SARAL-Bot: Autonomous Robot for Strawberry Plant Care

Arif Ahmed Ritvik Agarwal Gaurav Srikar Nathaniel Rose Parikshit Maini
University of Nevada Reno

***Abstract*—Strawberry farming demands intensive labor for monitoring and maintaining plant health. To address this, Team SARAL develops an autonomous robot for the 2024 ASABE Student Robotics Challenge, capable of navigation, unhealthy leaf detection, and removal. The system addresses labor shortages, reduces costs, and supports sustainable farming through vision-based plant assessment. This work demonstrates the potential of robotics to modernize strawberry cultivation and enable scalable, intelligent agricultural solutions.**

I. Establishment of Need and Benefit to Agriculture

The 2024 ASABE Student Robotics Challenge aims to simulate the task of strawberry farming, which presents an excellent opportunity to demonstrate the potential benefits of robotics in agriculture. Strawberries are extensively consumed worldwide and the Food and Agriculture Organization (FAO) reported the increasing production of strawberries worldwide [1]. Strawberry production is labor-intensive, requiring significant manual effort for tasks such as planting, monitoring plant health, removing unhealthy leaves, and harvesting. These tasks can be time-consuming, costly, and physically demanding for human workers. The development of autonomous robots capable of navigating strawberry fields, identifying healthy and unhealthy plant parts, and precisely removing unhealthy leaves and flowers has the potential to revolutionize the strawberry farming industry. By automating these tasks, farmers can:

1) Reduce labor costs: Autonomous robots can work continuously without the need for breaks, thereby reducing the number of human workers required and associated labor costs.
2) Improve efficiency: Robots can complete tasks faster and more consistently than human workers, leading to increased overall efficiency in strawberry production.
3) Enhance crop health: Robots equipped with advanced sensors can accurately identify unhealthy plant parts and remove them promptly, preventing the spread of diseases and ultimately improving crop health and yield.
4) Enable precision agriculture: Robotics allows for targeted treatments and interventions, reducing the use of resources such as water, fertilizers, and pesticides, thus promoting sustainable agricultural practices.
5) Alleviate labor shortages: As the agricultural industry faces increasing labor shortages, robots can help fill the gap and ensure that critical tasks are completed on time.

By participating in the 2024 ASABE Student Robotics Challenge, our team aims to contribute to the development of innovative robotic solutions that address the needs of the strawberry farming industry. Our robot demonstrates the potential for autonomous systems to improve efficiency, crop health, and sustainability in agriculture, ultimately benefiting farmers, consumers, and the environment.

In order to satisfy the strawberry market demands, it is obvious that we need to increase the production of strawberries. Recent robotics research on agriculture reported that robotic solutions can help monitor plants and increase the overall yield of strawberries [2], [3]. To get a high yield of strawberries it is important to identify if the plants are suffering from nutrient deficiencies, diseases, and/or incorrect soil pH. Interestingly, only by inspecting the leaf-color it is possible to identify several factors.[1] For example, the yellow or pale leaves indicate that the plant had moisture stress due to over-watering. When the leaves start to show such characteristics, it is necessary to prune those unhealthy leaves so that the symptoms do not spread to other leaves. Similarly, there are some symptoms such as fungal attack, airflow problem, cold snap issue etc. Those problems can be easily identifiable using the robotic solutions [3].

Fig. 1. Simulated ASABE 2024 Competition Arena (preparatory stage)

II. Approach and Originality

Our team has developed an innovative solution for the advanced division of the 2024 ASABE Student Robotics Challenge, leveraging the capabilities of an off-the-shelf robot, the HiWonder ArmPi Pro [2], and building on top of it with custom hardware and software.

[1]https://www.epicgardening.com/strawberry-problems/
[2]https://www.hiwonder.com/products/armpi-pro

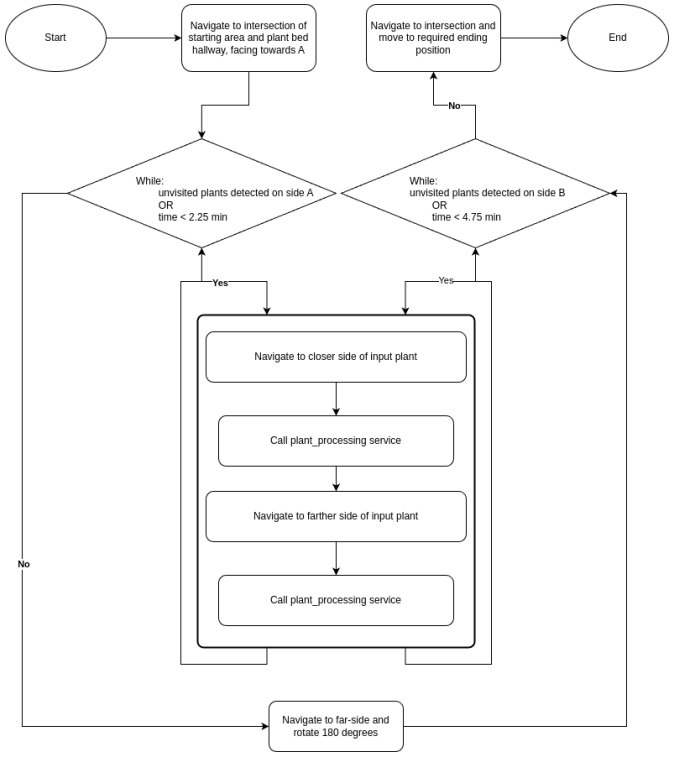

Fig. 2. Flowchart of SARAL-Bot Behaviour Coordinator Logic Coordinator

1) Hardware Enhancements: To improve the robot's localization capabilities while adhering to the size constraints, we have mounted an Intel D435i [3] RGBD camera on a linear actuator at the rear of the ArmPi Pro, facing forward. The linear actuator allows the depth camera (i.e., D435i RGBD camera) have better and increased field of view. Moreover, this design configuration also helps us perform a robust Visual Inertial Odometry (VIO) using a VIO algorithm called Real-Time Appearance-Based Mapping (RTABMAP) [4].

2) Software Architecture: Our software stack is built on ROS2 Humble, the latest LTS version, ensuring compatibility and stability.[4] We have developed a modular architecture consisting of three main systems (all in ROS2 Humble):

1) Behavior Coordinator - responsible for task sequence
2) Navigation System - responsible for mobility control
3) Plant Processing System - responsible for plant trimming

These subsystems then communicate with the provided ArmPi Pro's control nodes via a ROS1 Node made by us which exposes various REST API endpoints via a webserver. Finally, in order to ensure reliable and repeatable deployment, we use Docker for environment management[5].

3) Robot Behaviour and Task Strategy: The robot behaviour is focused on coordinating our navigation and plant processing systems. Given the static nature of the arena, we have optimized our navigation strategy by limiting planning to pre-defined obstacle-free straight-line movements.

[3]https://www.intelrealsense.com/depth-camera-d435i/
[4]https://docs.ros.org/en/humble/index.html
[5]https://www.docker.com/

4) Arm Control and Inverse Kinematics: For controlling the ArmPi Pro's chassis and arm, we utilize the onboard Raspberry Pi, running a Flask server with a ROS Melodic node. This allows seamless integration of the existing hardware with our ROS2-based software stack. Inverse kinematics for the robotic arm is handled using MoveIt 2, a powerful motion planning framework. This enables our robot to accurately reach leaves and flowers at varying heights and distances.

5) Computer Vision Techniques: To detect and differentiate between healthy leaves, unhealthy leaves, and flowers, we employ HSV-based color thresholding followed by contour detection. This computer vision technique ensures robust performance under varying lighting conditions.

Our unique combination of hardware enhancements, modular software architecture, optimized navigation strategy, and advanced computer vision techniques demonstrates our team's innovative approach to tackling the challenges presented by the competition.

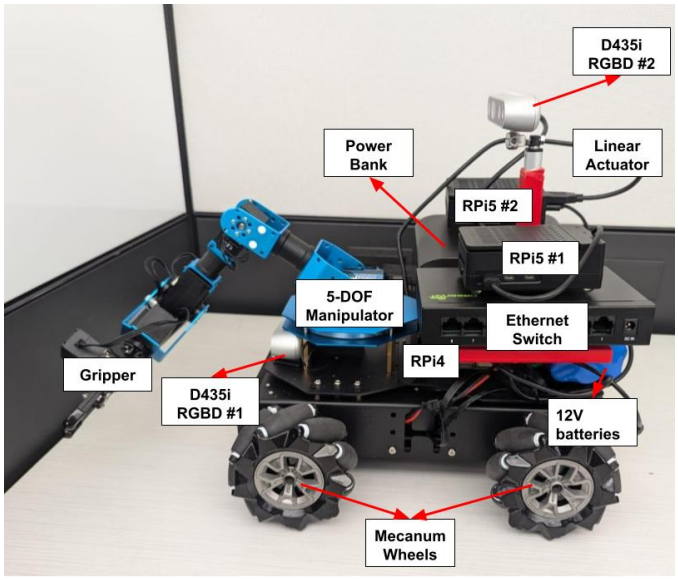

Fig. 3. The Robot Design: Stacking various hardware components to handle specific computational tasks. In this image, an RPi4 is mounted directly on top of the robot chassis, with a power bank, Ethernet switch, and two RPi5 units stacked on top of it.

III. Definition of Design Objectives and Criteria

Our team established clear design objectives and criteria to guide the development of our robot for the 2024 ASABE Student Robotics Challenge. These objectives were based on a thorough understanding of the competition rules, constraints, and challenges, ensuring that our robot would be well-suited to perform the required tasks effectively.

1) Alignment with Competition Requirements: Our primary design objective was to develop a robot capable of autonomously navigating the arena, inspecting plants, and performing selective harvesting while adhering to the size constraints specified in the competition rules. We aimed to create a compact and efficient design that could complete the tasks within the allotted time frame.

2) Performance Goals: We set specific performance goals for our robot, including:

- Achieving a navigation speed of at least 10 cm per second
- Maintaining a localization accuracy within 2 centimeters of the ground truth
- Detecting plant health with an accuracy of 95 percent or higher
- Successfully harvesting at least 90 percent of the target leaves and flowers

These quantitative goals helped us focus our efforts and evaluate the success of our design.

3) System Architecture and Component Selection: Our design objectives influenced our choice of system architecture and hardware components. We selected the ArmPi Pro as our base robot for its holonomic navigation capabilities, and we added an Intel D435i RGBD camera on a linear actuator to enhance localization and object detection. We chose ROS2 Humble as our software framework for its stability, modularity, and community support.

4) Modularity, Scalability, and Ease of Use: We prioritized modularity and scalability in our design, allowing for easy integration of new features and algorithms as the project progressed. We also aimed to create a user-friendly system that could be easily set up, operated, and maintained by our team members.

5) Reliability and Robustness: Ensuring the reliability and robustness of our robot was a critical design objective. We aimed to develop a system that could perform consistently and handle unexpected situations gracefully. This involved extensive testing and validation of our hardware and software components. For example, to increase the computational power of our system while adhering to the imposed budget, we decided to go for a distributed network of Raspberry Pis rather than a single higher-end computer.

6) Cost and Resource Optimization: Finally, we sought to optimize cost and resource utilization throughout the design process. We carefully selected components that balanced performance and affordability, and we leveraged open-source software and libraries to reduce development time and cost. By defining clear and specific design objectives and criteria, we established a strong foundation for the development of our robot, ensuring that it would be well-equipped to tackle the challenges of the 2024 ASABE Student Robotics Challenge.

IV. Parts List and Table

We show the parts list and prices in Table I in the bill of materials. The table categorizes components (robot, computational resources, sensors, power supplies, 3D printed parts and miscellaneous items) and gives the details for each component within a category. We've also emphasized that the total compute resources used on the robot adhere to the imposed control components budget, i.e. $ 500.00.

V. Hardware Description

A. Chassis and the Arm: HiWonder Arm Pi Pro

The ArmpiPro chassis is fully manufactured with a hard aluminum alloy to keep it durable while also being lightweight.

Item	Cost Per Unit
HiWonder Arm Pi Pro	**$ 529.00**
1 × Aluminum alloy chassis (256mm × 298mm)	
4 × Aluminum mecanum wheels (50mm × 100mm)	
1 × LX-225 servo (included in ArmPi Pro)	
3 × LX-15D servo (included in ArmPi Pro)	
1 × ID 1 servo (included in ArmPi Pro)	
1 × Gripper (included in ArmPi Pro)	
4 × Motor encoders	
1 × Arm Pi Pro 7.4V lithium ion battery (6000mAh)	
1 × Raspberry Pi 4 (included in ArmPi Pro)	
Added Computational Resources	
2 × Raspberry Pi 5	$ 90.00
1 × TP-Link 5 port Ethernet switch	$ 18.00
Sensors	
2 × Intel D435i depth camera	$ 327.00
Power Suppliers	
1 × KBT 12V lithium ion battery (2400mAh)	$ 23.00
1 × Krisdonia 5V power bank (25000mAh)	$ 89.50
Added Hardware	
1 × 12V Linear actuator(180mm length w/125mm stroke)	$ 28.00
1 × 12V 2-Channel Relay Switch	$ 8.99
Custom 3D Printed Parts	
1 × 3D printed chassis back (225.5mm×112mm×140mm)	$ 18.80
1 × 3D printed custom gripper (100mm length)	$ 0.94
TOTAL COMPUTE RESOURCES	**$ 260.00**

TABLE I
Bill of Materials with Prices

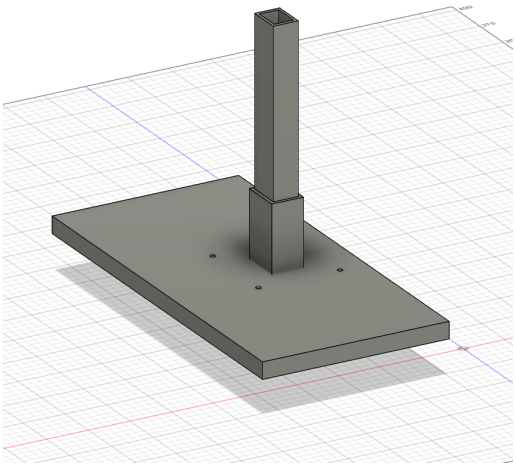

Fig. 4. Custom Chassis Back: This custom design is attached to the back of our aluminum chassis in order to increase the space we can utilize for components and to securely hold the linear actuator in place. The custom back is 10mm thick, 225.50mm wide, and 112.00mm long.

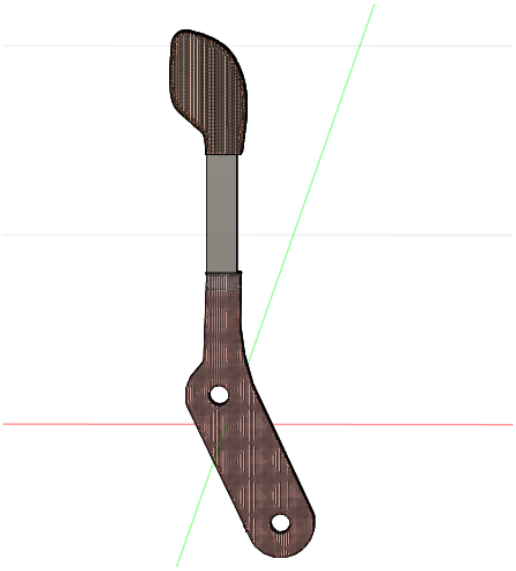

Fig. 5. Custom Gripper: This custom gripper design is utilized to extend the reach of our original gripper from 60mm to 100mm.

The chassis is 256mm wide, 298mm long, and 148mm tall, which puts it within the boundaries of the competition's size limitations. The chassis has several threaded holes throughout to allow for easy customization and easy wire management. The SARAL Bot's chassis has been customized with the use of a custom 3D printed back to stabilize the linear actuator while also adding space for additional components. New Mecanum wheels have been chosen to replace the initial ones that come with the Arm Pi Pro. These new wheels are made of aluminum alloy, minimizing risks of damage, are heavier, meaning they provide better stability to the mobile platform, and have fully rubber rollers, allowing for better traction and therefore control while traversing the peat moss filled terrain.

Fig. 6. Aluminum Mecanum Wheels: These new mecanum wheels are fully manufactured from aluminum alloy, making them more durable and heavier than the original plastic mecanum wheels. They also utilize fully rubber rollers allowing for better traction.

The manipulator mounted on our saral_bot features a 5 Degree of Freedom arm, powered by servo motors. Initially, the orientations of each joint in the arm were configured to perform pitch operations, except for the 1st and 5th joints, which were yaw-based, and the end effector. This configuration limited the arm's reach, particularly in accessing unhealthy leaves located at the back of a plant model. To enhance its reach, we altered the orientation of the 4th joint to perform rolling operations. This adjustment enables the arm to extend around either side of a plant, bend the forward arm sideways, and precisely reach the leaves with the gripper. The modified orientation configuration of the arm is detailed in table II.

The Armpi Pro robot's built-in gripper was too short to reach leaves behind the plant without disturbing the surrounding healthy leaves. We developed a longer fork gripper to grasp unhealthy leaves effectively without impacting the adjacent foliage. The designs of these grippers are depicted in Fig. 5. The initial reach of the arm was 363mm. With the extended gripper, the new total reach is 403mm.

The Armpi Pro robot was originally equipped with a built-in camera located on the 5th link of the arm. However, for this challenge, an eye-in-hand configuration was not ideal and was effectively reducing the power output of the end-effector, so we removed the pre-mounted camera.

When not in use for manipulation, the manipulator is positioned in a default pose to avoid obstructing the sensing perspective from the top of the linear actuator.

B. Computational Resources and Networking

Our robot's computational resources consist of three Raspberry Pi (RPi) single-board computers: two RPi 5 models and one RPi 4 which camer integrated with the ArmPi Pro. This distributed computing architecture allows us to efficiently handle the various computational tasks required for the robot's operation. The two RPi 5 models are dedicated to running our custom ROS2 nodes, which handle high-level decision making, navigation, and plant processing. These nodes communicate with each other using ROS2's built-in communication infrastructure, leveraging the publish-subscribe and service-client patterns. The RPi 5 models were chosen for their enhanced processing power and improved I/O capabilities compared to previous Raspberry Pi generations, ensuring smooth and responsive performance of our ROS2 system while adhering to the compute budget. The RPi 4 model, which comes integrated with the ArmPi Pro, is responsible for running the low-level control software for the robot's motors and servos. This control software, based on ROS1 Melodic, communicates with the higher-level ROS2 nodes running on the RPi 5 models through a custom ROS1-ROS2 bridge implemented using a Flask web server. This is further discussed in the ROS1-Flask Server subsection in Section VI. To facilitate communication between the three RPi boards, we employ a TP-Link 5-port Ethernet switch. The switch establishes a local area network (LAN) connecting the RPis via IPs configured by us, enabling them to exchange data and messages efficiently. The choice of a wired Ethernet connection, as opposed to wireless communication, ensures reliable and low-latency communication between the computational resources, which is critical for real-time robot

control and coordination. By distributing the computational workload across multiple RPi boards and establishing a robust wired network for communication, we have created a powerful and flexible computational infrastructure for our robot. This architecture allows us to efficiently process sensor data, perform complex calculations, and coordinate the robot's various subsystems, ultimately enabling it to navigate, inspect plants, and perform selective harvesting tasks with high accuracy and reliability.

C. Sensors

We are using two Intel D435i depth cameras for visual perception (See Fig. 3). The D435i is a high-performance, low-power stereo vision camera that provides accurate depth information. It features a pair of depth sensors, an RGB sensor, and an infrared projector, enabling it to capture high-resolution 3D images in various lighting conditions. One camera is mounted on a linear actuator at the rear end of the robot which will expand at the start. This placement provides a more isometric view of the plants while allowing the arm to function without occluding them. The other RGDB camera is mounted in the front and enables the robot to perform accurate localization and mapping.

In addition, the motor encoders on wheels and the arm servos are being used to achieve the desired outputs. Three different types of servos are utilized: LX-225, LX-15D, and ID 1 servo. More details on the servo motors are described in table II, as well. All these servos perform the manipulation with 0.3° accuracy.

Joint ID	Servo Type	Orientation	Torque	Limits (In Radians)
Joint 1	LX-15D	Yaw	$15kg/cm$ with $6V$ $17kg/cm$ with $7.4V$	$-\frac{\pi}{2} to \frac{\pi}{2}$
Joint 2	LX-225	Pitch	$25kg/cm$ with $7.4V$	$-\frac{\pi}{2} to \frac{\pi}{2}$
Joint 3	LX-15D	Pitch	$15kg/cm$ with $6V$ $17kg/cm$ with $7.4V$	$-\frac{\pi}{2} to \frac{\pi}{2}$
Joint 4	LX-15D	Roll	$15kg/cm$ with $6V$ $17kg/cm$ with $7.4V$	$-\frac{\pi}{2} to \frac{\pi}{2}$
Joint 5	ID 1	Yaw	$8kg/cm$ with $7.4V$	$-\frac{\pi}{2} to \frac{\pi}{2}$

TABLE II
SERVO SPECIFICATIONS FOR ROBOT JOINTS IN "ZERO STATE"

D. Power Supply

The system design utilizes 3 different power supplies due to the varying power requirements of different components. The first power supplier is provided with the Arm Pi Pro and is a 7.4V lithium ion polymer (Lipo) battery with 6000mAh which is used to power the 7.4V motors in the mobile platform and manipulator. The 7.4V Lipo battery also powers the RPi 4 through the use of a step down chip to bring it to the intended 5V voltage. The next power supplier is another Lipo battery that is 12V with 2400mAh and is used to power the linear actuator through the use of a 5V to 12V 2-channel relay switch which gives the correct 12V inputs to the linear actuator depending on the 5V inputs it receives, thus allowing precise control over the linear actuator, which is crucial to positioning the rear-mounted depth camera for plant inspection and manipulation. The final supplier is a 5V power bank with 25000mAh which powers both RPi 5 modules, both D435i depth cameras, and the Ethernet switch.

VI. SOFTWARE OVERVIEW

We select the Robot Operating System 2 (ROS2) Humble instead of ROS1 for all our sensing and planning tasks. Overall, ROS2 is a more reliable and high-quality framework compared to ROS1 [5]. ROS2 uses highly secure DDS communication compared to the TCP/UDP used in ROS1. Besides, ROS2 supports multiple nodes per process, whereas ROS1 allows only a single node per process. However, the existing control software for the wheel and arm motors, provided by ArmPi Pro is based on ROS1 Melodic. To avoid reinventing the wheel, we built competition-specific software on top of this foundation. Our software architecture consists of three main modules: Behaviour Coordinator, Navigation System, and the Plant Processing System. These modules together achieve the desired robot functionality in the competition.

A. ROS1-Flask Server

In order to interface the ROS1 system and the ROS2 system built on top of it, we created a ROS1 Node that also runs an internal web server. This server exposes REST API endpoints which when hit by our ROS2 system with desired motor inputs as parameters, publishes the equivalent ROS1 messages to control the wheel and arm motors.

We opted for the Flask REST API instead of a ROS2 bridge due to the lack of support for ROS2 (Dashing) on Ubuntu 18.04 running on the Armpi Pro's Pi. Thus, the Flask REST API provides a reliable channel for communication between ROS1 and ROS2.

B. Behaviour Coordinator

The Behaviour Coordinator (as shown in Fig 2) is responsible for coordinating the overall robot behavior and managing the high-level decision-making process. It communicates with the Navigation System and Plant Processing System using custom ROS2 messages to trigger the appropriate actions based on the current state and sensor data. The Behaviour Coordinator implements the following high-level logic:

1) Navigate to the intersection of the starting area and plant bed hallway, facing towards plant bed A.
2) While there are unvisited plants detected on side A or the elapsed time is less than ~ 2.25 minutes, navigate to the nearest side of the next detected plant on side A and invoke the Plant Processing System.
3) Navigate to the far side of the arena and rotate 180 degrees.
4) While there are unvisited plants detected on side B or the elapsed time is less than ~ 4.75 minutes, navigate to the nearest side of the next detected plant on side B and call the Plant Processing System.
5) Navigate to the intersection and move to the required ending position.

The front mounted camera on the robot provides the necessary data for robot localization with respect to the global frame.

The rear-end front facing camera, which is elevated along the vertical axis when the linear actuator is extended, is accessed by the Plant Processing System for visually analyzing the plants. The identified targets i.e. unhealthy leaves and unwanted flowers, and their 3D locations are then utilized to perform the arm positioning and the plucking maneuver.

C. Navigation System

The Navigation System (see Fig. 7) is responsible for localizing the robot within the arena and planning paths to reach the desired goal positions. It relies on sensor data from the Intel RealSense D435i depth cameras and uses RTABMAP for visual SLAM (Simultaneous Localization and Mapping), thus constructing an accurate map of the environment. As the robot behaviour progresses, navigation goal pose is decided and conveyed by the Behaviour Coordinator, which ensures that there are no obstacles in the straight line path connecting the current pose to the goal pose. Now, a simple PID controller is used to minimize the error between the two poses. On completion the control goes back to the Behaviour Coordinator.

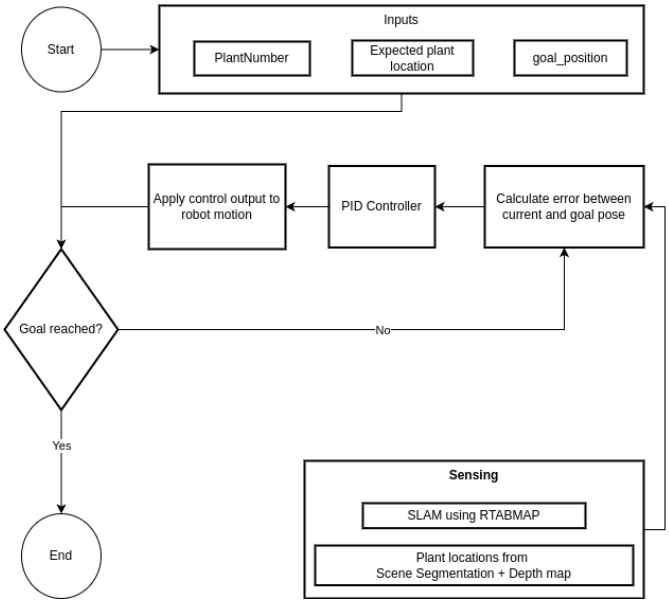

Fig. 7. Flowchart of SARAL-Bot Navigation System

D. Plant Processing System

The Plant Processing System is activated by the Behaviour Coordinator when the robot is positioned near a plant. It utilizes the rear-end front-facing depth camera on the linear actuator to capture RGBD images of the plant. A vision pipeline to detect and localize the leaves and flowers of each plant is put in place. The pipeline transforms RGB into the HSV (Hue, Saturation, Value) color space and identifies healthy leaves, unhealthy leaves, and flowers based on the color thresholds derived from the information provided in the competition rules. To reduce false positives, a planarity check is performed using the RANSAC algorithm [6], discarding any detected objects that are not planar.

The depth information from the camera is then used to localize the detected leaves and flowers with respect to the camera frame. After detecting, identifying, and localizing the plant parts, the 'Plant Processing System' performs the following tasks:

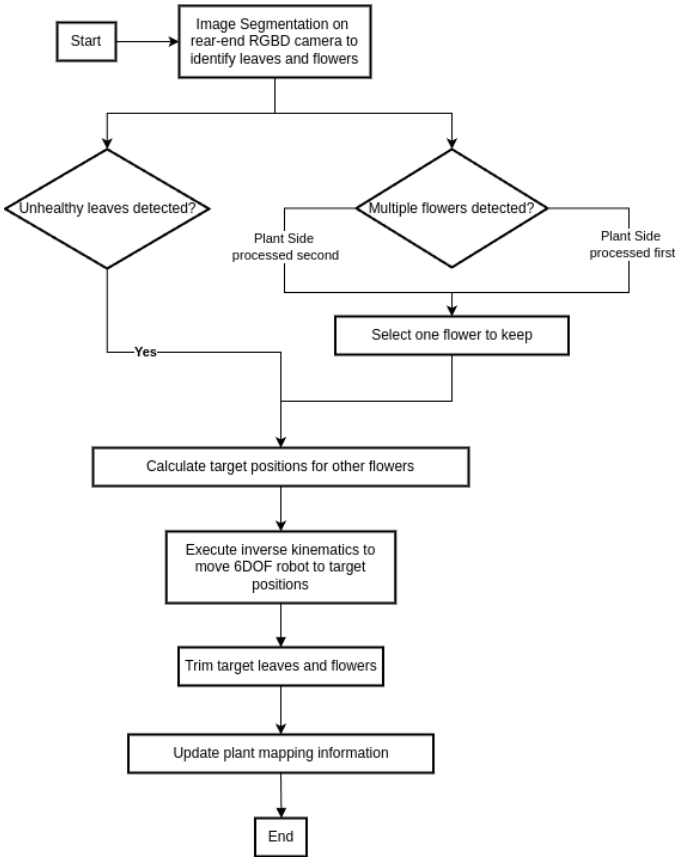

Fig. 8. Flowchart of the SARAL-Bot Plant Processing System

1) Publish the counts and poses of the plant parts using a custom ROS2 message.
2) Transform the poses of the unhealthy leaves and flowers to the manipulator's base link.
3) Based on the input from the behaviour node, if we are processing the nearer side of the plant, we skip one flower from trimming. On the other hand, if we are processing the farther side of the plant, depending on whether there is 0 or 1 flower left on the nearer side, we trim all but one or all flowers, respectively.
4) Send manipulation commands to the ArmPi Pro's joint state control node using REST API requests, thus reaching the required pose for the modified gripper to trim the target.

By breaking down the software architecture into these three main modules, we ensure a clear separation of concerns and enable efficient communication between the subsystems. The Behaviour Coordinator manages the high-level decision-making, the Navigation System handles localization and motion control, and the Plant Processing System focuses on detecting, localizing, and manipulating through plant parts.

As previously mentioned, the Armpi Pro's Raspberry Pi is equipped with ROS Melodic, which manages manipulation-related topics along with other operational topics and services

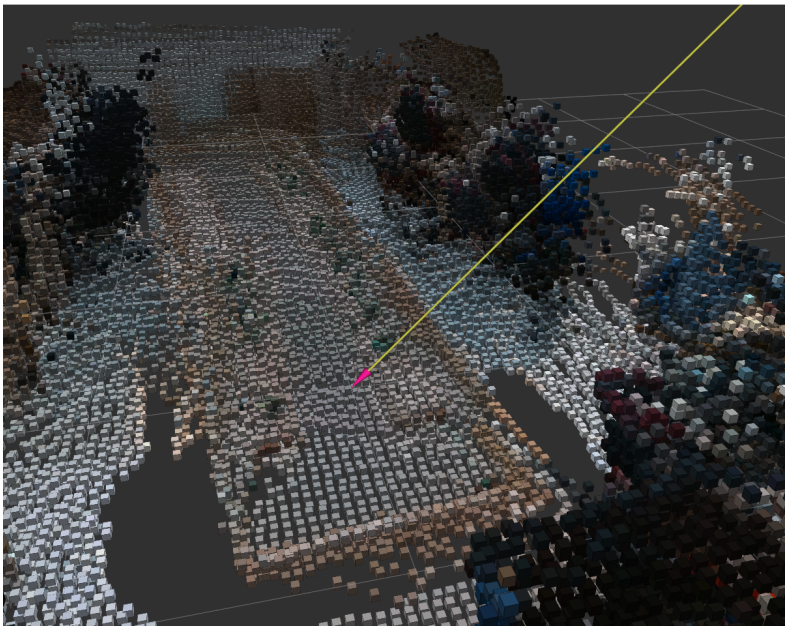

Fig. 9. Map Reconstruction Retrieved from RTABMAP Database

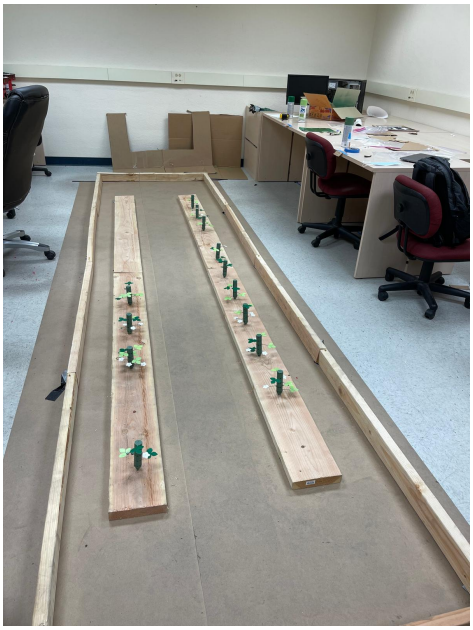

Fig. 10. Ground Truth of the Arena

for robot hardware. The manipulation topic processes an array of integers corresponding to the number of arm joints (ranging between 0 to 1000), with each integer representing the rotation of a joint in radians.

On the ROS2 end, our focus is on performing inverse kinematics and obstacle avoidance. During inverse kinematics, the desired goal location ($3D$ position and orientation) is received detection and localization. We then calculate the inverse kinematics to model the poses of each joint in the arm so that the end-effector reaches the desired location. The estimated poses, expressed in radians, are scaled between $0-1000$ and passed to the ROS1 topic via the aforementioned REST API call.

Initially, we utilized the Robotics Toolbox (RTB) for inverse kinematics, but found that RTB sometimes estimated poses beyond the mechanical limits of the joints, leading to unrealistic and unstable maneuvers. Moreover, RTB lacked pre-defined packages for obstacle avoidance. Obstacle avoidance is crucial to prevent the arm from striking hardware components at the rear of the saral_bot and to avoid collisions with the plant and its healthy structure.

Moveit2 is an easy-to-use, well-structured, stable, and reliable platform for robotic manipulation. Its built-in and ready-to-use functionalities, such as inverse kinematics and obstacle avoidance, enable realistic and stable maneuvering of the arm.

To utilize Moveit2 packages, we created the URDF (Unified Robot Description Format) file of our robot, which describes each part of our arm including links, joints, and their properties. Using the URDF file, we generate an SRDF (Semantic Robot Description Format) file along with several configuration files for collision settings, kinematics, and controllers. These files are initially used for simulation purposes. In the simulation phase, we employ ROS2 control as the controller to manipulate our arm to achieve the desired goal pose. In real-time operations, we use the Armpi Pro's built-in packages as the controller unit.

The inverse kinematics operations are performed with ease, avoiding maneuvers that would exceed the joint limits. In addition to managing joint limitations, the arm must also avoid collisions with the computation block located at the rear and the healthy parts of the plant. To facilitate this, we added a few objects into the planning scene for visualization purposes; for instance, a cuboid object behind the arm and a cylindrical object at the center of the plant. This setup helps in planning the manipulation path to avoid these objects and successfully reach the goal pose.

VII. Appropriateness of Tests and Performance Data

We first test each of the modules we described in this paper. Then, we test the complete competition pipeline in the simulated competition environment (See Fig 1,10).

A. Navigation

In this subsection, we evaluate the performance of our navigation system. During our tests, our robot autonomously navigated the simulated arena by following the static set of waypoints using PID based control, while collecting RGBD and IMU(inertial Measurement Unit) data from the Intel RealSense D435i depth camera. RTABMAP processed the incoming data in real-time, constructing a 3D map of the environment (Fig. 9) and estimating the robot's pose within the map.

These results demonstrate the robustness and reliability of our navigation system. With more thorough testing and validation of our navigation system in the simulated environment, we will be even more confident in its ability to perform well in the actual competition. The positive results obtained from these tests underscore the appropriateness of our chosen approach and the readiness of our robot to tackle the real challenge.

B. Detection

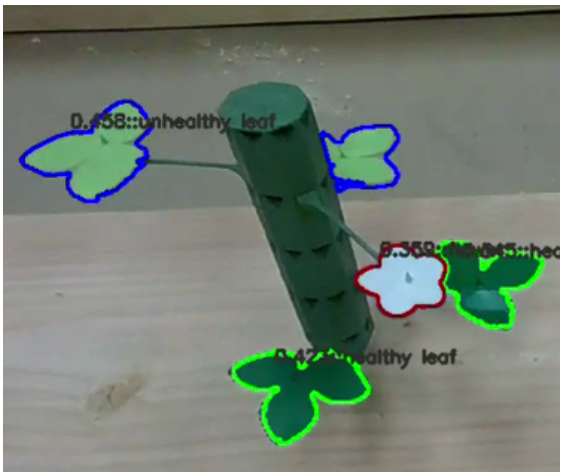

Fig. 11. Detected Healthy, Unhealthy Leaf-clusters, and Flowers

In Fig 11, we show our single plant detection result. We use accuracy as a metric for evaluating our HSV-color thresholding method for detecting healthy or unhealthy leaf clusters and flowers. Under different viewpoints and lighting conditions, our preliminary results show good accuracy but also exhibit false positives. For example, non-plant objects were detected as either flower or leaf-cluster due to having similar color properties.

When we include a planarity check, then it eliminates false positives. We visualize the plane of a detected flower object of a plant tested on our simulated test environment in Fig. 12. Here, the fitted plane contains 251 inliers.

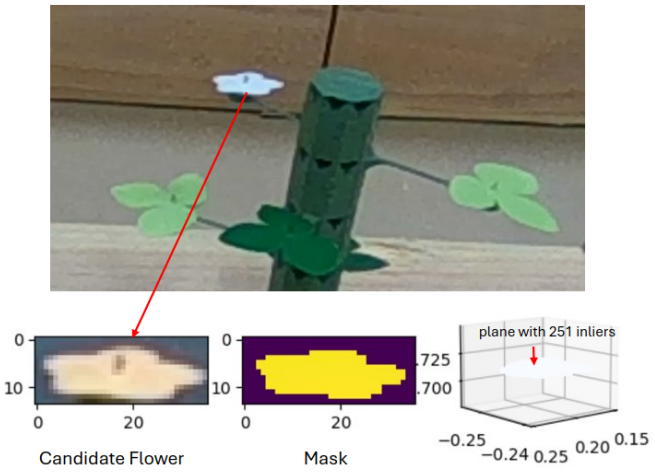

Fig. 12. Visualizing the Plane of a Detected Flower Object with 251 Inliers

C. Manipulation

To validate the performance of our manipulation system, we conducted a series of tests in both simulated and real-world environments. The tests aimed to assess the accuracy, reliability, and efficiency of our Inverse Kinematics (IK) solver and obstacle avoidance algorithms. Initially, we utilized the Robotics Toolbox (RTB) Python by Dr. Peter Corke for

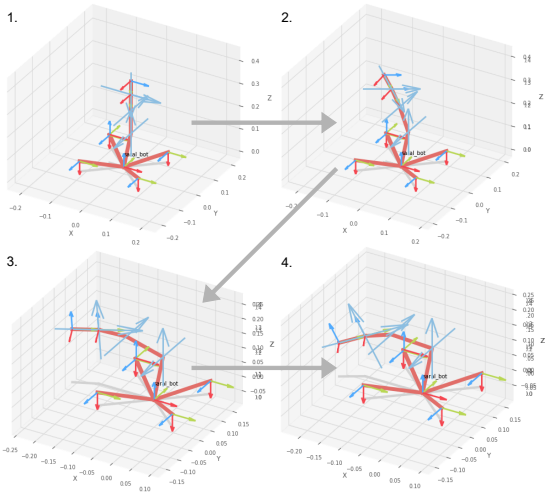

Fig. 13. Simulation showing the Plant Processing System Positioning Itself to Trim an Unhealthy Leaf-cluster.

inverse kinematics and tested the manipulation pipeline in a simulated environment. However, we encountered issues with RTB estimating poses beyond the mechanical limits of the joints, leading to unrealistic and unstable maneuvers. Moreover, RTB lacked integrated planning in presence of obstacles, which is crucial for preventing self-collisions and those with plants. To address these limitations, we transitioned to using Moveit2, a powerful and sophisticated platform for the plant manipulation tasks. Moveit2's built-in functionalities for inverse kinematics and obstacle avoidance enable us to perform realistic and stable maneuvering of the arm. In the simulation phase, we employed ROS2 control as the controller to manipulate the arm and achieve the desired goal pose. For all the relevant test cases that we expect to encounter throughout the course of the robot's operation, target poses are reached without violating any joint limits or self-collisions. To test the efficacy of our obstacle avoidance pipeline, we introduced virtual objects in the planning scene, representing the computation block at the rear of the robot and the healthy parts of the plant. We then tasked the arm with reaching target poses while avoiding these obstacles. After validating the manipulation pipeline in simulation, we deployed it on the physical robot and conducted real-world tests. Our system could successfully grasp and trim unhealthy leaves and flowers from the plant models to be used in the competition, thus demonstrating the effectiveness of our manipulation system. The transition from RTB to Moveit2 proved to be a significant improvement, enabling us to overcome the limitations of RTB and achieve stable and reliable maneuvers. The comprehensive testing and validation process, spanning both simulation and real-world experiments, gives us confidence in the readiness of our manipulation system for the competition.

VIII. Achievement of Objectives

Throughout the development process of our robot, we have maintained our focus on achieving the objectives and design criteria established in Sec III of this report. By leverag-

ing innovative hardware enhancements, a modular software architecture, and advanced algorithms, we have created a robot that successfully addresses the challenges posed by the competition. Now, we will discuss what we achieved from our two primary objectives.

Our first objective was to develop a robot capable of autonomously navigating the competition arena while adhering to the specified size constraints. The integration of the ArmPi Pro chassis with custom hardware additions, such as the linear actuator and depth cameras, has allowed us to create a compact and maneuverable platform. The use of RTABMAP for simultaneous localization and mapping, combined with efficient path planning and motion control algorithms, enables our robot to navigate the arena with precision and reliability.

Our second objective was to achieve accurate and robust plant inspection and selective plucking/trimming. The incorporation of an RGBD camera mounted on the linear actuator at the rear of the robot provides a clear view of the plants, allowing for efficient detection and localization of leaves and flowers. Our HSV-based color thresholding and contour detection algorithms, coupled with point cloud processing, enable the robot to distinguish between healthy and unhealthy plant parts with high accuracy. The integration of Moveit2 for inverse kinematics and obstacle avoidance ensures precise and collision-free manipulation of the robot arm during the trimming process. In addition to the competition objectives, we also aimed to create a modular and scalable system that could be easily adapted and maintained. The use of ROS2 as the primary software framework facilitates a modular and reusable codebase, allowing for the seamless integration of new features and algorithms. The separation of concerns between the Behaviour Coordinator, Navigation System, and Plant Processing System promotes a clean and maintainable software architecture. Furthermore, the utilization of Docker for environment management ensures consistent and reproducible deployments across different systems.

Throughout the development process, we've been conducting extensive tests and evaluations to validate the performance and reliability of our robot. The achievement of these objectives is a testament to the hard work, dedication, and technical expertise of our team. The successful integration of navigation, perception, and manipulation capabilities in a compact and efficient package demonstrates the potential for autonomous robots to revolutionize the way we approach tasks such as plant inspection and selective harvesting. Looking ahead, we believe that the knowledge and experience gained through this project will serve as a foundation for further advancements in agricultural robotics. The modular and scalable nature of our system opens up possibilities for adapting the robot to various agricultural applications beyond the scope of the competition. By continuing to refine and expand upon the capabilities demonstrated by our robot, we aim to contribute to the development of innovative solutions that address the challenges faced by modern agriculture, ultimately promoting efficiency, sustainability, and improved crop health.

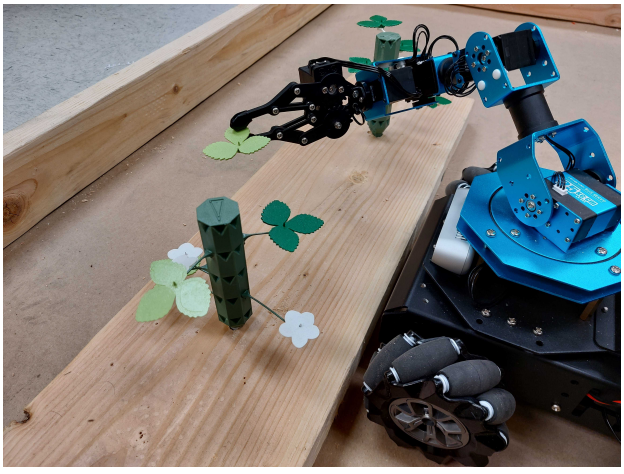

Fig. 14. Plant Processing System Trimming an Unhealthy Leaf-cluster with a 5-DoF Arm and 1-DoF Lateral Movement

ACKNOWLEDGMENT

We would like to express our deepest gratitude to all those who have supported and contributed to the development of our robot for the 2024 ASABE Student Robotics Competition. First and foremost, we extend our sincere thanks to our faculty advisor, Dr. Parikshit Maini, for his unwavering guidance, expertise, and mentorship throughout the project. Their valuable insights and encouragement have been instrumental in helping us navigate the challenges and achieve our goals. We are grateful to the ASABE (American Society of Agricultural and Biological Engineers) for organizing this competition and providing a platform for students to showcase their skills and innovations in agricultural robotics. The competition has been a source of inspiration and motivation for our team, driving us to push the boundaries of our knowledge, skills and capabilities. We also extend our appreciation to the open-source community, particularly the developers and contributors of ROS2, RTABMAP, Moveit2, and other libraries and tools that have been integral to our robot's software stack. Their dedication to creating accessible and powerful software has greatly facilitated our development process.

REFERENCES

[1] S. Park and J. Kim, "Design and implementation of a hydroponic strawberry monitoring and harvesting timing information supporting system based on nano ai-cloud and iot-edge," *Electronics*, vol. 10, no. 12, p. 1400, 2021.

[2] S. G. Defterli *et al.*, "Review of robotic technology for strawberry production," *Applied Engineering in Agriculture*, vol. 32, no. 3, pp. 301–318, 2016.

[3] G. Ren, H. Wu, A. Bao, T. Lin, K.-C. Ting, and Y. Ying, "Mobile robotics platform for strawberry temporal–spatial yield monitoring within precision indoor farming systems," *Frontiers in Plant Science*, vol. 14, p. 1162435, 2023.

[4] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of field robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[5] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074

[6] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.