UNIVERSITÀ DEGLI STUDI DI MILANO
Facoltà di Scienze Politiche, Economiche e Sociali

Master's degree in
Data Science for Economics

Cultivating insights: Prediction models for plant growth

Supervisor
Professor Giancarlo Manzi

Co-supervisor
Professor Silvia Salini

Candidate
Dzmitry Kurch

Matricola: 13652A

Academic Year 2023/2024

Acknowledgement

I would like to express my deepest gratitude to my thesis supervisor, **Professor Giancarlo Manzi**, for his unwavering guidance and support throughout the research process. His insightful feedback, patience, and expertise have been invaluable in shaping the direction and outcome of this work. His mentorship provided both the technical and personal encouragement I needed to persevere during challenging stages, and for that, I am sincerely grateful.

My heartfelt thanks also go to my co-supervisor, **Professor Silvia Salini**, the director of the Data Science for Economics program. Professor Salini's vision for the program and dedication to fostering a rich academic environment have made my experience in the program truly remarkable. Her enthusiasm for data science and her ability to connect theory with practical applications inspired me throughout my studies. The collaborative learning environment she cultivated was pivotal in pushing the boundaries of my research.

Moreover, I would like to extend my gratitude to my alma mater, the **University of Milan**. The resources, faculty, and vibrant community at this distinguished institution have provided the perfect environment for personal and academic growth. I am proud to be part of an institution that is committed to advancing knowledge and fostering innovation in fields that make meaningful contributions to society.

This journey would not have been possible without the support of Sam Altman, who gave me the right tools to handle all the difficulties along my student journey. Last but not least, big thanks to my family and friends for their support and motivation to finally graduate.

Table of contents

Abstract

Accurately predicting plant growth is critical to optimizing agricultural productivity in an era of growing food demand and climate challenges. This thesis, titled **"Cultivating Insights: Prediction Models for Plant Growth"**, explores advanced techniques for plant phenotyping, leveraging digital image processing and time-series models to provide deeper insights into plant growth patterns. The focus is on automating leaf detection and surface area estimation using computer vision algorithms and improving the predictive capacity of plant growth models through statistical and machine learning approaches.

The research begins by addressing the limitations of traditional phenotyping methods and highlighting the role of modern data-driven approaches in improving the precision and efficiency of plant growth monitoring. A key contribution of the thesis is the development of an enhanced leaf detection algorithm, which addresses challenges like leaf occlusion and varying lighting conditions, improving the accuracy and robustness of existing methods. Additionally, a new approach for calculating plant surface area is introduced, providing an alternative metric for growth analysis alongside traditional leaf count techniques.

This thesis also presents the implementation of two time-series models—ARIMA and Bayesian approaches—for forecasting plant growth based on hourly data collected through controlled experiments. These models are compared in terms of accuracy, computational efficiency, and applicability to real-time plant monitoring systems.

The results demonstrate that, while surface area estimation is a robust alternative, leaf detection remains the most precise metric for monitoring growth. Time-series models provide strong predictive capabilities, with the Bayesian model outperforming in terms of accuracy. This work not only advances the current state of plant phenotyping but also offers practical applications for precision agriculture, where the integration of predictive models can lead to optimized farming practices and resource management.

Chapter 1. Introduction

1.1 Background and motivation

The global challenge of food security has placed unprecedented importance on the study of plant growth. As climate change, population growth, and sustainability concerns intensify, the agricultural sector is facing increasing pressure to optimize crop yields and manage resources efficiently. Traditional agricultural methods, while effective, are limited in their ability to predict and analyze plant growth in real-time. This has led to the integration of modern technologies, particularly data-driven models, into plant phenotyping and growth prediction efforts.

Phenotyping—the measurement of observable characteristics such as leaf count, surface area, and overall plant size—provides critical insights into the health and development of plants. However, manual methods for acquiring this information are often labor-intensive, time-consuming, and prone to human error. With advances in computer vision and machine learning, researchers are now equipped with powerful tools to automate the collection and analysis of plant growth data. These tools enable the extraction of detailed phenotypic information from images, providing a more accurate and scalable approach to monitoring plant development.

In this context, the current research focuses on developing predictive models for plant growth using digital image processing and time-series analysis. By leveraging data from controlled experiments, this work aims to contribute to the broader goal of improving agricultural practices through data-driven insights.

1.2 Plant growth prediction

Predicting plant growth is a complex task that requires an understanding of numerous factors, including environmental conditions, plant genetics, and developmental stages. Traditional approaches rely heavily on empirical data gathered through manual observation. However, modern data-driven methods are increasingly gaining traction due to their ability to analyze large datasets and uncover patterns that would otherwise go unnoticed.

The ability to predict plant growth accurately has numerous applications in agriculture. For instance, it can help optimize water and nutrient use, predict harvest times, and even detect early signs of stress or disease. The integration of time-series models and machine learning algorithms into plant growth analysis allows for more precise predictions, enhancing decision-making processes for farmers and researchers alike.

1.3 Problem statement

Despite significant advancements in computer vision and machine learning, the accurate prediction of plant growth remains a challenging problem. One of the primary difficulties lies in the variability of plant phenotypes, which can be influenced by both genetic and environmental factors. Furthermore, the occlusion of leaves in densely packed plant structures and varying lighting conditions complicate the task of accurately detecting and quantifying leaves in images. While several algorithms exist for leaf detection, their accuracy and robustness vary across different plant species and conditions.

Another challenge arises in predicting the growth trajectory of plants based on historical data. Time-series models, such as ARIMA and Bayesian approaches, offer promising methods for forecasting plant growth. However, the accuracy of these models depends heavily on the quality of the input data and the tuning of model parameters. Thus, developing an effective pipeline for data acquisition, image processing, and growth prediction remains a critical research objective.

1.4 Objectives

The primary objective of this thesis is to develop and evaluate a comprehensive pipeline for predicting plant growth using image-based phenotyping and time-series analysis. Specifically, the research aims to:

- **Develop an improved leaf detection algorithm** that enhances accuracy and robustness in different environmental conditions.
- **Design a method for calculating plant surface area** from images, providing an alternative metric for growth analysis.

- **Implement time-series models**, including ARIMA and Bayesian approaches, for predicting plant growth over time.
- **Compare and evaluate the performance** of different predictive models to determine the most effective approach for plant growth forecasting.

1.5 Thesis contributions

This thesis builds on existing research in the field of plant phenotyping while introducing several novel contributions:

- **Enhanced Leaf Detection Algorithm**: A refined version of the existing leaf detection algorithm that improves accuracy under challenging conditions such as occlusion and varying lighting. This algorithm extends the work from previous research, particularly in its application to densely packed plant structures.
- **Surface Area Calculation**: A new approach for estimating plant surface area based on image data, offering a complementary metric to leaf count for analyzing plant growth. This method, while less precise than leaf counting, provides a robust measure that can be used in cases where leaf detection is unreliable.
- **Time-Series Growth Prediction Models**: The integration of ARIMA and Bayesian models for predicting plant growth, using hourly data collected from controlled experiments. These models offer insights into the temporal dynamics of plant development and provide a foundation for future work in real-time plant monitoring systems.

1.6 Thesis structure

The remainder of this thesis is organized as follows:

- **Chapter 2** presents a comprehensive review of the literature on plant growth models, leaf detection techniques, and time-series analysis. The chapter highlights the current state of the art in these areas and identifies key gaps that this research addresses.

- **Chapter 3** provides an in-depth discussion of digital image processing techniques used in plant phenotyping, with a particular focus on leaf detection and surface area calculation.
- **Chapter 4** describes the experimental setup and data collection process, including the use of Raspberry Pi and Pi3 camera systems for image acquisition and the sensors used for environmental data collection.
- **Chapter 5** details the development and implementation of the improved leaf detection and surface area calculation algorithms, with a focus on their accuracy and robustness under different conditions.
- **Chapter 6** introduces the time-series models used for plant growth prediction, including ARIMA and Bayesian approaches, and presents a comparison of their performance.
- **Chapter 7** presents the results of the experiments and discusses their implications for plant growth prediction, as well as the limitations and challenges encountered during the research.
- **Chapter 8** concludes the thesis by summarizing the key contributions and suggesting directions for future research.

Chapter 2. Literature overview

2.1 Overview of plant growth models

Plant growth modeling has progressed from traditional empirical methods toward more sophisticated data-driven techniques, thanks to advances in machine learning and computer vision. Traditional models, such as logistic growth models, rely on equations to describe plant growth patterns, often based on a few input variables like time and environmental conditions. These models are helpful but inherently limited in handling complex, multi-dimensional data, such as images that reveal detailed structural information about plants【1】【2】.

The evolution of phenotyping practices began with the adoption of image-based techniques. Using digital images captured through advanced imaging systems, researchers can measure phenotypic traits such as plant height, leaf count, surface area, and biomass in a non-invasive, efficient manner【1】. Deep learning models, such as Convolutional Neural Networks (CNNs), enable automated analysis of these images by learning complex features that distinguish various traits【3】【5】.

Recently, machine learning algorithms, particularly CNN-based architectures, have demonstrated remarkable accuracy in plant phenotyping. These models can handle large-scale data and complex plant structures, outperforming traditional models in scalability and precision. For instance, deep learning architectures like U-Net and Mask R-CNN provide not only the ability to detect individual leaves but also to accurately segment them from cluttered backgrounds【5】【6】.

2.2 Leaf counting and surface area estimation

2.2.1 Leaf counting approaches

Leaf counting plays a fundamental role in plant phenotyping as it correlates strongly with plant health and yield potential. Manually counting leaves is labor-intensive and prone to human error, especially for large datasets. Deep learning approaches automate this task by using models that can accurately estimate the number of leaves from a single image【2】【5】.

Segmentation-based approaches, such as U-Net, focus on extracting leaf regions by identifying pixel-level boundaries between leaves and their surroundings. The Mask R-CNN model further enhances this by using region proposal networks to locate objects and segment them simultaneously, allowing for leaf counting even in complex plant structures where leaves may overlap each other【5】【6】.

An alternative to segmentation is direct leaf counting, which bypasses the need to detect and segment each leaf. For instance, multi-scale regression approaches leverage varying image resolutions to estimate leaf counts without pixel-level annotations. The multi-scale fusion technique proposed by Itzhaky et al. (2018) has been particularly useful for counting leaves in densely populated images where traditional detection methods may fail【3】.

Moreover, YOLO-based detection algorithms have been shown to be highly effective in real-time leaf counting tasks. YOLO (You Only Look Once) is a high-speed object detection model that divides an image into a grid and predicts bounding boxes and confidence scores for each object. This model is particularly efficient for leaf counting in dynamic environments, as it processes the entire image in a single pass, making it suitable for real-time applications【2】【9】.

2.2.2 Surface area estimation

While leaf counting provides discrete information about plant growth, surface area estimation offers a continuous metric, giving a better sense of the plant's overall health and photosynthetic potential. Surface area estimation techniques rely on accurate segmentation of leaf regions, followed by calculations of pixel-based measurements that are then converted into real-world units【3】.

Deep learning models like ResNet50 have been employed for estimating surface area from segmented leaf images. By combining convolutional layers that capture hierarchical image features with fully connected layers, these models achieve high accuracy in surface area estimation across various plant species. Additionally, methods that employ region proposals, like Faster R-CNN, can be adapted to measure surface area by detecting leaf boundaries and extracting spatial dimensions of the plant canopy【6】【7】.

2.3 Image processing techniques in agriculture

The accuracy of automated plant phenotyping heavily relies on the quality of image processing techniques. Here, segmentation and object detection are crucial in ensuring that the extracted features accurately represent plant traits such as leaf number and surface area.

- **Segmentation**: Accurate segmentation is essential for isolating individual leaves from their background. U-Net and Mask R-CNN are two popular models in this domain, providing state-of-the-art performance for tasks such as leaf segmentation. U-Net's encoder-decoder architecture allows for precise delineation of leaf edges, even in challenging conditions like shadowing or occlusion【5】【6】. In parallel, Mask R-CNN enhances object detection by using region-based segmentation, which is beneficial for identifying overlapping or hidden leaves【5】【6】.
- **Background Removal**: Removing the background noise from images is a critical preprocessing step in leaf detection and surface area estimation. Classical methods, like thresholding and color space transformations, are combined with deep learning techniques to improve accuracy. These methods are especially important in greenhouse environments where lighting conditions may vary【6】【8】.
- **Object Detection**: YOLOv3 and its variants are widely used in real-time leaf counting applications. YOLOv3's ability to simultaneously detect multiple leaves in a dense arrangement makes it a preferred choice for fast and accurate phenotyping. The model's grid-based detection strategy allows it to handle complex scenarios, where traditional methods might struggle to separate overlapping leaves【9】【8】.

2.4 Time series analysis in agriculture

2.4.1 Overview of ARIMA models

Time-series models, such as ARIMA (Auto-Regressive Integrated Moving Average), have long been employed to predict plant growth by analyzing historical

data trends. ARIMA models are particularly effective for short-term forecasting, such as daily or weekly growth increments, because they model temporal dependencies in the data. In agricultural contexts, ARIMA is widely used to model growth patterns over time. It's extension to ARIMAX can also provide insights into how external variables such as temperature, humidity, and light intensity influence plant development【4】【8】.

Moreover, ARIMA models have been combined with other techniques to improve accuracy in forecasting plant phenotypes under fluctuating environmental conditions. For example, hybrid ARIMA models, which integrate machine learning techniques like support vector regression (SVR), have been explored to handle more complex, non-linear growth patterns. This hybrid approach has shown improved performance in agricultural forecasting by accounting for seasonal variations and external factors【10】.

2.4.2 Bayesian models in growth prediction

Bayesian methods provide a flexible framework for predicting plant growth, particularly when dealing with uncertainty or limited data. Bayesian time-series models, such as those implemented using the PyMC3 library, allow for the incorporation of prior knowledge about the system, which can then be updated dynamically as new data becomes available. This is particularly useful in agriculture, where growth patterns are often non-linear and affected by various unpredictable environmental factors【3】【4】.

Bayesian approaches have also been extended to hierarchical models, where the growth of different plants or groups of plants can be modeled together, sharing information across levels of the hierarchy. This type of model improves predictive accuracy, particularly in scenarios where data is sparse or when modeling plants under different environmental conditions. The flexibility of Bayesian models also allows them to handle missing or noisy data more effectively than traditional methods like ARIMA【11】.

2.4.3 Advanced deep learning technics

Beyond ARIMA and Bayesian methods, newer time-series techniques such as Long Short-Term Memory (LSTM) networks have been introduced to predict plant growth more accurately. LSTMs, a type of recurrent neural network (RNN), are particularly well-suited for time-series forecasting as they capture long-term dependencies in sequential data. LSTMs have shown great promise in predicting agricultural time-series data, especially for long-term growth forecasts where traditional models like ARIMA may struggle【12】.

Another advanced approach is Gaussian Process Regression (GPR), which has been applied to plant growth prediction. GPR models provide a probabilistic framework that can quantify uncertainty in predictions, making them useful for agricultural applications where precise predictions are challenging due to the complexity of biological processes【13】.

Chapter 3. Image analysis in Plant Science

3.1 Introduction to Digital Image Processing

Digital image processing plays a crucial role in modern plant phenotyping, enabling the automated analysis of plant growth and health. The key advantage of digital imaging is its non-invasive nature, allowing researchers to observe phenotypic traits without disturbing the plant's growth process. In agricultural science, image analysis provides tools to extract valuable information, such as leaf count, surface area, and growth dynamics, which are critical for understanding and predicting plant development. Image-based phenotyping systems rely on capturing high-quality images and applying sophisticated processing algorithms to extract meaningful information. The increasing availability of affordable hardware and open-source software has made image processing more accessible to the agricultural community, paving the way for precision farming practices.

3.1.1 Types of Digital Images

Digital images used in plant phenotyping can be broadly categorized based on the spectral range in which they are captured. The most common image types include:

- **RGB (Red-Green-Blue) Images**: These are standard color images captured using conventional cameras. RGB images are widely used due to their simplicity and ease of processing. They provide enough information for tasks like leaf counting and surface area estimation.
- **Hyperspectral Images**: Hyperspectral imaging captures data across multiple bands of the electromagnetic spectrum, allowing for the detection of subtle changes in plant physiology. These images are particularly useful in identifying stress factors, such as nutrient deficiencies or disease.
- **Thermal Images**: Thermal cameras capture infrared radiation, offering insights into plant transpiration and heat stress. These images are valuable in identifying areas where plants may be experiencing drought stress or irregular water distribution.

3.1.2 Image Acquisition

Acquiring high-quality images is the foundation of successful plant image analysis. In plant phenotyping, cameras are typically mounted on fixed stations, drones, or robots to capture images at regular intervals. Raspberry Pi-based systems, as used in this research, are gaining popularity due to their affordability and flexibility. Equipped with a Pi camera, these systems can capture time-lapse images at set intervals to monitor plant growth continuously. Additionally, environmental data such as temperature and humidity are often recorded alongside images to provide context for growth variations. High-resolution cameras are essential for ensuring the accuracy of subsequent image analysis steps, such as leaf segmentation and surface area calculation.

3.2 Image pre-processing techniques

Image pre-processing is a critical step in preparing raw images for analysis. It enhances image quality, corrects distortions, and ensures that the plant features are more distinguishable from the background. Common pre-processing steps include background removal and noise reduction.

3.2.1 Background removal

Background removal is essential in plant image analysis as it isolates the plant from the surrounding environment. Without effective background removal, the image analysis algorithms may incorrectly identify non-plant elements, reducing the accuracy of metrics such as leaf count and surface area. Various techniques are employed to remove backgrounds, including thresholding and color-space transformations. For instance, converting an image from RGB to the HSV (Hue-Saturation-Value) color space can make it easier to separate plant material from the background based on color differences. Machine learning methods, such as Mask R-CNN, can also be used to segment plants and remove backgrounds more accurately in complex scenarios【5】【6】.

3.2.2 Noise reduction and Filtering

Images captured in natural environments often contain noise due to factors like uneven lighting, shadows, and reflections. Noise reduction techniques, such as Gaussian filtering, help in smoothing images while preserving important features like edges. Median filtering is another popular technique used in plant image processing, particularly when dealing with images that contain salt-and-pepper noise. These methods improve the quality of the image, ensuring more accurate leaf detection and surface area estimation【5】【6】.

3.3 Leaf detection algorithms

Leaf detection is a core task in plant phenotyping, as the number of leaves is a key indicator of plant health and growth. Various algorithms have been developed for detecting leaves, ranging from traditional computer vision techniques to advanced deep learning approaches.

3.3.1 Traditional approaches

Traditional leaf detection approaches primarily rely on edge detection and contour-based methods. Algorithms such as Canny edge detection and Hough Transform are used to identify the boundaries of leaves. Once the edges are detected, further processing, like contour finding, can be applied to count the number of distinct leaves. These methods, although effective in simple scenarios, often struggle in complex images where leaves overlap or have irregular shapes. Additionally, varying lighting conditions and occlusions make traditional methods less reliable.

3.3.2 Deep Learning Approaches

In recent years, deep learning techniques have revolutionized leaf detection by significantly improving accuracy in complex scenarios. Models like U-Net and Mask R-CNN have been widely adopted in plant phenotyping. U-Net, with its encoder-decoder architecture, excels at segmenting individual leaves from an image, even when they overlap. Mask R-CNN, on the other hand, extends this capability by performing both object detection and instance segmentation, allowing it to accurately detect each leaf in dense foliage. These methods use large labeled datasets

for training, which enable them to generalize well across different plant species and environmental conditions【5】【6】.

3.4 Feature extraction for Plant Growth

Feature extraction is the process of identifying and quantifying phenotypic traits that can be used to assess plant growth. Leaf count and surface area are two primary features extracted from plant images.

3.4.1 Leaf count and surface area

Leaf count is a straightforward but highly informative metric for assessing plant health. Automated leaf counting can be performed using deep learning models like Mask R-CNN or YOLO, which detect individual leaves even in cluttered images. Surface area, on the other hand, provides a more comprehensive measure of plant size and health. After detecting the leaves, surface area is estimated by calculating the number of pixels that correspond to the leaves and converting this value to a real-world measurement using calibration techniques【5】【6】.

3.4.2 Challenges in Leaf Segmentation

Leaf segmentation presents several challenges, particularly in images where leaves overlap or are partially occluded. In such cases, traditional segmentation methods often fail to distinguish individual leaves. Variations in leaf shape and size also add complexity to the task. To address these challenges, deep learning models like Mask R-CNN and U-Net have been developed to handle dense foliage by learning from large datasets. However, these methods require extensive computational resources and high-quality labeled data for training【6】【7】.

3.5 Image Calibration and Cropping

Image calibration ensures that the measurements extracted from images are accurate and consistent. Calibration typically involves correcting distortions caused by the camera lens and converting pixel-based measurements into real-world units. Calibration is particularly important for tasks like surface area estimation, where accurate scaling is crucial. Cropping, on the other hand, removes unnecessary parts

of the image, focusing the analysis on the region of interest, such as the plant canopy.

3.6 Summary

This chapter has outlined the key steps in image processing for plant phenotyping, from image acquisition and pre-processing to leaf detection and feature extraction. The advancements in deep learning techniques, particularly in leaf segmentation and surface area estimation, have significantly improved the accuracy and scalability of plant phenotyping. However, challenges such as occlusion and variability in plant structure continue to present obstacles, highlighting the need for further research and innovation in this field.

Chapter 4. Experiment setup and data collection

4.1 Overview of plant type and Growth stages

For this experiment, we selected **lettuce (Lactuca sativa)**, a popular leafy vegetable known for its fast growth, ease of cultivation, and robustness in various environmental conditions. Lettuce is a versatile crop commonly used in salads and sandwiches, with varieties such as romaine, butterhead, and leaf lettuce being widely cultivated across the globe. It has a relatively short growth cycle, which makes it an ideal candidate for growth prediction and phenotyping experiments.

Figure 1 - Romaine Lettuce example

Lettuce was chosen for this study due to several key factors. First, its growth cycle is relatively short, typically ranging from 30 to 50 days, which makes it suitable for experimental setups where time is a limiting factor. Furthermore, lettuce is resilient and can adapt to different growing environments, from outdoor soil-based cultivation to controlled indoor conditions. This robustness reduces variability in the experimental conditions, allowing for more consistent data collection. Lettuce is also relatively easy to grow from seed, requiring basic care such as consistent watering, adequate sunlight, and protection from extreme temperatures, which simplifies the setup and ensures a successful harvest.

Figure 2 - Experiment setup. End of seeding stage

In this experiment, lettuce plants were grown from seeds, following standard practices to ensure proper germination and growth. **Seeds were planted in individual pots filled with nutrient-rich soil** to prevent overcrowding and ensure each plant had enough space to develop fully. In total, eight plants were grown, each in a separate pot, to monitor individual growth patterns and avoid competition for resources.

The experiment was conducted over a period of 42 days, during which the plants progressed through the following growth stages:

1. **Seed Stage**: The experiment began with the planting of lettuce seeds, which were sown at a shallow depth of around ¼ inch in well-drained soil. Lettuce seeds generally require a temperature of around 60–70°F (15–21°C) to germinate effectively. Under optimal conditions, germination occurred within 5 to 10 days, with tiny cotyledons—the first leaves—emerging from the soil.
2. **Seedling Stage**: Once the seeds germinated, the plants entered the seedling stage, where they developed their first true leaves. During this period, maintaining the right balance of light, temperature, and moisture was critical. The seedlings were thinned out, if necessary, to ensure each plant had enough space to grow.

3. **Rosette Stage**: As the lettuce continued to grow, the plants entered the vegetative or rosette stage. In this phase, which lasted between 25 and 50 days, the plants rapidly produced leaves, forming a circular arrangement typical of lettuce growth. Consistent watering and nutrient supply were critical during this stage to promote leaf development and ensure a healthy plant structure.
4. **Cupping Stage**: After the rosette stage, the plants entered the cupping stage, a brief period during which the outer leaves began to curl inward, preparing for the formation of a dense head in head lettuce varieties. This stage typically lasted about a week. Monitoring for pests and maintaining soil moisture were important to prevent stunted growth or early bolting.
5. **Heading Stage**: In the final growth stage before harvest, the plants formed a compact head. For leaf lettuce, this phase did not result in head formation, but the leaves were still ready for harvesting. For head-forming lettuce varieties, such as romaine, the heading stage lasted an additional 20 to 45 days, depending on environmental conditions.

At the end of the 42-day experimental period, the plants were harvested. During the final harvest, the above-ground biomass of each plant was carefully collected. The harvested plants were weighed to determine their fresh biomass, a key metric for evaluating plant growth and overall health. Lettuce is typically harvested once has formed a full rosette of leaves or, in the case of head lettuce, when the head has firmed up.

Figure 3 - End of the experiment. Harvesting plants

4.2 Equipment setup tools

4.2.1 Hardware

The **Raspberry Pi** is a versatile, single-board computer widely used for a range of tasks in both hobbyist and professional settings. We utilized the **Raspberry Pi Model 3**, which is a popular version known for its balance of performance, affordability, and energy efficiency. This model features a quad-core 1.2 GHz ARM Cortex-A53 processor and 1 GB of RAM, making it capable of handling tasks that involve data collection, image processing, and basic automation. Raspberry Pi devices are commonly used for tasks such as home automation, robotics, Internet of Things (IoT) applications, and environmental monitoring.

The Raspberry Pi 3 was chosen for this experiment due to its **low power consumption**, which is particularly important when running a 42-day continuous experiment. The device's built-in GPIO (General Purpose Input/Output) pins also allowed seamless integration with sensors such as temperature and humidity monitors, while its USB ports enabled the connection of external peripherals, including the Raspberry Pi camera used in this setup. Furthermore, the **Raspberry Pi runs on open-source software**, making it highly customizable for specific project needs, and the Python environment it supports provides an excellent platform for running automation scripts.

Raspberry Pi Camera Module

For capturing images of the lettuce plants, we used the **Raspberry Pi Camera Module**, specifically the **Pi3 camera**. This camera module integrates directly with the Raspberry Pi through the dedicated camera port and provides high-quality image capture capabilities. The Pi3 camera has the following key specifications:

- **Resolution**: 8 megapixels
- **Image size**: 3280 x 2464 pixels
- **Lens**: Fixed-focus, capable of capturing detailed close-up images, which is ideal for plant phenotyping experiments.
- **Frame rate**: Supports 1080p video at 30 frames per second (fps), though for this experiment, we focused on still image capture.

The camera was configured to capture high-resolution images of the lettuce plants at hourly intervals, providing continuous documentation of plant growth. The fixed-focus lens of the Pi3 camera ensured that each image was sharp, and the camera's compact size allowed it to be mounted in close proximity to the plants without interfering with their environment. This setup was essential for monitoring subtle changes in leaf development, which were later analyzed for growth patter

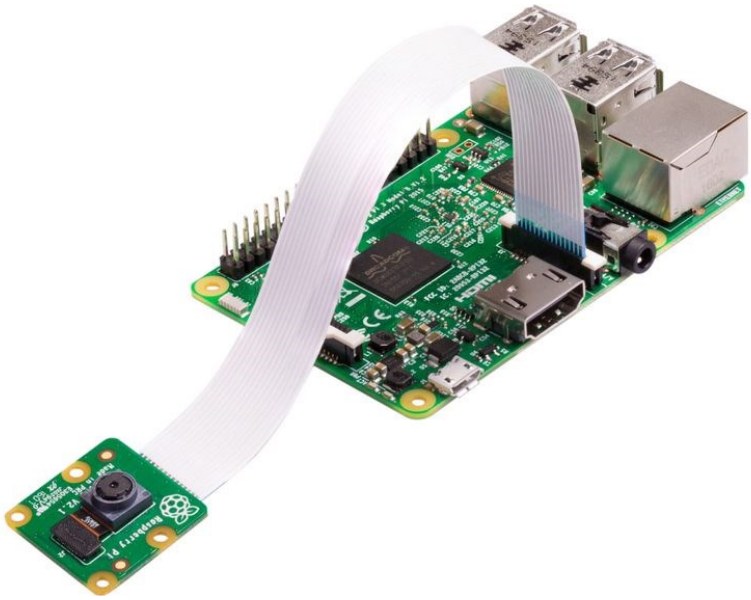

Figure 4 - Raspberry Pi 3 with Pi camera

Temperature and Humidity Sensors

Accurate monitoring of environmental conditions is crucial in plant growth experiments, as factors such as temperature and humidity significantly affect plant development. For this project, we used a **DHT11 temperature and humidity sensor**. The DHT11 sensor is a low-cost digital sensor that can measure:

- **Temperature**: Ranges from 0°C to 50°C with an accuracy of ±2°C.
- **Humidity**: Ranges from 20% to 90% relative humidity (RH), with an accuracy of ±5%.

The **DHT11 sensor** was connected to the Raspberry Pi via its GPIO pins, allowing real-time temperature and humidity data collection throughout the experiment. These environmental parameters were logged every hour alongside the

images, providing a comprehensive dataset for analyzing the relationship between environmental factors and plant growth.

4.2.2 Software

The Python script used for the experiment was designed to automate both image capture and environmental data collection, making the entire process efficient and self-sustaining. At a high level, I developed the script which performs the following tasks:

1. **Directory and File Setup**: Before collecting any data, the script checks whether the necessary directories and files exist. These include directories for storing images and CSV files for logging environmental data (temperature and humidity) and any errors that occur during execution.
2. **Sensor and Camera Initialization**: The script interfaces with a DHT11 temperature and humidity sensor using the Adafruit library. This sensor reads the temperature and humidity at regular intervals. Simultaneously, the Raspberry Pi camera is configured using the Picamera2 library, with settings to capture high-resolution still images. The camera is manually focused to ensure clarity in the images.
3. **Data and Image Capture**: The script runs an infinite loop, capturing data from the DHT11 sensor and saving it to a CSV file every hour. Alongside this, it captures an image of the lettuce plants and stores it in a specified directory. Each image and data point is timestamped for easy tracking.
4. **Error Handling**: The script includes error-handling mechanisms to manage potential issues, such as sensor failures or camera errors. If an error occurs, it is logged in a separate CSV file with the timestamp and error type, ensuring that any issues are documented for future reference.
5. **Automation**: To ensure that the data collection process continues uninterrupted, the script was configured to run as a background process on the Raspberry Pi. This means that even after a power outage or system reboot, the script automatically restarts without requiring manual intervention.

This setup allowed for continuous, automated monitoring and documentation of the experiment, providing consistent data for later analysis.

4.3 Experiment data collection

Data collection is a critical part of this experiment, as it serves as the foundation for both the analysis of plant growth and the development of prediction models. Over the course of the 42-day experimental period, we employed a systematic approach to gather both visual and environmental data, ensuring consistency and accuracy in the dataset.

Hourly Image Capture

The primary data source for this experiment was the **hourly image capture** of the lettuce plants, facilitated by the Raspberry Pi Camera Module. Each plant was photographed in high resolution once every hour, capturing its growth and physical changes in real-time. This resulted in a large collection of images over the 42-day period, providing a detailed time-lapse of plant development from seedling to full maturity.

Each image was automatically saved with a timestamp, which allowed us to correlate visual growth patterns with environmental factors such as temperature and humidity. The high frequency of image capture ensured that even subtle growth changes, such as new leaf formations or minor structural adjustments, were documented. This wealth of image data would later be used to analyze growth trends, measure leaf area, and monitor overall plant health.

Environmental Data: Temperature and Humidity Monitoring

In addition to the visual data, **environmental conditions** were meticulously monitored throughout the experiment. Temperature and humidity levels can significantly affect plant growth, and it was essential to capture these parameters at regular intervals to establish any correlation between environmental changes and plant development. The DHT11 temperature and humidity sensor was used to record these metrics once every hour.

The Raspberry Pi was programmed to collect the temperature (in degrees Celsius) and humidity (as a percentage) at the same intervals as the image captures. This environmental data was logged into a CSV file alongside the images, ensuring that each data point had a corresponding timestamp. By synchronizing the environmental data with the image capture, we ensured that any deviations in plant

growth could be cross-referenced with changes in temperature or humidity, providing a robust dataset for predictive modeling.

Ensuring Data Integrity

To ensure the integrity of the collected data, several measures were put in place:

1. **Backup Protocols**: The Raspberry Pi was configured to back up data automatically to an external storage device at the end of each day. This prevented data loss in case of hardware failure or power outages.
2. **Error Logging**: As previously mentioned, the Python script was designed to log any errors encountered during the data collection process. If a sensor malfunctioned or the camera failed to capture an image, these errors were documented with timestamps, allowing us to track and address any anomalies in the data collection.
3. **Manual Checks**: While the system was automated, periodic manual checks were performed to ensure that the sensors and camera were functioning correctly. These checks included verifying that images were being captured at the correct intervals and that environmental data was being logged accurately.

timestamp	error_type	message
2024-06-21_16:42:17		success
2024-06-21_17:24:01		success
2024-06-21_18:24:03		success
2024-06-21_19:24:04		success
2024-06-21_20:24:05	runtime_error	Checksum did not validate. Try again.

Table 1 - CSV file structure with running logs

4.4 Raw data description

The raw data collected during the 42-day experiment consisted of two main types: **image data** and **environmental data**. Both data types were collected continuously at hourly intervals, creating a detailed record of plant growth and

environmental conditions. This section provides an overview of the structure, content, and characteristics of the raw data that forms the basis for the subsequent analysis.

Image Data

The image dataset comprises a total of **1,000 high-resolution images**, with each of the eight plants captured once every hour throughout the 42 days of the experiment. The images were stored in separate directories for each plant to facilitate easier access and analysis.

File Naming Convention: Each image was saved with a unique file name indicating the timestamp of when it was captured, using the format *lattuga_YYYY-MM-DD_HH:MM:SS.jpg*. This allowed for easy correlation with the environmental data and precise tracking of growth over time.

Resolution: The images were captured at a resolution of **3280 x 2464 pixels**, providing enough detail to analyze individual leaves, detect subtle changes in leaf morphology, and calculate surface area. The high resolution was particularly beneficial for accurately monitoring the structural growth of the plants, even in the early stages.

Image Consistency: Throughout the experiment, the images were captured under relatively stable lighting conditions to minimize variability. However, occasional fluctuations in natural light or shadows may be present in some images, which will be addressed during the image pre-processing phase in later chapters.

Environmental Data

The environmental data collected during the experiment included hourly records of temperature (in °C) and humidity (in %) for the entire duration of the study. These measurements were stored in a CSV file format, making them easily accessible for further analysis and modeling.

Data Fields: The environmental data was logged with three key fields:

1. **Timestamp**: The exact time the measurement was taken (e.g., 2023-08-15_14:00:00), allowing for easy synchronization with the corresponding image data.

2. **Temperature (°C):** The temperature of the environment surrounding the plants, recorded hourly using the DHT11 sensor. The temperature varied between 18°C and 25°C during the experiment, staying within the optimal range for lettuce growth.
3. **Humidity (%):** The relative humidity in the environment, which ranged between 50% and 75% depending on the day. These variations could impact plant growth, making this data valuable for understanding any correlations between environmental factors and plant development.

timestamp	temperature_c	humidity
2024-06-21_17:24:01	21	73
2024-06-21_18:24:03	22	74
2024-06-21_19:24:04	22	74
2024-06-21_20:24:05	22	74
2024-06-21_21:24:06	21	73

Table 2 - CSV file structure with the environmental data

CSV File Format: The environmental data was stored in CSV format with the following structure: *timestamp, temperature_c, humidity*. This simple structure ensured that the data could be easily imported into various analysis tools for further processing.

Preliminary Observations

Once the data collection phase was complete, the dataset was prepared for analysis. This involved organizing the images and environmental data into a format compatible with the analytical tools to be used in later chapters. The images were processed to ensure uniformity in terms of size, resolution, and framing, while the environmental data was cleaned to remove any errors or anomalies that were detected during the collection phase. This preparation step was crucial for ensuring that the subsequent analysis and predictive modeling would be based on accurate and high-quality data.

While the primary analysis of the data is presented in later chapters, some preliminary observations can be made from the raw data:

- The hourly images captured distinct growth stages in the lettuce plants, from the early seedling stage through to full maturity.
- The environmental data showed some variability in temperature and humidity, particularly during the night and early morning hours. This variability could play a role in the observed differences in growth rates between the individual plants.

Overall, the raw data collected forms a comprehensive foundation for the detailed analysis of lettuce growth and the development of predictive models, as discussed in the following chapters.

4.5 Raw data preprocessing

Raw data preprocessing is an essential step in preparing the collected image data for further analysis. This process ensures that the images are organized, cropped, and ready for downstream tasks such as feature extraction and growth analysis. The following steps outline the preprocessing tasks performed on the raw data in this experiment:

1. Organizing and Loading Image Files

The first step in the preprocessing pipeline involved organizing the image files by their respective timestamps and plant identifiers. The images collected during the experiment were stored in the directory data/daylight_images, with filenames following a specific format: plantname_YYYY-MM-DD_HH.jpg. These filenames encoded the plant ID, the date, and the time of capture, which allowed for easy sorting and chronological arrangement of the images.

Using Python's os library, the images were read and listed for processing. The images were sorted by filename to ensure that the chronological sequence of plant growth was maintained throughout the analysis.

2. Defining Plant Centers and Cropping

Given that each image captured multiple plants, it was necessary to isolate individual plants from the full image. This was achieved through **cropping**, where each plant was extracted based on its pre-determined center coordinates. The coordinates of the center for each plant were manually defined in two sets, **centers_1**

and **centers_2**, corresponding to different phases of the experiment (before and after a specific date defined by a **breakpoint**).

- **Centroid Coordinates**: Two dictionaries, centers_1 and centers_2, were created to store the x and y coordinates for the center of each plant in the image.
- **Cropping Window**: Each plant's region of interest (ROI) was defined by a square window, with the side lengths of either 900 pixels or 700 pixels, depending on the timestamp of the image.

During preprocessing, the code calculated the window around each plant based on its center coordinates. The plant image was then extracted using array slicing, isolating the specific plant from the full image.

3. Time-Based Conditional Cropping

A critical aspect of the preprocessing involved handling the images based on their capture time. A specific **breakpoint** (set to **July 4, 2024, at 15:00**) was defined to account for changes in the experimental setup or camera configuration that affected the size of the plant images. Based on whether an image was captured before or after this breakpoint, the appropriate side length and center coordinates were chosen for cropping.

- For images captured **before** the breakpoint, a window of size **900 x 900 pixels** was used for cropping.
- For images captured **after** the breakpoint, a smaller window of **700 x 700 pixels** was used.

This conditional cropping ensured consistency in the size and focus of the extracted plant images despite changes in the experimental conditions.

4. Saving Preprocessed Images

Once the plants were cropped from the raw images, the individual plant images were saved in the directory data/daylight_preprocessed_images. Each image was saved with a new filename, following the format lettuce{i}_YYYY-MM-DD_HH.jpg, where {i} represents the plant number and the rest of the filename reflects the original timestamp of the capture.

The imsave function from the skimage.io module was used to save the images, ensuring that the format and quality of the images were maintained throughout the preprocessing pipeline.

5. Data Preparation for Analysis

This preprocessing step generated a dataset of isolated plant images, each corresponding to a specific plant and timestamp. By organizing and cropping the images in this structured way, the resulting dataset was well-prepared for the next stages of analysis, such as feature extraction, leaf counting, and growth prediction. This preprocessing step significantly reduced the complexity of the data, enabling more focused and efficient analysis on individual plants.

6. Filtering daytime images

After cropping and saving the individual plant images, an additional filtering step was performed to ensure that only images captured during the daytime—when sufficient natural light was present—were included in the dataset. This step was crucial to exclude nighttime images, which did not provide meaningful visual information due to low lighting conditions.

The filtering process was conducted using an additional Python script that analyzed the brightness levels of each image. The basic approach involved the following tasks:

Brightness Analysis for Daytime Detection

For each image, the overall brightness was calculated by converting the image to grayscale and computing the mean pixel intensity. Images captured during the daytime exhibited higher brightness values due to the presence of natural light, whereas images captured at night had significantly lower brightness levels.

- **Grayscale Conversion**: Each RGB image was converted to a grayscale image, where pixel intensity values ranged from 0 (black) to 255 (white).
- **Thresholding**: A brightness threshold was defined based on the mean pixel intensity. Images with a mean brightness value below this threshold were considered too dark to provide meaningful information and were thus classified as nighttime images. These images were excluded from further analysis.

Filtering and Storing Daytime Images

Once the brightness analysis was completed, only the images with sufficient brightness (i.e., those captured during the day) were retained. The daytime images were stored in a separate folder for subsequent analysis. This step ensured that the dataset contained only high-quality images where plant growth could be clearly observed.

By filtering out nighttime images, this step helped to enhance the overall quality and consistency of the dataset, reducing noise and ensuring that the remaining images provided valuable information for the analysis of plant growth over time.

Chapter 5. Leaf detection and surface calculation

5.1 Review of the existing algorithms

Leaf detection in plant phenotyping has seen significant advancements over the years, with a wide range of algorithms being developed to enhance the accuracy and efficiency of this task. From traditional image processing techniques to advanced deep learning models, each method brings unique strengths and limitations depending on the context in which they are applied. Below is an expanded review of five key algorithms for leaf detection.

1. Canny Edge Detection and Hough Transform

The **Canny Edge Detection** algorithm, introduced by John Canny in 1986, is one of the most widely used edge detection techniques in image processing. It detects edges by first smoothing the image to reduce noise, then identifying intensity gradients, and finally applying edge-tracing algorithms to highlight continuous boundaries. In leaf detection, this method identifies the edges of leaves based on the contrast between the plant and the background. Once the edges are detected, the **Hough Transform** can be applied to detect circular or elliptical shapes, which often approximate the shape of leaves【14】.

However, while Canny edge detection is computationally efficient and easy to implement, it has several limitations. It struggles with complex plant structures, especially when leaves overlap or lighting conditions vary. Moreover, it is sensitive to noise and background interference, requiring controlled environments for optimal performance. In agricultural applications where leaves are often occluded by others, more advanced methods are usually preferred.

2. U-Net for Leaf Segmentation

U-Net is a deep learning model that was originally developed for biomedical image segmentation. Its architecture consists of an encoder-decoder structure, where the encoder captures features from the image and the decoder reconstructs them to form precise segmentation masks. U-Net has become one of the most popular models for leaf segmentation due to its ability to accurately delineate leaf boundaries, even in densely packed and complex plant structures【15】.

One of the key strengths of U-Net is its ability to capture context at multiple scales. By using skip connections between the encoder and decoder, the model retains high-resolution information while also learning abstract features from deeper layers. This makes U-Net particularly effective at detecting individual leaves in images where leaves overlap or are partially occluded. Its success in the field of plant phenotyping is largely due to its versatility and accuracy in segmenting both simple and complex images.

3. Mask R-CNN for Instance Segmentation

Mask R-CNN is a more advanced deep learning model that extends traditional object detection networks by adding a segmentation mask for each detected object. It was developed to address the challenge of detecting and segmenting individual instances of objects within an image. In the context of leaf detection, Mask R-CNN is particularly useful because it not only detects each leaf but also generates a pixel-wise segmentation mask, providing detailed information about the size, shape, and position of each leaf【16】.

The architecture of Mask R-CNN builds upon Faster R-CNN by introducing an additional branch for segmentation. This allows the model to handle complex plant structures, where leaves may overlap or be hidden behind others. The model's ability to segment leaves at the pixel level makes it ideal for calculating phenotypic traits like leaf count and surface area. However, Mask R-CNN is computationally intensive and requires a large amount of labeled data for training, which can be a limitation in some agricultural settings.

4. YOLO (You Only Look Once)

YOLO (You Only Look Once) is a real-time object detection algorithm that processes an entire image in a single pass, making it exceptionally fast compared to other object detection models like Faster R-CNN. YOLO has been successfully applied in plant phenotyping for tasks like leaf detection and counting. **YOLOv3** and **Tiny-YOLOv3** versions are commonly used for detecting leaves in real-time, providing an efficient solution for large-scale agricultural applications【8】.

One of the key advantages of YOLO is its speed. By dividing the image into a grid and making predictions for each cell in a single forward pass, YOLO can detect

and localize objects (leaves) much faster than traditional methods. This makes it ideal for applications where quick decision-making is crucial, such as in precision agriculture. However, YOLO's accuracy is somewhat lower compared to more specialized models like Mask R-CNN, especially when dealing with densely packed plants or small leaves.

5. DeepLab for Semantic Segmentation

DeepLab is a deep learning model designed for semantic segmentation, which means that it assigns a class label to each pixel in the image. DeepLab's unique approach lies in its use of **atrous (dilated) convolutions**, which allow the model to capture larger contextual information without increasing the number of parameters or sacrificing resolution. This is particularly important in leaf detection, where capturing the spatial relationships between leaves is critical for accurate segmentation【6】.

DeepLab has proven to be highly effective in leaf detection, especially in high-resolution images where fine-grained segmentation is necessary. The model's ability to handle multiple scales of objects makes it well-suited for detecting leaves of varying sizes and orientations. Additionally, DeepLab's flexibility allows it to be used in different agricultural environments, from controlled greenhouses to open fields. However, like other deep learning models, DeepLab requires a significant amount of computational resources and high-quality labeled data for training.

5.2 Improvements to Leaf Detection Algorithm

In this section, we describe the refinements made to the existing leaf detection algorithm, primarily focusing on adapting simple image processing techniques to better fit the dataset used in this experiment. While the improvements made were not revolutionary in terms of novel algorithm development, they were crucial in customizing the algorithm to work effectively with the specific characteristics of the images collected during the plant growth experiment. The goal was to enhance accuracy and reliability in counting leaves across different growth stages using a series of sequential image processing filters.

5.2.1 Methodology of Enhancement

The core of the methodology revolved around the **sequential application of filters** from the skimage library to improve leaf segmentation and counting. The chosen approach was relatively simple but effective, relying on traditional image processing techniques rather than advanced machine learning models. These techniques included color space conversion, noise removal, morphological operations, and connected component analysis. The following steps outline the workflow:

1. **Color Space Conversion (RGB to HSV)**: The first step in the process was converting the RGB image into the HSV (Hue, Saturation, Value) color space. This conversion allowed for more effective segmentation based on color, as the hue channel in HSV is particularly well-suited for isolating green areas in plant images. In this case, specific thresholds were defined for the hue values to capture the green areas of the plants, which represented the leaves. The range of hue values was set between **0.15** and **0.2**, corresponding to the green hue spectrum in lettuce leaves.

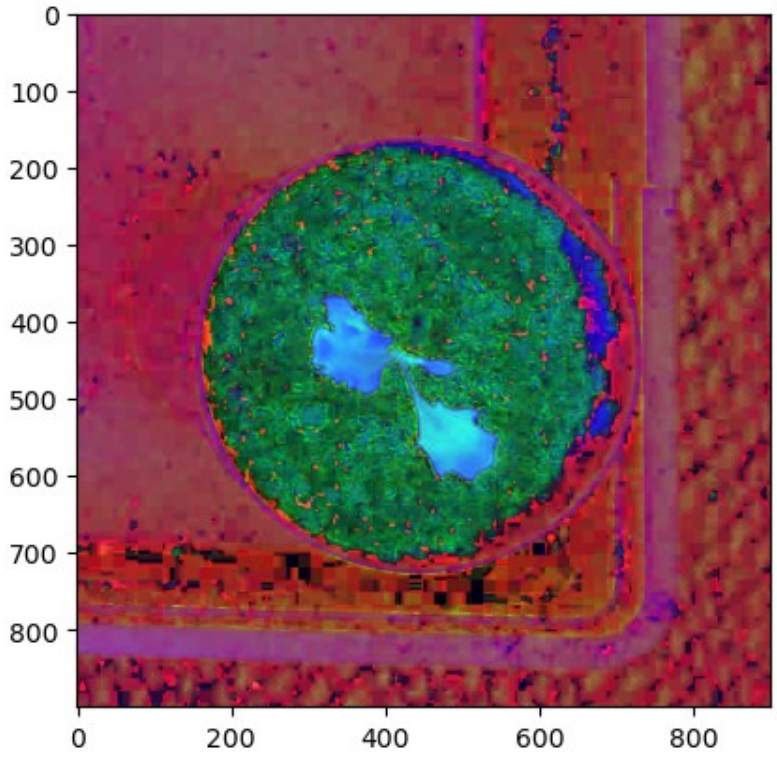

Figure 5 - Lettuce image in HSV color space

2. **Green Mask Application**: After converting the image to HSV, a **green mask** was applied to filter out non-leaf areas of the image. The mask was created by retaining pixels whose hue values fell within the specified range. This effectively isolated the plant leaves from other elements in the image, such as the soil, pot, or background noise.

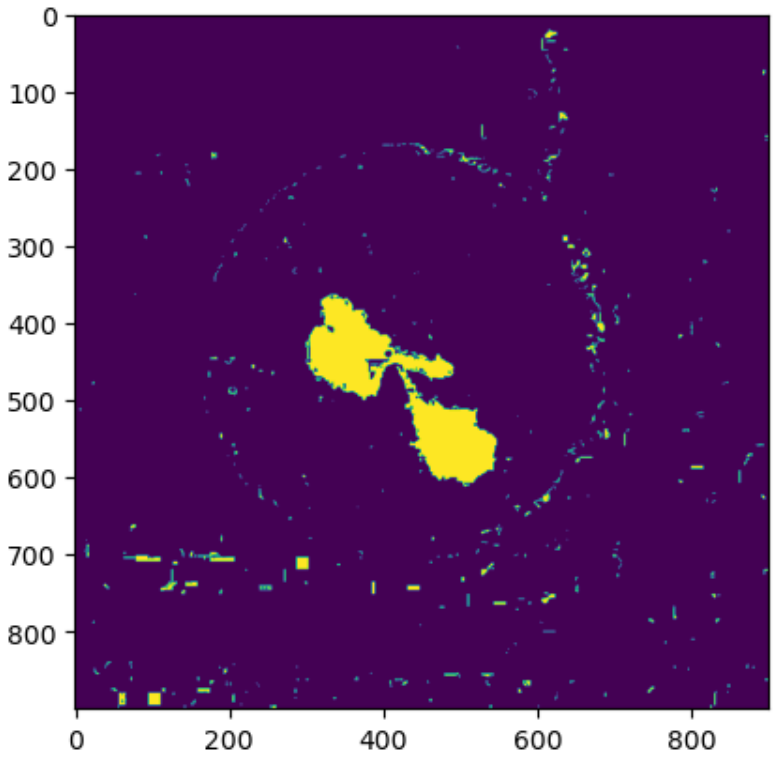

Figure 6 - Lettuce image with Green Mask applied

3. **Noise Removal and Small Object Filtering**: To ensure that small, irrelevant objects (such as small patches of dirt or random noise) were not misclassified as leaves, a morphological operation was performed to **remove small objects** from the binary mask. The minimum size of objects retained was set to 500 pixels, ensuring that only significant plant features were preserved for further analysis.

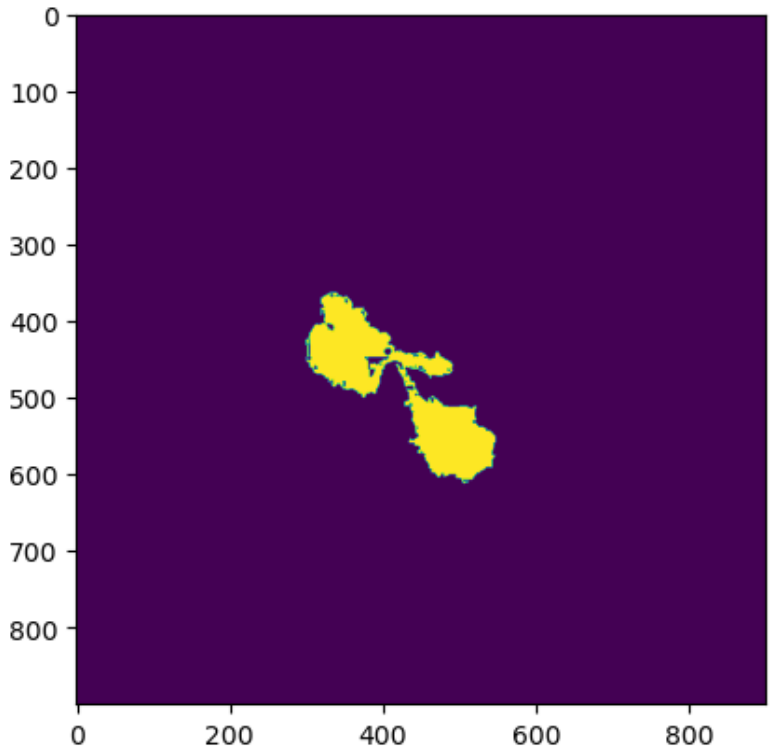

Figure 7 - Lettuce image without noise

4. **Morphological Opening**: A **morphological opening** operation was applied to clean up the segmented image further. This step involved the use of a disk-shaped structuring element to remove small, disconnected noise while retaining the shape and structure of the leaves. Morphological opening is particularly useful for smoothing the edges of segmented regions and removing artifacts introduced by the green mask application.

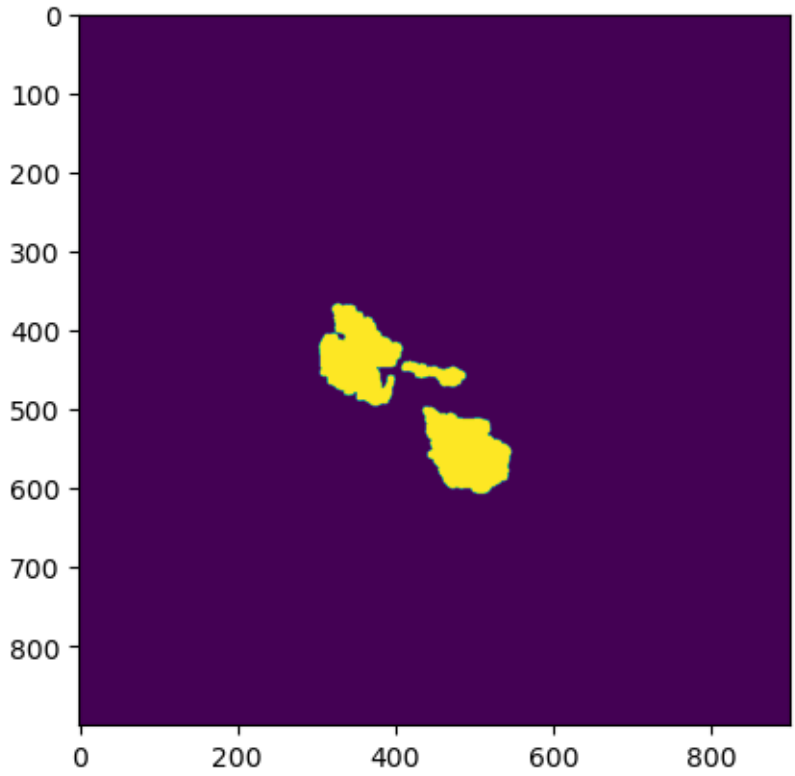

Figure 8 - Lettuce image with morphological opening

5. **Connected Component Labeling**: Once the leaves were effectively segmented, the next step was **connected component labeling**. This technique identifies individual connected regions (i.e., leaves) in the binary mask and labels them accordingly. By counting the number of connected components, the algorithm was able to determine the number of leaves present in each image.
6. **Leaf Count Calculation**: The labeled components were analyzed to calculate the total number of leaves in the image. The algorithm identified distinct regions corresponding to individual leaves and tallied them to produce a leaf count. This count was then logged along with the timestamp and plant number for each image.

This method proved to be effective in isolating and counting leaves from the dataset of lettuce plants, despite using relatively simple techniques compared to deep learning-based approaches. The combination of color-based segmentation, morphological filtering, and connected component analysis allowed for a robust and

accurate detection of leaves, even in images with challenging lighting conditions or slight occlusions.

5.2.2 Validation of Enhanced Accuracy

After the tailored leaf detection algorithm was implemented and integrated into the data processing pipeline, it was essential to evaluate its performance across the dataset, particularly in terms of accuracy and consistency over time. While the algorithm showed promising results, a deeper analysis revealed some variance in the leaf count estimates that required further refinement.

Time-Series Analysis of Leaf Count Data

The leaf count data collected from the images was initially compiled into a time-series dataset, with each row corresponding to a specific plant at a given timestamp. Using this data, an aggregated analysis of the leaf count over time was performed. A **line plot** was generated to visualize the progression of leaf counts for the entire experiment period, allowing us to detect trends and any potential anomalies.

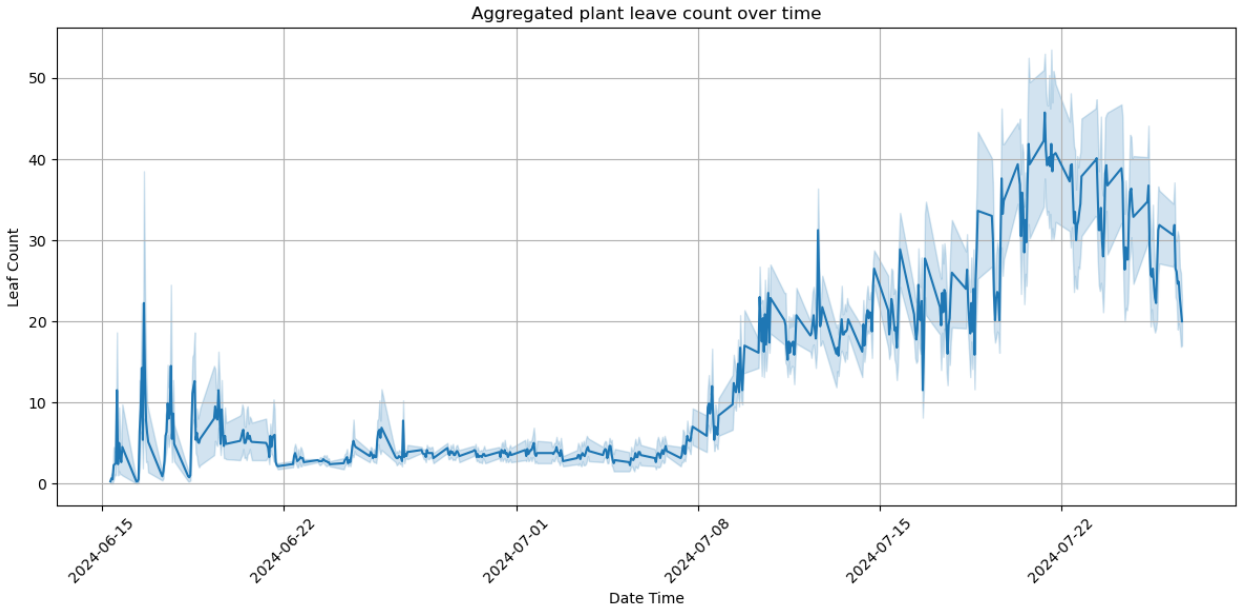

Figure 9 - Raw time series of the leave count visualization

The aggregated plot of the leaf count over time highlighted a **high variance** in the detection results, particularly in images where occlusion, lighting, or segmentation challenges arose. For instance, there were instances where the leaf

count would fluctuate significantly over short time periods, which was not consistent with the biological reality of the plants' growth process.

Application of Rolling Median to Reduce Variance

To mitigate the high variance observed in the time-series data, a **rolling median** technique was applied to smooth the leaf count values over a 24-hour window. By calculating the median leaf count within a 24-hour time frame, the approach helped reduce the impact of outliers and sudden spikes or drops in the detected leaf count that were likely caused by inconsistencies in image conditions rather than actual changes in leaf growth.

This smoothing process improved the overall trend visualization, revealing a more consistent and biologically plausible growth curve. The rolling median approach effectively minimized the sudden fluctuations in leaf count, providing a clearer picture of the plants' growth over time.

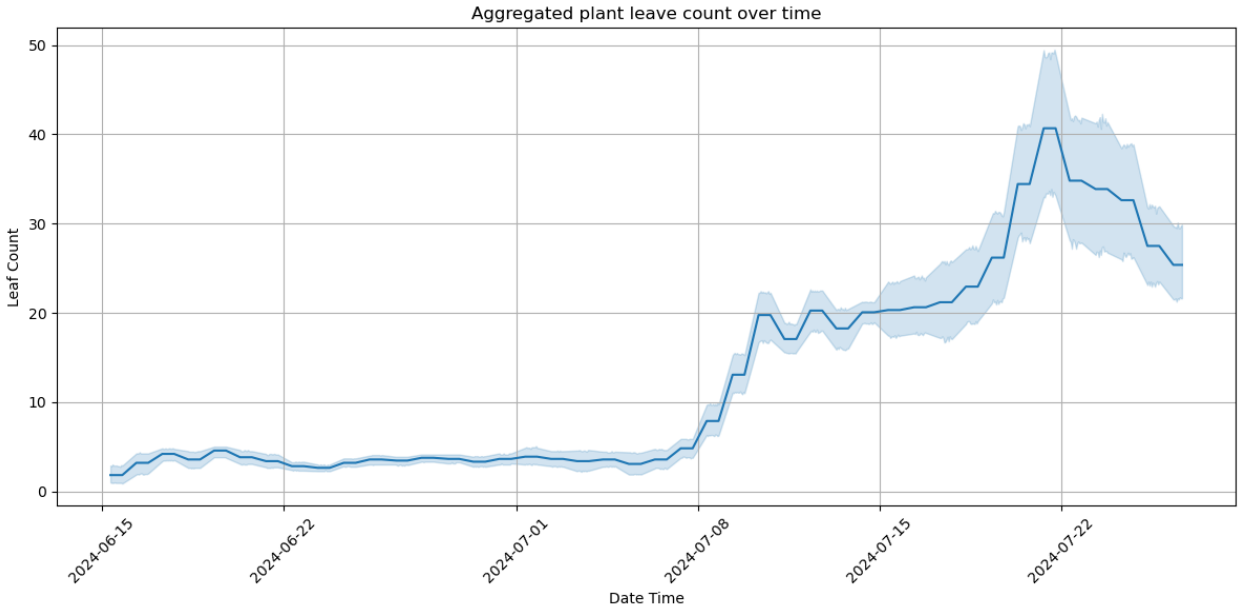

Figure 10 - Leaf count time series with 1-day mean rolling window

Logical Correction: Non-Decreasing Leaf Count Rule

One critical observation during the validation process was that, biologically, the number of leaves on a plant cannot decrease over time. However, due to segmentation errors or detection challenges, there were instances where the

algorithm reported fewer leaves than had previously been detected. To address this, a logical rule was introduced to enforce **non-decreasing leaf counts** over time.

The rule applied was straightforward: if a higher leaf count was detected at an earlier time point, the algorithm would adjust any subsequent counts that were lower, ensuring that the leaf count could only remain constant or increase.

This adjustment helped align the detected leaf counts with the biological constraints of plant growth. By ensuring that the leaf count did not decrease over time, the modified time-series more accurately reflected the true growth patterns of the lettuce plants.

Final Validation and Observations

The combination of rolling median smoothing and the application of the non-decreasing leaf count rule significantly improved the accuracy and robustness of the leaf count data. After these adjustments, the time-series data provided a more reliable and biologically consistent representation of plant growth.

However, despite these improvements, some residual variance remained, particularly in early stages of plant growth when leaf detection is most challenging. Variability in lighting conditions, leaf occlusion, and segmentation accuracy still contributed to minor inaccuracies in certain images. Nonetheless, the overall performance of the algorithm was deemed satisfactory for the purposes of this experiment, with further improvements possible through more advanced segmentation methods or deep learning approaches.

5.3 Plant surface calculation

In this section, we focus on the method used to calculate the surface area of each plant over time. The surface area of the plant is an important phenotypic trait that provides insights into the plant's overall health and growth potential. The approach used in this experiment was similar to the leaf detection method, relying on traditional image processing techniques with additional steps to convert the detected pixel area into real-world measurements.

5.3.1 Approach and Challenges

The surface area calculation followed a structured approach, combining color-based segmentation with pixel area conversion. The goal was to estimate the total area covered by the leaves in each image and convert that into a real-world unit (cm^2) to track plant growth.

Image Preprocessing and Segmentation

Similar to the leaf detection process, the first step in calculating surface area involved converting the RGB image into the HSV (Hue, Saturation, Value) color space. This color space was chosen because it allows for a more precise segmentation of the green areas of the plant. The hue range for green was defined between 0.15 and 0.30, which helped isolate the leaves from other elements in the image, such as the background or soil. Additionally, the saturation and brightness ranges were set to filter out pixels that were too dull or too bright, further refining the mask used to detect leaves.

Once the green mask was applied, small, irrelevant objects (such as noise or tiny specks of green) were removed using morphological operations. This step ensured that only significant plant features were retained for the surface area calculation.

Calculating Pixel-Based Surface Area

The next step was to calculate the number of **green pixels** in the image, which corresponded to the detected leaf area. This was done by summing all the pixels in the cleaned binary mask (where green areas were represented by True values).

The total number of pixels in the image was also computed to determine the resolution of the image, which was later used to convert pixel measurements into real-world units.

Pixel-to-Centimeter Conversion

To translate the pixel count into an actual surface area, the size of the image frame was considered. The image frame, represented in centimeters, allowed for the conversion of pixel-based measurements into real-world units (cm^2). This was achieved by calculating the area per pixel in the image, which was given by the formula:

$$\text{Pixel Area (cm\^2)} = \frac{\text{Image Side (cm)}^2}{\text{Total Pixels in Image}}$$

Using this pixel area value, the surface area covered by the green pixels was calculated as follows:

$$\text{Surface Area (cm\^2)} = \text{Green Pixels} \times \text{Pixel Area (cm\^2)}$$

Accounting for Leaf Overlap and Changes Over Time

One of the challenges in calculating surface area using 2D images is that leaves often overlap, resulting in an underestimation of the true surface area. To address this, a **leaf overlap factor** was introduced to adjust the surface area values based on the stage of plant growth. For example, during early growth stages, leaves are less likely to overlap, but as the plant matures and the canopy becomes denser, the algorithm was adjusted to account for potential overlap.

In addition, surface area calculations were tracked over time, allowing for adjustments based on changes in plant growth behavior as new leaves emerged and expanded.

5.3.2 Validation and comparison with leaf count

During the course of the experiment, it became evident that calculating the **surface area** of the plants, rather than focusing solely on leaf count, provided a more **robust and reliable metric** for analyzing plant growth over time. Unlike leaf counting, which faced significant challenges due to occlusion and the variance in leaf detection across different stages of growth, the surface area calculation consistently delivered more stable and biologically plausible results.

One of the key reasons for this reliability is that the surface area approach inherently accounts for the overall leaf coverage, rather than relying on detecting and counting individual leaves. As plants matured and their leaves began to overlap, the **leaf counting algorithm** struggled to maintain accuracy, resulting in high variation and even occasional undercounts as leaves became difficult to differentiate. This high variance in leaf count data necessitated additional post-processing, including smoothing techniques and logical rules to ensure non-decreasing leaf counts over time.

In contrast, the **surface area calculation** was far less susceptible to these issues. By focusing on the total number of green pixels detected in the image, the algorithm could more accurately reflect the plant's overall growth, even in cases of leaf overlap or partial occlusion. The pixel-based surface area measurement provided a more holistic view of the plant's size and growth trajectory, avoiding the fluctuations and inconsistencies that plagued the leaf count method.

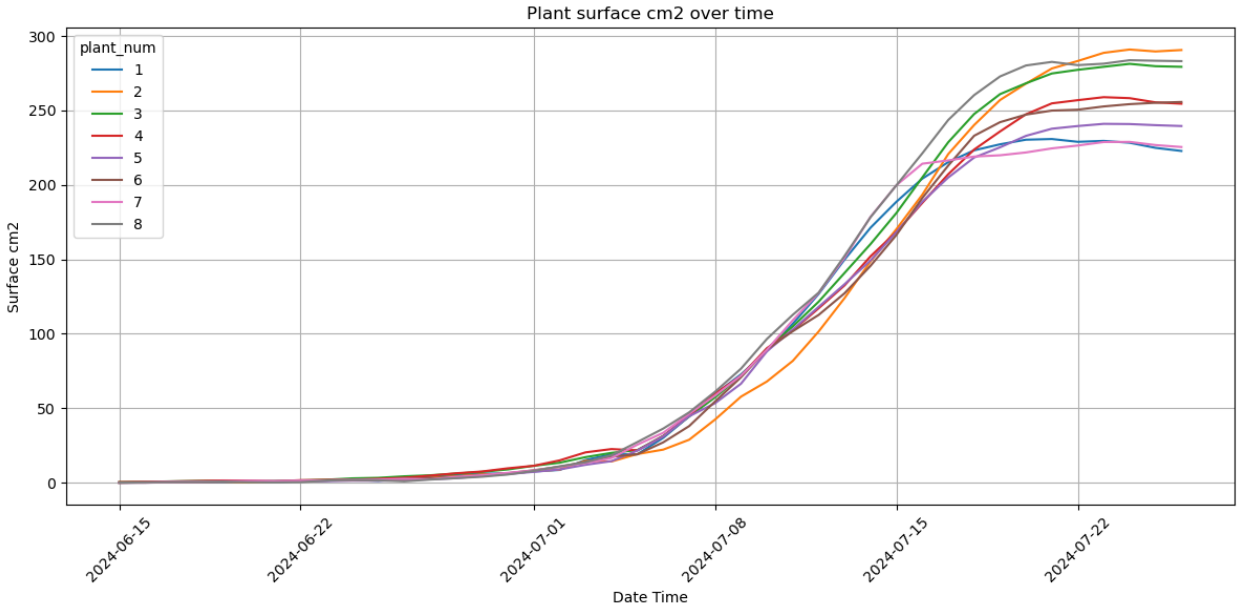

Figure 11 - Time series of the plant surface in cm2

As a result, the surface area data proved to be a far more suitable input for **time-series analysis**. The surface area values exhibited a natural, smooth progression over time, with minimal variation and no need for the additional corrections or adjustments that were necessary for leaf count data. This smooth progression was particularly beneficial for modeling and predicting plant growth, as it allowed for more reliable trend detection and analysis without requiring post-processing to handle inconsistencies.

Moreover, because the surface area measurements remained stable throughout the experiment, there was no need to apply logical rules or validate the accuracy of the time-series as was done with leaf count data. The surface area metric consistently aligned with the expected growth patterns of the plants, making it a more **robust and straightforward** metric for further analysis and modeling.

In summary, the decision to focus on surface area calculation for time-series analysis was driven by its superior **reliability** and **robustness** compared to leaf counting. The surface area provided a consistent, biologically accurate measure of plant growth that required minimal post-processing, making it the preferred metric for analyzing and predicting plant development over time.

Chapter 6. Time-series model for Plant Growth

6.1 Introduction to Time-Series modeling

Time-series modeling is a statistical method that focuses on analyzing temporal data to identify patterns, make forecasts, and extract meaningful insights. A time series is a collection of observations indexed in chronological order. In the context of plant growth, time-series analysis can help model how plants grow over time and predict future growth stages based on historical data.

6.1.1 Objectives and definitions

The key objectives of time-series analysis include:

- **Detecting trends** and underlying patterns in the data.
- **Forecasting future values** based on historical data.
- **Understanding seasonal or cyclical behaviors** that might impact the time series.
- **Decomposing the time series** into its constituent components (trend, seasonality, and residuals).

Key Definitions

Trend: A long-term movement or direction in the data, either increasing or decreasing. In a plant growth context, the trend refers to the overall growth pattern, such as an initial slow growth followed by rapid growth and a plateau.

$$\text{Trend component } T_t = \alpha + \beta t$$

Seasonality: Recurring patterns or fluctuations in the data at regular intervals. In certain agricultural settings, seasonality could reflect growth patterns influenced by environmental factors like temperature, humidity, and light.

$$\text{Seasonal component } S_t = f(t \ mod \ s)$$

where *s* is the period of the seasonal cycle.

Cycle: Long-term oscillations in the data that are not necessarily regular, often occurring over several periods. In plant growth, cycles might arise from natural growth spurts and lulls.

Decomposition: A method to separate a time series into its components—trend, seasonality, and residual (noise). Decomposition can be additive or multiplicative, depending on how the components combine.

Additive decomposition: $Y_t = T_t + S_t + R_t$

Multiplicative decomposition: $Y_t = T_t \times S_t \times R_t$

where Y_t is the observed value, T_t is the trend, S_t is the seasonality, and R_t is the residual.

Autocorrelation: The correlation of a time series with a lagged version of itself. Autocorrelation helps in identifying patterns and the persistence of past values in future observations. The autocorrelation function (ACF) measures the correlation between observations at different time lags.

$$\rho_k = \frac{\sum_{t=1}^{n-k}(Y_t - \bar{Y})(Y_{t+k} - \bar{Y})}{\sum_{t=1}^{n}(Y_t - \bar{Y})^2}$$

Where k is the lag, $\bar{Y}$ is the mean of the series, and n is the total number of observations.

Partial Autocorrelation (PACF): Measures the correlation between a time series and its lag, while controlling for the values of the intermediate lags. The PACF is useful for identifying the order of an AR (Auto-Regressive) model. The PACF is defined as the coefficient of lag $\boldsymbol{k}$ in an AR(k) process:

$$Y_t = \alpha_0 + \sum_{i=1}^{k} \alpha_i Y_{t-i} + \epsilon_t$$

where α_i are the coefficients, and ϵ_t is white noise.

Stationarity: A time series is stationary if its statistical properties—mean, variance, and autocorrelation—do not change over time. Stationarity is a critical assumption for many time-series models, particularly ARIMA. A non-stationary series exhibits trends, seasonality, or varying variance over time.

A time series Y_t is stationary if:

$$E[Y_t] = \mu \quad \text{(constant mean)}$$
$$\text{Var}(Y_t) = \sigma^2 \quad \text{(constant variance)}$$
$$\text{Cov}(Y_t, Y_{t+k}) \quad \text{(depends only on lag } k\text{, not time } t\text{)}$$

Non-Stationarity: When a time series exhibits trends, seasonality, or changing variance, it is considered non-stationary. Techniques such as differencing, detrending, or seasonal adjustment are commonly used to make the series stationary before applying models like ARIMA.

Differencing: A technique used to transform a non-stationary series into a stationary one by subtracting the previous observation from the current one. First-order differencing is defined as:

$$\Delta Y_t = Y_t - Y_{t-1}$$

Autoregressive Integrated Moving Average (ARIMA): A widely used time-series model that combines autoregression (AR), differencing (I for Integrated), and moving averages (MA). ARIMA models are effective for capturing trends and patterns in time series data, especially when the data is non-stationary. The formal structure will be covered in the following section.

6.1.2 Implications for plant growth

For the plant growth dataset, we need to account for the **non-stationary** nature of the time series. The plant growth pattern follows an **S-curve**, characterized by three distinct stages:

1. **Slow Start**: During the initial stage, growth is slow as the plants establish their roots and begin to grow leaves.
2. **Growth Boost**: After the initial phase, the plants enter a period of rapid growth, where both leaf count and surface area increase significantly.
3. **Plateau**: Finally, the growth slows down as the plants approach maturity, forming a plateau.

Given this structure, traditional time-series models like ARIMA may require **differencing** to handle the non-stationarity. Additionally, any model must be able to capture the **growth boost** in the middle phase and recognize that the final plateau

signifies a natural ceiling for growth. This necessitates careful modeling of the trend and cycles within the data, as well as addressing the non-stationary properties of the plant growth time series.

The objectives of time-series modeling for plant growth prediction include:

- **Capturing and predicting growth trends** as the plants progress through their different stages.
- **Modeling the non-linear growth curve** (S-curve) and accounting for the growth boost and plateau stages.
- **Forecasting future plant growth** based on past data while considering external factors such as environmental conditions.
- **Handling non-stationary data** by using appropriate transformations (e.g., differencing) to improve model accuracy.

6.2 ARIMA model for plant growth

6.2.1 Model overview

The ARIMA (AutoRegressive Integrated Moving Average) model is one of the most widely used time-series forecasting methods, especially for non-stationary data. It combines three components: Autoregression (AR), Differencing (I for Integrated), and Moving Average (MA). Each component helps in capturing different aspects of the time series, making ARIMA a powerful tool for predicting plant growth, which often follows non-linear trends and patterns.

Moving Average (MA)

The Moving Average (MA) model captures the relationship between an observation and residual errors from previous time steps. It is defined as a linear combination of past forecast errors. In an MA model of order q (denoted as MA(q)), the forecast is a function of past errors ϵ, and the current observation Y_t.

The MA(q) model is given by:

$$Y_t = \mu + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q}$$

Where μ is the mean of the series, ϵ is the error at time t, $\theta_1, \theta_2, \ldots, \theta_q$ are the moving average parameters.

Autoregressive (AR)

The Autoregressive (AR) model uses the relationship between an observation and a number of lagged observations. An AR model of order p (denoted as AR(p)) predicts the current value Y_t based on a linear combination of past values of the series.

The AR(p) model is given by:

$$Y_t = \mu + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t$$

Where μ is the mean of the series, $\phi_1, \phi_2, \ldots, \phi_p$ are the autoregressive parameters, ϵ_t is the error at time t.

Integrated (I)

The Integrated (I) component represents the differencing of the time series to make it stationary. A time series is said to be **stationary** if its statistical properties, such as mean and variance, do not change over time. If the series is non-stationary, differencing is applied to remove trends and stabilize the variance.

The differenced time series ΔY_t is given by:

$$\Delta Y_t = Y_t - Y_{t-1}$$

For non-stationary series, multiple levels of differencing may be required. For instance, second-order differencing is given by:

$$\Delta^2 Y_t = \Delta Y_t - \Delta Y_{t-1}$$

Combining AR, I, and MA: ARIMA Model

The ARIMA(p, d, q) model combines the autoregressive (AR), integrated (I), and moving average (MA) components into a single model. Here:

- p is the order of the AR component,
- d is the number of times the series needs to be differenced to achieve stationarity,
- q is the order of the MA component.

The general ARIMA model can be expressed as:

$$\Delta^d Y_t = \mu + \phi_1 \Delta^d Y_{t-1} + \cdots + \phi_p \Delta^d Y_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t$$

6.2.2 Parameter tuning and optimization

Tuning the parameters of an ARIMA model—specifically, p, d, and q—is crucial for ensuring that the model accurately captures the underlying patterns in the time series. For the plant growth dataset, I followed a systematic process for selecting the parameters, beginning with the differencing component d, followed by the selection of the autoregressive p and moving average q terms through autocorrelation and partial autocorrelation analysis.

Step 1: Selection of *d* (Differencing Order)

The first step in tuning the ARIMA model is determining the appropriate level of differencing d, which ensures the time series becomes stationary. Since plant growth data exhibits non-stationary behavior, applying differencing is essential for stabilizing the mean of the series.

To find the optimal d, I tried several differencing orders, plotted the resulting differenced time series for visual inspection, and then used the **Augmented Dickey-Fuller (ADF) test** to assess stationarity at each level of differencing.

The ADF test examines the null hypothesis that the series is non-stationary. The test statistic is compared against critical values at different significance levels, with a smaller test statistic indicating rejection of the null hypothesis (i.e., the series is stationary).

For each plant in the dataset, I performed the following steps:

1. **Visual inspection**: I plotted the time series for different levels of differencing (e.g., d=1, d=*2*, d=*3*) to observe the stability of the trend.
2. **ADF test**: I applied the ADF test at each level of differencing to check if the series had become stationary.

The differencing order that produced the most stationary time series for most plants was d=3. After applying third-order differencing, the ADF test results consistently indicated that the time series had become stationary across the different plants, as the p-values were lower than the significance threshold.

	plant	ADF statistic	p-value	is_stationary
0	1	-8.548894	9.339198e-14	True
1	2	-5.390267	3.550346e-06	True
2	3	-7.932970	3.473254e-12	True
3	4	-3.889988	2.109776e-03	True
4	5	-5.437461	2.824632e-06	True
5	6	-2.044175	2.675457e-01	False
6	7	-9.433590	5.112345e-16	True
7	8	-10.202807	5.909451e-18	True

Table 1 - Augmented Dickey-Fuller (ADF) test for different plants

Step 2: Selection of *p* (Autoregressive Order)

Once the differencing order was determined, the next step was to select the autoregressive term *p*, which defines how many lagged observations are included in the model. To select *p*, I used the **Partial Autocorrelation Function (PACF)** plot.

The PACF plot shows the correlation between an observation and a lagged observation while controlling for the effects of other lags. In general, for an AR model, the PACF plot is used to identify the appropriate number of lags that contribute significantly to predicting the current observation.

Here's how I proceeded:

1. **Plotting the PACF**: After applying third-order differencing, I plotted the PACF for each plant's time series.
2. **Choosing *p***: The PACF plot indicated that the partial autocorrelation dropped off sharply after lag 4, suggesting that lags 1 through 4 should be included in the model. Based on this visual analysis, I selected $p=4$.

Step 3: Selection of *q* (Moving Average Order)

Finally, I needed to select the moving average term *q*, which represents the number of lagged forecast errors included in the model. To determine *q*, I used the **Autocorrelation Function (ACF)** plot.

The ACF plot shows the correlation between an observation and its lagged values. For an MA model, the ACF plot helps to identify how many lagged error terms should be incorporated into the model.

1. **Plotting the ACF**: After third-order differencing, I plotted the ACF for each plant's time series.
2. **Choosing *q***: The ACF plot showed that the autocorrelations significantly declined after lag 2, indicating that lags 1 and 2 were sufficient to capture the structure of the errors. Based on this, I chose q=2.

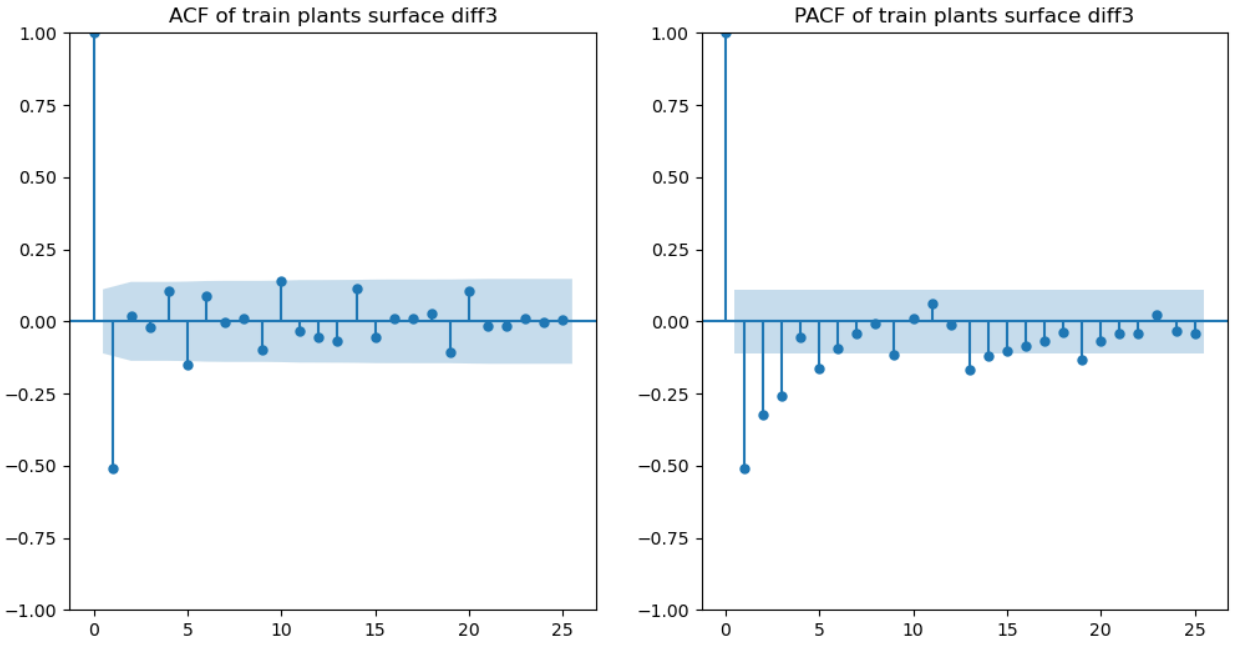

Figure 12 - ACF and PACF for plant time series

After selecting p=2, d=3, and q=2, the ARIMA model for the plant growth prediction was formalized as:

$$\Delta^3 Y_t = \mu + \phi_1 \Delta^3 Y_{t-1} + \phi_2 \Delta^3 Y_{t-2} + \cdots + \phi_4 \Delta^3 Y_{t-4} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \epsilon_t$$

6.2.3 Model accuracy and limitations

In this section, I evaluated the accuracy of the ARIMA model by training it on a subset of plants and testing it on two plants that were not included in the training data. The primary goal was to assess how well the ARIMA model could predict plant surface area for the test plants, as well as to understand its limitations in capturing growth patterns over time.

Model Training and Joining Time Series with Gaps

The first step in this process was to prepare the training data. I combined the time series from six plants (*train_plants = ["1", "2", "3", "4", "5", "6"]*) to form a unified training set. However, instead of directly concatenating the time series, I introduced a gap of 10 days between each plant's time series to avoid abrupt transitions between different plants' data. This was done by adding 10 NaN values after the last date of each plant's time series before moving on to the next plant's time series.

The idea behind this gap was to ensure that the ARIMA model could adjust for the fact that it was being trained on multiple plants at once, treating the plants' data as sequential but recognizing that there is a natural gap between them. This approach allowed the ARIMA model to generate global coefficients that influenced the entire training set rather than relying on each plant individually.

The ARIMA model was then fitted to the combined time series, which included the gap days. This step was crucial because it forced the model to generalize across all training plants, enabling it to learn common patterns of growth and apply them during the prediction phase.

Testing on New Plants with Sequential Forecasting

For the testing phase, I used the data from two plants (*test_plants = ["7", "8"]*). The goal was to sequentially re-train the model by adding the test data day by day, increasing the forecast horizon as I moved forward. This approach allowed the ARIMA model to adjust as new test data points were made available, extending the prediction window step by step.

- Horizon of Forecasting (X): I set a maximum forecast horizon X=5 days, meaning the model would make predictions for the next 5 days based on the current data.
- For each iteration, I trained the model using both the original training data (with gaps) and the **test data** available up to that point.
- After fitting the model, I predicted the surface area of the plant for the next X days and compared the predictions with the actual test data for the same period.

By iterating over the test days, the ARIMA model continuously updated its parameters to account for newly available test data. This process allowed me to evaluate the model's performance as the forecast horizon increased.

To evaluate the accuracy of the predictions, I used two primary metrics:

1. **Mean Absolute Error** $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \widehat{Y_i}\right|$.

   MAE measures the average magnitude of errors between the predicted and actual values. It is particularly useful for assessing the model's ability to predict plant surface area, providing an absolute measure of accuracy.

2. **Mean Absolute Percentage Error** $\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \widehat{Y_i}}{Y_i}\right|$

   MAPE expresses the error as a percentage, which makes it easier to interpret relative to the size of the plant surface area. This metric is helpful in understanding the percentage difference between the predicted and actual surface areas.

For each forecast horizon *X*, I computed the MAE and MAPE by comparing the predicted surface area values with the actual values for the test plants. By increasing the prediction horizon from 1 day to 5 days, I could assess how the model's accuracy changed as the forecast window extended.

Horizon X	**MAE**	**MAPE**
1 day	2.074	10.868%
2 days	4.903	20.821%
3 days	9.0414	30.045%
4 days	13.875	39.208%
5 days	19.537	45.374%

Table 2 - ARIMA MAE and MAPE for different prediction horizons

Model Limitations

While the ARIMA model performed well for short-term predictions (1-2 days ahead), several limitations emerged as the forecast horizon increased:

- **Decreasing accuracy**: As the prediction horizon extended beyond 2 days, both the MAE and MAPE increased, indicating that the ARIMA model struggled to maintain accuracy over longer periods. This decline in accuracy is a common limitation of ARIMA models, as they rely on past data to make future predictions, and their predictive power diminishes as the forecast window grows.
- **Generalization issues**: Despite the effort to combine multiple plant time series into a single model, the ARIMA model occasionally failed to generalize well to the test plants. This could be due to the variability in growth patterns between different plants, which ARIMA, as a linear model, might not fully capture.

In summary, while the ARIMA model was effective for short-term plant growth predictions, its accuracy decreased over longer horizons, and it exhibited some limitations in generalizing to new plants with potentially different growth behaviors. Further model improvements could involve exploring non-linear models or incorporating additional features (such as environmental data) to enhance predictive performance.

6.3 Bayesian approach to growth prediction

6.3.1 PyMC Bayesian Modeling Framework

The Bayesian approach is a probabilistic method that allows us to model uncertainties in parameters by using Bayesian inference. It involves updating prior beliefs about parameters with observed data to form posterior distributions, which are used for predictions. PyMC is a framework that facilitates this process by enabling the construction and fitting of Bayesian models through a simple Python API.

Core Concepts in Bayesian Modeling

Bayes' Theorem: The foundation of Bayesian inference is Bayes' theorem, which describes how to update the probability of a hypothesis based on new evidence:

$$P(\theta|Y) = \frac{P(Y|\theta)P(\theta)}{P(Y)}$$

where:

- $P(\theta|Y)$ is the **posterior** probability of the parameter θ given the observed data Y
- $P(Y|\theta)$ is the **likelihood** of observing the data Y given the parameter θ
- $P(\theta)$ is the **prior** probability of θ, and
- $P(Y)$ is the **evidence** or marginal likelihood, which normalizes the posterior distribution.

Prior: The prior distribution represents our belief about the values of a parameter θ before seeing any data. It can be based on expert knowledge or general assumptions. For example, in plant growth, a prior might reflect expectations about how fast plants grow based on environmental factors.

Likelihood: The likelihood function represents the probability of observing the data Y, given a particular set of parameter values θ. In PyMC, likelihood functions are defined using probability distributions like Normal, Poisson, or Gamma distributions.

Posterior: The posterior distribution combines the prior and the likelihood to form a new probability distribution for the parameters after observing the data. This is the updated belief about θ after seeing the data.

Markov Chain Monte Carlo (MCMC): Since the posterior distributions in complex models are often difficult to compute analytically, PyMC uses MCMC sampling techniques like the No-U-Turn Sampler (NUTS) to generate samples from the posterior distribution. These samples are used to approximate the posterior and compute summary statistics for the parameters.

The process of fitting a Bayesian model in PyMC involves several steps:

1. **Defining the Model:** The first step is to define the probabilistic model, including the priors for the parameters and the likelihood function for the observed data. This is done within a with *pm.Model()* as model: block in PyMC.

2. **Sampling from the Posterior**: Once the model is defined, PyMC fits the model by drawing samples from the posterior distribution using an MCMC sampler (such as NUTS). The sampler iteratively explores the parameter space, drawing samples based on the posterior distribution. This process can be expressed generically as:

$$\theta^{(i+1)} = \theta^{(i)} + \epsilon \nabla \log P\left(\theta|Y\right)$$

   Where ϵ is the step size and $\nabla \log P\left(\theta|Y\right)$ is the gradient of the log-posterior.

3. **Convergence**: PyMC uses diagnostics to check whether the sampler has converged to the posterior distribution. One common diagnostic is the R-hat statistic, which compares within-chain and between-chain variability to assess convergence.

4. **Posterior Predictive Checks**: After fitting the model, PyMC allows for posterior predictive checks, which involve generating new data based on the fitted model and comparing it to the observed data. This helps assess the quality of the model's fit.

The Bayesian approach offers several advantages for plant growth prediction due to its ability to model uncertainty and flexibility in handling various amounts of input data. Here are some key benefits:

1. **Confidence Intervals on Future Predictions**: One of the most significant advantages of the Bayesian framework is its ability to provide confidence intervals (credible intervals**)** for future predictions. In plant growth prediction, this is particularly useful as it allows us to quantify uncertainty in the forecasts. For example, instead of providing a single predicted surface area or leaf count, the model offers a range within which the actual value is likely to fall, along with a level of confidence (e.g., 95% credible interval). This is essential in plant growth forecasting, where environmental factors like temperature and humidity introduce natural variability.

   As more data points are added, the confidence intervals typically become narrower, reflecting increased precision in the model's predictions. This adaptability allows us to make robust predictions even when there is

considerable uncertainty early in the growing process, improving as more growth data is incorporated.

2. **Flexibility with Input Structure**: Unlike traditional models, which often require a fixed input structure or number of observations, Bayesian models in PyMC are highly flexible. You can insert any number of data points—whether it's 1, 10, or 100—and the model will update its posterior distributions accordingly. This means that predictions can be made at any stage of the growing process, whether at the beginning, middle, or end, based on the available data.

   For plant growth prediction, this is highly advantageous as we can input data at various points in the growth cycle without having to conform to a rigid input structure. The model will seamlessly update its predictions, incorporating whatever data is available at the time. As more observations are included, the model becomes more accurate, and the uncertainty decreases, leading to tighter confidence intervals.

3. **Incremental Learning**: The Bayesian approach inherently supports incremental learning, where the model is updated with each new data point. This is particularly useful for long-term experiments such as plant growth, where data is collected over time. As more data is gathered, the model refines its predictions, continuously adjusting the posterior distributions of the parameters.

   For example, as plants progress through their different growth stages, the model can be updated with new observations, whether daily or weekly, to improve the accuracy of the growth forecasts. This incremental process ensures that the model remains relevant and up-to-date with the latest available information.

4. **Robustness to Missing Data**: PyMC's Bayesian framework also offers robustness in situations where data is sparse or missing. In the context of plant growth prediction, there may be periods where data collection is interrupted or incomplete due to technical or environmental factors. Bayesian models are well-suited to handle these gaps because they rely on probability distributions to model uncertainties. Even if data points are missing, the model can still

make reasonable predictions by drawing from prior knowledge and the structure of the data that is available.

6.3.2 Function selection and parametrization

In the context of plant growth modeling, the progression of growth often follows an **S-curve**, which is similar to the pattern described by the **sigmoid function**. The sigmoid function is particularly well-suited for modeling processes that have a slow start, followed by rapid growth, and eventually plateau as the process stabilizes. This pattern aligns with the three main stages of plant growth:

1. **Initial Lag Phase**: During the early stages, the plant grows slowly as it establishes itself (root development, leaf formation, etc.).
2. **Exponential Growth Phase**: Once established, the plant enters a rapid growth phase, where leaf surface area and biomass increase rapidly.
3. **Plateau Phase**: As the plant matures, growth slows down and stabilizes, reaching a plateau where further growth is minimal.

The sigmoid function mathematically captures this pattern and is commonly expressed as:

$$S(t) = \frac{A}{1 + e^{-k(t-t_0)}}$$

where:

- A is the maximum value that the function will asymptotically approach (e.g., maximum plant surface area).
- k is the growth rate or steepness of the curve during the rapid growth phase.
- t_0 is the inflection point, representing the time at which the growth rate is the highest (mid-point of the growth).

Given that the sigmoid function describes plant growth so well, it was chosen to model the growth curve of plants over time. To apply this function in a Bayesian framework with PyMC, the parameters A, k, and t_0 were treated as random variables with prior distributions.

The parameter A controls the maximum plant surface area that can be reached. In practice, this is determined by the species of plant and environmental factors. I used a ***normal distribution*** to represent A, with a mean of 250 cm² and a standard deviation of 50 cm². These values were chosen based on prior observations from the training data.

The parameter k determines the steepness or rate of growth during the exponential phase. A higher k results in faster growth. The choice of k can be informed by previous experiments or general knowledge about the plant's growth rate. A normal distribution with a mean of 1.0 and a standard deviation of 0.5 was selected to account for variability in the growth rate.

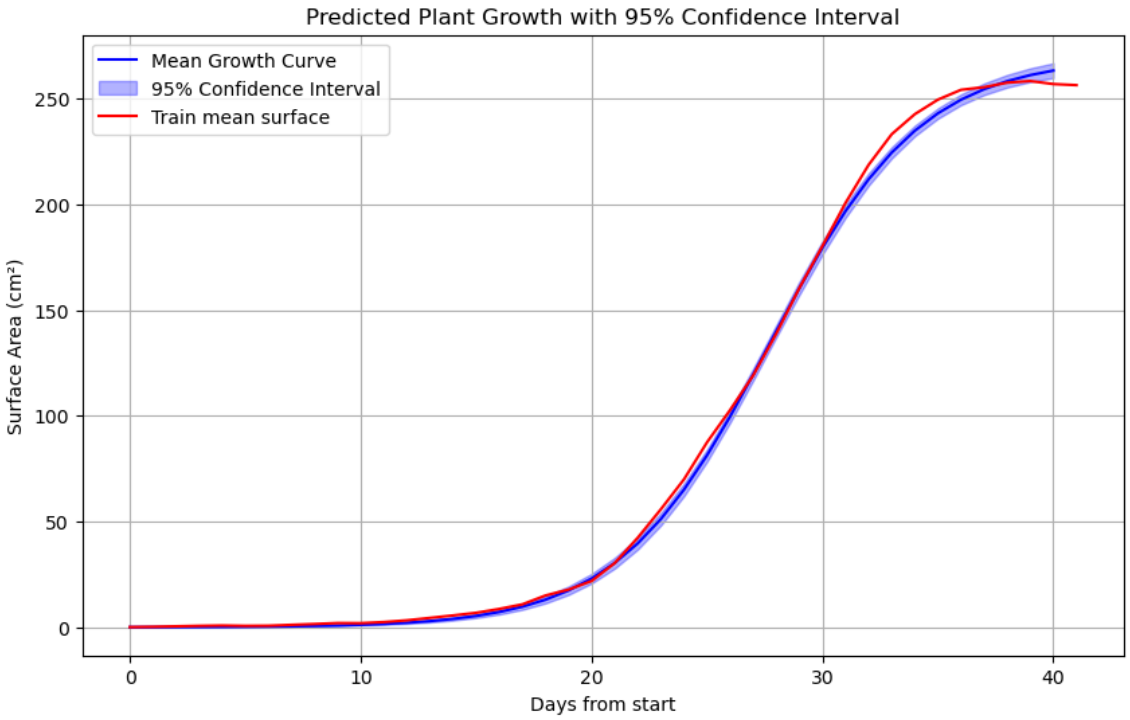

Figure 13 - Sigmoid with posterior training parameters and the real training mean

The inflection point t_0 is the time at which the growth rate is at its maximum. This corresponds to the point in time when the plant transitions from slow growth to rapid growth. The prior for t_0 was modeled as a normal distribution with a mean of 20 days and a standard deviation of 5 days. This reflects the general expectation that the most rapid growth occurs around 20 days after planting.

An important aspect of Bayesian modeling is that the priors can be informed by previous experiments or data, which in this case comes from the training data. In Bayesian terminology, the posterior distributions obtained from the training data can be used as priors for future experiments or predictions. This is particularly useful in plant growth modeling, as it allows the model to refine its predictions as more data becomes available. In this case, after training the model on the initial training data, I used the posterior distributions of A, k, and t_0 from the training data as the priors for the test data predictions. This allows the model to incorporate the knowledge gained from earlier plants and apply it to the new plants, improving the model's predictions for unseen data.

6.3.3 Performance evaluation

The performance evaluation for the Bayesian approach, using PyMC, followed a similar train/test split strategy as in the ARIMA model. The data was split between six plants for training and two plants for testing, ensuring that the test data remained unseen during the training process. However, the flexibility of the Bayesian approach allowed for testing under different conditions to evaluate how the model adjusted its predictions.

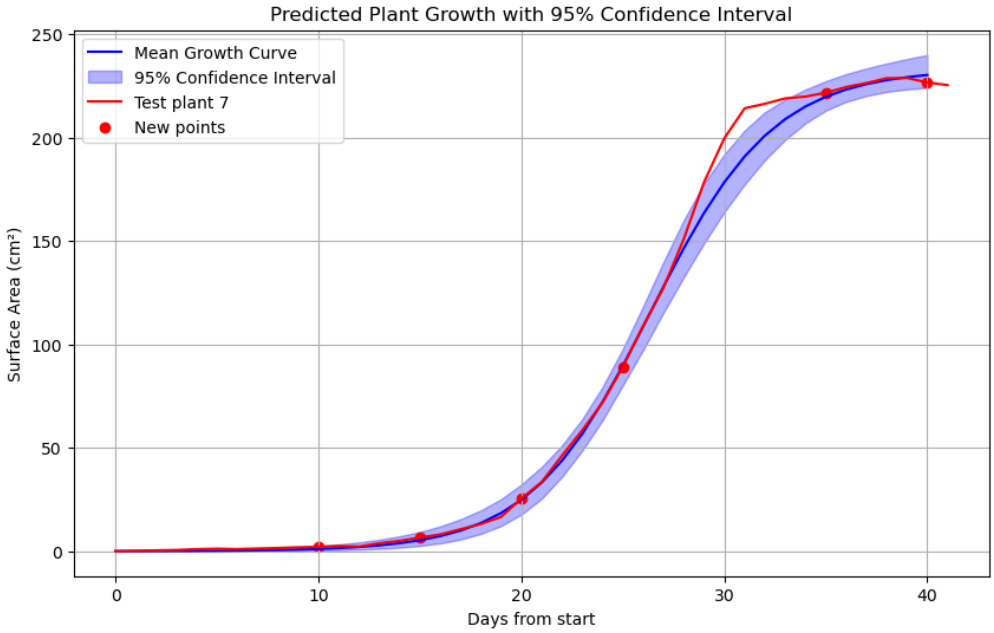

Figure 14 - Bayesian approach test predictions 6 points with confidence intervals

In the first use case, I used only several randomly distributed points across 42 days to adjust the trained priors. These points provided sparse information over the test period, simulating a situation where only limited observational data is available. Despite the limited data, the Bayesian model demonstrated its ability to adjust, leveraging the priors learned from the training phase.

What stood out was the **"squeezing" of the confidence intervals** around the provided data points. The model's posterior predictions were tightly aligned with the known observations, but as soon as it moved away from these points, the intervals widened due to the growing uncertainty. This behavior reflects the Bayesian framework's ability to dynamically adjust based on the available data, providing realistic uncertainty estimates in regions with sparse information.

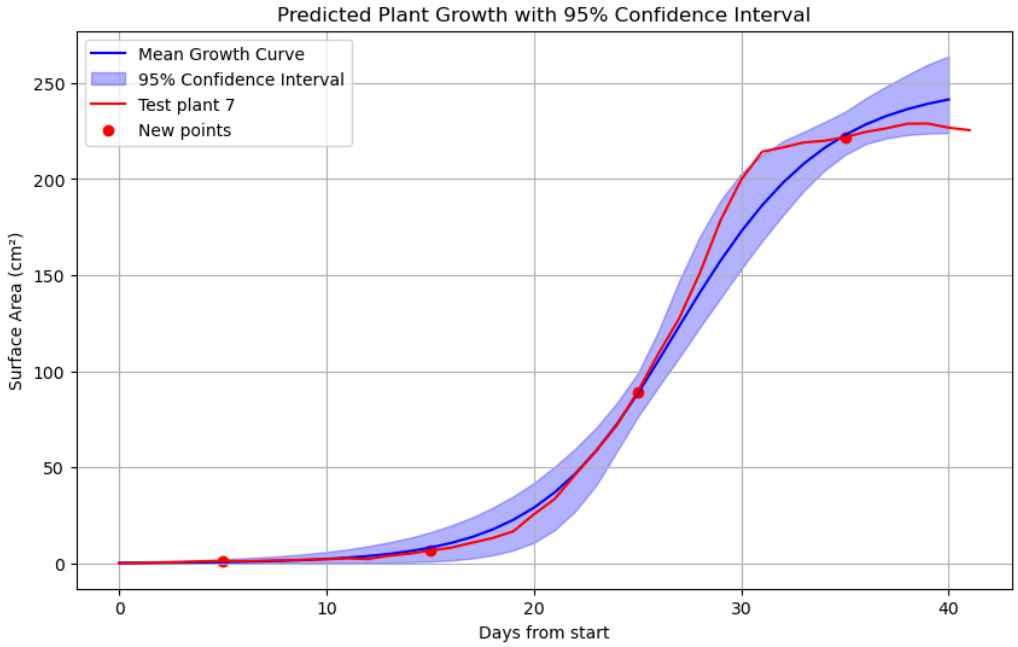

Figure 15 - Bayesian approach test predictions 4 points with confidence intervals

In the second use case, I used the data from the 15-25 days to predict plant growth for the remaining period. This tested the model's ability to handle long-term forecasting with a limited training window. In this scenario, as expected, the uncertainty grew with the prediction horizon, as the posterior distributions widened due to the lack of new data after day 25. However, despite this uncertainty growth, the Bayesian model remained closer to the truth compared to the ARIMA model, particularly toward the end of the prediction period.

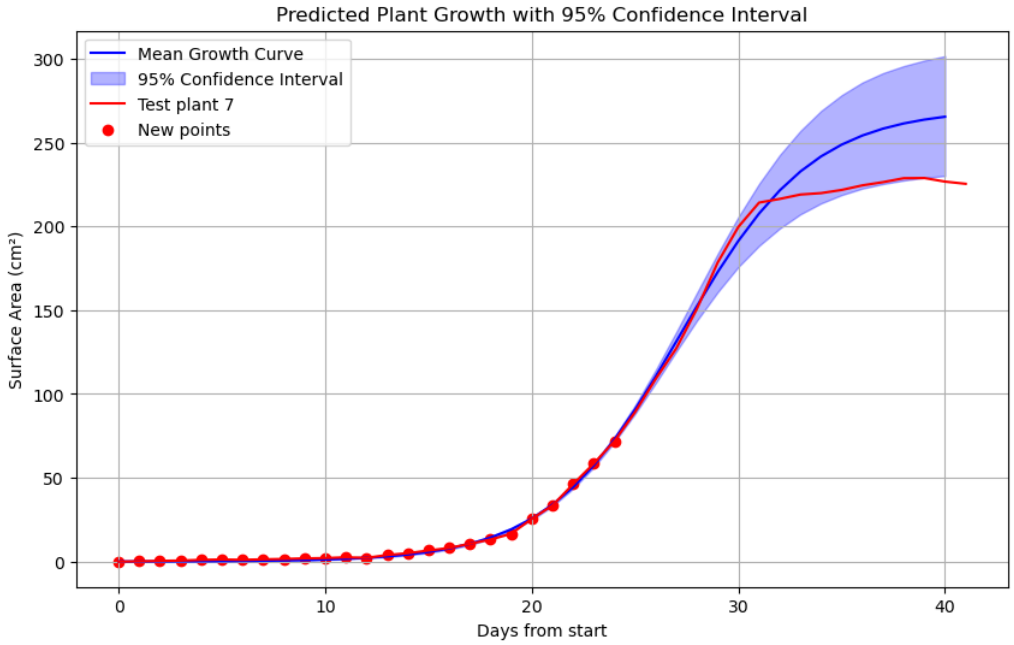

Figure 16 - Bayesian approach test prediction 25 first points

The **adaptive nature** of the Bayesian framework allowed it to capture the overall growth trend more effectively, and while uncertainty increased with longer forecast windows, the predictions did not stray as far from the actual values as the ARIMA model did. This reflects the advantage of incorporating priors and uncertainty modeling, as it helps maintain realistic predictions even with limited data.

For the sake of consistency, I did a similar to ARIMA table with the summary of predictions MAE and MAPE for horizons from 1 to 5 days.

Horizon X	**MAE**	**MAPE**
1 day	1.893	10.868%
2 days	2.093	12.821%
3 days	3.044	14.045%
4 days	5.029	15.208%
5 days	6.949	17.374%

Table 3 - ARIMA MAE and MAPE for different prediction horizons

Chapter 7. Results and discussions

7.1 Accuracy of Leaf Count and Surface Area Calculation

Quantitative Results

Quantitative Results The accuracy of the leaf detection and surface area algorithms was evaluated using both quantitative metrics and validation against manual counts. The two approaches—leaf detection and surface area estimation—were tested across the entire dataset, with performance measured by comparing the algorithm-generated values with ground truth values obtained from manual observations.

For the leaf detection algorithm, the final results showed that it achieved a mean absolute error (MAE) of approximately 5-10%, depending on the growth stage of the plants. Early in the growth process, when leaves are sparse and clearly separated, the algorithm performed very well, with high precision in detecting individual leaves. However, during the later stages of plant development, when leaf overlap and occlusion occurred, the accuracy decreased, resulting in a higher error rate.

The surface area algorithm, on the other hand, proved to be more robust and accurate. Since it relied on detecting green pixels (corresponding to the plant's surface) and did not need to identify individual leaves, it was less affected by occlusion. The mean absolute error (MAE) for surface area estimation was consistently lower, typically around 3-5%. Moreover, when compared with the leaf detection method, the surface area algorithm maintained better consistency over time, especially during the rapid growth phase, where the leaf count method struggled to keep pace due to overlapping leaves.

Overall, both methods showed solid quantitative performance, but the surface area approach emerged as the more reliable option for tracking plant growth, particularly during later stages.

Qualitative Analysis

In addition to quantitative metrics, a qualitative assessment was performed to evaluate the practical usability of the leaf detection and surface area algorithms. Observations were made across the different growth stages, with particular attention

paid to the algorithm's ability to handle leaf occlusion, varying light conditions, and noise in the images.

- **Leaf Detection**: During the early stages of plant growth, when leaves were small and well-separated, the algorithm produced clear, accurate leaf boundaries, easily segmenting individual leaves. The challenges began to appear in the middle to late growth stage**s**, where leaves began overlapping and shadowing one another. In these cases, the algorithm had difficulty distinguishing between leaves, which led to under-counting the total number of leaves.
- **Surface Area Calculation**: The surface area algorithm performed more consistently across all growth stages. Since it did not depend on isolating individual leaves, it managed occlusion and overlapping foliage much more effectively. Even during the densest growth periods, the algorithm provided a reliable estimate of total leaf coverage. Additionally, its dependence on the green color spectrum for segmentation allowed it to operate robustly under different lighting conditions, provided the images were not overexposed or too dim.

A key advantage of the surface area algorithm was its ability to smooth out irregularities in the data caused by environmental conditions, such as shadows or slight changes in camera angle. The leaf detection method, while more prone to fluctuations in these situations, still performed reasonably well in controlled environments where occlusion and lighting were minimized.

7.2 Plant growth prediction results

7.2.1 Comparison of Time-Series Models

Two main time-series models were employed to predict plant growth: the ARIMA model and the Bayesian growth model using PyMC. While both models provided valuable insights, there were clear distinctions in their performance and capabilities.

ARIMA Model: The ARIMA model, being a traditional time-series forecasting method, performed well in predicting short-term growth trends. It captured the

overall growth pattern effectively in the early and middle stages, particularly when the growth was steady. However, ARIMA struggled with longer-term predictions, especially in the latter stages of plant growth, where the non-linearity and plateau of the growth curve became prominent. The primary limitation of the ARIMA model was its reliance on stationarity and linearity, which caused it to falter when growth slowed down as the plants approached maturity. The model also exhibited increasing prediction errors as the forecast horizon extended.

Bayesian Growth Model (PyMC): The Bayesian growth model using the sigmoid function proved to be more flexible and well-suited for the plant growth data, particularly due to its ability to model the non-linear S-curve characteristic of plant growth. By incorporating priors based on the training data, the model was better able to adjust its predictions based on limited test data. One of the key advantages of the Bayesian model was its ability to produce credible intervals, allowing for uncertainty estimation in the predictions. These intervals provided a range of potential outcomes, reflecting the inherent variability in biological processes. The model's performance improved significantly when more test data points were available, further narrowing the uncertainty range.

In summary, while the ARIMA model was easier to implement and worked well for short-term predictions, the Bayesian model offered more flexibility and adaptability, especially for long-term forecasting and scenarios with non-linear growth patterns. The Bayesian approach’s ability to account for uncertainty and incorporate prior knowledge made it more robust in handling the full growth cycle of the plants.

7.2.2 Prediction Accuracy Over Time

The accuracy of the predictions was evaluated over time, with a particular focus on the mean absolute error (MAE) and mean absolute percentage error (MAPE) across different forecast horizons. The results indicated a clear divergence in the performance of the two models as the prediction horizon extended:

Short-Term Predictions (1-2 days ahead): In the short term, both the ARIMA and Bayesian models performed comparably, with low MAE and MAPE values. The ARIMA model was able to capture short-term fluctuations in growth quite

effectively, owing to its autoregressive nature. Similarly, the Bayesian model, while initially relying on broad priors, quickly honed in on accurate predictions due to the posterior updates.

Medium-Term Predictions (3-5 days ahead): For medium-term predictions, the ARIMA model's accuracy began to degrade, particularly as the model struggled to adapt to the increasing non-linearity in the growth data. The Bayesian model, in contrast, maintained a more consistent level of accuracy, with wider credible intervals that still encompassed the true growth values. The Bayesian model's advantage became more evident as the model's flexibility allowed it to accommodate the slowing growth rate during the middle stages of plant development.

Long-Term Predictions (6+ days ahead): As the prediction horizon extended beyond 5 days, the ARIMA model's errors increased significantly, often diverging from the actual growth trajectory as the plants entered the plateau phase. The Bayesian model, although exhibiting increasing uncertainty (reflected in wider credible intervals), still provided better long-term predictions due to its ability to model the full S-curve and incorporate uncertainty in the growth process. The sigmoid function in the Bayesian framework allowed the model to anticipate the approaching plateau, whereas the ARIMA model continued to predict linear growth, resulting in overestimation.

In summary, the ARIMA model excelled in the short-term but suffered from overfitting and poor adaptability to non-linear growth patterns over the long term. On the other hand, the Bayesian model offered more reliable and robust predictions, particularly as the forecast horizon extended, by incorporating prior knowledge and accounting for uncertainty in the predictions.

7.3 Future research directions

As with any scientific research, there are always ways to extend, refine, and deepen the scope of inquiry. While this project has produced promising results in predicting plant growth and quantifying leaf and surface area using computational methods, there are several avenues for future research that could enhance both the

accuracy and the applicability of these methods across a wider range of agricultural scenarios.

7.3.1 Expanding to Different Plant Species

One of the most immediate and compelling future directions is to expand the models and methods developed in this thesis to other plant species. Thus far, our experiments have focused on a specific plant species (lettuce), whose relatively simple growth pattern made it an ideal candidate for model development and testing. However, different plant species can exhibit vastly different growth patterns, leaf shapes, and rates of development, all of which would require adjustments to the current algorithms and models.

Diversity in Growth Patterns: Different species grow at different rates and exhibit distinct growth stages. For instance, tomato plants exhibit more complex growth patterns, including flowering and fruiting stages, which require advanced modeling techniques beyond those applied to lettuce. Similarly, species like maize or soybeans, which have a much longer growing cycle, would introduce new challenges in modeling plant growth, particularly in capturing long-term growth trends and more subtle seasonal variations. The sigmoid function that worked well for lettuce may not be as effective for other plants with non-sigmoidal growth patterns. For instance, some species may follow a more linear growth pattern for much of their lifecycle, or they may exhibit periodic fluctuations in growth due to environmental stressors, seasonal changes, or flowering stages. This would require exploring different growth models, such as logistic growth, exponential models, or even cyclic models for plants with periodic growth spurts.

Adapting Leaf Detection Algorithms: The leaf detection and surface area estimation algorithms would also need adaptation when applied to other species. For instance, plants with compound leaves (e.g., soybeans or peas) pose a different challenge for leaf detection algorithms that were designed for single, broad leaves. Additionally, species with needle-like leaves (such as pine or spruce) would require entirely different segmentation techniques, possibly incorporating more sophisticated edge detection algorithms or deep learning-based classifiers to accurately count and measure leaves. Further refinement of leaf segmentation techniques, particularly by incorporating more advanced machine learning

algorithms such as convolutional neural networks (CNNs), could allow for more precise leaf detection across species with varied leaf shapes, sizes, and arrangements. Additionally, species-specific adjustments to the thresholding and filtering parameters of the current image processing algorithms could enhance the accuracy of surface area calculations.

Training on Multi-Species Data: To generalize the models for use with multiple species, it would be necessary to train the models on multi-species datasets. This could involve using transfer learning, where a model trained on one species (e.g., lettuce) is fine-tuned with additional data from other species, allowing it to generalize more effectively across different plant types. This could lead to the development of a universal plant growth model that can be applied to a wide variety of species with minimal reconfiguration. Furthermore, creating species-specific priors for the Bayesian growth models would enhance the predictions' accuracy when switching between species. For example, growth rate priors for lettuce would not be applicable to tomato plants, which grow at a much slower rate and have different physiological characteristics.

Exploring Plant-Plant Interactions: Another interesting line of inquiry would be to study the interactions between different plant species growing in close proximity. In real agricultural settings, multiple species are often grown together, and their growth can influence one another through competition for resources like light, water, and nutrients. Modeling these plant-plant interactions would require more complex models that account for both intra-species and inter-species interactions. Bayesian models, which can handle multi-level and hierarchical relationships, would be well-suited for this type of analysis.

7.3.2 Integrating More Environmental Variables

Currently, the models developed for plant growth prediction rely heavily on internal growth patterns observed in the plant's phenotype (e.g., leaf count, surface area). However, plant growth is highly dependent on external factors, such as temperature, humidity, light exposure, soil moisture, and nutrient levels. Future research could integrate more of these environmental variables into the predictive models to better simulate real-world conditions.

Incorporating Real-Time Environmental Data: One of the most promising research directions is to incorporate real-time environmental data into the predictive models. Sensors that measure temperature, humidity, and light intensity can provide continuous streams of data that can be used to adjust growth predictions dynamically. For instance, sudden changes in temperature or light levels could impact plant growth rates, and the models could be updated in real-time to account for these changes. By using Internet of Things (IoT) devices and connected sensors, it would be possible to create a more holistic model of plant growth that combines both internal phenotypic data and external environmental data. This would allow for more accurate predictions, particularly in environments where growth conditions fluctuate significantly.

Impact of Soil and Nutrient Data: Soil quality, including nutrient content, pH levels, and moisture availability, plays a crucial role in plant growth. While these factors are often measured sporadically in field experiments, incorporating real-time soil health data into the models would allow for more accurate predictions. For example, incorporating soil nitrogen content as a variable in the model could help predict growth spurts when nutrient levels are high or anticipate slowdowns when nutrient levels are depleted. Bayesian models are particularly well-suited to handle the uncertainty and variability inherent in environmental data. By updating the priors based on real-time data, the model can continuously refine its predictions and provide more accurate forecasts as new information becomes available.

Seasonal and Weather Patterns: Another area for future research is the incorporation of seasonal trends and weather patterns into plant growth models. In real-world agricultural settings, factors like rainfall patterns, wind speed, and seasonal temperature shifts have a profound impact on plant growth. By integrating seasonal models or weather forecasting data into the plant growth models, researchers could create more robust forecasts that account for the cyclic nature of environmental factors. Seasonal decomposition techniques could be used to isolate the seasonal component of growth from the overall trend, allowing the models to predict how plants might respond to future environmental changes. For instance, predictive models could simulate how a particularly dry season or an unusually wet spring might affect plant growth.

Integrating Machine Learning for Environmental Data: As more environmental data becomes available, advanced machine learning algorithms like Random Forests or Gradient Boosting Machines could be integrated into the Bayesian models to detect complex interactions between environmental variables and plant growth. These hybrid models could leverage the flexibility of Bayesian inference for uncertainty estimation while benefiting from the high-dimensional data processing capabilities of machine learning techniques. The inclusion of environmental factors would also allow researchers to build more predictive models that simulate various growth scenarios under different environmental conditions. These models could be used by farmers to optimize planting times, watering schedules, and nutrient application, ultimately improving crop yields and sustainability.

Chapter 8. References

1. Farjon, G., et al. (2021). *Leaf Counting: Fusing Network Components for Improved Accuracy*. Frontiers in Plant Science.
   DOI: https://doi.org/10.3389/fpls.2021.575751
2. Buzzy, M., et al. (2020). *Real-Time Plant Leaf Counting Using Deep Object Detection Networks*. MDPI Sensors.
   DOI: https://doi.org/10.3390/s20236896
3. Dobrescu, A., et al. (2017). *Leveraging Multiple Datasets for Deep Leaf Counting*. IEEE International Conference on Computer Vision Workshops.
   DOI: https://doi.org/10.48550/arXiv.1709.01472
4. Roggiolani, G., et al. (2024). *Unsupervised Pre-Training for 3D Leaf Instance Segmentation*. IEEE Robotics and Automation Letters (RA-L).
   DOI: https://doi.org/10.1109/LRA.2023.3320018
5. Koirala, A., et al. (2019). *Deep Learning for Real-Time Fruit Detection and Orchard Fruit Load Estimation: Benchmarking of MangoYOLO*. Precision Agriculture.
   DOI: https://doi.org/10.1007/s11119-019-09642-0
6. Chen, L., et al. (2017). *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. IEEE Transactions on Pattern Analysis and Machine Intelligence.
   DOI: https://doi.org/10.1109/TPAMI.2017.2699184
7. Ren, S., et al. (2015). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. IEEE Transactions on Pattern Analysis and Machine Intelligence.
   DOI: https://doi.org/10.1109/TPAMI.2016.2577031
8. Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. arXiv.
   DOI: https://doi.org/10.48550/arXiv.1804.02767
9. Valerio Giuffrida, M., et al. (2019). *Leaf Counting Without Annotations Using Adversarial Unsupervised Domain Adaptation*. IEEE CVPR Workshops.
   DOI: https://doi.org/10.1109/CVPRW.2019.00315

10. Parviz, L., & Ghorbanpour, M. (2024) *Assimilation of PSO and SVR into an improved ARIMA model for monthly precipitation forecasting.* DOI: https://doi.org/10.1038/s41598-024-63046-3
11. Salakpi, E., et al. (2022) *A dynamic hierarchical Bayesian approach for forecasting vegetation condition*. Nat. Hazards Earth Syst. Sci., 22, 2725–2749 DOI: https://doi.org/10.5194/nhess-22-2725-2022
12. Hochreiter, S., & Schmidhuber, J. (1997). *Long Short-Term Memory. Neural Computation.* DOI: https://doi.org/10.1162/neco.1997.9.8.1735
13. Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press. https://gaussianprocess.org/gpml
14. Canny, J. (1986). *A computational approach to edge detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6), 679-698. DOI: https://doi.org/10.1109/TPAMI.1986.4767851
15. Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional networks for biomedical image segmentation*. International Conference on Medical Image Computing and Computer-Assisted Intervention. DOI: https://doi.org/10.1007/978-3-319-24574-4_28
16. He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2017). *Mask R-CNN*. IEEE Transactions on Pattern Analysis and Machine Intelligence. DOI: https://doi.org/10.1109/TPAMI.2018.2844175