INTELLIGENT AGRICULTURE ROBOT FOR TEA PLANTATION PRESERVATION : TEABOT

Hettiarachchige Thulya Madhawee Premathilake

IT20265410

BSc. (Hons) Degree in Information Technology Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

September 2023

INTELLIGENT AGRICULTURE ROBOT FOR TEA PLANTATION PRESERVATION : TEABOT

Hettiarachchige Thulya Madhawee Premathilake

IT20265410

Final Report Documentation Submitted in Partial Fulfillment of the Requirements for the Bachelor of Science Information Technology Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

September 2023

DECLARATION

I declare that this is my own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor: Date:

ABSTRACT

Tea cultivation stands out as one of the most crucial export commodities, making a substantial contribution to the Gross Domestic Product (GDP). Over the past few decades, there has been a gradual shift in the workforce towards various other occupations. Unlike many other crops, tea cultivation demands costly and meticulous maintenance. With a shortage of labor, maintaining tea estates has become increasingly challenging, leading to reduced yields. In response, tea estate owners have transitioned to cultivating low-maintenance crops. Enter the TeaBot, an advanced robot designed to replace human labor in tasks such as watering and fertilizing vast tea estates. What sets the TeaBot is its ability to operate effectively in rugged outdoor terrain and navigate through tea plantations without the need for specialized infrastructure. Considering that tea plants require continuous hydration and nutrients for optimal crop yield, the TeaBot plays a pivotal role in enhancing efficiency while reducing resources wastage. The key to its operation lies in translating provided linear and angular velocities into precise motion for its four motor-powered wheels, achieved through a sophisticated motor control algorithm. Additionally, an autonomous navigation method and stem identification method have been developed, employing two distinct approaches: a deep learning-based computer vision method and a classical computer vision-based approach. The deep learning-based autonomous navigation employs the U-Net architecture, while the stem identification model has been trained using the MobileNetV2 architecture. These models accurately identify the path and the stem, but some limitations remain, such as difficulties in controlling various environmental factors and the diverse growth patterns of tea plant stems. The choice of the classical computer vision method is based on its notable attributes, including high precision, minimal resource usage, and optimal efficiency. As a result of that, the liquid fertilization process has become highly efficient, marking a significant advancement in tea cultivation.

Keywords: Precision Agriculture, Computer Vision, Robotics, Machine Learning, Autonomous Navigation

ACKNOWLEDGEMENT

This research project would not have been possible without the guidance and support of our supervisor and co-supervisor. We would like to express our heartfelt gratitude for the immense support, guidance, and motivation provided by Mrs. Shashika Lokuliyana, who serves as our supervisor, and Mrs. Narmada Gamage, our co-supervisor. We would also like to take this opportunity to extend our deep appreciation to our external supervisor, Mr. Rajitha De Silva, for his tremendous guidance, leading this research from its inception to its successful completion. Even though he was pursuing a Ph.D. in the United Kingdom, he always made time for our meetings and played a pivotal role in enhancing this project to its fullest potential.

My team and I are particularly grateful to Mr. Thiwanka Cholitha Hettiarachchi, a graduate of SLIIT, who generously assisted and guided us. He motivated us by sharing his experiences like a brother. Additionally, we are thankful to Mr. B.V. Dhanayake for graciously providing access to his tea estate (Magedara Estate) without hesitation during our testing phase.

I would also like to extend my appreciation to my fellow research group members, with whom I had the pleasure of working on this project. Special thanks are due to Imalka Gunawardana, the leader of our research group, for his exceptional leadership in guiding this project to success.

Furthermore, I am deeply thankful to my family for their unwavering patience, support, and provision of necessary resources to meet our needs and cover other expenses. Finally, I would like to express my gratitude to everyone who collaborated with us, contributing to the effectiveness and success of this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
GDP	Gross Domestic Product
CNN	Convolutional Neural Network
EDB	Export Development Board
ML	Machine Learning
ResNet	Residual Neural Network
ROS	Robot Operating System
PC	Personal Computer
VGG	Visual Geometry Group
CSV	Comma-Separated Values
RGB	Red Green Blue
HSV	Hue Saturation Value
OpenCV	Open Source Computer Vision Library
NumPy	Numerical Python
SciPy	Scientific Python
JSON	JavaScript Object Notation

LIST OF TABLES

LIST OF APPENDICES

1. INTRODUCTION

1.1. Background & Literature Survey

Agriculture is one of the world's many essential sectors. It has a main key role in the production of food, fiber as well as fuel for the world's growing population [1]. The process of growing and raising animals for human consumption, among other things, is part of agriculture. Over time, agriculture has evolved and adopted new technologies and practices to increase productivity and efficiency. The population would be increased in the coming years in an unexpected way. Hence, due to that situation, all of the wants and needs are going to be led to an increase in demand. Most importantly in food. Considering this fact this would mean that the agricultural field is going to be more important and necessary even more important than today [1]. Agriculture is an essential part of Sri Lanka's economy, providing food, income for farmers and rural communities, and the job sector. The rice and tea industries are making a huge impact on the country's well-being. More specifically in the country's economy [2]. People who have better knowledge in the agricultural field could manage sustainably this sector. As a result of that, it could lead to a positive impact on the environment [3]. Not only those specialized people, but as part of the environment everyone has opened up situations to come up with novelty ideas for improving productivity, food security, and environmental sustainability in the agricultural region [1], [3].

1.1.1. Challenges in Sri Lanka's tea industry

When considering Sri Lanka's agriculture sector, the tea industry is one of the main important components. The International Labour Organization claims that Sri Lanka ranks as the world's fifth-largest producer among all countries [4]. Also, more than a million people are employed in this industry. Under this people category, not only the tea state workers but also small-scale owners, tea pluckers, and collectors can be defined [4]. According to the Sri Lanka EDB, the tea sector is a major revenue source in terms of foreign exchange earnings. Almost 11% of all export revenues are gained through the tea industry [5]. The tea industry is facing numerous issues [6]. One of the pressing issues in the farm sector is related to labor availability [6]. In addition, there are maintenance problems due to continuous removal of nutrients from soil through harvesting, which makes fertilizer application necessary for maintenance of yield [7].

1.1.2. Agricultural practices in tea plantations

Utilizing organic, inorganic, green manure, and bulk manure fertilizer treatments, integrated plant nutrition management has been practiced on tea plantations [8]. Farmers were permitted to start using synthetic fertilizers on tea and a few other crops while continuing to forbid their use on others, but by that time, much of the harm had already been done [9]. The disruption this created to Sri Lanka's agricultural industry couldn't have happened at a worse moment. Numerous crises had already been causing damage on the tourism sector, the primary source of foreign cash and the foundation of the nation's economy [10]. Irrigation is an essential factor in tea production, especially in the low elevation tea growing region of Sri Lanka, which produces 60% of the national production [11]. The most limiting factor affecting productivity in this region is the short but intense inter-monsoonal dry period. Irrigation, along with other methods like cultivar selection, is likely to improve productivity [11]. The trials showed that irrigated tea plants had better root and canopy structure than non-irrigated plants, resulting in higher yields [11]. In recent years, IoT-based drip irrigation has been introduced for tea plantations in Sri Lanka [12]. This technology uses precision agriculture and IoT to optimize water usage and increase crop yield. However, it is unclear how widely this technology has been adopted on Sri Lankan tea estates. Traditional tank irrigation systems are one the old system that have been using for watering. It is unclear how much these traditional tank irrigation systems have evolved specifically for use on modern-day tea estates. Overall, while there have been some advancements in irrigation technology for tea production in Sri Lanka, it is unclear how widespread these advancements are across all tea estates. Traditional tank irrigation systems continue to be used alongside more modern technologies like drip irrigation which has several numbers of disadvantages [13]. Drip irrigation systems require a lot of money to set up and also need a lot of money for upkeep. To make sure that tea plantations are well-maintained, it is important to find new and better ways to take care of them. This is where innovative sensor-driven methods come in, which use information technology [14] to improve the way tea plantations are cared for. One of the ways this has been done is by using intelligent robots [15] that can water and fertilize the plants [16]. However, it [16] can be challenging to get the robot to move

around accurately in large fields. Moreover, using Lora can be expensive, and giving instructions in large fields can be complicated [17].

1.1.3. Methods currently in use for tea conservation

Although there have been several agricultural robots designed for various purposes [18], [19]. Those have not been able to address the gap highlighted in the previous issues. Developing a robot for liquid fertilization of tea plantations is a complex task due to the challenges posed by accurately identifying the stems in different environmental and weather conditions. Utilizing advanced machine learning techniques to accurately identify the end of the stem may prove to be the most effective approach [20], [21]Previous studies have primarily concentrated on the sustainability of forest operation [22], weeding [23], harvesting [24], [25] and seed cutting [26]. However, there has been a lack of research on identifying the end of stems for liquid fertilization processes. This deficiency is evident when compared these studies, as indicated in Table 1.1.

The “TeaBot” project aims to address this research gap by utilizing advanced machine learning techniques such as image processing to identify the end of the stem accurately. It is worth noting that the stem identification task performed by “TeaBot” is a critical component of the fertilization process. Therefore, the project places great emphasis on identifying the end of the stem with utmost precision, as it plays a crucial role in the overall functionality of the robot. The stem identification process under “TeaBot” has garnered significant positive effective reasons and has proven to be highly beneficial for improve the preservation of this tea plants. The advantages of this component can be summarized as follows:

1. Improved crop quality: The precision of the stem identification ensures that fertilizers are applied only where needed, which can result in higher crop quality. Also it ensures that tea plants receive the nutrients they need to grow and produce high-quality leaves.

2. Reduced fertilizer usage: By accurately identifying the end of the stem, TeaBot ensures that the liquid fertilizer is applied only where it is needed. This minimizes the amount of fertilizer used, which can significantly reduce the cost of fertilization and

help preserve natural resources. Additionally, the reduced fertilizer usage can also minimize the risk of nutrient runoff, which can have negative environmental impacts.

3. Improved water conservation: “TeaBot” can also be used to apply liquid fertilizer directly to the root zone of tea plants, which can help conserve water. Traditional methods of fertilization involve spraying water and fertilizer over a wide area, which can result in significant water wastage. By using “TeaBot” to apply liquid fertilizer directly to the root zone, tea state owners can ensure that their tea plants receive the necessary nutrients while also conserving water.

4. Reduced labor costs: The stem identification process mainly led to automate the fertilization process, reducing the need for manual labor. This can result in significant cost savings for the tea state owners, particularly in areas with labor shortages or high labor costs.

5. Improved safety: By reducing the need for manual process of liquid fertilizer, “TeaBot” able to improves safety for workers on tea plantations. This reduces the risk of accidents and injuries associated with manual labor.

In summary, the “TeaBot” stem identification provides several advantages for reducing resource wastage, including reduced fertilizer usage and improved water conservation. These benefits can help tea plantation owners preserve natural resources, reduce costs, and contribute to more sustainable plantation practices. The system's precision, efficiency, and sustainability benefits make it an indispensable tool for tea state owners seeking to optimize their operations and increase their crops while also reducing their environmental impact.

1.2. Research Gap

The use of computer vision and deep learning techniques for detecting and tracking tree and plant stems has been extensively studied in recent years. However, there is a lack of research specifically focused on identifying the end of tea plant stems for the purpose of optimizing tea production. As mentioned before such as Real-Time Computer Vision for Tree Stem Detection and Tracking [22] and Grape stem detection using regression convolutional neural networks [24], have demonstrated the potential of computer vision and deep learning techniques in accurately detecting and tracking tree and plant stems. However, these studies do not focus on the specific problem of identifying the end of tea plant stems for the purpose of controlling a robot for liquid fertilizing. Similarly, Stem localization of sweet-pepper plants using the support wire as a visual cue [25] and X-Ray-Based Stem Detection In An Automatic Tomato Weeding System [23] focus on stem localization for the purpose of supporting plants or weed control, respectively. Sugarcane stem node detection and localization for cutting using deep learning [26] is focused on detecting and localizing nodes on sugarcane stems for harvesting purposes. Thus, there is a clear research gap in the specific area of identifying the end of tea plant stems for the purpose of optimizing tea production. The purpose of this research is to develop a machine learning model specifically trained on tea stem images and the design and implementation of a robot control system that uses the end of the tea stem coordinates as input for liquid fertilizing the plant. Additionally, there are specific challenges to be addressed in this research, such as variability in tea plant growth patterns and environmental factors, the need for real-time processing, and the need for accurate control of the robot for precise watering and fertilization. Overall, this research has the potential to make significant contributions to the field of precision agriculture and the optimization of tea production. By addressing this research gap and developing a practical solution for identifying the end of tea plant stems and controlling a robot for liquid fertilizing, it could be improved the efficiency and quality of tea production.

Table 1.1 : Comparison of former research

Research	Methodologies that used to identify the stem	Identify the end of the stem	Pass coordinators as the output	Aligned with liquid fertilizing process	Agricultural robot	Purpose
Real-Time Computer Vision for Tree Stem Detection [22]	CNN based (Object Detection)	✗	✓	✗	✗	Forest operations' sustainability
Stem detection in an automatic tomato weeding system [23]	X-ray based	✗	✗	✗	✓	Weeding system
Grape stem detection [24]	Segmented based	✗	✗	✗	✓	Harvesting
Stem localization of sweet pepper [25]	X-ray based	✗	✗	✗	✓	Harvesting
Sugarcane pre-cut seed [26]	Object detection and image processing	✗	✗	✗	✗	Seed cutting
TeaBot	CNN based	✓	✓	✓	✓	Liquid fertilizing

The proposed research solution will include features that were not proposed in previous studies. This solution will be unique and innovative, incorporating compared to the prior studies. The table above demonstrates that the Teabot has the capability to detect the end of the stem, which can optimize production in tea plantations. This highlights the potential of using robotics and advanced sensing technologies to improve large scale tea state management and increase efficiency.

1.3. Research Problem

As mentioned above, the tea cultivation industry is facing the challenge of maintaining consistent and precise liquid fertilizing of tea plants, which is essential for optimal plant growth and yield. According to the prior studies also there were not included any methodologies for identify the end of the stem and pass the correct coordinators. Tea plant structure is different another to another. Also, above discussed studies were not adaptive for variation of the stem. However, achieving this presents a significant challenge due to the variability in tea plant structures, as well as environmental factors that may affect the performance of the identification process. Therefore, the research problem is, How to give the coordinates of the detected tea stem through the developed algorithm in order to fulfill the liquid fertilizing process? Moreover, since this is the main research problem to fulfill this problem there are many other sub problems as well. Those are,

1. How to identify the end of the tea stem precisely?

2. How to identify different variations of the tea stem?

3. What type of different conditions needed to be tested to ensure its accuracy and reliability?

4. What methodologies can be used to pass the coordinators to the robot controller?

1.4. Research Objectives

1.4.1. Main objective

Automation watering mechanisms have been already implemented. Although drip [27] and center pivot irrigation systems [28] and GPS or lidar technologies have been used for this purpose, they are not resilient to varying environmental conditions, and their accuracy is limited. To address these challenges, TeaBot has been proposed to navigate tea plantations and accurately, apply liquid fertilizer to the precise location where it is needed. It uses computer vision-based path identification to move through the plantation and detects the end of the tea stem using image processing. TeaBot is equipped with pressurized water nozzles that enhance the accuracy and precision of the liquid fertilization process. It helps to reduce labor costs, preserve the health and productivity of tea plantations, and improve overall sustainability.

The identification of the end of the tea stem is a critical component of the TeaBot robot's functionality, as it enables the robot to precisely target and apply liquid fertilizer to the tea plantations. Without the ability to identify the end of the stem, the robot cannot accurately locate the correct position to liquid fertilizing. If the end of the stem is not identified, the robot may apply liquid fertilizing to the wrong location, resulting in the tea plant's damage and may leading to inefficient use of resources. By using advanced image processing techniques, the TeaBot can accurately detect the end of the stem and provide the corresponding coordinates to the robot controller. This allows the robot to apply liquid fertilizer directly to the roots, promoting healthy growth and optimal crop yields. This process mainly affected to the liquid fertilizing process. This is not only promoting efficient resource use but also ensures the sustainability of the tea plantations by reducing the risk of overwatering or under fertilization, which can result in long-term damage to the soil and plants. Overall, main objective can be defined as follow, develop an algorithm to precisely identify the end of the stem and pass the coordinators to the liquid spraying controller.

1.4.2. Sub objectives

In order to achieve the main objective, several subsidiary goals have been identified to facilitate the overall process. Foremost among these is the imperative task of tea stem identification, which is a pivotal prerequisite for the successful execution of the liquid fertilization process. The key consideration in this endeavor is to ensure the utmost accuracy while minimizing the time required for tea stem recognition. This is particularly critical, as the TeaBot necessitates watering while concurrently ascertaining, the real-time results of tea stem identification. Delayed result reporting is not a viable option, given the exigency of real-time processing. Furthermore, in the context of tea plantation management, it is crucial to acknowledge the variegated ages of the tea plants. This necessitates the formulation of the following specific sub-objectives:

1. The implementation of a highly precise algorithm for the detection of tea stems.
2. Rigorous testing and validation of the algorithm's performance across a range of diverse scenarios.
3. The fine-tuning of the algorithm, informed by the performance metrics gathered during testing and validation exercises.

These sub-objectives collectively form an integral part of the overarching strategy designed to meet the primary objective, ensuring efficient and effective tea stem identification for enhanced liquid fertilization process.

2. METHODOLOGY

TeaBot, the proposed research project, will be developed using the following defined approaches in order to provide an efficient and effective maintenance solution to tea plantations and preservation.

2.1 Requirement Analysis

First and foremost, the initial step involves acquiring a comprehensive understanding and knowledge by engaging in a thorough process of requirement analysis. This process is essential for gaining insight into the specific needs, preferences, and objectives relevant to the task. To gain a better understanding of large-scale tea agriculture, Conducted field visits and stakeholder interviews with tea state owners and workers. One major challenge facing these fields is the ongoing maintenance of tea plantations, which requires regular watering and fertilization which is referred to as liquid fertilization. However, this presents difficulties due to factors such as a shortage of human resources, resource wastage, and the high initial cost of automating the process. The TeaBot can address these challenges. With the intention of, gathering a considerable amount of data from people who are working in this field. Then analyzed the watering and fertilization process of current large-scale tea agriculture. As mentioned above, by identifying the end of the tea stem, the liquid fertilization process can be done in an efficient way as well as it is beneficial for use recourse without wastage. To do this, needed to develop an algorithm to identify the end of the tea stems precisely, collected a series of images captured at various growth stages, vividly illustrated in Figures 2.2, 2.3, and 2.4, allowing for a comprehensive visual representation of the developmental progression. Furthermore, conducted experiments under diverse lighting conditions, as visually presented in Figures 2.5 and 2.6, thereby enabling to examine the influence of varying lighting scenarios on our findings and results. The main functional and non-functional requirements of the stem identification component are illustrated in Figure 2.1.

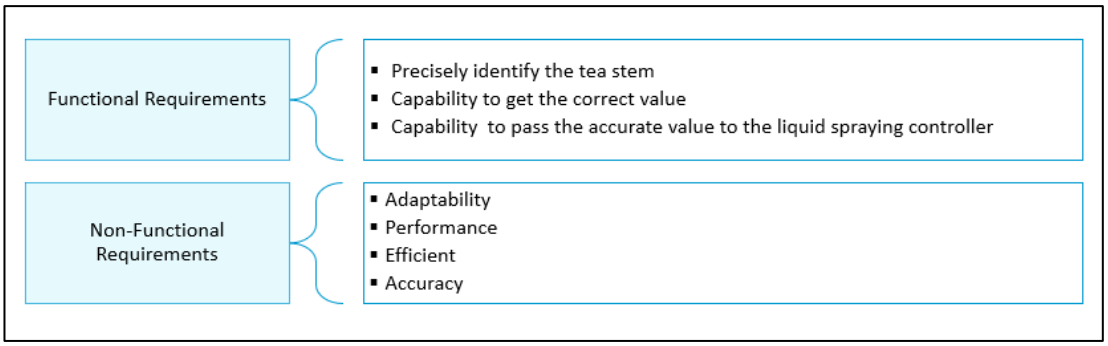


Figure 2.1 : Functional & non-functional requirements

Figure 2.2 : Stem structure of a small tea crop

Figure 2.4 : Stem structure of a middle tea crop

Figure 2.3 : Stem structure of a large crop

Figure 2.5 : Low sunlight

Figure 2.6 : High sunlight

2.2 Feasibility Study

The feasibility study for the requirement analysis of large-scale tea agriculture identifies technical, economic, and operational feasibility considerations. The study concludes that the image processing methodology developed appears feasible and

could help optimize plantation maintenance, increase yield, and profitability. However, further study is necessary to assess the economic feasibility of the project.

2.3 Overall System Architecture

TeaBot is capable of efficiently navigating both flat and sloped terrain, all without the need for specially designed tracks. This remarkable robot boasts dimensions of 12 inches in height, 20 inches in length, and 24 inches in width, with an impressive ground clearance of 4 inches. These dimensions enable TeaBot to effectively maneuver through the latter sections of tea plant rows. A Raspberry Pi serves as the main controller for the robot's operations, serving as the central controller. Real-time coordinates are provided by cutting-edge computer vision algorithms, ensuring precision in its movements. The primary goal of this robotic marvel is to reduce operational costs, enhance efficiency, and yield a higher return on investment within the tea plantation industry.

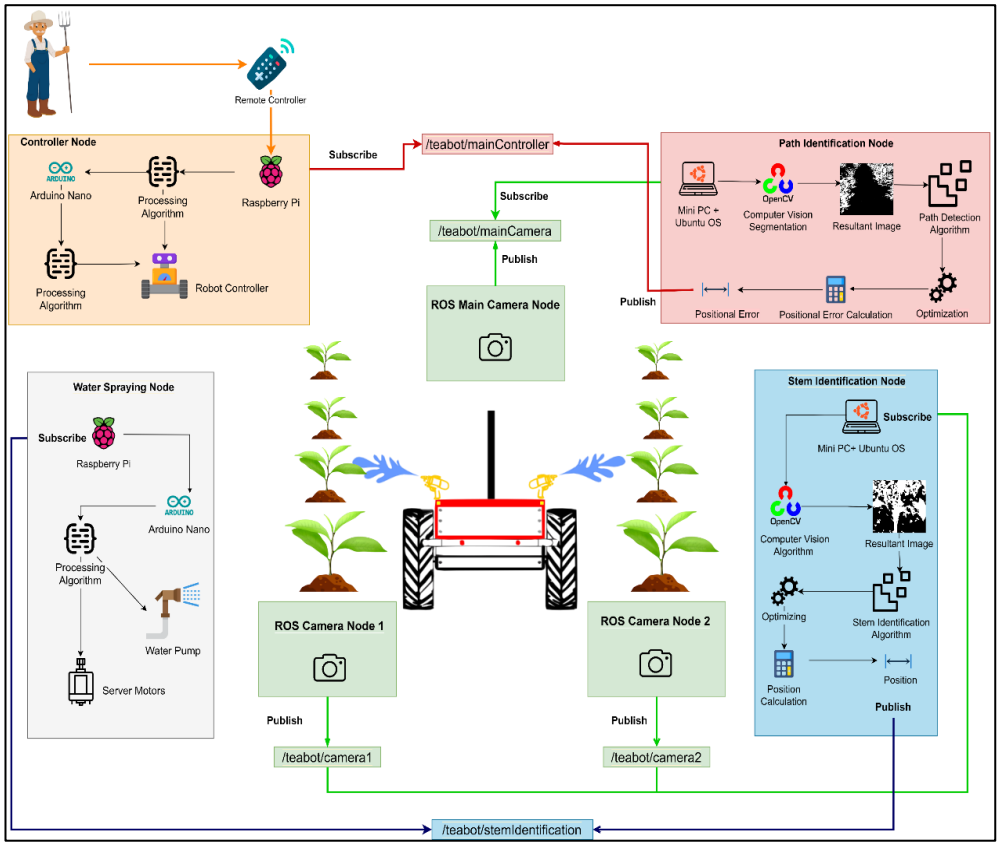


Figure 2.7 : System overview diagram

The proposed system architecture for TeaBot is neatly divided into four key components, as illustrated in Figure 2.7. The TeaBot excels in several specialties, including automated navigation, the ability to traverse challenging terrains, and pinpoint-accurate stem identification for precise fertilization. Deep learning algorithms power TeaBot's precise navigation and stem recognition capabilities. Motor control algorithms seamlessly convert velocities into precise wheel actions and translate stem coordinates into servo motor actions, ensuring the utmost precision. TeaBot's system components connectivity is illustrated in Figure 2.8.

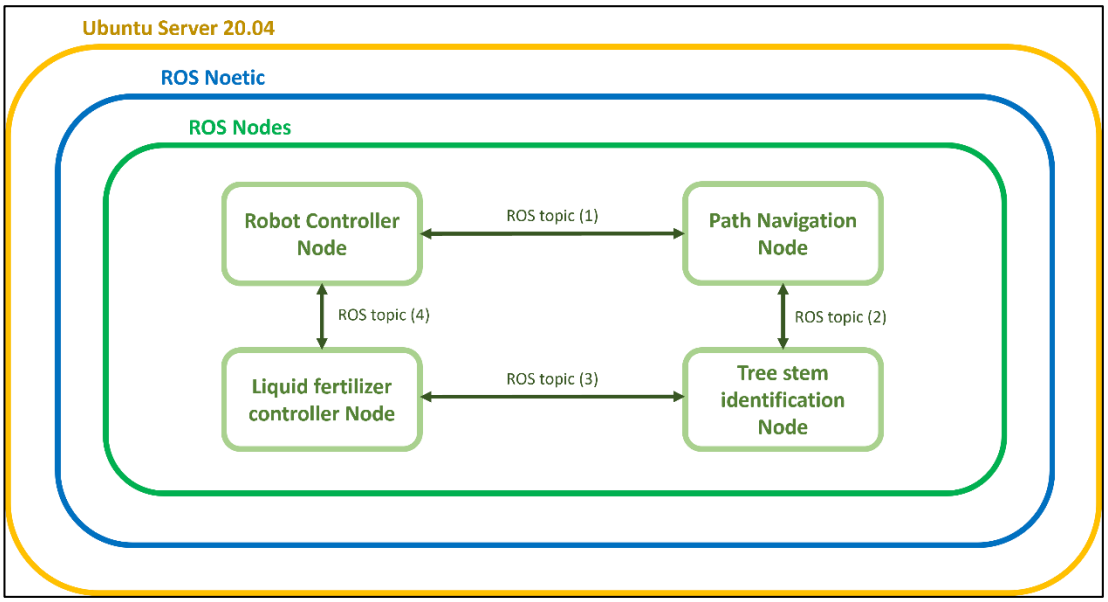


Figure 2.8 : System components

When discussing the implementation of TeaBot, it's crucial to underscore its key features, including navigation, deep learning-based stem identification, and path navigation algorithms. This emphasis is essential to underscore TeaBot's pivotal role in sustainable large-scale tea cultivation. It addresses labor shortages and optimizes yields, potentially revolutionizing tea estate management and reestablishing Ceylon Tea's global prominence. In particular, by showcasing the optimization of the path navigation algorithm, TeaBot enhances its accuracy and efficiency, further solidifying its potential to revolutionize the tea estate management industry and elevate Ceylon Tea's standing on the global stage.

2.4 Stem Identification Component

TeaBot is composed of four interconnected ROS nodes, as illustrated in Figure 2.8. These nodes encompass the following components: the robot controller node, the path navigation node, the tea stem identification node, and the liquid fertilizer controller node. The stem detection node is deployed within the mini-PC and utilizes a Logitech C310 web camera to capture video frames of the tea plants. Meanwhile, the front-facing web camera of TeaBot is responsible for capturing video frames while the robot is navigating.

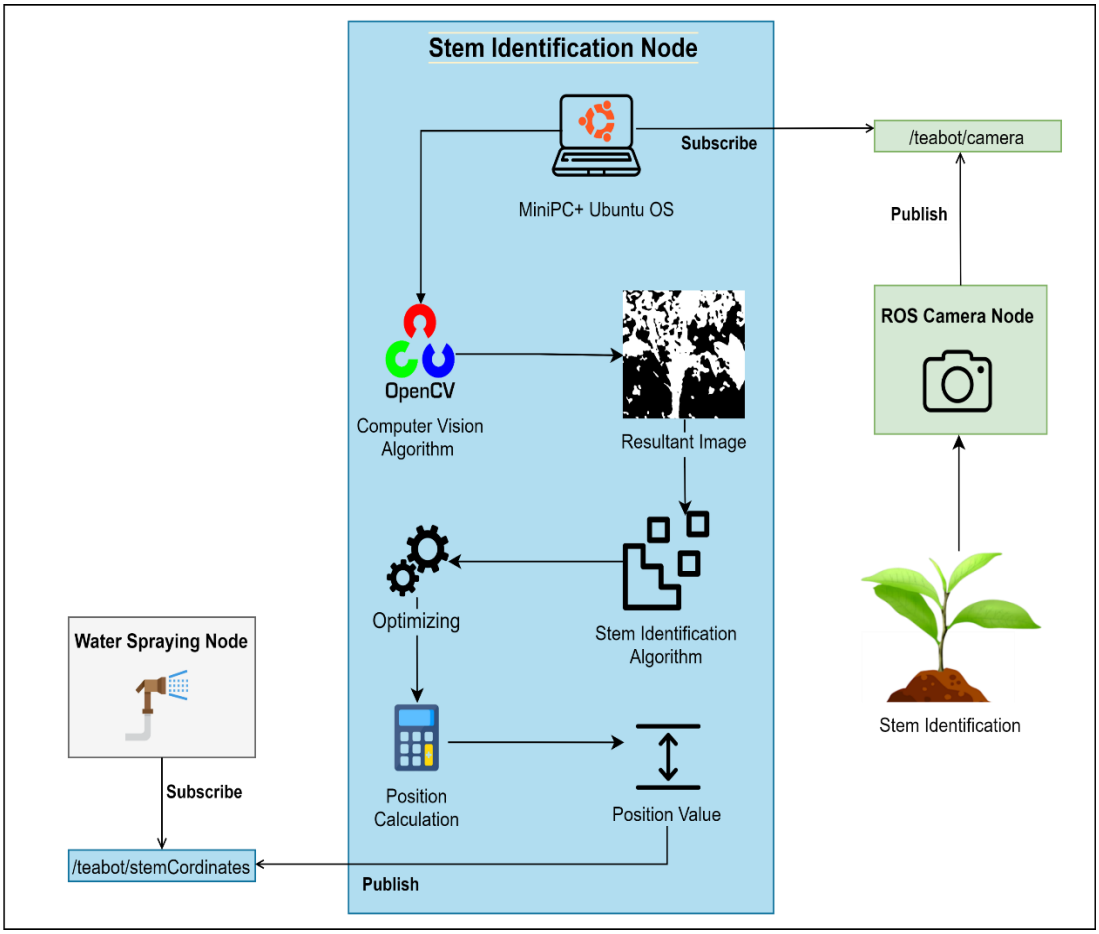


Figure 2.9 : Stem identification component overview

After the tea plant frames are captured, it can be used to utilized for predicting the location of the tea stem. The web camera node is also hosted on the mini-PC. This arrangement of the tea stem identification node and web camera on the mini-PC offers improved performance, a critical nonfunctional requirement. Initially, the stem identification process utilized an image processing algorithm. However, when this algorithm was deployed on the Raspberry Pi, it did not yield the intended results due to its high resource consumption. This is because all the web camera nodes were also deployed on the Raspberry Pi. Processing three web cameras simultaneously demands substantial computational power, which the Raspberry Pi was unable to provide, leading to delays in output. In response, a classic computer vision approach was adopted for the stem identification process, which significantly improved performance. The mini-PC handled all the cameras smoothly without any latency.

Once the camera captures the video, it will be transmitted to the computer vision algorithm that has been meticulously developed. Subsequently, this algorithm will be employed to generate a resulting mask. The resulting image is then fed into the computer vision algorithm, leveraging the power of classic computer vision techniques implemented in Python using the OpenCV library. This robust methodology will facilitate the identification of the tea stem, with a primary focus on utilizing the resultant mask obtained from the initial computer vision application as the key input for this computer vision-based algorithm.

The outcome of this algorithm will be a set of positional coordinates that pinpoint the precise location of the tea stem. These coordinates are subsequently relayed to the liquid fertilizer controller node, also known as the water spraying node. The transmission of this valuable positional data to the liquid fertilizer controller node plays a pivotal role in achieving accurate and targeted watering of the tea plant. This data exchange is facilitated through the use of the ROS. The overarching process described above is comprehensively illustrated in Figure 2.9.

2.4.1 Dataset creation for ML model requirements

There are numerous potential approaches for identifying the stem of a tea plant. Initially, it was contemplated that employing image processing techniques could lead to a successful solution for detecting the end of the tea stem. In order to effectively implement a machine learning model, the creation of a high-quality dataset becomes imperative. To achieve this, capture images throughout the process. Approximately 3000 images of tea stems gathered, as this is the typical quantity of images needed for training a machine learning model [29]. To ensure that the model remains unbiased, images collected at different growth stages, as depicted in Figures 2.2, 2.3, and 2.4. Moreover, it is necessary to collect a dataset that can identify the end of the tea stem under varying stem structure conditions. Images will be captured under different lighting conditions as well, including varying shadow conditions in different areas with different environmental effects, as shown in Figures 2.5 and 2.6. The accuracy of the dataset labeling process is crucial to the machine learning model [30], and it should be done using the correct methodologies. Finally, the end of the tea stem image dataset transformed and labeled in a way that is appropriate for the machine learning model. That process illustrated in the Figure 2.10.

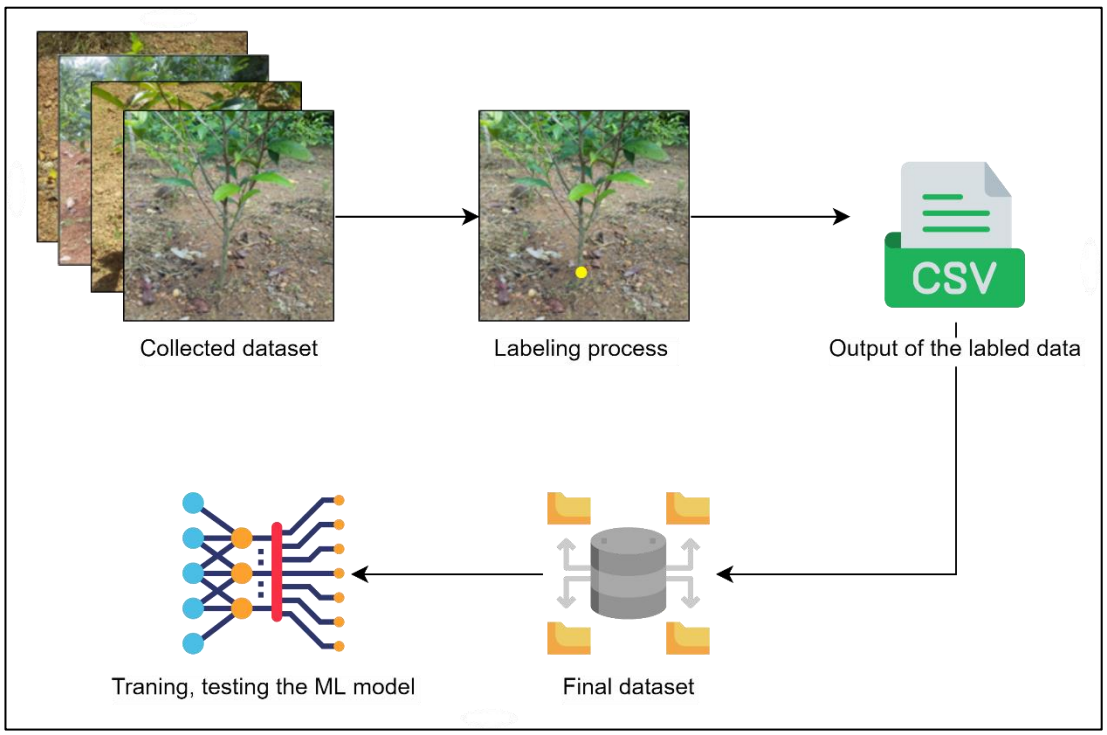


Figure 2.10 : Dataset creation

2.4.2 Development of the ResNet50 model

To identify the tea stem, several options are available. As mentioned earlier, one of the alternatives discussed involves developing an image processing algorithm as the primary choice. The challenge at hand involved the impracticality of assigning a distinct class label to every conceivable combination of (x, y) coordinates representing a tea stem in an input image. In contrast to classification tasks that result in discrete labels, regression offers the advantage of making predictions for continuous values within a predefined range [31]. The main aim is to leverage the ResNet50 model, which operates within a Convolutional Neural Network (CNN) architecture, to examine and pinpoint the endpoint of a tea stem. This ResNet50 model will serve as the foundational network for subsequent fine-tuning. In the context of this scenario, needed to define a CNN architecture that is well-suited for our objectives and has been pretrained on a substantial dataset. The preference for ResNet50 over VGG16 is substantiated by previous research, which noted that "although both models extract similar features from the same images, ResNet-50 exhibits greater sensitivity to the edges of target objects" [32]. Furthermore, a comparative analysis between the ResNet50 and VGG16 models reveals variations in accuracy and loss values, further supporting our choice [33].

The progression of the ResNet50 model's development adhered to a methodical and structured approach, which comprised a sequence of distinct sub-objectives designed with the overarching goal of enhancing the model's performance in terms of accuracy and precision. These sub-objectives encompassed a meticulously orchestrated set of pivotal steps, each contribute to the model's refinement.

- Dataset creation and data preprocessing
- ResNet50 model development and training
- Testing and model optimization
- Performance analysis

With these sequential steps, the ResNet50 model was methodically created and refined, ensuring that it met the desired objectives. The subsequent section will provide a detailed explanation of the ResNet50 model development process.

Developed a Python script designed for managing a dataset in preparation for training and testing a machine learning model. The script begins by importing several essential libraries, including TensorFlow, scikit-learn, Matplotlib, OpenCV, and pandas. These libraries provide functionalities for deep learning, data manipulation, and data visualization, all of which are critical components of machine learning projects.

Afterward, the script defines various paths to access the dataset and a CSV file containing labeled data. Specifically, it divides these paths into the base directory and image directory paths, respectively. Additionally, it establishes a path to the CSV file, which presumably contains information about the dataset, such as image filenames and corresponding labels. This CSV file typically stores annotations, and reading it is the initial step in gaining access to the dataset's information.

The code then proceeds to create three lists: 'data,' 'targets,' and 'filenames' as illustrated in Figure 2.11. The 'data' list is intended to store preprocessed image data, with each image converted into a NumPy array. The 'targets' list is used to hold the target values or annotations, which, in this case, appear to be coordinates (x and y). The 'filenames' list is utilized to maintain a record of the image filenames. The script systematically traverses the image directories, processing each image it encounters, and then seamlessly incorporates these images into the 'data' list. This iterative process ensures that every image is handled with precision and added to the collection, making it a essential step in the script's execution. Concurrently, it extracts the corresponding target values from the CSV file and appends the image filenames to the 'filenames' list. This process effectively pairs image data with its associated annotations and filenames.

The script subsequently directs its attention toward partitioning the dataset into distinct sets for training, validation, and testing. This data splitting procedure is a customary practice in machine learning, designed to guarantee the availability of distinct datasets for the purposes of training, model validation, and conclusive testing. It accomplishes this by dividing the data into training and testing segments, allocating 80% of the dataset for training, while reserving the remaining 20% for testing. The implementation of the partitioning the dataset can be seen in Figure 2.12.

```
data = []
targets = []
filenames = []
for image_dir in image_directories:
  # load the image and preprocess it
  image = load_img(image_dir, target_size=(224, 224))
  image = img_to_array(image)

  # update our list of data, targets, and filenames
  data.append(image)

  # get the image name from the image dir
  #filename = image_dir[]
  filename = os.path.basename(image_dir)

  row_df = label_df.loc[label_df['filename'] == filename]
  new_x, new_y = row_df['new_x'].values, row_df['new_y'].values
  targets.append(np.concatenate([new_x, new_y], axis = 0))
  filenames.append(filename)
```

Figure 2.11 : Implementation of the three lists

```
# convert the data and targets to NumPy arrays, scaling the input
# pixel intensities from the range [0, 255] to [0, 1]
data = np.array(data, dtype="float32") / 255.0
targets = np.array(targets, dtype="float32")

# partition the data into training and testing splits using 80% of
# the data for training and the remaining 20% for testing
split = train_test_split(data, targets, filenames, test_size=0.20,
  random_state=42)

# unpack the data split
(trainImages, valImages) = split[:2]
(trainTargets, valTargets) = split[2:4]
(trainFilenames, valFilenames) = split[4:]

split = train_test_split(valImages, valTargets, valFilenames, test_size=0.50,
  random_state=42)

# unpack the data split
(valImages, testImages) = split[:2]
(valTargets, testTargets) = split[2:4]
(valFilenames, testFilenames) = split[4:]


# write the testing filenames to disk so that we can use then when evaluating/testing
print("[INFO] saving testing filenames...")
f = open(TEST_FILENAMES, "w")
f.write("\n".join(testFilenames))
f.close()
```

Figure 2.12 : Implementation of the partitioning the dataset

When it comes to model fine-tuning, a pre-trained ResNet-50 model is employed for a specific task. Initially, the ResNet-50 model is loaded with pre-trained weights from the "imagenet" dataset, and its top layer is excluded, ensuring that only the feature extraction layers are retained. Furthermore, all layers of the ResNet model are frozen, preventing updates during the training process. The output tensor from the ResNet-50 model is stored. Subsequently, a series of convolutional layers is added on top of this output. These convolutional layers increase in number and gradually decrease the number of filters as they progress. These additional convolutional layers consist of 2048, 1024, 512, 256, and 128 filters, respectively, with ReLU activation functions, and padding is set to 'same.' Following the convolutional layers, the output is flattened to create a one-dimensional vector. This vector is then passed through a series of fully connected layers. The fully connected layers comprise one with 1024 units, followed by one with 256 units, another with 32 units, and the final one with 2 units, serving as the output for predicting the stem coordinates.

The model training process includes the following essential hyperparameters: the initial learning rate is set to 1e-4, the number of epochs is 30, and the batch size is 60. Likewise, the loss function for the model is specified as the Huber loss, and the optimizer is set to Adam with an initial learning rate. Part of the model layers can be found in Figure 2.13, 2.14 and 2.15. The model is then trained using the 'model.fit' method, which takes in the training images and their corresponding target values for regression. The validation data, batch size, number of epochs, and verbosity level are also defined at this stage. This is where the actual training takes place, and the model learns to make predictions based on the provided data. The training process continues for the specified number of epochs, and the model's performance is monitored during each epoch.

In the evaluation section, the code begins by evaluating the model's performance on a test dataset. It computes the loss and stores the result in the 'loss' variable, which is then printed to the console.

Moving on to the inference section, the code loads a pre-trained model from a specified location. It specifies the path to the model file and loads the model using TensorFlow's 'load_model' function. Following, a sample image for inference is defined. This image is read using OpenCV, resized to a 224x224 pixel size, and preprocessed by converting it to a NumPy array, scaling its values to a range of [0, 1], and expanding its dimensions to create a batch-like structure suitable for model prediction. The model is then used to predict the x and y coordinates. These predicted values are printed to the console. The image is reshaped to its original form and its pixel values are rescaled back to the [0, 255] range. A circle is drawn on the image at the coordinates specified by x and y with a radius of 4 pixels, and the circle is filled with a red color. This is done using OpenCV's 'circle' function. Finally, the resulting image is displayed using 'cv2_imshow.'

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input_1[0][0]']

 conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1_pad[0][0]']
                                )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1_conv[0][0]']
                                )

 conv1_relu (Activation)        (None, 112, 112, 64  0           ['conv1_bn[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1_relu[0][0]']
                                )

 pool1_pool (MaxPooling2D)      (None, 56, 56, 64)   0           ['pool1_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160        ['pool1_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2_block1_1_bn[0][0]']
 n)

 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2_block1_1_relu[0][0]']

 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block1_2_conv[0][0]']
 ization)

 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2_block1_2_bn[0][0]']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['pool1_pool[0][0]']

 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2_block1_2_relu[0][0]']
```

Figure 2.13 : ResNet50 model layer part 1

```
conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['pool1_pool[0][0]']

conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2_block1_2_relu[0][0]']

conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2_block1_0_conv[0][0]']
ization)

conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2_block1_3_conv[0][0]']
ization)

conv2_block1_add (Add)         (None, 56, 56, 256)  0           ['conv2_block1_0_bn[0][0]',
                                                                 'conv2_block1_3_bn[0][0]']

conv2_block1_out (Activation)  (None, 56, 56, 256)  0           ['conv2_block1_add[0][0]']

conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2_block1_out[0][0]']

conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block2_1_conv[0][0]']
ization)

conv2_block2_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2_block2_1_bn[0][0]']
n)

conv2_block2_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2_block2_1_relu[0][0]']

conv2_block2_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block2_2_conv[0][0]']
ization)

conv2_block2_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2_block2_2_bn[0][0]']
n)

conv2_block2_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2_block2_2_relu[0][0]']

conv2_block2_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2_block2_3_conv[0][0]']
ization)

conv2_block2_add (Add)         (None, 56, 56, 256)  0           ['conv2_block1_out[0][0]',
                                                                 'conv2_block2_3_bn[0][0]']

conv2_block2_out (Activation)  (None, 56, 56, 256)  0           ['conv2_block2_add[0][0]']
```

Figure 2.14 : ResNet50 model layer part 2

```
conv5_block3_out (Activation)  (None, 7, 7, 2048)   0           ['conv5_block3_add[0][0]']

conv2d (Conv2D)                (None, 7, 7, 2048)   37750784    ['conv5_block3_out[0][0]']

conv2d_1 (Conv2D)              (None, 7, 7, 1024)   18875392    ['conv2d[0][0]']

conv2d_2 (Conv2D)              (None, 7, 7, 512)    4719104     ['conv2d_1[0][0]']

conv2d_3 (Conv2D)              (None, 7, 7, 256)    1179904     ['conv2d_2[0][0]']

conv2d_4 (Conv2D)              (None, 7, 7, 128)    295040      ['conv2d_3[0][0]']

flatten (Flatten)              (None, 6272)         0           ['conv2d_4[0][0]']

dense (Dense)                  (None, 1024)         6423552     ['flatten[0][0]']

dense_1 (Dense)                (None, 256)          262400      ['dense[0][0]']

dense_2 (Dense)                (None, 32)           8224        ['dense_1[0][0]']

dense_3 (Dense)                (None, 2)            66          ['dense_2[0][0]']

==================================================================================================
Total params: 93,102,178
Trainable params: 69,514,466
Non-trainable params: 23,587,712
__________________________________________________________________________________________________
```

Figure 2.15 : ResNet50 model layer part 3

2.4.3 Development of the MobileNetV2 model

The decision to transition the model from ResNet50 to a more lightweight alternative on the Raspberry Pi was primarily motivated by the constrained computational resources available for this particular task. To mitigate this limitation and effectively identify the tea stem, according to the these references [34]–[36], which collectively indicated that MobileNetV2 was the most suitable lightweight replacement for ResNet50.

The principal adjustment made during this transition pertained to the architectural design of the layers within the MobileNetV2 model. In the subsequent section, it offers a comprehensive explanation of the specific changes made to the model. This exposition will provide a thorough insight into the implemented alterations. It's essential to emphasize that, apart from the modifications to the model's architecture, all other fundamental components and characteristics of the original ResNet50 model remained unaltered. This includes parameters such as the number of epochs, learning rate, and the loss function, among other things. This ensures that the model's essential qualities and capabilities have been preserved while accommodating the resource limitations imposed by the Raspberry Pi platform. Regarding the architecture of the MobileNetV2 layers, Figure 2.16 illustrates this architecture. The process involves loading a pre-trained MobileNetV2 model with weights derived from the 'imagenet' dataset.

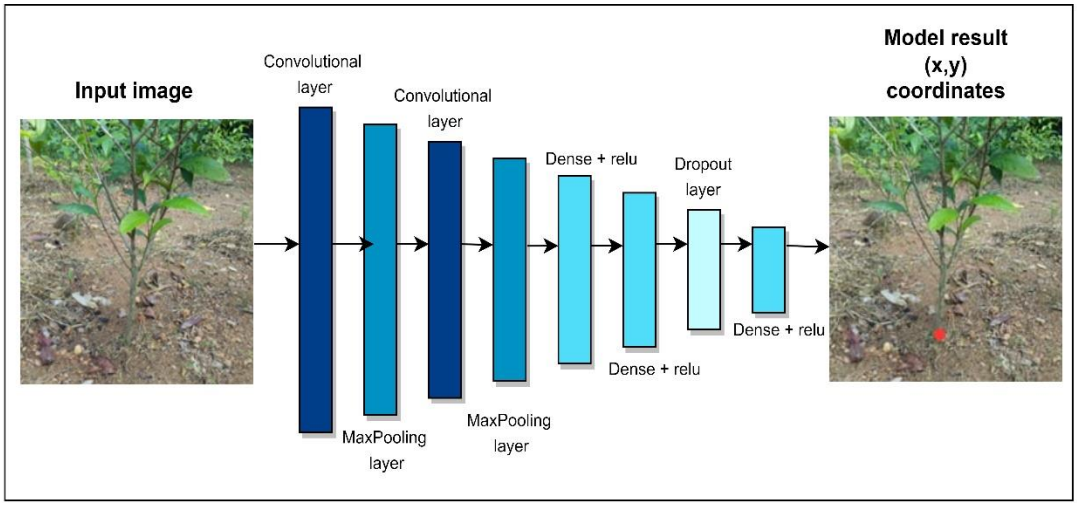


Figure 2.16 : MobileNetV2 architecture

The top layers of this loaded model are excluded. The input shape for this modified model is set at 224x224 pixels with 3 color channels (RGB). A custom regression head is added to this base model, which takes the output of the base model as its input and performs several operations. Firstly, it applies a global average pooling layer, which computes the average value of all the feature maps in the preceding layer. This step aids in reducing spatial dimensions while retaining critical features. Subsequently, three fully connected (Dense) layers are added, with 1024, 256, and 32 units, respectively, each employing the ReLU activation function. These layers contribute to feature extraction and transformation. Finally, there is an output layer with two units, responsible for predicting x and y coordinates. This model is constructed by specifying the inputs as the input layer of the base model and the outputs as the 'predictions' layer.

Furthermore, the code secures the layers of the MobileNetV2 model by setting their 'trainable' attribute to False. This means that during training, only the custom regression head's weights will be updated, while the pre-trained MobileNetV2 weights remain fixed. This approach is valuable when fine-tuning the model for a specific task while retaining the invaluable pre-trained knowledge in the base model. For a clearer representation of some of the model layers, refer to Figures 2.17, and 2.18.

```
Model: "model_6"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_7 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 Conv1 (Conv2D)                 (None, 112, 112, 32  864         ['input_7[0][0]']
                                )

 bn_Conv1 (BatchNormalization)  (None, 112, 112, 32  128         ['Conv1[0][0]']
                                )

 Conv1_relu (ReLU)              (None, 112, 112, 32  0           ['bn_Conv1[0][0]']
                                )

 expanded_conv_depthwise (Depth  (None, 112, 112, 32  288        ['Conv1_relu[0][0]']
 wiseConv2D)                    )

 expanded_conv_depthwise_BN (Ba  (None, 112, 112, 32  128        ['expanded_conv_depthwise[0][0]']
 tchNormalization)              )

 expanded_conv_depthwise_relu (  (None, 112, 112, 32  0          ['expanded_conv_depthwise_BN[0][0
 ReLU)                          )                                ]']

 expanded_conv_project (Conv2D)  (None, 112, 112, 16  512        ['expanded_conv_depthwise_relu[0]
                                )                                [0]']

 expanded_conv_project_BN (Batc  (None, 112, 112, 16  64         ['expanded_conv_project[0][0]']
 hNormalization)                )
```

Figure 2.17 : MobileNetV2 model layer part 1

```
block_15_depthwise_BN (BatchNo  (None, 7, 7, 960)   3840        ['block_15_depthwise[0][0]']
rmalization)

block_15_depthwise_relu (ReLU)  (None, 7, 7, 960)   0           ['block_15_depthwise_BN[0][0]']

block_15_project (Conv2D)       (None, 7, 7, 160)   153600      ['block_15_depthwise_relu[0][0]']

block_15_project_BN (BatchNorm  (None, 7, 7, 160)   640         ['block_15_project[0][0]']
alization)

block_15_add (Add)              (None, 7, 7, 160)   0           ['block_14_add[0][0]',
                                                                 'block_15_project_BN[0][0]']

block_16_expand (Conv2D)        (None, 7, 7, 960)   153600      ['block_15_add[0][0]']

block_16_expand_BN (BatchNorma  (None, 7, 7, 960)   3840        ['block_16_expand[0][0]']
lization)

block_16_expand_relu (ReLU)     (None, 7, 7, 960)   0           ['block_16_expand_BN[0][0]']

block_16_depthwise (DepthwiseC  (None, 7, 7, 960)   8640        ['block_16_expand_relu[0][0]']
onv2D)

block_16_depthwise_BN (BatchNo  (None, 7, 7, 960)   3840        ['block_16_depthwise[0][0]']
rmalization)

block_16_depthwise_relu (ReLU)  (None, 7, 7, 960)   0           ['block_16_depthwise_BN[0][0]']

block_16_project (Conv2D)       (None, 7, 7, 320)   307200      ['block_16_depthwise_relu[0][0]']

block_16_project_BN (BatchNorm  (None, 7, 7, 320)   1280        ['block_16_project[0][0]']
alization)

Conv_1 (Conv2D)                 (None, 7, 7, 1280)  409600      ['block_16_project_BN[0][0]']

Conv_1_bn (BatchNormalization)  (None, 7, 7, 1280)  5120        ['Conv_1[0][0]']

out_relu (ReLU)                 (None, 7, 7, 1280)  0           ['Conv_1_bn[0][0]']

global_average_pooling2d_6 (Gl  (None, 1280)        0           ['out_relu[0][0]']
obalAveragePooling2D)
```

Figure 2.18 : MobileNetV2 model layer part 2

2.4.4 Development of the classic computer vision algorithm

Even when switching from using ResNet50 to a lighter model called MobileNetV2, still faced the problem. The issue is that the Raspberry Pi's limited computing power makes it take a long time to obtain results. This makes it challenging to identify plant stems while TeaBot is in motion since this needs to be done in real-time. To address this issue, employed a computer vision approach. Developed solution simply illustrated in Figure 2.19.

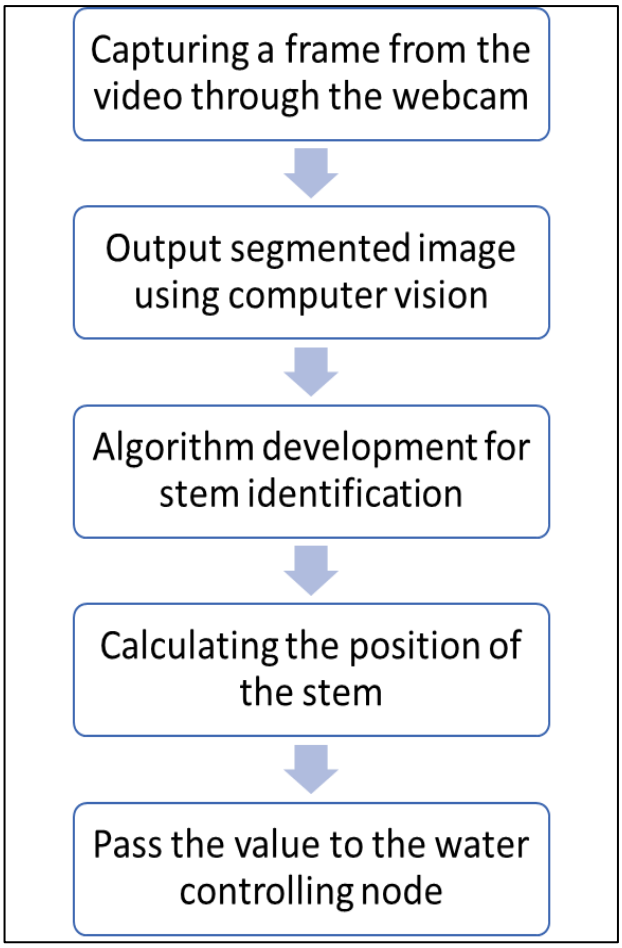


Figure 2.19: Developed solution

The computer vision algorithm used can be described as follows:

It is a Python script for a robot stem identification system. This script utilizes various libraries such as OpenCV, NumPy, and ROS for image processing and the control of a robotic system. Firstly, the script begins by importing necessary libraries and modules such as ROS, standard message types, geometry message types, SciPy signal processing functions, JSON handling, OpenCV for image processing, and other utility libraries like time, NumPy, and OS. Then, the function is defined, which is likely the main entry point for the robot's stem identification functionality. This function initializes various global variables. Afterward, the script sets up a video capture object to access the video feed from a camera and sets the frame dimensions and saturation properties. These settings are often used to configure camera parameters. More variables are defined to specify parameters for image processing and segmentation. These parameters control the size and segmentation of the captured image.

A continuous loop (`while True`) is used to capture frames from the camera. Inside the loop, the script reads a frame from the camera using a method and resizes it to a specified width and height using OpenCV's resize function. The frame is also split into its color channels, which are blue, green, and red. Histogram equalization is applied to each color channel independently to enhance the image's quality and improve contrast. This is done by splitting the image into its color channels, equalizing each channel, and then merging them back together.

Then, the script converts the BGR image to an HSV color space, defined as Hue, Saturation, Value. HSV is often used in computer vision to work with color information. A range of colors is defined using several variables in the HSV color space, creating a binary mask where white represents the target color within the specified range. This mask is generated using the inRange function. This is illustrate in Figure 2.20.

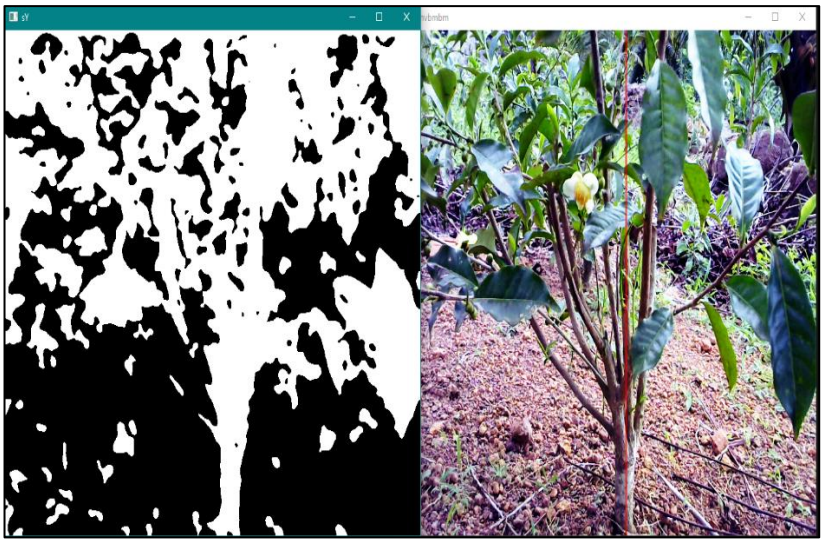


Figure 2.20 : Developed solution output

Then, an undisclosed mask is applied to the input frame. Following this, a copy of the original frame is created and stored in a variable. The image is converted to grayscale using a function, simplifying it to a single channel, facilitating specific types of image analysis. Next, the equalizeHist function is utilized to perform histogram equalization, enhancing the image's contrast for improved object distinction under varying lighting conditions. To reduce noise in the image, a 3x3 median blur filter is applied. Subsequently, Sobel filters are used to calculate the gradient in both the x and y directions, aiding in edge and contour detection. The image is then converted to an absolute value representation, typically to ensure positive pixel values. It is inverted, a useful step in certain image processing tasks. A binary threshold is applied with a value of 120, setting pixels below this threshold to 0 (black) and those equal to or above the threshold to 255 (white).

Morphological operations are employed on the image through erosion and dilation, further processing and cleaning the binary image by eliminating small artifacts and enhancing regions of interest. A Gaussian blur with a 25x25 kernel is then applied to the image, smoothing it and reducing noise. Another binary threshold, with a value of 100, is applied to further simplify the image and prepare it for connected components processing. The subsequent step involves connected components processing, which labels connected regions in the binary image, calculates statistics for each component, and identifies their centroids. Area statistics for each connected component are

extracted from the code, followed by the creation of a new binary image that retains only the connected components with an area greater than or equal to 100.

After the original image and initializing various variables for data storage and processing. It also sets up counters and lists for further analysis. Then, processes the image by iterating over its columns. It collects data for specific column segments and counts the white pixels in each column. This data is stored in arrays for further analysis, such as finding peaks in the pixel counts of columns.

The implemented code identifies peaks in the pixel count data, using a threshold calculated based on the average value of the peaks. This helps identify columns with notable pixel counts. It also sorts the data based on pixel counts in ascending order to better understand which columns have the lowest to highest pixel counts. Columns with counts above the average value are isolated for further analysis. Then updates its data to only contain the columns with significant pixel counts. The sorting is done to prepare the data for further processing. Following the sorting, an empty list is created to hold interim results. This list will store data as it's being processed in the following steps. Firstly, it goes through the sorted data. It checks the difference between two values and, if the difference meets a certain condition, it updates one of the values. Otherwise, it appends the current result to the interim list and resets it for further calculations.

After processing the data, the implemented code handles specific cases. It adds the last interim result to the list if it exists and meets certain conditions. Additionally, if there's relevant data and it meets certain criteria, the data is further sorted, and a particular value is calculated and used for graphical operations. It manages the transparency of images. It blends two images together with a specific level of transparency to create a final image. Then checks for specific conditions related to velocity. If these conditions are met, it updates parameters for a robotic system. Images are displayed using a library's function. Checks monitors keyboard input and checks for specific conditions. If these conditions are met, it stops the robot's movement and exits the processing loop.

Finally, controlling a robot's navigation and stem identification tasks using the ROS framework. It relies on global variables for maintaining the robot's state and

communicates with other components through ROS messages for coordinated control and operation.

2.4.5 Transfer coordinates to liquid fertilizer controller

The computer vision algorithm is designed to accurately identify the end of the tea stem by analyzing various features of the image. Once the precise coordinates of the stem end are determined, they are provided to the robot controller for further processing. The robot controller uses this values to adjust the position of the water spraying nozzles to ensure that they are aligned with the identified coordinates. The accuracy of this process is crucial in order to ensure the liquid fertilization to the tea plants. Any deviation from the identified coordinates can result in an incorrect process of the fertilization, which can adversely affect the growth and yield of the tea plants.

To ensure the accuracy of this task, proper communication is necessary between the computer vision based model and the robot controller. This involves ensuring that the correct data is transmitted to the liquid fertilizer controller, and that any errors or discrepancies are promptly identified and addressed. By maintaining clear and effective communication, the image processing model and the liquid fertilizer controller can work together seamlessly to achieve the desired outcome of accurate and precise liquid fertilization of the tea plants. The communication process illustrated in the Figure 2.21.

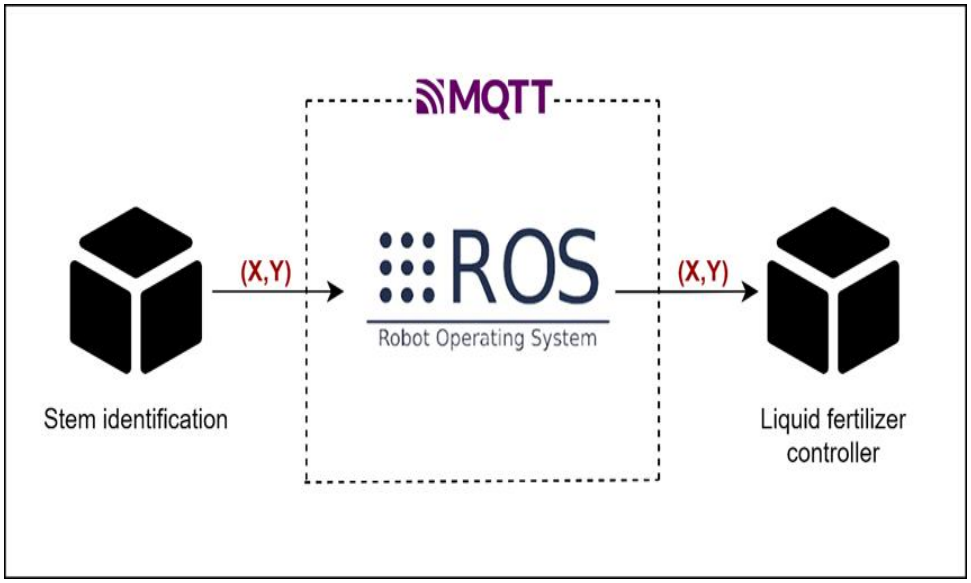


Figure 2.21 : Communication process

2.5 Commercialization Aspect of the Product

TeaBot, a groundbreaking agricultural innovation, offers a game-changing solution to the challenges of watering and fertilizing in the tea cultivation industry. With its automation, navigation, and deep learning-based stem identification capabilities, TeaBot streamlines and optimizes large-scale tea estate management. This robotic marvel not only addresses the pressing issue of labor shortages but also ensures precision in stem recognition and liquid fertilization. By embracing TeaBot's path navigation algorithm optimization, tea estate owners can revolutionize their operations, ultimately reclaiming Ceylon Tea's global prominence. This innovative product holds the potential to transform the tea industry, enhancing sustainability and yield optimization while reducing the reliance on manual labor.

Target Audience:

- Large-Scale Tea Estate Owners:
    - TeaBot is primarily aimed at owners and managers of large tea estates and plantations who face challenges related to labor shortages and optimizing yields.

- Agricultural Equipment Suppliers:
    - Companies that supply agricultural equipment can be potential partners in offering TeaBot as part of their product lineup.

- Government Agricultural Agencies:
    - Government bodies involved in promoting sustainable agriculture and modern farming techniques can also be part of the target audience as they might support the adoption of such innovative solutions.

Product Pricing:

The pricing strategy should be primarily value-based. This means the pricing should reflect the value that TeaBot brings to its customers in terms of labor savings, precision, and increased yields. The Pro version should be priced at a premium, justifying the added features and support it offers, while the Standard version should be positioned as a more budget-friendly option. The tiered pricing can be seen in Table 2.1.

Table 2.1 : Pricing plan

Pro Version	Standard Version
The Pro version of TeaBot is command a higher price, considering it includes advanced deep learning algorithms, comprehensive training, and premium technical support. The price for this package could range from a few $500 to $750, depending on the scale and needs of the tea estate.	The Standard version, which includes the essential TeaBot hardware and basic deep learning capabilities, should be competitively priced. It can target a broader market segment, with pricing ranging from a $200 to $350.

Bundled Pricing:

Consider offering bundled pricing options that include TeaBot along with maintenance and software updates. This can simplify the purchasing decision for customers and ensure they have ongoing support and access to the latest features.

Volume Discounts:

Implement volume-based pricing discounts for customers looking to deploy multiple TeaBots. This can incentivize large tea estates or cooperatives to invest in multiple units, driving economies of scale.

Upgrades and Add-Ons:

Provide options for customers to purchase add-on features or upgrade from the Standard version to the Pro version at any time. This allows customers to start with a more affordable option and scale up as their needs grow.

Competitive Analysis:

Continuously monitor the pricing of competing products in the market. Ensure that TeaBot's pricing remains competitive while still reflecting its value proposition.

Cost Structure Analysis:

Regularly review the cost structure of manufacturing, distribution, and support. This will help in maintaining healthy profit margins while remaining competitive in the market.

Effective pricing strategies will not only make TeaBot more attractive to potential customers but also ensure the product's sustainability and profitability in the market. Pricing should be regularly reviewed and adjusted as necessary to remain competitive and maximize the product's reach and impact.

2.6 Testing and Implementation

Multiple stages of testing conducted using various methodologies for the ResNet50 model and MobileNetV2 image processing models. Upon completion of dataset creation, it partitioned into two datasets in an 80:20 ratio for training and testing purposes. The model subsequently be implemented and trained using the training dataset.

1. Gather a set of representative test images:

Collect a set of test images that the model has not encountered during training.

2. Load the trained model:

Once the test data is created, load the trained ResNet50 and lightweight MobileNetV2 models using a deep learning framework such as TensorFlow or PyTorch.

3. Preprocess the test data (resize, normalize, format):

Both image processing models typically require preprocessing of the test data before feeding it into the model.

4. Run the test by passing images through the model:

After preprocessing the test data, run the test by passing the images through the ResNet50 model and obtain the model's predictions. Evaluate the following metrics: model accuracy, precision, recall, and F1 score. The same process is repeated for the MobileNetV2 model.

5. Analyze the results:

Visualize the model's predictions, calculate the performance metrics, and compare the results to the expected outcomes.

After thoroughly conducting testing for ResNet50 by inputting a range of different patterned growth images and images subjected to varying light conditions, the outcomes were meticulously documented. The accurate results are visually represented in Figure 2.22, 2.23, 2.24 and 2.25.

Figure 2.22 : Accurate result of the ResNet50 (1)

Figure 2.23 : Accurate result of the ResNet50 (2)

Figure 2.22 provides an illustration of the stem structure of a medium-sized tea crop, while Figure 2.23 depicts the stem structure of a smaller tea crop. Further visual representations and details can be found in the subsequent section below.

Figure 2.24 : Accurate result of the ResNet50 (3)

Figure 2.25 : Accurate result of the ResNet50 (4)

The aforementioned results accurately represent the outcomes obtained from the ResNet50 model. However, it should be noted that the results presented in Figure 2.26

and 2.27, as discussed in the following section, are found to be inconsistent with the expected performance of the ResNet50 model.

Figure 2.26 : Inaccurate result of the ResNet50 model (1)

Figure 2.27 : Inaccurate result of the ResNet50 model (2)

Due to the extensive output generated by the ResNet50 model, a transition was made to the more resource efficient MobileNetV2, known for its lightweight architecture. In terms of accuracy, the results obtained using MobileNetV2 closely resemble those presented in Figure 2.22, 2.23, 2.24, and 2.25. These findings have been thoroughly analyzed and are discussed in the subsequent sections. This summary highlights that in direct comparison to the ResNet50 model, MobileNetV2 exhibited superior accuracy in image processing. A visual representation of this comparison can be observed in Figure 2.28.

Figure 2.28 : ResNet50 vs. MobileNetV2 comparison

In Figure 2.28, the left side displays the output of the ResNet50 model, while on the right-hand side, the output of the MobileNetV2 model. Both outputs correspond to the same image. However, it is noteworthy that when utilizing MobileNetV2, it provides more accurate coordinates for the stem detection.

To address the limitations of both models in image processing, an alternative solution was developed using computer vision. The primary drawbacks encountered included a lengthy output time of approximately 4-5 seconds and the challenge of training from scratch on custom datasets. During the testing phase, numerous discussions and decisions took place.

Ultimately, after careful consideration, a computer vision algorithm was chosen as the preferred approach. The accurate results of the computer vision algorithm-based approach is defined in Figure 2.29 and 2.30.

Figure 2.29 : Accurate result of the computer vision-based approach (1)

Figure 2.30: Accurate result of the computer vision-based approach (2)

Even though the computer vision-based approach was chosen, it occasionally yielded flawed results, infrequently. These issues can be mitigated through code optimization. However, in comparison to image processing models, the computer vision-based approach holds substantial value due to its primary advantage of delivering results within 1 second. For reference, an example of flawed stem detection using the computer vision-based algorithm is presented in Figure 2.31.

Figure 2.31 : Inaccurate result of the computer vision-based approach

3 RESULTS & DISCUSSION

3.1 Results

According to the test results mentioned earlier, this section provides a detailed overview of the statistical data pertaining to the implemented solution. Also, serves to present the proper output and analysis derived from the testing process, offering a comprehensive understanding of the solution's performance and efficacy.

3.1.1 ResNet50 model vs MobileNetV2 model

The detection of the end of the tea stem optimizes the precise liquid fertilization process. A thousand frames were created using videos captured through TeaBot, and all frames were meticulously labeled. The end area of the tea stem was denoted by a single dot. The labeled images yielded an Excel file containing the corresponding x and y coordinates. These labeled frames and coordinates were then used to train the image processing model, leveraging the ResNet50 architecture. The model underwent 30 epochs of training with a learning rate set at 0.001, achieving an accuracy of 88%. A lightweight version of the model was developed using the MobileNetV2 architecture, which underwent training under the same constraints. Both models were subsequently deployed and tested within the tea plantation field, demonstrating commendable accuracy. The performance metrics for the stem detection algorithm are provided in Table I.

Table 3.1 : Stem identification model summary

Model	Accuracy	Precision	Recall	F1
ResNet50	0.88	0.90	0.89	0.91
MobileNetV2	0.90	0.91	0.89	0.92

The validation dataset demonstrated an accuracy rate of approximately 90% after undergoing multiple iterations of model training. Throughout the training phase, the loss consistently remained at 0.0125, with the validation loss converging to 0.25. The empirical findings of the study revealed a moderate level of performance when compared to previous works, as accuracy fluctuated between 60% and 90% across iterations in both models. Throughout the testing phase, it became evident that the stem detection algorithm displayed inaccuracies in specific scenarios, as illustrated in Figure 2.26 and 2.27. In particular, the algorithm's proficiency in identifying stems within smaller tea plantations was compromised. The algorithm's effectiveness exhibited fluctuations across a range of environmental conditions, encompassing situations with intense sunlight and shadow effects. To improve predictive precision, it was clear that the model's dataset should encompass a comprehensive representation of these diverse scenarios.

3.1.2 ML vs Classic computer vision algorithm

Both models' inference process demanded an average of 4 to 5 seconds for completion, and a notable observation was made regarding the models' relatively high resource consumption. The outcomes of the testing phase have been graphically presented in Figure 2.29 and 2.30 in the above section. The evaluation of performance primarily focused on a classic computer vision-based algorithm. Impressively, this algorithm was able to successfully detect tea stems in 82 out of the 100 test images conducted. A detailed account of this evaluation can be found in both Figure 2.29 and 2.30, while Figure 2.31 highlights instances of flawed tea stem detection.

In some specific cases, it became evident that the achieved results were not entirely accurate, primarily due to the inadvertent inclusion of other stems or branches in the captured images. This, in turn, resulted in a decrease in the algorithm's predictive accuracy, and it was observed that code optimization could potentially enhance this aspect. These less favorable outcome also vividly illustrated in Figure 2.31.

To provide a more comprehensive understanding of the results, graphs were generated for each tested image. The x-axis represented the column index of the image, while the y-axis depicted the pixel value summation for each vertical column. Accurately generated outputs displayed a distinct peak, as exemplified in Figure 3.1.

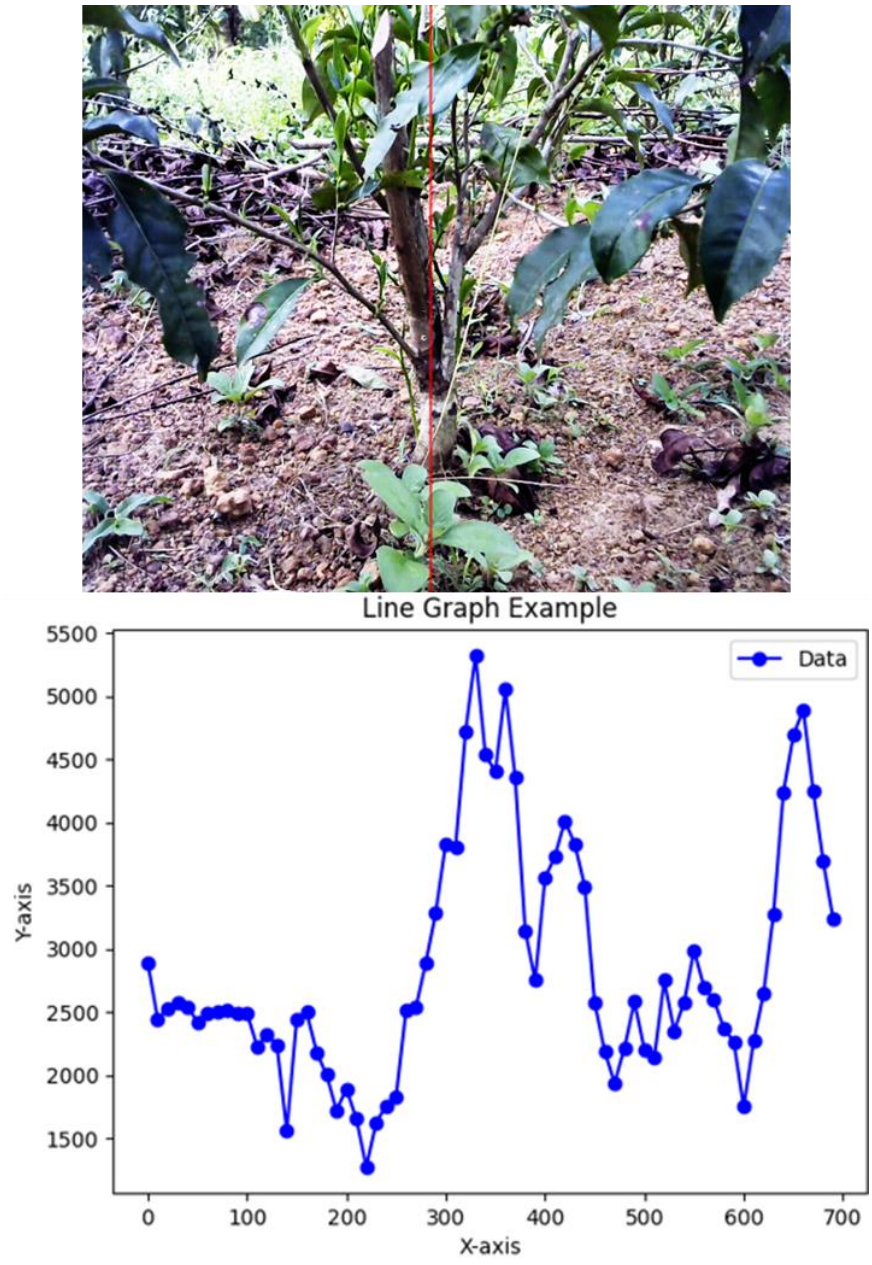


Figure 3.1 : Generated graph for accurate

However, in the case of poorly generated images, the highest value column was not clearly defined, and it often displayed two or even three places with the highest summation values, as evident in Figure 3.2. A comparative analysis of the generated graphs can be found in Figure 3.2, wherein the peaks in the graph signify the presence of a tea stem with a higher summation value. However, these characteristics were not as visible due to the scenarios, resulting in the generation of imprecise outcomes.

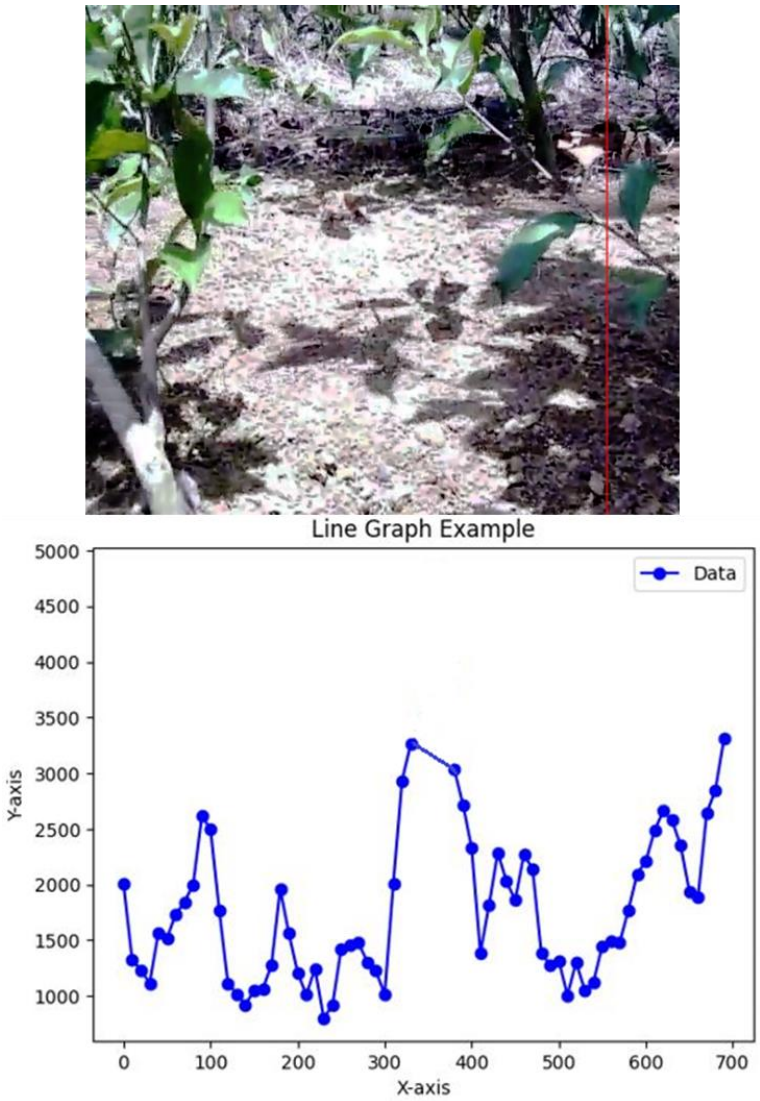


Figure 3.2 : Generated graph for inaccurate

3.2 Research Findings

This section provides an overview of the research findings from the TeaBot project as a whole and highlights specific research findings from the component of Communications. These findings were drawn from a combination of literature review and empirical test results detailed in previous sections of this document.

1. Incompatibility with Raspberry Pi:

- The ResNet50 and MobileNetV2 image processing models were found to be unsuitable for deployment on Raspberry Pi, as they couldn't provide real-time output due to their computational demands.

2. Superiority of Computer Vision-Based Approach:

- The computer vision-based approach outperformed image processing models in terms of accuracy and speed. It proved to be more efficient for tea stem identification during real-time navigation by a robot.

3. Real-Time Performance:

- The computer vision-based algorithm was capable of providing real-time output, ensuring that the robot could make instantaneous decisions during navigation.

4. Resource Constraint:

- Attempting to run image processing models simultaneously on Raspberry Pi led to significant delays due to resource limitations, which made them impractical for real-time applications.

5. Effective Data Exchange:

- The communication framework in TeaBot facilitated the use of ROS data nodes between the robot's components, ensuring coordinated operation during navigation and the liquid fertilization process.

6. Low Latency:

- The ROS nodes implemented in TeaBot were designed to minimize latency, ensuring rapid data transfer and decision-making, which is essential for real-time autonomous navigation and the liquid fertilization process.

7. Reliability:

- The communication component demonstrated a high degree of reliability in maintaining data flow and synchronization between the various subsystems of the robot.

3.3 Discussion

The focus of this study has been centered on addressing the challenges associated with models for stem identification based on image processing, which have often yielded suboptimal results. This issue has been recognized as a significant influencer leading to less than satisfactory outcomes. To overcome this challenge, developed a computer vision algorithm that significantly improved accuracy and reduced output time. This approach effectively mitigated the need for extensive exploration of larger datasets for pre-training the model, saving valuable time and resources.

The task of stem identification may involve using various parts of tea plants that grow in diverse environmental conditions and may also necessitate the utilization of stems from tea trees of different ages. This diversity in data sources is crucial for developing a robust and adaptable model that can handle the variability found in real-world scenarios.

Finally, to enhance the liquid fertilizer spraying process, it is imperative to establish stability in the spraying arm concerning the robot's relative motion. Achieving a synchronized and precise motion of the spraying arm is crucial for consistent and accurate fertilizer application. Additionally, the cessation of the motor's operation must be well-coordinated to avoid over-spraying or under-spraying.

4 CONCLUSION & RECOMMENDATIONS

The TeaBot, a groundbreaking innovation, has emerged as a revolutionary solution in the tea industry. This autonomous robotic system is designed to automate the vital tasks of watering and fertilization in extensive tea estates. Its advanced design, specifically tailored for off-road conditions, makes it a formidable solution to address the critical needs of the tea cultivation sector. In this exploration, will delve into the TeaBot's capabilities and the insights gained from evaluating its navigation methods, particularly in terms of image processing and resource consumption. Additionally it highlights the precision of the TeaBot's robotic arm and its potential to transform the tea industry by offering a cost-effective solution while reducing manual labor and preserving the integrity of tea estates.

The TeaBot's role in the management of extensive tea estates is pivotal. By automating the labor-intensive tasks of watering and fertilization, it not only boosts efficiency but also trims down operational costs while ensuring consistent and precise care for tea plants. What sets the TeaBot apart is its specialized design, crafted to thrive in off-road conditions, a common feature of tea plantations. This design is coupled with autonomous navigation capabilities, freeing it from constant human intervention.

While exploring the TeaBot's stem identification approaches, some fascinating insights came to light. The use of image processing exhibited an exceptional accuracy rate, ranging from 88% to 90%. However, this method was found to be resource-intensive, making it unsuitable for deployment on platforms with limited resources, such as the Raspberry Pi. In response to this limitation, the TeaBot adopted the classical computer vision method, ensuring reliability and efficiency in tea stem identification.

Beyond stem idetification, the TeaBot's precision is further demonstrated by its robotic arm, capable of accurately spraying water over significant distances when properly aligned. This precision guarantees that tea plants receive the necessary care without waste, promoting healthier and more sustainable tea estates. The economic advantages of the TeaBot are notable, as it offers a cost-effective solution by automating essential tasks and reducing the reliance on manual labor.

Furthermore, the TeaBot's introduction to tea estate management contributes to the preservation of the estates' integrity. Its precision, autonomy, and off-road capabilities ensure that the tea plants receive optimal care, leading to healthier crops and the maintenance of the estates' aesthetic appeal. In essence, the TeaBot is not only a cost-saving technology but also a guardian of the tea estate tradition and beauty.

By significantly reducing the need for manual labor, the TeaBot marks a shift in the industry. It eliminates the arduous and repetitive tasks, offering not only cost savings but also improving the well-being of workers. In conclusion, the transformative impact of the TeaBot on the tea industry is undeniable. It addresses vital needs for efficient estate management, cost reduction, and labor elimination. Its adaptability to various platforms and resource-efficient approach makes it accessible to a broader range of tea estate owners, promising a more sustainable and profitable future for the industry. As this innovative technology continues to evolve, its influence on the tea cultivation sector is likely to grow, offering a more efficient and sustainable future.

REFERENCE LIST

[1] H. C. J. Godfray *et al.*, “Food Security: The Challenge of Feeding 9 Billion People,” *Science (1979)*, vol. 327, no. 5967, pp. 812–818, Feb. 2010, doi: 10.1126/SCIENCE.1185383.

[2] “Sri Lanka - Agricultural Sector.” Accessed: Mar. 19, 2023. [Online]. Available: https://www.trade.gov/country-commercial-guides/sri-lanka-agricultural-sector

[3] D. Tilman, K. G. Cassman, P. A. Matson, R. Naylor, and S. Polasky, “Agricultural sustainability and intensive production practices,” *Nature 2002 418:6898*, vol. 418, no. 6898, pp. 671–677, Aug. 2002, doi: 10.1038/nature01014.

[4] *Future of work for Tea Smallholders in Sri lanka*. International Labour Organization, 2018. Accessed: Mar. 19, 2023. [Online]. Available: https://www.ilo.org/wcmsp5/groups/public/---asia/---ro-bangkok/---ilo-colombo/documents/publication/wcms_654641.pdf

[5] “Ceylon Tea Export Growth - EDB Sri Lanka.” Accessed: Mar. 19, 2023. [Online]. Available: https://www.srilankabusiness.com/tea/tea-export-growth.html

[6] C. Rathnayake, B. Malcolm, G. Griffith, B. Farquharson, and A. Sinnett, “Current Issues in the Farm Sector of the Sri Lankan Tea Industry Current Issues in the Farm Sector of the Tea Industry in Sri Lanka,” *Australasian Agribusiness Perspectives*, vol. 24, 2021.

[7] “EFFECTS OF ORGANIC AND INORGANIC FERTILIZER APPLICATION FOR TEA CULTIVATION IN SRI LANKA”, Accessed: Mar. 19, 2023. [Online]. Available: https://www.researchgate.net/publication/351943232_EFFECTS_OF_ORGANIC_AND_INORGANIC_FERTILIZER_APPLICATION_FOR_TEA_CULTIVATION_IN_SRI_LANKA

[8] “Balanced plant nutrition critical for sustainable tea cultivation | History of Ceylon Tea.” Accessed: Mar. 19, 2023. [Online]. Available: https://www.historyofceylontea.com/ceylon-publications/ceylon-tea-articles/balanced-plant-nutrition-critical-for-sustainable-tea-cultivation.html

[9] “Sri Lanka’s organic farming disaster, explained - Vox.” Accessed: Mar. 19, 2023. [Online]. Available: https://www.vox.com/future-perfect/2022/7/15/23218969/sri-lanka-organic-fertilizer-pesticide-agriculture-farming

[10] “How Organic Farming Worsened Sri Lanka’s Economic and Political Crisis | Time.” Accessed: Mar. 19, 2023. [Online]. Available: https://time.com/6196570/sri-lanka-crisis-organic-farming/

[11] S. N. Bandara, “Agronomy of Irrigated Tea in Low Elevation Growing Areas of Sri Lanka,” 1968.

[12] “IoT Based Drip Irrigation Method for Tea Plantation”, Accessed: Mar. 19, 2023. [Online]. Available: https://www.researchgate.net/publication/354077145_IoT_Based_Drip_Irrigation_Method_for_Tea_Plantation

[13] B. Jayant, K. Dahiya, A. Rukhiyar, R. Raj, and R. K. Meena, “A REVIEW OF THE DRIP IRRIGATION SYSTEM,” *Journal of Engineering Research and Application*, vol. 01, no. 01, 2022, doi: 10.55953/JERA.1103.

[14] D. D. Bochtis, C. G. C. Sørensen, and P. Busato, “Advances in agricultural machinery management: A review,” *Biosyst Eng*, vol. 126, pp. 69–81, Oct. 2014, doi: 10.1016/J.BIOSYSTEMSENG.2014.07.012.

[15] H. Tian, T. Wang, Y. Liu, X. Qiao, and Y. Li, “Computer vision technology in agricultural automation —A review,” *Information Processing in Agriculture*, vol. 7, no. 1, pp. 1–19, Mar. 2020, doi: 10.1016/J.INPA.2019.09.006.

[16] S. A. S, S. T. D, and M. Z. KURIAN, “MULTIPURPOSE AGRICULTURE ROBOT USING LORA,” *International Journal of Advanced Scientific Innovation*, vol. 2, no. 1, Jul. 2021, doi: 10.5281/ZENODO.5121003.

[17] J. M. Marais, R. Malekian, and A. M. Abu-Mahfouz, “LoRa and LoRaWAN testbeds: A review,” *2017 IEEE AFRICON: Science, Technology and Innovation for Africa, AFRICON 2017*, pp. 1496–1501, Nov. 2017, doi: 10.1109/AFRCON.2017.8095703.

[18] R. Rahmadian and M. Widyartono, “Autonomous Robotic in Agriculture: A Review,” *Proceeding - 2020 3rd International Conference on Vocational Education and Electrical Engineering: Strengthening the framework of Society 5.0 through Innovations in Education, Electrical, Engineering and Informatics Engineering, ICVEE 2020*, Oct. 2020, doi: 10.1109/ICVEE50212.2020.9243253.

[19] A. Botta, P. Cavallone, L. Baglieri, G. Colucci, L. Tagliavini, and G. Quaglia, “A Review of Robots, Perception, and Tasks in Precision Agriculture,” *Applied Mechanics 2022, Vol. 3, Pages 830-854*, vol. 3, no. 3, pp. 830–854, Jul. 2022, doi: 10.3390/APPLMECH3030049.

[20] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, “Machine Learning in Agriculture: A Review,” *Sensors 2018, Vol. 18, Page 2674*, vol. 18, no. 8, p. 2674, Aug. 2018, doi: 10.3390/S18082674.

[21] A. Sharma, A. Jain, P. Gupta, and V. Chowdary, “Machine Learning Applications for Precision Agriculture: A Comprehensive Review,” *IEEE Access*, vol. 9, pp. 4843–4873, 2021, doi: 10.1109/ACCESS.2020.3048415.

[22] L. A. Wells and W. Chung, “Real-Time Computer Vision for Tree Stem Detection and Tracking,” *Forests 2023, Vol. 14, Page 267*, vol. 14, no. 2, p. 267, Jan. 2023, doi: 10.3390/F14020267.

[23] R. P. Haff, D. C. Slaughter, and E. S. Jackson, “X-Ray Based Stem Detection in an Automatic Tomato Weeding System,” *Appl Eng Agric*, vol. 27, no. 5, pp. 803–810, 2011, doi: 10.13031/2013.39559.

[24] T. Kalampokas, E. Vrochidou, G. A. Papakostas, T. Pachidis, and V. G. Kaburlasos, “Grape stem detection using regression convolutional neural

networks," *Comput Electron Agric*, vol. 186, p. 106220, Jul. 2021, doi: 10.1016/J.COMPAG.2021.106220.

[25] C. W. Bac, J. Hemming, and E. J. Van Henten, "Stem localization of sweet-pepper plants using the support wire as a visual cue," *Comput Electron Agric*, vol. 105, pp. 111–120, Jul. 2014, doi: 10.1016/J.COMPAG.2014.04.011.

[26] W. Wang, C. Li, K. Wang, L. Tang, P. F. Ndiluau, and Y. Cao, "Sugarcane stem node detection and localization for cutting using deep learning," *Front Plant Sci*, vol. 13, p. 5181, Dec. 2022, doi: 10.3389/FPLS.2022.1089961/BIBTEX.

[27] B. Jayant, K. Dahiya, A. Rukhiyar, R. Raj, and R. K. Meena, "A REVIEW OF THE DRIP IRRIGATION SYSTEM," *Journal of Engineering Research and Application*, vol. 01, no. 01, 2022, doi: 10.55953/JERA.2022.1103.

[28] A. Shilpa, V. Muneeswaran, and D. Devi Kala Rathinam, "A Precise and Autonomous Irrigation System for Agriculture: IoT Based Self Propelled Center Pivot Irrigation System," *2019 5th International Conference on Advanced Computing and Communication Systems, ICACCS 2019*, pp. 533–538, Mar. 2019, doi: 10.1109/ICACCS.2019.8728550.

[29] "How Much Data Is Required for Machine Learning? – PostIndustria." Accessed: Apr. 16, 2023. [Online]. Available: https://postindustria.com/how-much-data-is-required-for-machine-learning/

[30] "The Startup Magazine 4 Main Benefits That Data Labeling Can Bring to Your Company | The Startup Magazine." Accessed: Apr. 06, 2023. [Online]. Available: https://thestartupmag.com/4-main-benefits-data-labeling-can-bring-company/

[31] "Object detection: Bounding box regression with Keras, TensorFlow, and Deep Learning - PyImageSearch." Accessed: May 10, 2023. [Online]. Available: https://pyimagesearch.com/2020/10/05/object-detection-bounding-box-regression-with-keras-tensorflow-and-deep-learning/

[32] “Localised feature comparison between VGG-16 and ResNet-50. Although... | Download Scientific Diagram.” Accessed: May 11, 2023. [Online]. Available: https://www.researchgate.net/figure/Localised-feature-comparison-between-VGG-16-and-ResNet-50-Although-both-models-extracted_fig6_328440990

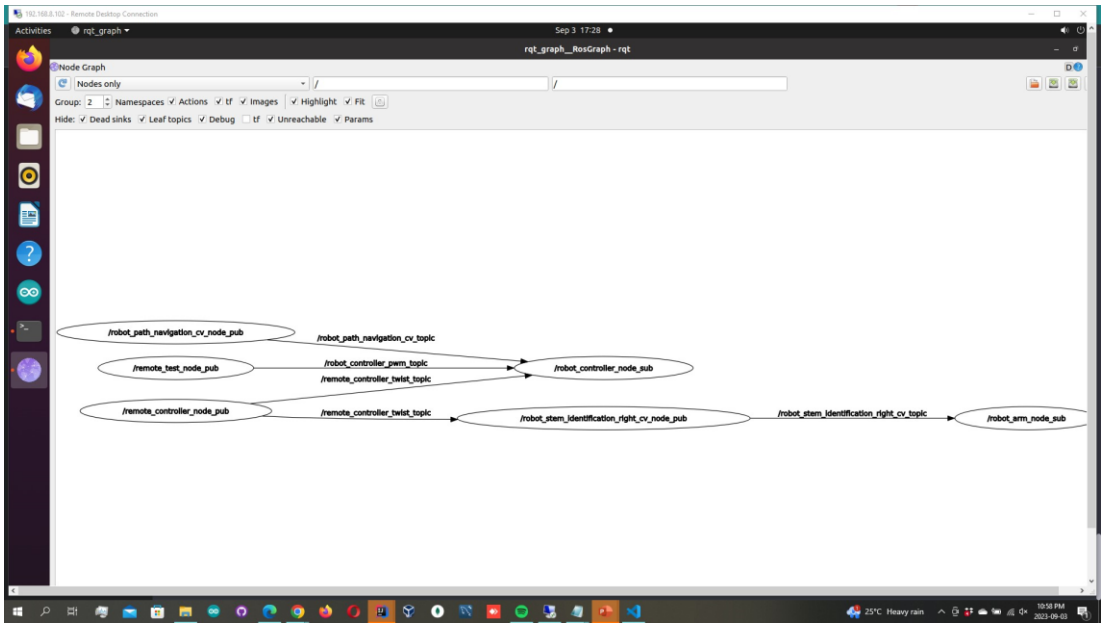
[33] “Accuracy and loss values for the VGG16 and ResNet50 models on the... | Download Scientific Diagram.” Accessed: May 11, 2023. [Online]. Available: https://www.researchgate.net/figure/Accuracy-and-loss-values-for-the-VGG16-and-ResNet50-models-on-the-GAN-augmented-dataset_fig1_352142241

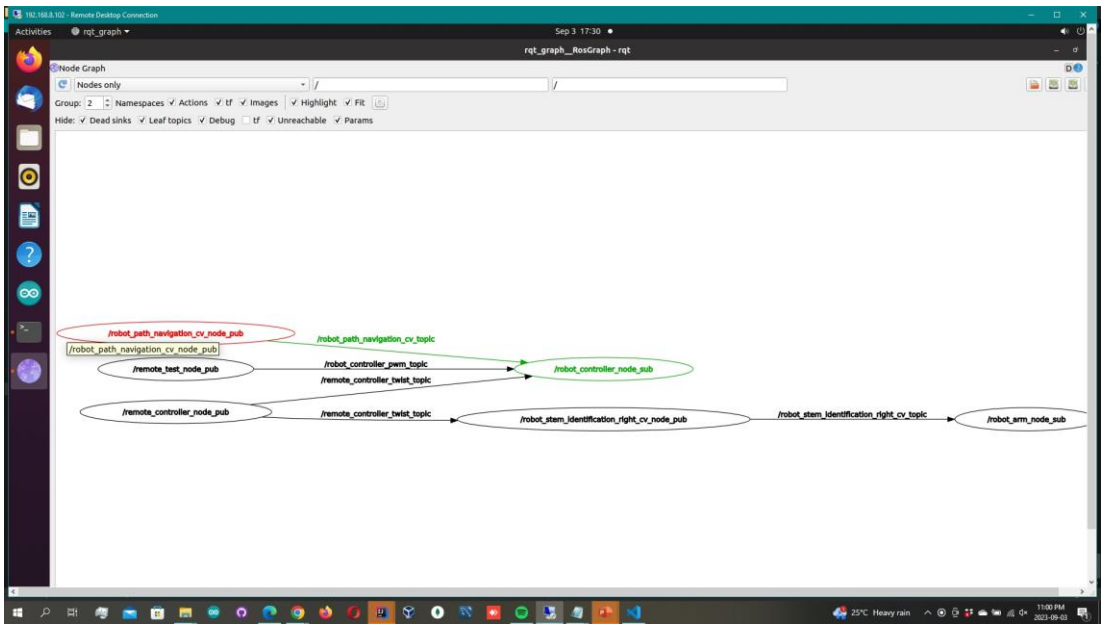
[34] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, Jan. 2018, doi: 10.1109/CVPR.2018.00474.

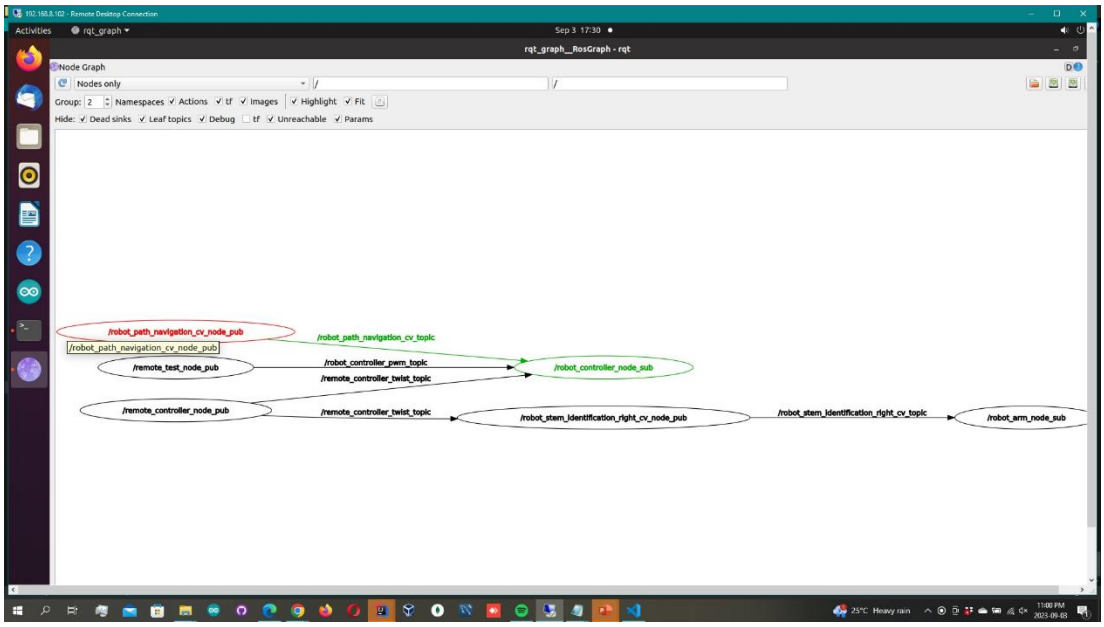
[35] “MobileNet vs ResNet50 - Two CNN Transfer Learning Light Frameworks.” Accessed: Aug. 08, 2023. [Online]. Available: https://analyticsindiamag.com/mobilenet-vs-resnet50-two-cnn-transfer-learning-light-frameworks/

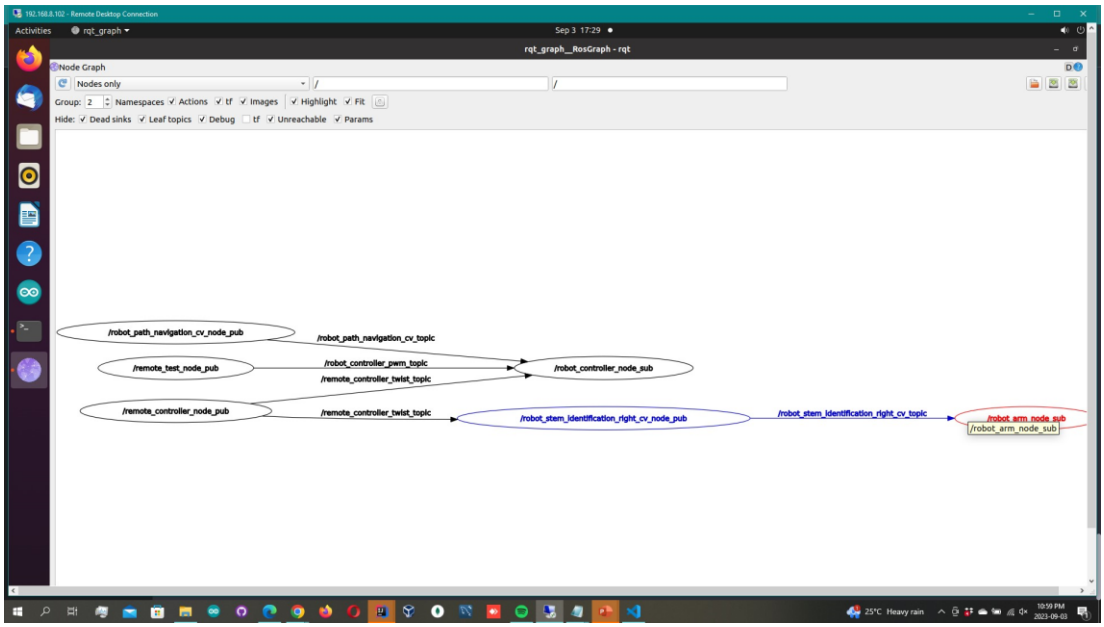
[36] “Why your MobileNetv2 model performs better than Resnet50 | by Imtiaz Ahmed | Jul, 2023 | Medium.” Accessed: Aug. 08, 2023. [Online]. Available: https://medium.com/@imtiaz.ahmed2206/why-your-mobilenetv2-model-performs-better-than-resnet50-2a9998fda4c7

Appendix A : ROS nodes graphs






Activities
rqt_graph
Sep 3 17:30
rqt_graph__RosGraph - rqt
Node Graph
Nodes only
Group: 2 Namespaces Actions tf Images Highlight Fit
Hide: Dead sinks Leaf topics Debug tf Unreachable Params
/robot_path_navigation_cv_node_pub
/robot_path_navigation_cv_topic
/remote_test_node_pub
/robot_controller_pwm_topic
/remote_controller_twist_topic
/robot_controller_node_sub
/remote_controller_node_pub
/remote_controller_twist_topic
/robot_stem_identification_right_cv_node_pub
/robot_stem_identification_right_cv_topic
/robot_arm_node_sub


Activities
rqt_graph
Sep 3 17:29
rqt_graph__RosGraph - rqt
Node Graph
Nodes only
Group: 2 Namespaces Actions tf Images Highlight Fit
Hide: Dead sinks Leaf topics Debug tf Unreachable Params
/robot_path_navigation_cv_node_pub
/robot_path_navigation_cv_topic
/remote_test_node_pub
/robot_controller_pwm_topic
/remote_controller_twist_topic
/robot_controller_node_sub
/remote_controller_node_pub
/remote_controller_twist_topic
/robot_stem_identification_right_cv_node_pub
/robot_stem_identification_right_cv_topic
/robot_arm_node_sub

Appendix B : Budget justification

Item	Quantity	Amount (LKR)
Rubber wheels	4	4,000
Sprocket, chain and wheel (gear system)	4	10,800
Iron frame	1	15,500
Motors	4	14,000
43A Motor drivers	4	5,400
Raspberry Pi	1	90,000
12V 65Ah battery	1	45,000
12V 8Ah battery	1	5,000
Camera	3	15,000
Liquid nozzles	2	3,000
Servo motors	8	9,600
Total		**220,000**