes and Flowers in the Environment**
   PyTorch’s Faster RCNN model pre-trained on the MS COCO dataset was fine-tuned on custom tomato and flower datasets similar to greenhouse environments. Fig. 11 and Fig. 12 show samples from the datasets used for training the model. The tomato dataset consists of 893 labeled images and the flower dataset consists of 745 labeled images.
   a. Kaggle Dataset

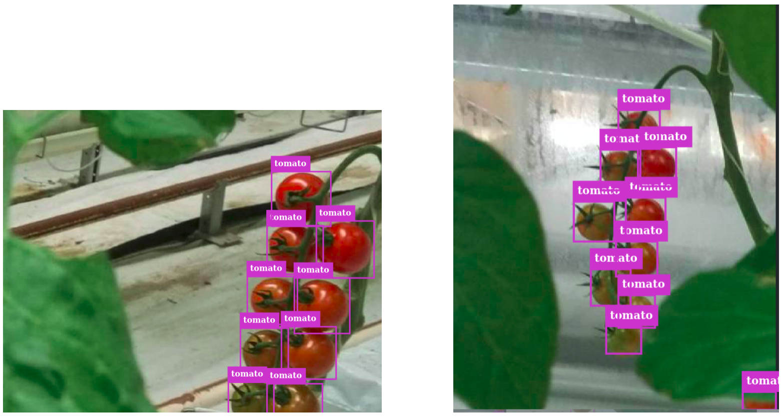


Figure 11: Kaggle Tomato Dataset

b. Tomato Flower Dataset [1]

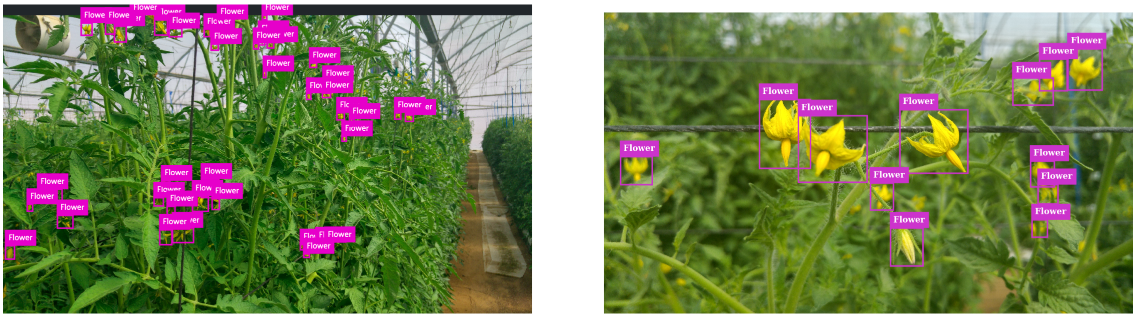


Figure 12: Tomato Flower Dataset

Two separate object detection models were trained, one for tomatoes and another for flowers. Since each model has only one object class to detect (tomato or flower), the last layer of the pre-trained model was modified to output class scores for only one class plus background. The tomato detection model was trained for 50 epochs and achieved 0.53 MAP during training. The flower detection model was trained for 10 epochs and achieved 0.33 MAP during training. The MAP was calculated on a test set that was from the same distribution as the training dataset. The detection model was deployed onto Jetson Xavier mounted on the Hello Robot and ran inference using the RGB images from the head camera. The resulting inference rate on Jetson is 2 frames per second. Fig. 13 shows the training metrics for tomato and flower detection model respectively.

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.537
 Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=100 ] = 0.917
 Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=100 ] = 0.544
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.421
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.588
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.676
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.121
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.487
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.608
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.517
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.657
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.726
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.335
 Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=100 ] = 0.742
 Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=100 ] = 0.234
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.006
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.174
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.393
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.037
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.238
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.427
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.011
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.274
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.487
```

Figure 13: Training Metrics

2. **Identifying the Clusters and Locating them in the Global Frame**
   The detections received from the model place bounding boxes around individual tomatoes. The bounding boxes from the detection model are converted into one bounding box per cluster. This is done by finding overlapping boxes and considering them as belonging to one cluster. After finding the clusters in the image, the centroid of each bounding box within the the cluster is calculated. In order to reject false positives, the calculated centroid is looked-up in the red HSV mask of the image to ensure that the selected point belongs to a tomato. The Z coordinate of the tomato is obtained by sampling at the centroid location in the depth image. The X and Y coordinates are then calculated using the image pixel coordinates and the camera intrinsics. Fig. 14 shows the cluster's bounding box in green and model's detections in blue. Fig. 15 shows the RGB image and the corresponding red HSV mask.

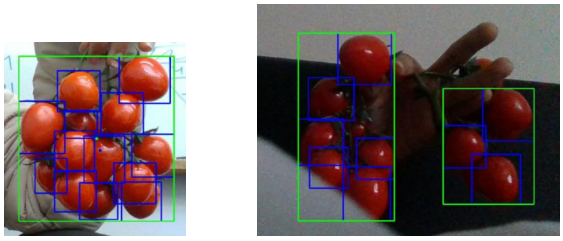


Figure 14: Bounding Boxes

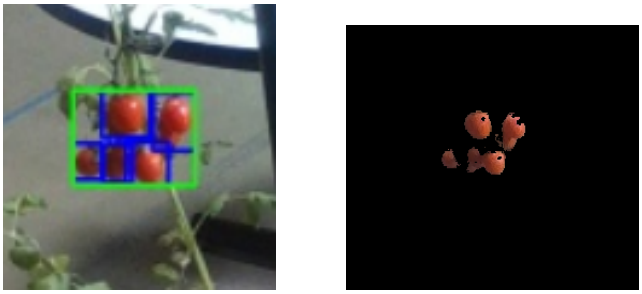


Figure 15: RGB Image and red HSV Mask

6.3.3 Manipulation

In the Spring Validation Demonstration, the manipulation subsystem actuates the end-effector to reach a point of interaction, which is an offset distance from the centroid of the localized cluster. The following are the works developed to meet the performance requirement.

1. **Inverse Kinematics Algorithm**

   It is imperative to first understand Hello Robot's Stretch RE1's joint configuration to design an inverse kinematics algorithm. Stretch has one linear joint that extends and retracts a horizontal telescoping arm, and a vertical mast that linearly lifts the telescoping arm up and down. These two linear joints create a Cartesian configuration of the robot when the robot is stationary. At the end of its arm, it has a wrist with a yaw rotational joint and the wrist can add roll and/or pitch with additional motors. The mobile base has a differential drive that translates or rotates the base. In the Spring semester, the assumption is that the robot is stationary where the arm points toward the target object in a normal direction, and the arm moves up/down and back/forth within the given Cartesian configuration space.
   Assuming a target point of interaction is given, it is possible to calculate the distance between the target point and the origin frame, which is the base link frame of the robot. However, because the robot's arm cannot be fully retracted and lowered to the base link frame, which is located underneath the robot's base, an additional step to calculate the distance between the origin of the base link frame and the origin of the first telescopic link at its most retracted and lowered status is needed. Using ROS TF, the offset distance between the two frames are calculated.
   ROS TF also provides the distance between the origin of the base link frame and the target point. As previously mentioned, Stretch has linear joints, implying that the required joint values are equivalent to the distance. Thus, by subtracting the calculated offset distance from the required joint values, the final joint values are returned. These are inputted to move_to_pose function provided from Stretch ROS package to locate the arm to the target point as shown in Fig. 16.

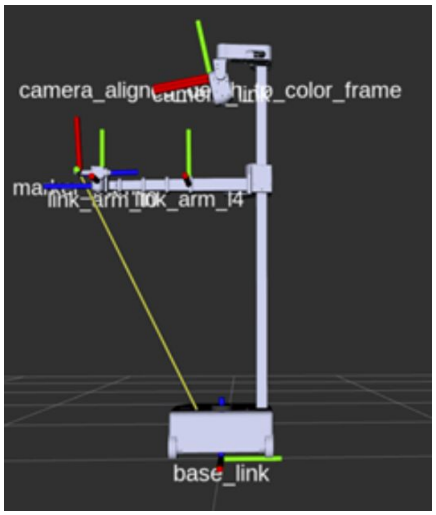


Figure 16: Coordinate Transformation for Inverse Kinematics

2. **Subscription to Point of Interaction**

   The manipulation subsystem subscribes to the point of interaction, which the perception subsystem first publishes to the navigation subsystem, and the navigation publishes to the

manipulation. The point of interaction provided in the Spring semester is the centroid of the bounding box of a tomato cluster and tomato flower cluster with respect to the odom frame as seen in Fig. 17 (a). Prior to the system integration, the manipulation system temporarily subscribed to ArUco marker for testing and integration as shown in Fig. 17 (b).

(a) (b)

Figure 17: Subscription to Point of Interaction

3. **Direct Offset Calibration**

   After the basic function of the inverse kinematics was complete, the robot still required some calibration on its positional accuracy as shown in Fig. 18.

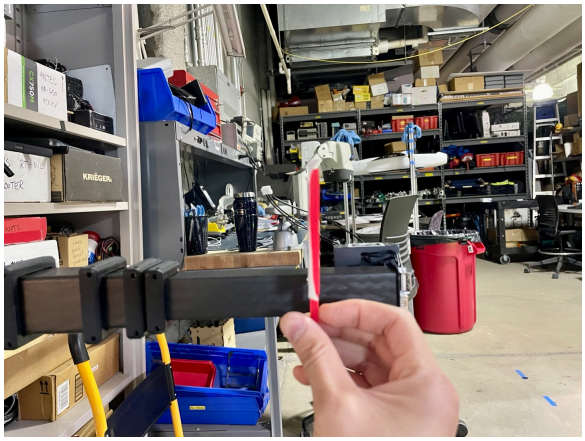


Figure 18: Positional Inaccuracy before Direct Calibration

The arm still extended a little further and lower than where it should be. After adding/subtracting a certain offset distance additionally, the arm was able to reach out to the desired position.

6.3.4 Customized tools

The customized tool is an end-effector to the hello robot to perform multiple farm tasks such as harvesting and pollination. There are two customized tools – namely the harvester and the pollination. They are described below in detail.

1. **Harvester**

   The aim of the harvester tool is to harvest cherry tomato bunches in a greenhouse setting. It is assumed that the tomatoes plants are pruned, and tomatoes are not occluded (this is the case in most greenhouses). The harvester is made of two parts – a gripper and a

cutter. The gripper should be able to hold weight of up to 500 gm without dropping it. The cutter should be able to cut the tomato stem, right above the bunch in less than two passes. Functioning of the harvester is shown in Fig. 19.

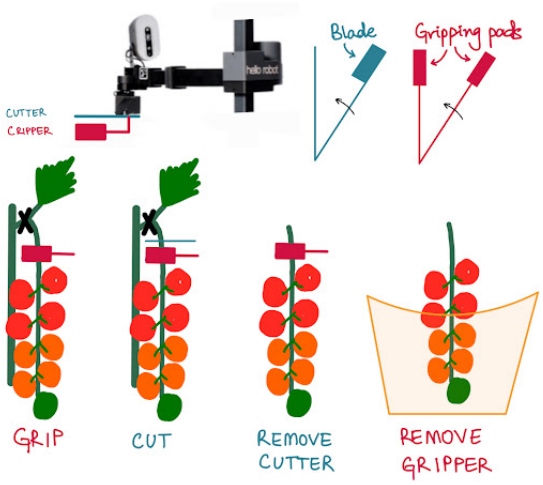


Figure 19: Harvesting Depiction

- Structure
  The cutter and the gripper are both powered by Dynamixel XL430-W250-T motors. They are situated in such a way that the axis of rotation for the cutter and the gripper are the same. Both the cutter and the gripper have a stationary part while the moving parts are connected to the motor shaft. The moving part of the cutter has a sharp blade attached to it which can cut through the tomato stem. The total length of the tool is 180 mm long and it has a 60-degree cutting span and 100-degree gripping span. The motor mounts and other parts of the tool are 3D printed using PLA. Extra supports are provided to the stationary parts of the tool as they must withstand the stall torque of the motor. The structure of the harvester can be seen in Fig. 20.

Figure 20: Harvester Tool Structure

- Attachment to the robot
  The harvester is attached to the wrist yaw of the robot. The yaw motion of the wrist is not used to control the harvester. The harvester is attached via a connector that is common to both tools. This allows for easy and accessible tool change. The connector is shown in both Fig. 20 and Fig. 22. It is 3D printed using PLA.

- Working
  Once the POI is obtained, the arm with the tool attached reaches the POI. Here, both the cutter and the gripper are in an open position, that is, the gripper is open by 100 degrees and the cutter is open by 60 degrees. At the correct POI, the stem is between the gripper and the cutter, this is the ideal cut position. The gripper first closes, gripping the stem between the gripping pads. Here, the torque of the motor is not disabled, and the motor applies a torque of 1.5 Nm. Once the gripper holds the stem, the cutter closes by a swinging action. This constitutes one pass of the cutting action. The cutter opens and closes again (second pass) ensuing that the stem has been cut. Once this motion is completed, the arm retracts with the gripper still holding the cut bunch of tomatoes. The lift of the arm is lowered, and the gripper is then opened at a lower height so that no harm comes to the tomatoes.

2. **Pollinator**
   The main objective of the pollinator tool is to perform the pollination of tomato flowers which is usually done using bumble bees or manually, using a brush-like tool. Successful pollination consists of visible shaking of the flower, or the stem attached to it. Minimal contact pressure has to be applied during this process, ensuring no harm comes to the flower so that the fruits produced are of good quality. Pollination action is depicted in Fig. 21.

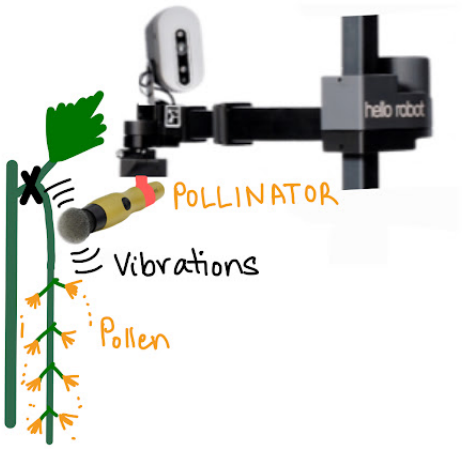
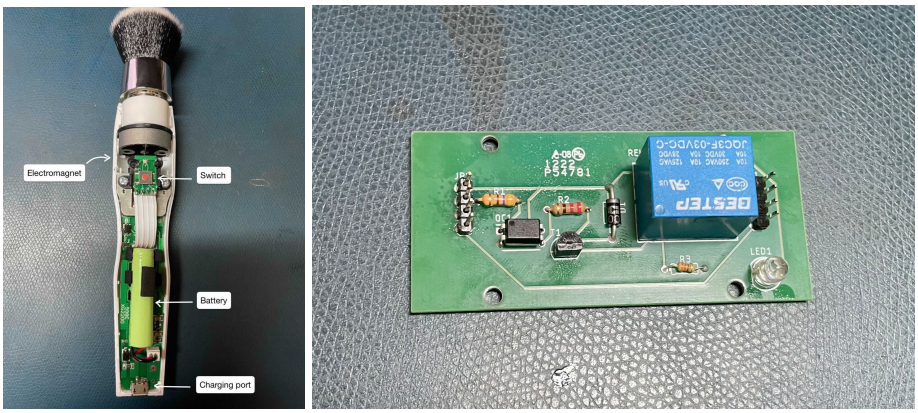


Figure 21: Pollination Depiction

- Structure
  The pollinator tool consists of a flat head vibrating brush. The diameter of the end of the brush is 40 mm. The pollinator has a custom mount which is 3D printed using PLA material. The mount is attached to the wrist yaw link of the robot using a connector. This connector is common to both tools. The pollinator tool structure can be seen in Fig. 22.

Figure 22: Pollinator Tool

- Working
  Once the POI for pollination is obtained, the arm with the end effector reached the POI such that the bristles of the pollinator touch the flower. The pollinator is switched on and vibrating, the vibrations cause the pollen in the flowers to fall out. Wrist yaw of the robot is used to rotate the entire tool by 30 degrees on either side of the initial position. This ensures that, even if the initial position did not have contact with the flowers, the coverage due to the 60-degree motion would shake the flowers effectively. Once the motion is completed, the arm is retracted to its initial position.
- PCB
  The pollinator comprises of an electromagnetic coil that is controlled by a switch as seen in Fig. 23 (a). If the switch was used, the pollinator would have to be switched on for the entire duration of the operation. Apart from being a waste of energy, unnecessary vibrations will also harm the plant. A PCB was designed to migrate from control via the switch to control by our robot as shown in Fig. 23 (b). In this way, the pollinator is switched on only when needed. The schematic and board are in Appendix B.



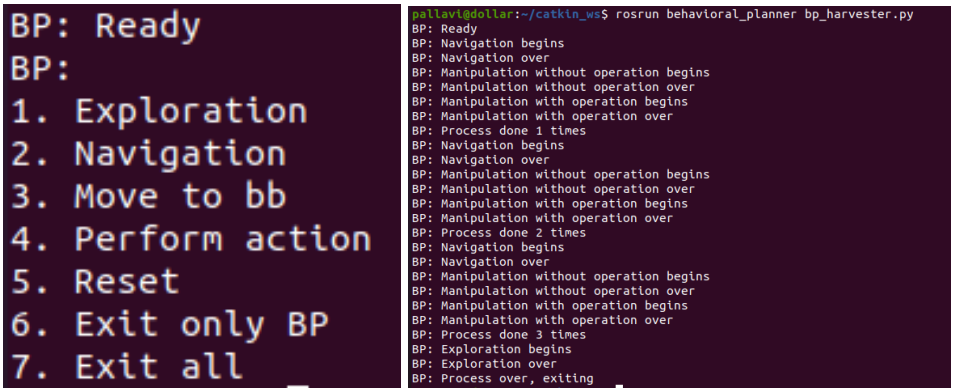
(a) (b)

Figure 23: (a) Pollinator Disassembled (b) PCB Designed for the Pollinator

6.3.5 Integration

Integration was done via a behavior planner. The behavior planner was constructed such that the robot can perform the action from start to finish i.e. traverse the setup, stop at the bounding box, perform the necessary actions, and continue. This was done via server-client services. As shown in Fig. 24 (a), a menu was created such that each subsystem could be controlled

independently for testing as well. The subsystems were modified by adding the server-client functions and operations. The entire pipeline is shown in Fig. 24 (b).



(a) (b)

Figure 24: (a) Menu for Ease of Control (b) Pipeline of System Operation

All the packages are launched - perception, navigation, and manipulation. The perception package detects and localizes clusters, then publishes the cluster locations. All cluster locations are continuously being published. The navigation package subscribes to the cluster locations. If there are no clusters, the robot continues to traverse the environment. When a cluster location is received, the robot navigates to it. If there are multiple clusters, the closest cluster is chosen. The navigation package also publishes this chosen cluster. The manipulation package actuates the arm and lift joints to the location and performs the required action. The behavior planner specifically controls and switches between the navigation and manipulation package depending on which actuation is needed to reach the cluster - the navigation package is first controlled, and once the base reaches the cluster, the manipulation package is controlled so that the arm and lift reach the cluster. The flow is summarized in Fig. 25.

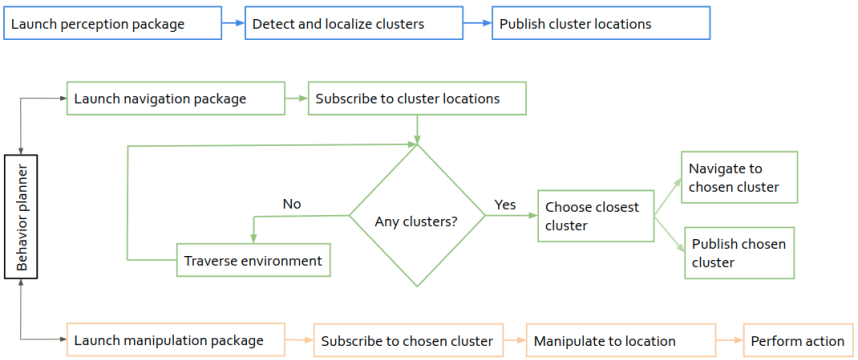


Figure 25: Integration via Behavior Planner

6.4 Modeling, Analysis, and Testing

6.4.1 Navigation

The overall performance was sufficient; however, there is a lot of room for improvement from a structural and software engineering perspective. The highlight of the system is the LiDAR-based Pot Detection sub-system. The algorithm works well in different situations. Fig.

26 shows the pot detection system in action and Fig. 27 shows the system after detecting all the pots in the environment. This is when the robot has been teleoperated and the odometry has not drifted by a large margin. The test environment consists of 16 pots and it can detect all 16 every time. The two main failure modes are when:

1. The environment has non-circular pots/grow bags.
2. The pots are black in color which does not reflect the laser well. See Fig.28.

Aside from these failure modes, a shortfall of the system but not an obstacle to operation is the drift in the odometry. When the odometry drift is too large the same pot when visited again at a different time may appear to be a new pot in the fixed odometry frame. This happens since the association between identical pots is evaluated by a Euclidean distance threshold. While this is not a neat implementation, the module being reactive continues to track the currently detected pots despite the inferred location of that pot in the odom frame being different.

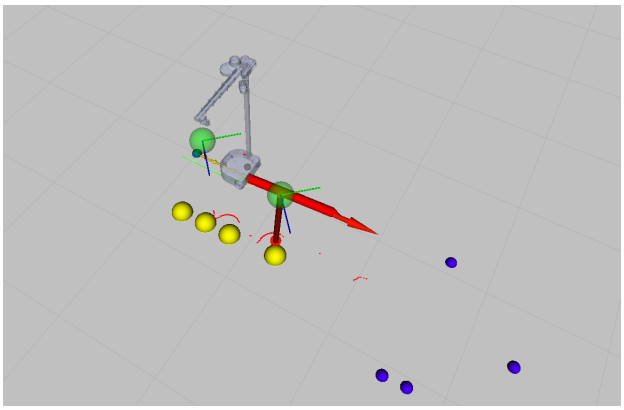


Figure 26: Pots Being Detected (Yellow Spheres) as the Robot Navigates Forward along a Row.

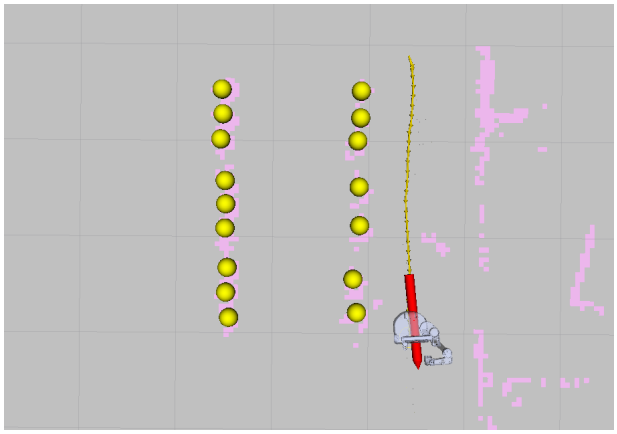
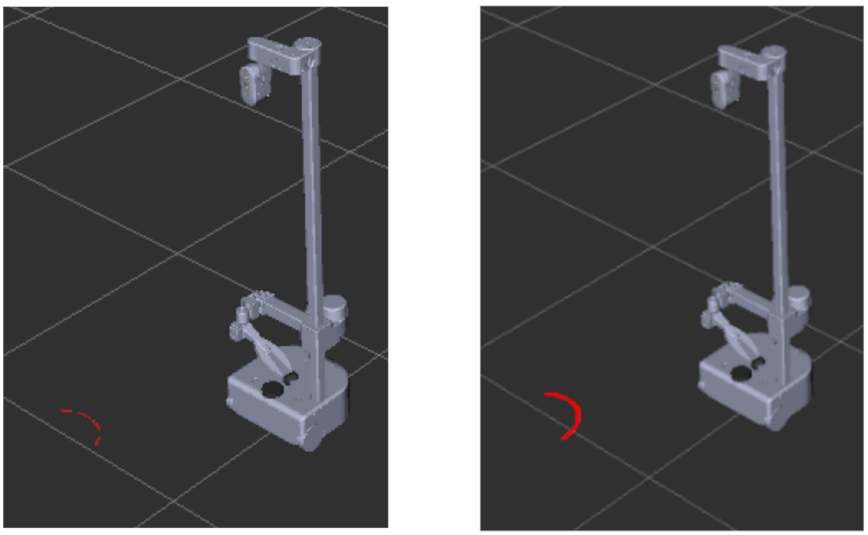


Figure 27: Detected history of Pots (Yellow Spheres) after complete Row Traversal (Tele-operated)



Initial Performance with Sparse LiDAR Points due to Black Pots

Performance showng Dense LiDAR Points after Repainting Pots

Figure 28: LiDAR Sensor Output from Two Different Coloured Pots (a) Black pots and (b) Green pots

Currently, the LiDAR data is utilized to generate the parallel line and maintain a safe distance from the pots. Aside from this, there is no active obstacle detection. All target goals are simply projected onto this line away from the plant rows. In practice, the robot can track this line within +-10 cm.

The current implementation of end-of-row detection and switching is robust. The end is marked by Aruco markers as explained earlier and the localization of these markers as well as the turn way-point generation with respect to the robot is accurate and reliable.

On the whole, the navigation system works but can benefit from modularization of the subsystems and better software engineering practices. Improvements in the odometry can also be useful for the comparison of measurements with ground truth. A local controller that avoids the implementation of move-base could also potentially help make the movements smoother rather than jerky.

6.4.2 Perception

For the perception system the main parameters that have to be tuned are as follows:
1. **confidence_threshold:** Minimum confidence score required to register a model prediction as tomato. This parameter helps remove false detections from the model.
2. **min_detections:** Minimum detections of the same point required to register as new cluster. This parameter helps in rejecting one-off cluster detections.
3. **max_depth:** Maximum depth at which a cluster can be registered. This parameter needs to be tuned based on the depth accuracy of the camera. Clusters locations that are far away will be inaccurate.
4. **cluster_threshold:** Minimum distance required between two clusters to be registered as different clusters.

In order to tune these parameters, the robot was tested in an environment with the ground truth positions of the cluster known. Based on the lighting conditions in the environment, con-

fidence_threshold was tuned just right to reject false detections while detecting all the tomatoes in the environment. min_detections and max_depth were tuned together. If max_depth is reduced, min_detections required is increased and vice-versa. There is a trade-off between cluster location accuracy and detection rate. If max_depth increases the location accuracy is reduced, hence more detections are required to register a cluster. cluster_threshold was tuned by continuously reducing the distance between two clusters in the environment until it gets detected as one cluster. Through testing, it was found that the clusters need to be at least 25 cm apart to get detected as unique clusters. Figure 30 shows the test environment setup used for unit-testing the perception subsystem.

Figure 29: Perception Unit Testing

6.4.3 Manipulation

The manipulation system is considered successful in the Spring Validation Demonstration if the arm reaches out within 10cm of the point of interaction. The subsystem testing was performed first, followed by the integrated system testing.

For the subsystem testing, the main test was to track and reach out to ArUco markers in different locations. After trial and error in getting the right offset direct calibration, the robot's arm was successfully positioned at the desired locations as shown in Fig. 30

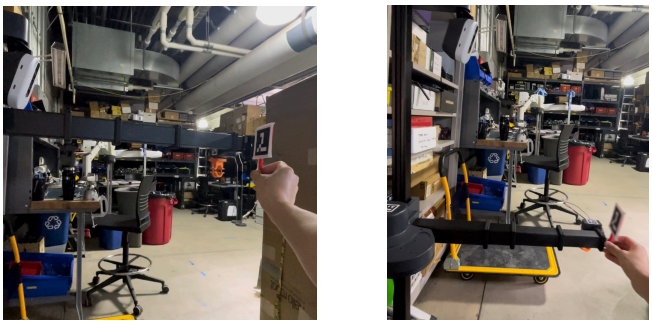


Figure 30: Manipulation Subsystem Testing

For the integrated system testing, however, it became clear that the base orientation adjustment is required in order to accurately locate the end-effector at the point of interaction. In the Spring Validation Demonstration, the team focused on getting the robot arm to the POI within 10cm. This was also achieved after going through trials and errors of calibration. The results are that the end-effector was located within 5cm as shown in Fig 31

Figure 31: Manipulation Integrated System Testing

6.4.4 Customized tool

1. Harvester tool

   Successful harvesting is termed when the tomato bunch (up to 500 gm) in weight is gripped efficiently and is cut from the parent stem in 2 or less passes. Fig. 32 shows the variation of weight and stem thickness for successful cuts. The current success rate of tomato cutting is 80 %. Fail cases occur when the cutter motor fails to provide enough torque for cutting or when the robot exhibits synchronous movements.

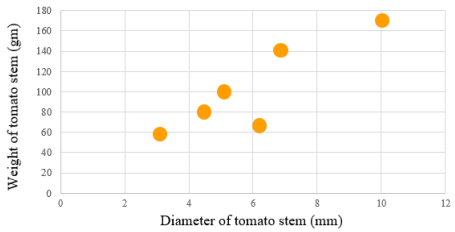


Figure 32: Graph Showing Successful Cuts of Tomato Stems

   - Further improvements

     Further improvements to the design can include decreasing the distance between the cutter and the gripper to make the tool more compact. Incorporating feedback to make sure that the stem is cut is a necessary addition. Apart from this, the robot movements need to be asynchronous (only one process runs at a time) and high torque needs to be provided for cutting all time.

2. Pollinator tool
   - Successful pollination is termed as visible shaking of the flower or the stem due to contact with the brush. If the POI provided is correct, there is successful pollination for 90% times. Fail cases occur when the robot exhibits synchronous movements.
   - Further improvements

     A gripper can be included in the pollinator device, just about the brush such that the gripper can hold the stem of the flower while the brush pollinates the flowers.
3. PCB
   - The tool was successfully integrated with the PCB via Arduino control as shown in Fig. 33.

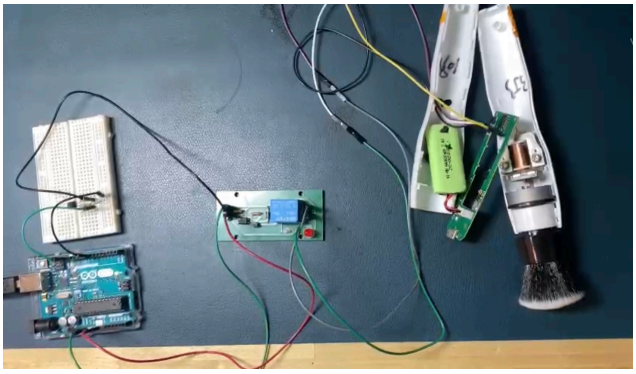


Figure 33: Pollinator Control via PCB and Arduino

- The wacc extension header was giving unpredictable outputs. By default, the input pins are pulled HIGH. However, one of the input pins was always pulled LOW instead. While the output pin control worked for the first command push, the next command pushes within the same code did not seem to have an effect. Following this, the wacc failed the system check. Since the wacc and the wrist are connected, the wrist did not home. The output from the system check is shown in Fig. 34. Following this, a replacement wacc had to be ordered and put in place. Besides causing a delay in schedule, a decision was made to not use this approach as the wacc is sensitive and the connected wrist is integral to tool operations.

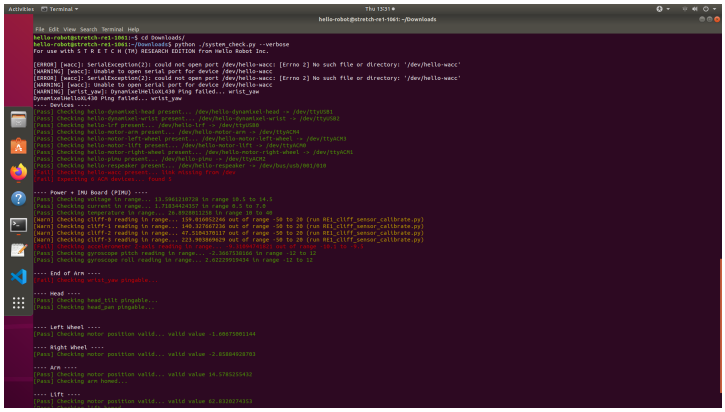


Figure 34: Wacc Expansion Header and Wrist Not Recognized during System Check

- Future improvement
  The pollinator PCB can alternatively be controlled by the USB hub on the wrist of the bot.

6.4.5 Integration

The architecture and structural planning of the behavior planner proved to be interesting. A dummy navigation and manipulation package was created to test whether the performance was working before integration with the actual packages. The subsystems were modified by adding the server-client functions and operations. The dummy packages were then substituted with the real packages. Apart from constructing the entire pipeline, each subsystem had to be looked at individually to ensure a clear definition and use of communication. Currently, there is sufficient communication between the behavior planner and subsystems. However, control is passed over to the subsystem until a particular task is finished. Ideally, control of the system should rest with the behavior planner entirely in case of need for emergency stops. This will

have to be changed in the future. Apart from this, the behavior planner does not keep a check on any of the subsystems, so even if it could intervene, the planner would not know when to. A heartbeat or watchdog timer must be added in the future.

Upon integration, the robot started to hang quite often. Apart from this, upon continuous usage of the bot, the motors do not run at full torque. This is a big risk because the robot then stops operating/displays unpredictable behavior and has to be completely restarted once this happens. This has to be looked into and fixed on priority. Each subsystem and its integration with other subsystems will have to be inspected further to find the cause of latency.

6.5 Performance Evaluation against Spring Validation Demonstration

The performance evaluation has been summarized as shown in Table 1.

During the SVD, the integrated testing of perception, navigation, manipulation, and customized tools was conducted to evaluate their performance. Starting with the perception test, the robot successfully localized all tomato and tomato flower clusters (M.P.3, M.P.5). On RVIZ, there seemed to be some false positives detected, but those were just the updated location of newly detected clusters and the old ones being accumulated. The perception was also successful during the SVD Encore.

The robot's navigation initially failed to avoid all the pots and collided with one of them during the SVD. After improving the system, the robot successfully navigated all the rows, conducted successful row switching and U-turn (D.P.1), and avoided collision (M.P.2) during the SVD Encore.

At the first row of the farm, the robot was commanded to stop at each identified tomato and flower cluster to conduct manipulation. For both SVD and SVD Encore, the end-effector successfully reached within 10cm (Euclidean) distance.

The customized tool system then operated the tools to pollinate flowers or harvest tomatoes. Some tomatoes were not cut in less than 2 passes because the stem was replaced with a thicker one during the test setup. This was an unfavorable condition as the stem was as thick as the main stem of the tomato plant. During the SVD Encore, the customized tool testing was performed in both tool subsystem testing and integrated system testing. At a hanger where a cluster of flowers and tomatoes were hung, the robot reached out its arm to operate the pollinator and harvester. In one of the trials in the subsystem testing, the harvester did not work completely because the robot hang. However, in the integrated testing, both tools successfully pollinated flowers and harvested tomatoes, while preserving flowers 60% of the time (D.P.4) and cutting the stem in two or less passes (D.P.6).

Overall, there were some room left for improvement during the SVD, but the SVD Encore was successful in meeting all the SVD Criteria.

Table 1: SVD Performance Evaluation Summary

Performance Requirements	SVD	SVD Encore
Will perform collision avoidance by maintaining minimum 15 cm from rows.	F	P
Will identify flowers with 60% success rate.	P	P
Clusters of flowers localized to within 10 cm (Euclidean) distance.	P	P
Will preserve flowers 60% of the time.	P	P
Will identify ripe tomato bunches with 60% success rate.	P	P
Clusters of tomatoes localized to within 10 cm (Euclidean) distance.	P	P
Will cut the tomato bunches in less than 2 passes.	F	P

6.6 Strong/Weak Points

6.6.1 Strong Points

1. Detection model's recall is high. That means it is able to detect all the tomatoes in the environment.
2. True positive rate for cluster detection is high (>60%). 5 out of 7 clusters get detected.
3. harvester tool grips and cuts efficiently at full motor torque.
4. The LiDAR-based pot detection accuracy is good which facilitates to accurate row traversal.
5. Navigation is reactive which allows for robust operation even if there is drift in the odometry.

6.6.2 Weak Points

1. Detection model's precision is low. That means the model predicts a lot of false detections leading to poor mAP.
2. Inference rate is very low (2 FPS on Jetson). This leads to poor cluster detection accuracy when the bot is in motion.
3. False positive rate for cluster detection is high (50%). 5 out of 10 detections are false. This is mainly due to bad odometry which is resulting in a previously detected cluster being registered as a new one.
4. Dropping motor commands cause synchronous movements of arm and lift, and disable motor torque.
5. Robot hangs due to overload.
6. The odometry is poor due to ridges in the concrete floor.

7 Project Management

7.1 Work Breakdown Structure

The system's work breakdown structure in Fig. 35 describes the top-down breakdown of the work in vertical farming automation to be performed. The partitioning of the work is depicted as four subsystems development, one system integration, and one project management in the second level. The breakdown of perception, customized tool, manipulation, and navigation subsystem have unit testing and integration in common, and work packages specific to each subsystem. Systems integration includes environment setup, behavior planner integration, system integration, and system validation. Note that project management tasks are continuously and indefinitely occurring.

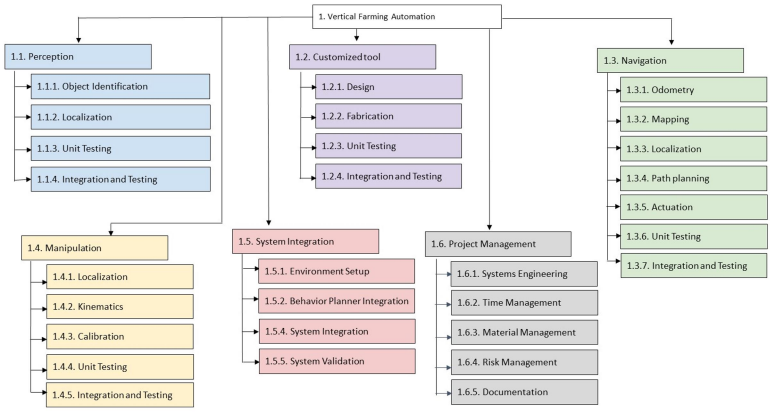


Figure 35: Work Breakdown Structure

7.2 Key Milestones and Schedule

Key Milestones for the Fall 2022 semester are largely divided into two phases: subsystems development to add functions until mid-October and one month of system integration to enhance the robustness and reliability of the system until Fall Validation Demonstration. Since the MRSD progress reviews are biweekly, internal milestones have been aligned with progress reviews.

7.2.1 Key Milestones

In the Fall semester, the main objective is to enhance the robot's navigation, localize the tomato cluster stem, adjust its orientation to align the Cartesian configuration normal to the point of interaction, harvest tomato clusters with a more improved tool, and integrate the system with high robustness and reliability. Corresponding key milestones are described in Table 2.

Table 2: Key Milestones

Key Milestones for the Fall 2022 Semester	
Perception	Achieve tomato stem localization Improve detection accuracy
Navigation	Custom Planner for Row Switching Improved Odometry
Customized Tool	Improve tool attachment design Redesign tool to compact form factor
Manipulation	Add base joint movement for orientation alignment Visual Servoing
Integration	Enhance robustness and reliability Visual Servoing

7.2.2 Schedule

For each progress review, there are about two weeks of time for development, which is illustrated in Fig. 36. The main goal in schedule management is to accomplish subsystem development before progress review 10 in mid-October to acquire more than 4 weeks being allocated for robust system integration. Note that test environment setup lead time has decreased to a great extent as the team decided to create a fake plant environment than nurturing real tomatoes. This is because rapid and repetitive testing interactions with plants are required. As the project accomplished the basic pipeline of the robot in the Spring semester, there is no delay

in the schedule as of now.

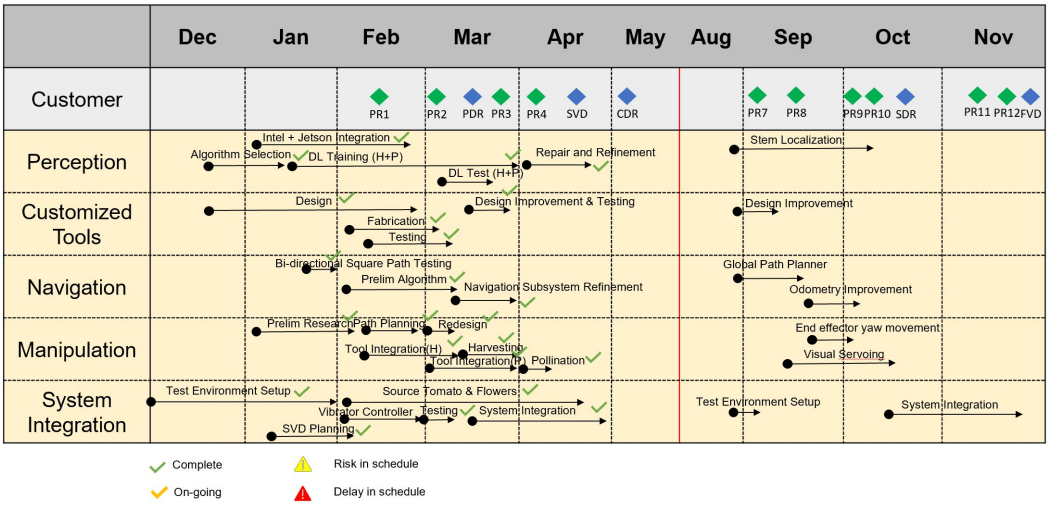


Figure 36: Master Plan

7.3 Test Plan

7.3.1 Subsystem and System Testing Activities

The important subsystem and system testing activities of the project are shown below. #1 to #7 are subsystem tests that will unit test elements and benchmark unit performance. #8 to #10 are system tests that will test integration and benchmark system performance.

1. Environment Setup, Customized Tool: Test cutting and gripping capability of the harvesting tool with different fake materials that have a likeness to real stems to determine which material is to be used
2. Perception: Test stem localization accuracy
3. Navigation: Test the performance of three methods for odometry estimate improvement including replacing the rear wheel with caster, fusing IMU with wheel encoders, and loop closure using the association between detected pots during u-turn.
4. Customized Tool: Test URDF and actuation functionality of harvesting tool via end-of-arm class
5. Customized Tool: Test cutting and gripping capability of newly designed harvesting tool
6. Customized Tool: Test torque and angle feedback from the tool
7. Manipulation: Test actuation accuracy of arm, lift and base given a location
8. Navigation Perception Integration: Test detections while robot is moving
9. Manipulation Navigation Integration: Test actuation accuracy with base joint included
10. Manipulation Perception Integration: Test visual servoing of base and arm using detected stem location

7.3.2 Milestones for Fall-semester Progress Reviews

Capability milestones and system testing for each of the fall-semester Progress Reviews have been summarized in Table 3.

Table 3: High level Test Plan

Date	PR	Milestone(s)	Test(s)
Early Sept	07	Improve existing test setup with fake plants instead of real ones.	Test 1
Mid-Sept	08	Improve Detections Final modular tool attachment with reduced spacing Extend End of Arm tool Class	Test 8 Test 5 Test 4
Early Oct	09	Include base alignment with manipulation and iterate to get above 80% successful harvests in test setup. Improve odometry estimation.	Test 7, 9 Test 3
Mid-Oct	10	Stem localization Incorporation of feedback for robot-environment interactions	Test 2 Test 6, 10
Mid-Nov	11	System Integration	TBD
End-Nov	12	System Integration	TBD

7.3.3 Fall Validation Demonstration (FVD)

The goal of the FVD is to demonstrate the robust autonomous operation of the harvesting operation on a row of plants. The highlighted major capabilities of the robot are the following:

1. Accurate representation of detections in the global frame through better odometry estimates.
2. Accurate alignment of the robot with target point of interaction on the stem.
3. Increased repeatability and accuracy of the harvesting operation in more complicated positions and orientations of the target clusters.

At the present time, the test will follow a similar procedure as that demonstrated during the SVD. The testing environment will be redesigned using fake plants that better capture the real farming environment. The robot will be running an updated software stack with refinements that implement the goals listed above.

The demonstration will take place on the basement level of Newell-Simon Hall outside the caged area. The required equipment will be the Hello stretch RE1 robot mounted with the harvester tool, wrist camera and Jetson, the testing racks with the fake plants for harvesting with tomato clusters attached, and media equipment for performance visualizations. The hello stretch RE1 robot platform with the Bob the Farmer software stack will be wheeled out to the test site and placed in front of the first row. Once the software package is launched, the robot will traverse the row, stop at the locations where 5 different tomato clusters have been identified, and harvest them. The robot will continue to traverse the farm till the end of the rows and terminate the program at the end of traversal. The biggest difference between SVD and FVD is in the use of a wrist camera for fine adjustment of the robot while targeting the cluster. This is shown in Fig. 37 where the wrist is extended and the stem location is used to re-orient and align the base for optimal grasp.

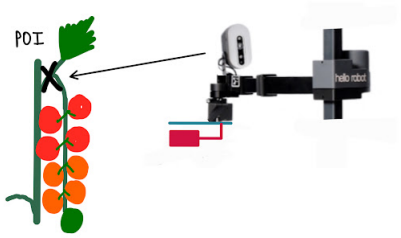


Figure 37: Wrist Camera Extension for Stem Detection and Localization

7.4 Parts List and Budget

Table 4 lists parts acquired through the MRSD budget for the project. Table 5 lists parts acquired through labs or MRSD inventory for the project. Table 6 is an overview on the budget and budget management.

Table 4: Parts List from Budget

No.	Part Name	Qty	Unit price	Total price	Comments
1	Black Nursery Pots	1	39.99	39.99	
2	Fake tomatoes	4	8.99	36	
3	Fake vine	1	11.89	11.89	
4	Fake vine tomatoes	1	12.99	12.99	
5	Antifungal	1	16.99	16.99	
6	Walmart- header racks(4), growth lights(3)	1	149.19	149.19	In person buy
7	Timer switch	2	10.19	20.38	
8	Water pump	1	42	42	
9	Water pump type 2	3	39.99	119.97	
10	Vibrating brush	2	24	48	
11	Bullet massager	1	9.71	9.71	
12	Soil	1	13.99	13.99	
13	Clothing rack	1	18.36	18.36	
14	3V relay	1	15.99	15.99	
15	Dynamixel motors	3	48.90	145.8	Shipping costs are not included
16	Car rentals	3	–	205.16	Yarnick's farm visit Walmart for acquiring test setup material (2)
17	Electronics	9	–	141.23	Includes keyboard, mouse, wifi adapter, batteries
18	Pesticide	2	11	22	
19	Flowers	1	186.93	186.93	For SVD Encore

Table 5: Parts list Procured

No.	Part Item	Qty.	Procurement means
1	Hello Stretch RE1 Robot	2	Manipulation lab, Prof Oliver Kromer
2	Intel RealSense Camera -D435	2	MRSD inventory
3	Desktop	2	MRSD inventory (returned 1)
4	Keyboard	2	MRSD inventory (returned)
5	Mouse	2	MRSD inventory (returned)
6	Extension cords	4	MRSD inventory (returned)
7	NVIDIA Jetson	2	MRSD inventory

Table 6: Budget Management

Total Budget	**$ 5000**
Budget used till now	$ 1411.14 (28.22%)
Remaining Budget	**$ 3588.2**
Future expenses (for FVD)	$ 1500 (approximate)

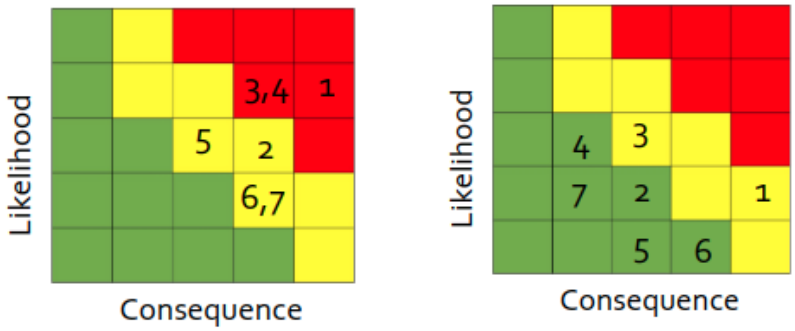
7.5 Risk Management

Some of the major risks associated with the project are outlined in Table 7. Technical risks include parts of the robot failing, subsystems being insufficient to complete the functional requirements associated, and the inability to produce a test setup with the intricate features needed. The Risk Likelihood-Consequence Table during identification of risks is shown in Fig. 38(a)

and the Risk Likelihood-Consequence Table following mitigation plan addressed is shown in Fig. 38(b). Earlier, Risk #4-6 were identified. They were encountered during SVD. By applying the mitigation plans, #5 and #6 were successfully mitigated. New risks #1-3 that had not been identified earlier were analyzed.

Table 7: Risk Management Table

No.	Risk	Category	Description	Likelihood	Consequence	Mitigation Plan
1	Robot hanging due to overload	Technical/ Schedule	Robot hangs due to overload - almost always due to operation of both cameras, sometimes due to to HDMI connected/RViz running sometimes proceeded by actuation inaccuracies and slow process, especially seen with integrated subsystems	4	5	Connect wrist camera to Jetson Run ROS on multiple computers for RViz Prefer to SSH into the bot Could be related to Risk #3 Reduce subsystem memory load
2	Inaccurate Wheel Odometry	Technical/ Schedule	Since this is the reference frame, even if the system works, not with respect to real world/ground truth.	3	4	Use ArUco markers on the pot as landmarks Identify pots that have been encountered Fusing IMU data with wheel encoders
3	Motor commands drop	Technical	Dropping motor commands - Causes synchronous movements Causes disabled motor torque	4	4	Use sleep commands Restart between runs Better mode switching
4	False Positives from detection model	Technical	Identify non-targets as targets	4	4	Use detections over multiple frames to filter out false positives
5	Unsuccessful cut	Technical/ Schedule	Stem is not cut in two passes	3	3	Add more passes or improve the tool design
6	Malfunctioning of the tools	Technical/ Schedule	Pollinator does not work due to its fragility or impact from wrist extension	2	4	Have other tool replacement options available Use the USB port instead of the header expansion
7	Manipulator collision/ manipulator stuck	Technical	Manipulator gets stuck in the plant while reaching the target points	2	4	Robot stops at an offset then approaches with a lower speed Change tool angle and approach again Visual servoing to avoid obstacles



(a) (b)

Figure 38: A Risk Likelihood-Consequence Table (a) During Identification (b) After Mitigation

8 Conclusions

The over-arching goal of the spring semester was two-fold. To implement a basic autonomy pipeline that would allow the robot to navigate to target locations within the farm environment and to get the end-effector within $\pm 10cm$ of the detected clusters. At the end of the Spring semester, the team believes that a basic working pipeline has been developed. Currently, the system is able to navigate along the rows while maintaining a safety offset of 15 cm. The robot can detect clusters with a mAP of 60% and it is able to get its end-effector within the required limits of ± 10 cm. The priorities for the Fall semester are to make the existing system cleaner in terms of software engineering as well as refining the kinks present in the existing system. Pollination as a project goal will be replaced with an increased emphasis on just harvesting. This will make the project more relevant to the needs of the indoor tomato harvesting operation as farmers consider bumblebees a more cost-effective solution. In terms of refinement, the team will implement techniques to improve odometry, include the base as a joint in the manipulation strategy, which will be coupled with visual servoing, and refine the tool to be more compact for the harvesting operation.

9 References

[1] Oppenheim, Dor Shani, Guy Edan, Yael. (2020). Tomato Flower Detection Using Deep Learning. 10.13140/RG.2.2.19486.56647.

[2] Manya Hubert, Schadeck, Dick, Marcel. (2020). Tomato Fruit Detection and Counting in Greenhouses Using Deep Learning. 10.3389/fpls.2020.571299

[3] ”Tsironis V., Bourou S., Stentoumis C. (2020). tomatOD: Evaluation of object detection algorithms on a new real-world tomato dataset. In ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Available from https://github.com/up2metric/tomatODhttps://github.com/up2metric/tomatOD ”

[4] Afonso, Manya; Fonteijn, Hubert; Polder, G. (Gerrit); Wehrens, Ron; Lensink, Dick; Faber, Nanne; et al. (2021): Rob2Pheno Annotated Tomato Image Dataset. 4TU.ResearchData. Dataset. https://doi.org/10.4121/13173422.v3

10 Appendix

A. Functional requirements

The system shall

M.F.1. Localize itself in the environment
M.F.2. Perform collision avoidance
M.F.3. Identify tomato flowers
M.F.4. Disperse pollen
M.F.5. Identify ripe tomato bunches
M.F.6. Cut the identified tomato bunches The system shall
D.F.1. Map obstacles in the environment
D.F.2. Autonomously navigate around rows in the test environment
D.F.3. Localize pollination point
D.F.4. Preserve flowers
D.F.5. Localize harvesting point
D.F.6. Preserve the quality of tomatoes

B. PCB

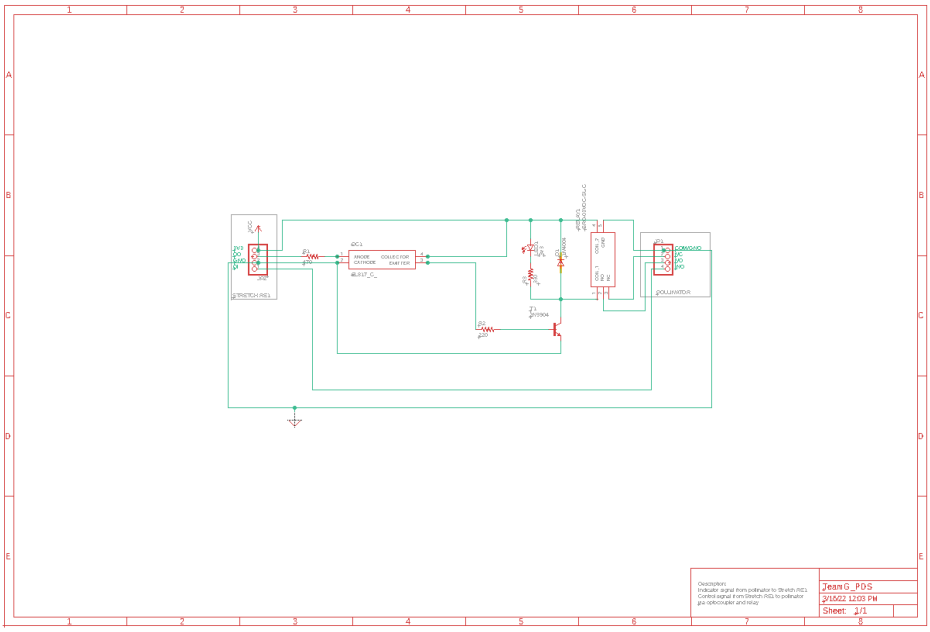


Figure 39: Pollinator Schematic

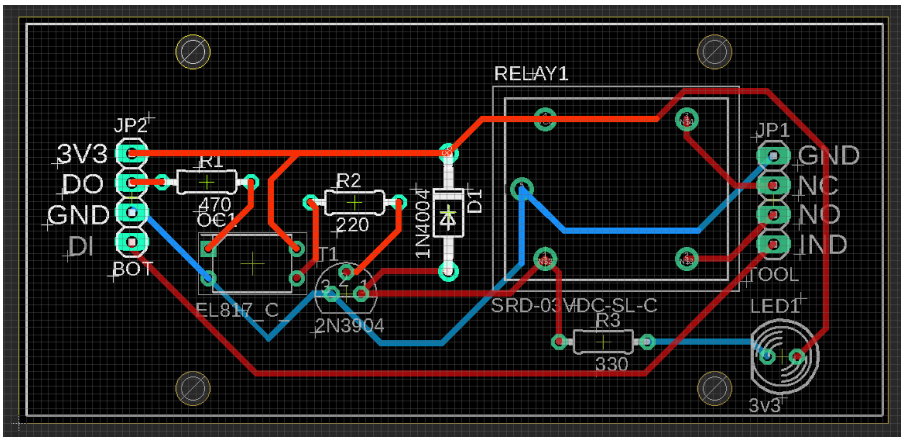


Figure 40: Pollinator Board