AlphaGarden: Leveraging Simulation in Developing an Autonomous Real-Sim-Real Pipeline for Polyculture Gardening

Rishi Parikh

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-120
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-120.html

May 11, 2023

Acknowledgement

I would like to thank my advisor Professor Ken Goldberg. Thank you for your support and mentorship for the past three years in AutoLab.

I would like to thanks my mentors: Simeon Adebola, Ellen Novoseller, and Daniel Brown who have guided me through this process. I would also like to thank some of my collaborators: Mark Presten, Satvik Sharma, Shrey Aeron, and Sandeep Mukherjee.

Last, and most importantly, I would like to thank my family: my mom, dad, and brother who have supported me throughout my life and for inspiring me every step of the way.

AlphaGarden: Leveraging Simulation in Developing an Autonomous Real-Sim-Real Pipeline for Polyculture Gardening

by

Rishi Parikh

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ken Goldberg, Chair
Professor Pieter Abbeel

Spring 2023

AlphaGarden: Leveraging Simulation in Developing an Autonomous Real-Sim-Real Pipeline for Polyculture Gardening

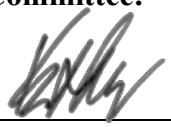
by Rishi Parikh

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:



Professor Ken Goldberg
Research Advisor

11 May 2023

(Date)

* * * * * * *

Professor Pieter Abbeel
Second Reader

5/10/23

(Date)

AlphaGarden: Leveraging Simulation in Developing an Autonomous Real-Sim-Real Pipeline for Polyculture Gardening

Abstract

AlphaGarden: Leveraging Simulation in Developing an Autonomous Real-Sim-Real Pipeline for Polyculture Gardening

by

Rishi Parikh

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Ken Goldberg, Chair

The AlphaGarden is an automated testbed for indoor polyculture farming which combines a first-order plant simulator, a gantry robot, a seed planting algorithm, plant phenotyping and tracking algorithms, irrigation sensors and algorithms, and custom pruning tools and algorithms.

AlphaGardenSim is a custom, fast, polyculture garden simulator which was used to learn various planting, irrigation, and pruning policies which were later evaluated in real.

Using an overhead camera and soil sensors to collect data from a physical scale garden testbed, the autonomous system utilizes a learned Plant Phenotyping convolutional neural network and a Bounding Disk Tracking algorithm to evaluate the individual plant distribution and estimate the state of the garden each day. From this garden state, AlphaGardenSim selects plants to autonomously prune and how much to irrigate each plant. A trained neural network detects and targets specific prune points on the plant.

The pruning pipeline is experimentally evaluated through four controlled 60-day garden cycles. Results suggest the system can autonomously achieve 0.94 normalized plant diversity with pruning shears while maintaining an average canopy coverage of 0.84.

Last, this thesis systematically compares the performance of the AlphaGarden to professional horticulturalists on the staff of the UC Berkeley Oxford Tract Greenhouse, adding closed loop variable irrigation. The humans and the machine tend side-by-side polyculture gardens with the same seed arrangement. We compare performance in terms of canopy coverage, plant diversity, and water consumption. Results from four 60-day cycles suggest that the automated AlphaGarden performs comparably to professional horticulturalists in terms of coverage and diversity, and reduces water consumption by as much as 44%.

to my family

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisor Professor Ken Goldberg. Thank you for your support and mentorship for the past three years in AutoLab.

I would like to thanks my mentors: Simeon Adebola, Ellen Novoseller, and Daniel Brown who have guided me through this process. Simeon, your positive attitude was always contagious, and I'm glad to have been able to work with you. I also appreciate your support and willingness to give advice at every step of the way. Ellen, I am thankful to have grown as a researcher under your mentorship. I would also like to thank some of my collaborators: Mark Presten, Satvik Sharma, Shrey Aeron, and Sandeep Mukherjee.

Through the years, I also got the chance to get close to many lab members who I now consider some of my closest friends. I will miss the post-paper deadline movie nights, playing IM frisbee, and lab lunches.

Last, and most importantly, I would like to thank my family: my mom, dad, and brother who have supported me throughout my life and for inspiring me every step of the way.

This thesis focuses on AlphaGarden, an ongoing project as part of the AutoLab at Berkeley. I would like to thank all of my collaborators and co-authors on these project.

Chapter 1

Introduction

In 1950, Alan Turing considered the question "Can Machines Think?" and proposed a test based on comparing human vs. machine ability to answer questions. Comparing how machines do on predefined tasks compared to humans improves our understanding of said tasks and serve as a rule of thumb for when the task is ripe for large-scale automation. This thesis considers the question "Can Machines Garden?" based on comparing human vs. machine ability to tend a real polyculture garden. Precision polyculture agriculture can reduce pesticides and water usage but requires more labor than monoculture farming. With the current state of world hunger, drought across cities worldwide, and a growing population, there is a pertinent need to sustainably produce enough food for everyone, even with limited water resources. Could robots help?

Polyculture farming is the practice of growing multiple crop types or plant species in a single area, rather than monoculture farming where only a single crop type is grown. Additionally, polyculture gardens reduce the depletion of soil nutrients while exhibiting many benefits compared to monoculture gardens, such as reduced demand for fertilizers, soil support, and crop support. A robot could change the way agriculture is practiced by increasing yield and improving resource efficiency.

This thesis outlines key components to automating polyculture farming in an indoor setting and compares the robot policy versus professional horticulturalists on three key metrics: plant coverage, plant diversity, and water use. AlphaGarden combines a custom first order POMDP simulator AlphaGardenSim and a real world test bed AlphaGarden built on top of a farmbot gantry robot.

The AlphaGarden tackles four challenges to successful automation:

1. Simulation: The AlphaGardenSim is a fast, first order simulator, that incorporates parameterized individual plant growth models, companion plant effects and inter-plant dynamics.

2. Real to Sim: To interface with AlphaGardenSim, we explore computer vision models and algorithms to translate between the physical garden and simulation.

Figure 1.1: **Overhead Garden Image** The above is a photo of the latest garden cycle taken at day 60. The red line splits the garden in half, where one side was tended by human experts, and the other by AlphaGarden. Each side started off with initial symmetric configurations reflected against the center line. Can you guess which side is which?

3. Sim to Real: To enact both learned and analytical policies from AlphaGardenSim into the real world, this thesis presents custom hardware and action planning algorithms.

4. Real World Experiments: We present 8, 60 day garden cycles, where we compare different pruning and irrigation policies versus expert human horticulturalists.

Chapter 4 summarizes AlphaGardenSim as well as dives into detailed modifications and experiments conducted in simulation. Chapter 5 and 6 detail algorithms designed to bridge the real-sim-real gap. Sensor readings from the physical garden test bed are converted to a state representation for the simulator, which allows us to use a lower fidelity simulator policy on the high fidelity image data. Next, sim-real algorithms are used to translate simulator policy to the real garden test bed. Chapter 7 discusses analyzes the results of 8, 60 day garden cycles carried out.

Chapters 8 and 9 describe the current limitations of deploying AlphaGarden at scale and future work and project direction.

In this thesis, I present work based on the following papers:

1. Learning Seed Placements and Automation Policies for Polyculture Farming with Companion Plants [1]

2. Simulating Polyculture Farming to Learn Automation Policies for Plant Diversity and Precision Irrigation [2]

3. Automated Pruning of Polyculture Plants [3]
4. **Can Machines Garden? Systematically Comparing the AlphaGarden vs. Professional Horticulturalists [4]**

My primary contributions to this project have been: formulation and assessment of new policies in simulation [2], a computer vision model to detect plant type from overhead images [1], [3], and integration of the real to sim portion of the autonomous pruning pipeline [3]. Last, this thesis will focus on [4], which I was a co-first on and is currently a finalist for outstanding paper at IEEE ICRA 2023. In this paper I helped develop new policy in simulation including staggered planting and variable irrigation, and aided in debugging physical experiments against expert horticulturalists.

Chapter 2

Related Work

2.1 Polyculture Farming

Polyculture farming, where multiple plant species are intercropped simultaneously and in close proximity, is a form of agricultural cultivation used for centuries that has been shown to enhance pest control, reduce weeds, limit soil erosion, and provide better use of light, water and soil nutrients [5, 6, 7, 8]. It is known that specific mixtures of cultivated plant species can result in higher overall yield [9]. Examples of mutually beneficial polycultures developed prior to industrial agriculture include maize-bean mutualisms, where maize provides a structural scaffold for the nitrogen-fixing leguminous vines [10], and intercropping of deep rooted native shrubs into grain cultivation, which improves water availability in arid regions [11]. More contemporary examples include shade-grown coffee, where species diverse agroforestry practices that can included cacao and banana intercropping provide not only canopy shade for coffee, increasing yields, but also provides needed habitat for birds, butterflies and other species [12].

Monoculture farming, as typically practiced in large-scale, industrial applications, is often characterized by heavy agrichemical inputs, such as chemical fertilizers and pesticides [13, 14], and increased vulnerability to disease and pestilence. The lack of long-term sustainability of industrial agriculture [15], and its implications for human food security, has sparked renewed interest in polyculture [16, 17, 18]. However, the advancements and automation seen in modern day monoculture farming have not yet scaled to its polyculture counterpart. One form of polyculture farming is indoor farming is a class of Urban Agriculture where crops can be grown inside a building [19], [20]. There is also increasing commercial interest in indoor farming [21].

One drawback is that polyculture farming requires more human labor than monoculture farming. In a polyculture setting, experts carefully select placements of seeds to optimize for companion growth, overall diversity, and coverage. Plants are irrigated to ensure optimal water uptake for overall plant health. Lastly, plants are carefully pruned to ensure smaller, slower growing plants have room to grow, remove dead or dying leaves, and promote plant

growth.

2.2 Automated Agriculture

Modeling and simulation can allow us to simulate plant growth and test many potentially useful policies at a much faster rate than natural growth. While monoculture plant simulators are well known [22] [23] [24], in prior work we presented the AlphaGardenSim [16], a polyculture plant simulator that uses first-order models of single plant growth, simulating inter-plant dynamics and competition for water and light. This thesis customizes and tunes the original implementation of AlphaGardenSim for real world applications.

Prior work has used the Farmbot, a robot gantry system, to interface with humans to help with seed planting, watering, and plant monitoring routines [25]. Robotic systems have also tended gardens such as work from Correll et al. [26], who designed a distributed autonomous gardening system with mobile manipulators that detect plants, irrigate, and grasp fruit. Botterill et al. studied pruning of grape vines using a six degree of freedom robot arm [27]. Also, Hernandez et al. [28] developed an autonomous urban garden that monitors soil moisture, temperature, and humidity. Some previous work has also compared the performance of a robotic system designed for agriculture to that of a human. Hayashi et al. developed a strawberry harvesting robot and indicated that its execution time, while better than previous studies [29, 30], took 2.5-3 times longer to harvest than a human [31]

Corbett-Davies et al. leveraged classification and search algorithms to make pruning decisions on simulated vines and indicated their system outperformed a human pruner with a skill level above the typical vineyard worker on this simulated domain [32]. However, to the best of our knowledge, there is no systematic comparison between an autonomous robotic system and horticulturalist(s) tending two identically planted gardens.

The current state of polyculture agriculture implements several strategies to promote more natural and optimal plant growth such as relay planting and intercropping, which is planting the next cycle of crops before the end of the first cycle and adding plants in empty spaces of the garden [33]. In AlphaGardenSim, we build on this by dynamically predicting vacancies in the garden created either through pruning or decay of plants and determining which seed to plant to promote coverage and diversity.

While in traditional farming, both rainfed and irrigated agriculture is discussed [34], in indoor farming, most of the focus has to be on irrigated agriculture. Moreover, in the face of drought in parts around the world, irrigated agriculture is still the world's major water user, using more than 70 percent of global water. Therefore, there is research into sustainable irrigation for agriculture [35]. In AlphaGardenSim, irrigation simulation is also being improved to further optimize coverage and diversity while reducing water usage. We present some of our ongoing work with irrigation in this paper.

Closed-loop robotic irrigation allows for better water management. There exists prior work using mobile robots for irrigation. For example, Zhen et al. prototyped a wireless network for closed-loop drip irrigation that used soil sensors as input [36]. We apply closed-loop

drip irrigation in a polyculture setting by interfacing a custom Arduino board (Farmduino [37]) on the Farmbot with soil moisture sensors through a wireless server.

Chapter 3

Automated Polyculture

3.1 Real to Sim to Real

AlphaGarden is an autonomous system that leverages simulation to tend to a real world polyculture garden. The pipeline consists of 5 components: Sensors, State Estimation, Simulation, Action Planning, and Execution as shown in fig 3.1.

Finding an optimal policy is a challenging task as the long time constants for real-world experiments makes this a data limited regime. A recent paradigm in robotic tasks is using simulation to model real world behavior and learn policy in a lower fidelity setting. This thesis presents a real-sim-real pipeline that leverages simulation to learn policies and real world algorithms to enact them.

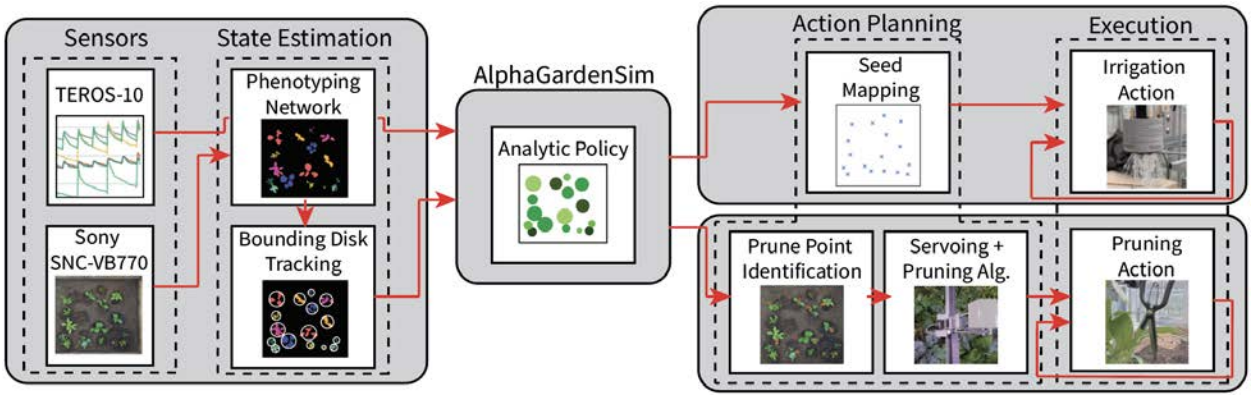


Figure 3.1: **AlphaGarden Autonomous Pipeline.** The leftmost box encompasses the Real2Sim phase, while rightmost shows Sim2Real. The overhead Sony camera and TEROS-10 soil moisture sensors gather data. This is followed by a state estimation process to identify individual plants and translate them to the simulator representation. AlphaGardenSim determines appropriate actions in real time, which is followed by an action planning and execution on the physical garden.

Chapter 4 details AlphaGardenSim, a first order simulator tuned with real world data to learn policy from a low fidelity representation of plant data. Chapter 5 and 6 describe a sequence of algorithms to translate between simulation and real as shown in fig: 3.1. At a high level sensors including an overhead camera and soil moisture sensors are used to extract a garden state. This garden state is then translated from a high fidelity state of pixels and sensor readings to a simulator representation. Custom pruning and irrigation tools are then used to enact the simulator policies into the real world.

3.2 Problem Statement

In each garden cycle, several plant types are planted over a specified number of days with the goal of increasing coverage, plant diversity, and reducing water usage. The problem statement below is consistent with [3], but adds water usage measurements and comparison with human experts.

Each garden has a total of n plants, placed within a planter bed of size (w, h), in cm. For each plant $i \in [0, n)$, the plant has its center coordinates (x_i, y_i), current radius r_i, both in cm. Each plant also has a corresponding plant type, p_i, which dictates the estimated germination time g_i, maturation time m_i, and maximum radius R_i. The lifecycle of each plant i is defined by five stages: germination, vegetative, reproductive, senescence, and death similar to [16]. Each stage corresponds to a target volumetric water content (VWC). A garden state on day t includes all information described above for every plant $i \in [0, n)$ at day t. Thus, a garden state is defined as:

$$s(t) = [p_i : ((x_i, y_i), r_i), ...], i \in [0, n)$$

We define coverage, $c(t)$, as the sum of all plant types canopy coverage, $c(t) = \sum_i c_i(t)$, over total area $w \cdot h$ at day t for each plant type i. We define garden diversity as:

$$\mathbf{v}(t) = [c_i(t) \cdot (R/R_i)^2, ...], \forall k$$
$$d(t) = H(\mathbf{v}(t))$$

where $H(\cdot)$ is an entropy function, $\mathbf{v}(t)$ is a vector of normalized plant types coverage, and R is the average maximum radius over all plant types. Multiplying $c_i(t)$ by $(R/R_i)^2$ normalizes each plant type's canopy coverage. We normalize because smaller, less dominant species are less likely to have the same coverage as much larger, faster-growing species. We aim to minimize water consumption while maximizing $c(t)$ and $d(t)$ through irrigation and pruning actions.

3.3 Physical Setup

FarmBot Gantry Robot: AlphaGarden is a $3.0m \times 1.5m$ raised planter bed located in the UC Berkeley greenhouse. A commercial FarmBot [37] gantry robot is installed over the

planter bed frame. This CNC robot may travel to any location in the garden from the soil level to $0.4m$ above. The FarmBot also features a magnetic universal tool mount (UTM) on its Z-axis that can automatically swap between tools stored on the west side of the bed.

Overhead Camera We mounted a Sony SNC-VB770 digital camera [38] with a $20mm$ Sony lens [39] $2m$ above the garden bed to monitor AlphaGarden. The camera's major requirements include (1) resolution, (2) image distortion, (3) power delivery, and (4) remote data accessibility. The VB770 satisfies these needs. It has a DSLM $35mm$ sensor with a maximum 4240×2832 resolution (1.4x higher than 4K) image mode. The $20mm$ lens minimizes distortion and allows us to capture the entire garden. We vetted the $20mm$ lens for characteristics including barrel distortion and lateral and axial chromatic aberrations. Together, the lens and sensor capture the entire garden bed with high clarity. This minimizes two problems that normal lenses introduce: occlusion and changes in relative size. Using a normal lens, plants at the edges of the field of view would often be occluded by plants in the center and distortion would affect how a plants center and radius is computed. The combined camera and lens automatically records photos every hour.

Onboard Camera A snake inspection camera [40] is located adjacent to the UTM on the Z-axis. It allows for close-up images of plants and soil. The onboard camera integrates with the FarmBot OS.

Soil Moisture Sensors To measure soil moisture and model soil dynamics, we distributed six TEROS-10 soil moisture sensors [41] throughout AlphaGarden. These measure the Volumetric Water Content (VWC) of the soil with a $430mL$ volume of influence. The sensors connect to a ZL6 Data Logger [42], which publishes readings every 30 minutes.

Drip Emitters Shrubbler drip emitters can be used to vary the water supply to a zone of the garden or to individual plants. They allow the adjustment of how much water each plant receives by varying the number of turns on the emitter head and for how long water flows through the emitter. More turns are equivalent to a higher flow rate and running for a longer time means each zone or plant receives more water. In moving from sim to real, we take advantage of this unique property of Shrubbler drip emitters and scale the water requirement for each plant from our experiments in simulation to ensure plant growth. A solonoid valve is used to turn on and off the drip emitters.

Custom Pruning Hardware

Rotary Pruner We built a custom pruning tool, dubbed the Rotary Pruner, that is lightweight, integrates with the FarmBot universal tool mount, and mounts automatically. Inspired by the traditional weed whacker, our first generation model utilizes thin, flexible blades rotating at high speeds to cut plants. We selected an SM Tech 775 Brushed 24V DC

Figure 3.2: **Pruning tools. Left:** Physical model of Rotary Pruner with a high speed motor and trimming blades. **Right:** Physical model of Pruning Shears with three servos to control closing, tilt, and orientation.

motor capable of 12000 rpm to achieve this. The motor's high power needs (>5V) mandated an external voltage source separate from the Farmbot's power rail. Thus, we designed a spring pin mechanism that allows the external power rail to automatically connect to the tool. We also designed a motor housing that inter-operates with the FarmBot UTM. The electrical control includes a relay circuit that governs motor power and uses GPIO to integrate with the FarmBot OS. The FarmBot does not rotate along the Z-axis, so we designed two such rotary pruning tools with different orientations: one that cuts along the X-axis and another that cuts along the Y-axis.

The Rotary Pruner that is chosen has a cutting direction that is closest to being orthogonal to the vector from the plant's center to the prune point, and is autonomously mounted using the tool rack and FarmBot UTM. To estimate the height of the plant and find the distance to the target leaf d, we mounted a Sharp infrared distance sensor [43] adjacent to the FarmBot UTM pointing towards the soil surface. After arriving at the prune coordinates and measuring d, the Rotary Pruner is then toggled on, and the FarmBot is lowered to $d + 5cm$; the system overestimates the depth of the leaf in order to ensure a cut. The Rotary Pruner is then toggled off and returned to its home position.

The Rotary Pruner faced fundamental limitations, primarily with its aggressive method of operation (the high speed blades would cause debris to fly), which could pose a danger to objects and people around the garden.

Pruning Shears Although the Rotary Pruner proved useful for many of the initial pruning actions, it spotlighted a few shortcomings that we wished to fix with a redesigned pruning attachment. Firstly, since the Rotary Pruner uses two separate attachments, the autonomous system had to regularly switch these attachments, adding unwanted power consumption and increasing the likelihood of mechanical failure. Secondly, due to the Rotary Pruner's relatively aggressive method of operation, it would frequently damage the target leaf (as well as surrounding plants) when attempting a prune action. This caused a reduction in plant health and an increase in water consumption.

For a quieter, more precise and delicate pruning tool, we motorized a pair of Japanese topiary shears. A pair of Niwaki Topiary Shears [44] were fastened directly to the FarmBot's gantry rails. A YANSHON Digital 360° servo motor is able to close the shears by winding a high strength steel cable attached to one handle of the shears onto a spool; the shears reopen with a spring mechanism when the cable is unwound. This assembly is mounted to a 2-axis servo gimbal (using BETU Digital 270° servo motors). The gimbal is able to position the shears vertically, horizontally, or at any intermediate angle as well as rotate the shears a full 180° to account for any leaf direction, allowing the FarmBot to trim with greater precision as well as reach the tops of plants. The servos connect to the FarmBot PWM header and integrate seamlessly with the FarmBot OS.

Control of the shears is executed through the three servos: one for tilt, one for cut angle, and one for shear closure. The Pruning Shears are at default open and stored horizontally to avoid collisions with plants below. The shears require calculating the orthogonal vector to the vector spanning from the center of the plant to the prune point. The servo that controls cut angle is then activated to position the shears along the orthogonal vector. The tilt servo then swivels the shears to a vertical position. The shears are then lowered to $d + 5cm$ and activated. Once a cut is complete, the shears return to their default positioning.

Chapter 4

AlphaGardenSim

AlphaGardenSim is a custom simulator for polyculture plant growth that has been calibrated using real-world measurements obtained from a physical test bed. The goal of the simulator is to accurately model the dynamics and relationships between various plant species and their surrounding environment. This simulator is designed to operate as an open-source platform, which allows users to access and modify the code base as required, in order to tailor the simulation for their specific needs. This section presents AlphaGardenSim 2.0 which builds from the original simulator presented in [16] and modified in [2] [1].

4.1 Polyculture Garden as a POMDP

AlphaGardenSim is designed to model a lush and diverse polyculture garden at a fraction of natural growth time. This is accomplished by representing the garden as a discrete HxW grid, which contains N plants that are sampled from a set of k plant types. The objective

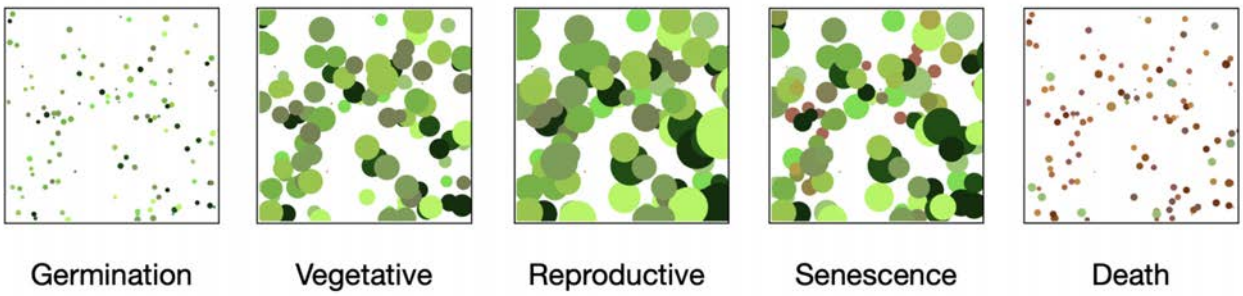


Figure 4.1: **Plant Life Cycle** Above is snapshots from a garden simulation across different stages in a plant lifecyle. At each stage, the plant interacts with its environment differently. AlphaGardenSim is able to simulate thousands of garden cycles in the time it takes to grow just one.

of the simulation is to cultivate the garden over a given growing period T while minimizing the need for irrigation.

This objective can be formulated as a Partially Observable Markov Decision Process (POMDP), which is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O})$. This framework provides a systematic way of modeling the decision-making process required to cultivate the garden successfully.

State $(\mathcal{S})$ The state space S represents the set of possible garden configurations. The parameters of the state $s(t)$ at timestep t are plant type, plant health, water amount, vacancy, and seed location.

Actions $(\mathcal{A})$. Action $a(t) \in \mathcal{A}$ is an action selected from the set of feasible actions that can be taken at each time t. At each timestep, an agent can decide what plants to prune, how much water to irrigate, and whether or not to seed new plants. Hueristic and learned policies are described in Section 4.3

Transitions $(\mathcal{T})$. At each timestep t, AlphaGardenSim executes a sequence of updates across the garden: irrigation, lighting, water use and plant growth according to the models described in Section 4.2.

Rewards $(\mathcal{R})$. As the objective is to achieve a diverse garden with maximal yield and water efficiency, we seek to optimize diversity reward $r_d(t)$, coverage reward $r_c(t)$, and water use reward $r_w(t)$.

Observations $(\mathcal{O})$. The observation function O defines the relationship between the underlying state of the garden and the observations that are made by the simulator. To simulate sensor precision limitations, we define $\mathbf{o}(x, y, t)$, a sector of size $\frac{H}{10} \times \frac{W}{10}$ centered at point (x, y) representing the area observable at timestep t.

By framing the cultivation problem as a POMDP, AlphaGardenSim provides a powerful tool for simulating the complex ecological processes that occur within a polyculture garden. This enables users to test and evaluate different strategies for optimizing interplant dynamics and water use, among other factors, in a controlled and realistic environment.

4.2 Garden Dynamics

AlphaGardenSim models garden dynamics as a set of first order equations for fast and efficient approximation of relationships within the garden including: plant and interplant dynamics, water content, and sunlight.

AlphaGardenSim abstracts a plant as a tuple of seed location, plant radius, and height, which builds from a model proposed by Price et al. [45]. AlphaGardenSim models resource competition for light and water use building from related plant modeling literature: [46], [47], [48], [49]. Light allocation is modeled as a decaying function of plant height, where i^{th} tallest plant at point (x, y) in the grid receives $(\frac{1}{2})^i$ amount of light from point (x, y). The total light a plant accumulates is defined as l_u. Total light allocation is used to determine maximum water uptake: $w_{max} = \frac{c_2}{c_1}\sqrt{l_u}$, where c_1 and c_2 are plant-specific parameters that control a plant's water and light efficiency respectively. Plants with the same zone of influence must compete for the same resources as shown in fig 6.4.

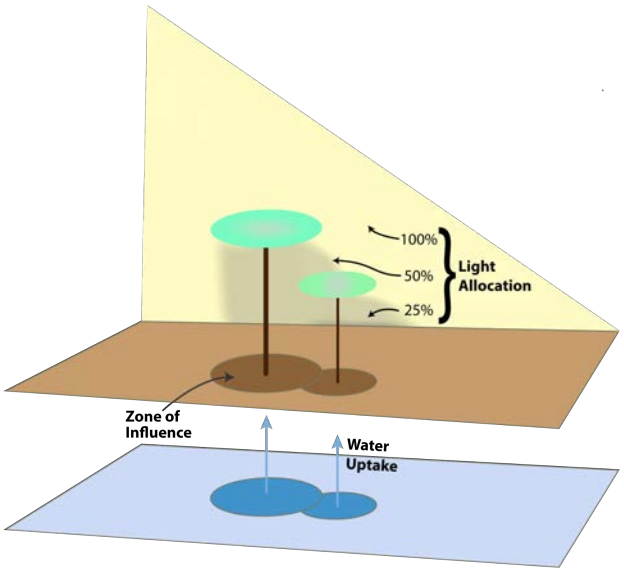


Figure 4.2: **Light and Irrigation Models.** Each plant receives light based on the size of its unoccluded leaf area in the grid, i.e., the number of grid points visible overhead, while occluded points allocate light in an exponentially decaying fashion. The plant's water uptake is then drawn from its neighboring grid points, to fulfill its growth potential. The plant is limited by the amount of light it intercepts and the amount of water available in its zone-of-influence.

A plant's growth is modeled as a logistic curve [50] based on resources available at said timestep and maximum growth:

$$\tilde{g} = c_1 \cdot \min(w, w_{max}) \cdot (1 - \frac{r_t}{r_1})$$

where w is the actual amount of water this plant was able to adsorb, r_t is the plant's current radius and r_1 is the plant's growth potential.

Plant interrelationships are defined within the relationship matrix $\mathbf{C} \in R^{k \times k}$, where k is the number of plant types in the garden. Here, $\mathbf{C}_{i,j}$ stores a value that describes the companionship between plants of type i and j.

The $\mathbf{C}$ matrix was populated by analyzing the growth curves of individual plants in the physical test bed relative to neighboring plants. One-hundred and twenty growth curves were created by annotating daily images of the garden with a plant's center and outermost radius. By comparing a plant's growth curve to the average growth curve of its type, we can discover if neighboring plants promote or hinder growth. Positive and negative scalar values were assigned and then tuned to minimize the MAE between simulated and real world individual plants.

The relationship matrix $\mathbf{C}$ is then used to calculate the companionship factor c. For a

Plant Type	g_0	g_1	m_0	m_1	r_1	c_1	$c(35)$	$e(35)$
Borage	7	7	49	55	60	0.09	3107	6.61
Kale	3	7	62	55	65	0.10	7450	5.41
Swiss Chard	7	7	53	50	47	0.11	5536	9.93
Turnip	3	7	42	47	53	0.11	3961	10.04
Green Lettuce	7	9	43	52	27	0.08	232	7.46
Arugula	5	8	45	52	40	0.10	1133	5.50
Sorrel	7	15	53	70	8	0.08	59	9.58
Cilantro	7	10	53	65	20	0.09	23	10.76
Red Lettuce	5	12	45	50	28	0.09	10	11.61
Radicchio	5	9	83	55	53	0.09	53	9.28

Table 4.1: **Growth Analysis:** Where g_0 (days) is original germination time, g_1 (days) is tuned germination time, m_0 (days) is original maturation time, m_1 (days) is tuned maturation time, r_1 is growth potential, c_1 is the biomass accumulation parameter, $c(35)$ (cm^2) is the simulated canopy coverage on day 35, and $e(35)$ (cm) is the mean absolute error on day 35 between simulated and average real world radius. Original values were taken from published plant tables [54]. Growth time is found by subtracting g_1 from m_1.

given plant i,

$$c_i = \sum_{j \in [1,\cdots,N], j \neq i} \frac{\mathbf{C}_{p(i),p(j)}}{\|l(i) - l(j)\|_2^2}$$

where $p(i)$ is the plant type of seed i and $l(i) = (x_i, y_i)$ as the location of seed i. The new daily radial growth parameter is defined to be $g = \tilde{g} \cdot c$.

AlphaGardenSim also models a plant's life cycle consisting of five non-overlapping stages: germination, vegetative, reproductive, senescence and death [51, 52] as shown in 4.1. The number of timesteps between consecutive stages is a random variable sampled from a plant-specific discritized Gaussian distribution, assuming that plants of the same type share transition times between stages [53]. Each stage has a different dynamics model and resource allocation.

4.3 Policies

Baseline Policies

AlphaGardenSim defines five baseline policies in [16], which are used to experiment and motivate real world implementation and experiments. Additional experiments such as adaptive sector sampling, dynamic, and staggered planting build on the five baseline policies.

1. Fixed, a policy that irrigates according to a fixed schedule and prunes all plants uniformly.

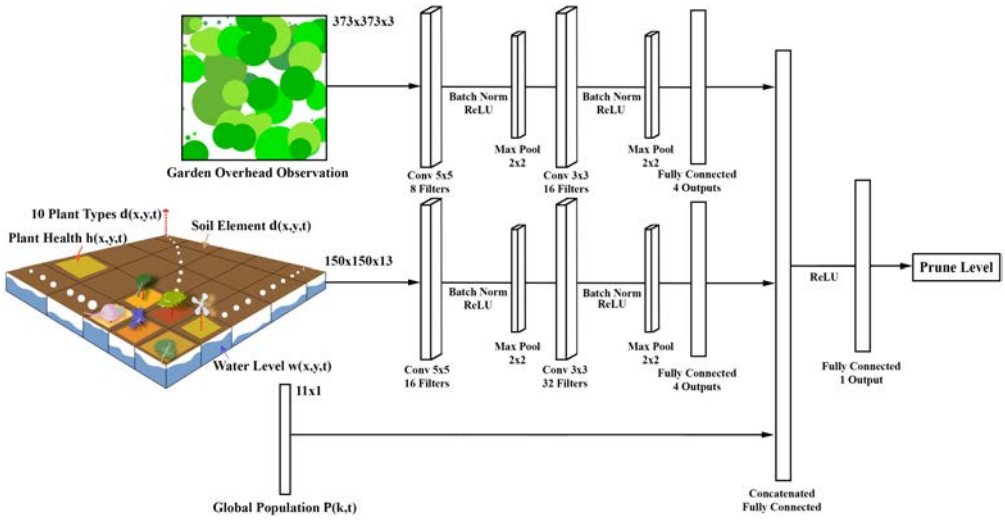


Figure 4.3: **Pruning policy network architecture** A deep convolutional neural network. The network takes three inputs: 1) an RGB image of the full garden; 2) a matrix of $h(x, y, t)$, $w(x, y, t)$ and $\mathbf{d}(x, y, t)$ for all (x, y) in the garden; 3) the global population distribution $\mathbf{P}(k, t)$ to predict a prune level for each observation using demonstrations from variable pruning.

2. Analytic, a policy that irrigates and prunes plants with a fixed pruning level based on water availability, plant health and garden diversity.

3. Look-ahead, a policy that selects a pruning level $p \in \mathcal{P}$ for each day t from a discrete set of pruning levels $\mathcal{P}$.

4. Learned, a deep supervised learned policy that learns from the lookahead policy prune level demonstrations to predict prune levels over 1500X faster than the look-ahead policy as seen in fig. 4.3

Staggered Planting

In a polyculture farm setting, plants can grow at different rates, some faster and some slower. To ensure faster-growing plants do not initially out-compete slower-growing ones, we introduce staggered planting, or planting in stages.

Staggered planting allows us to plant slower-growing plants earlier in the planting cycle while faster-growing plants are planted later in the cycle. We adapt staggered planting in the real world to account for variability when plants fail to germinate by transplanting a similar seedling in its place.

Staggered Planting in Simulation Given an initial seed placement, AlphaGardenSim models the interplant relationships over the course of a garden cycle. We use these dynamics

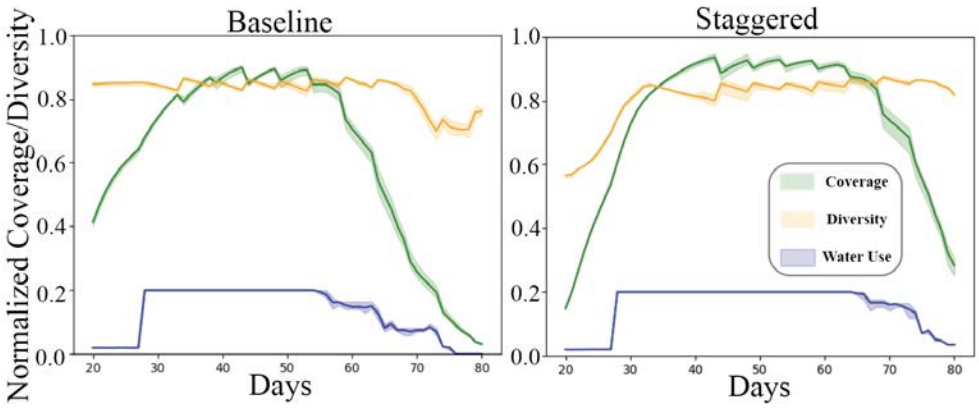


Figure 4.4: **Coverage and Diversity simulation results.** We compare the results of *(left)* normal and *(right)* staggered planting in simulation repeated 5 times. We find higher and sustained coverage in staggered planting through days 30-60.

and tuned growth models to predict the diversity and coverage of a garden and compare the results of the two garden cycles. One implements staggered planting and the other does not. The results of 5 randomized trials are shown in Figure 4.4. At day 50, the average normalized canopy coverage is 0.71 and 0.90 respectively for the normal and staggered simulations. Both methods had normalized diversity of 0.86.

Staggered Planting in Real To evaluate the results of staggered planting, we compare four 60-day gardens. In the first garden cycle, all plants are seeded on day 1, while in the other garden cycle, the fast growing plants are seeded after 10 days. In the cycle with staggered planting, this allows for the slower-growing plants to get a head start and reach their potential, whereas previously they may have been outcompeted by the faster-growing plants. In real-world experiments, staggered planting can also reduce overall water usage, as the larger plants only need to be watered for 50 days.

Dynamic planting

Dynamic planting is an extension of staggered planting that uses the new planting action to obtain continuous coverage over longer garden periods, past the days of when plants seeded on day 0 live. We wish to seed plants in locations that minimize inter-plant competition for light and water so we provide the policy vacancy scores $e(x, y, t)$ for all (x, y) in each $\mathbf{o}(x, y, t)$. If any $e(x, y, t)$ in $\mathbf{o}(x, y, t)$ is above a threshold, and the maximum number of plants the policy can seed each day has not been reached, the policy seeds a plant at that location.

Dynamic planting has several benefits over other policies that use stagnant seed placements. Dynamic planting has potential to limit plant competition and achieve higher diversity due to the fact that smaller, slow growing plants can be seeded prior to larger, fast

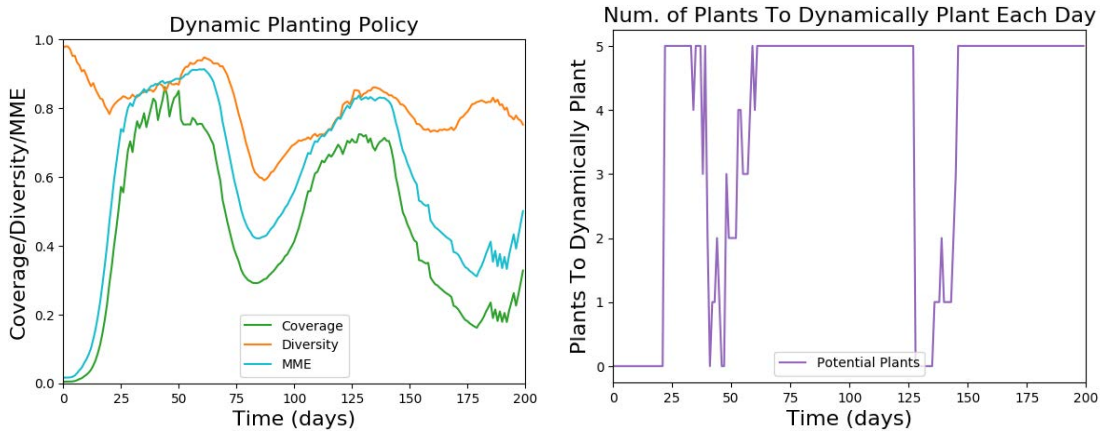


Figure 4.5: **Dynamic Planting Policy.** The policy seeds up to 5 new plants every day after day 20. During periods where coverage is high in the garden, there is little vacant space to seed new plants. As a result, the number of plants selected to be dynamically planted drops during days 35 to 61 and days 128 to 146. After these high coverage periods, up to 5 new plants are seeded every day resulting in a resurgence in coverage after the new plants germinate and mature.

Policy	Coverage	Diversity	MME	Water Use (liters)
Dynamic Planting	0.50	0.82	0.63	158.98

Table 4.2: **Dynamic Planting** policy averaged across 10 test gardens with 100 initial plants and the ability to seed up to 5 new plants every day after day 20. Evaluation metrics are averaged across all 200 days of garden simulation. Results show that replanting seeds can lead to sustained growth and diversity across indefinite periods of time.

growing plants when the garden period begins. Furthermore, a garden period is no longer constrained by constant companionship relations; new plants that are seeded can be chosen through a combination of optimizing local companionship relations and to improve global diversity and coverage.

We conduct dynamic planting experiments on a general setting initially consisting of 100 plants from 10 types. To evaluate how well dynamic planting can sustain garden growth, we simulate a growing period of 200 days. We follow the observation method from [16] to allow the policy to observe locations away from seed points $s(x, y)$. Dynamic planting begins seeding plants after day 20, which is when most of the original plants have reached the vegetative stage. The policy uses a vacancy threshold of $e(x, y, t) = 8$cm and can seed a maximum of 5 plants every day. Results averaged across 10 test gardens with random seed placements are visualized in Fig. 4.5. Since Dynamic planting only seeds new plants in locations that are sufficiently vacant, during periods where coverage is high in the garden, the number of plants seeded every day drops below 5. Once plants begin to die and coverage decreases, the garden becomes sparser, allowing Dynamic planting to find locations where

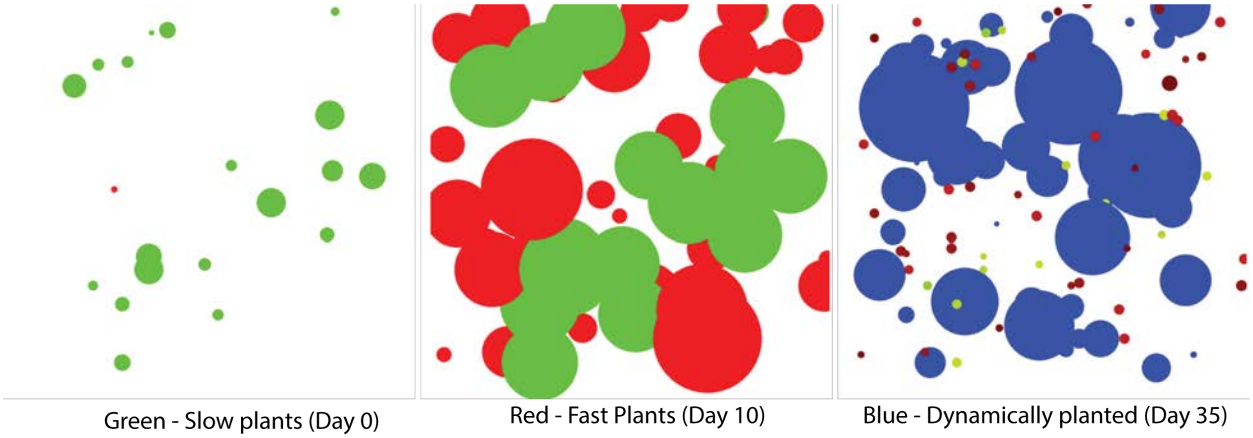


Figure 4.6: **Dynamic Planting Visualization:** This figure shows 3 snapshots of a garden cycle, where slow growing plants were planted on day 0, fast growing plants on day 10, and dynamically seeded plants starting from day 35. The center image shows that at its peak, staggered planting allows for a more diverse garden, with both slow and fast growing plants. The right image shows that as the initial plants are dying, newer plants take their place.

vacancy $e(x, y, t) \geq 8$cm. After the new plants germinate and mature, coverage rebounds.

Figure 4.6 shows an example simulation of staggered and dynamic planting.

Adaptive Sector Sampling

AlphaGardenSim samples observations using a sector sampling method which, at every timestep, samples m sectors centered at each $s(x, y)$ and an additional $\frac{m}{10}$ sectors centered at non-seed points. However, as seen in sectors can overlap due to plants seeded close to each other. During irrigation, both sectors may be watered, resulting in extra water usage. Additionally, multiple pruning actions may be used instead of one to prune all plants in the overlapping area. To address this, we create clusters of seed locations $s(x, y)$ that are within a distance c_d of each other. We center observations at the centers of these clusters to encompass all plants within that cluster. We create two sets of clusters: the seed locations of germinating plants that are within $c_{d,germ}$ of each other, and the seed locations of growing plants that are within $c_{d,grow}$ of each other. To further reduce the number of actions, we do not cluster, and consequently do not irrigate or prune, the seed locations of plants in Senescence or Death as these two stages are irreversible.

Chapter 5

Pruning

5.1 Plant Phenotyping

To estimate the garden state, we use a learned semantic segmentation neural network to label plant types from an overhead image. Plant phenotyping directly influences the success of Bounding Disk Tracking, and provides information on plant growth, diversity, and coverage.

For each day, the latest image from the an overhead camera is pulled from the server to be used for phenotyping. The images used are taken in the evening when there is uniform lighting without shadows. We trained a model using UNet architecture [55] and ResNet34 [56] backbone to output a $1630 \times 3478 \times (i_{total} + 1)$ array L of plant likelihood per pixel per label type, where i_{total} is the total number of plant types. The network is trained on six hand-labeled overhead images from previous garden cycles. Each image is split into 512×512 RGB patches and augmented via shifting and rotating. We extract leaf masks from various stages in the garden and overlay these leaves on top of the existing patches to augment the data set [1].

Accurate segmentation for plants after day 30 becomes increasingly important to be able to determine canopy coverage and pruning actions. However, a plant may look very different at germination compared to its mature state due to the distribution shift of a plant over its lifespan (as well as due to occlusions), which causes a drop in performance starting on day 40.

To address this, we introduce a prior probability distribution based on seed placement and plant maximum radius given from our tuned simulator [2]. We define a variable R_t^k, and c_t^k as the maximum radius and center of plant k at timestep t, and a $1630 \times 3478 \times (i_{total} + 1)$ occupancy grid, O defined as:

$$O(x, y, i) = \alpha * (2 - r/R_k)$$

if $r \leq R_k$ and c_k is of plant type i, where $\alpha = 5$, and r is the distance from c_k to (x, y), else 1.

We use this location based occupancy grid as a prior probability, and compute a new likelihood grid L' as an element-wise multiplication of the original segmentation output, L,

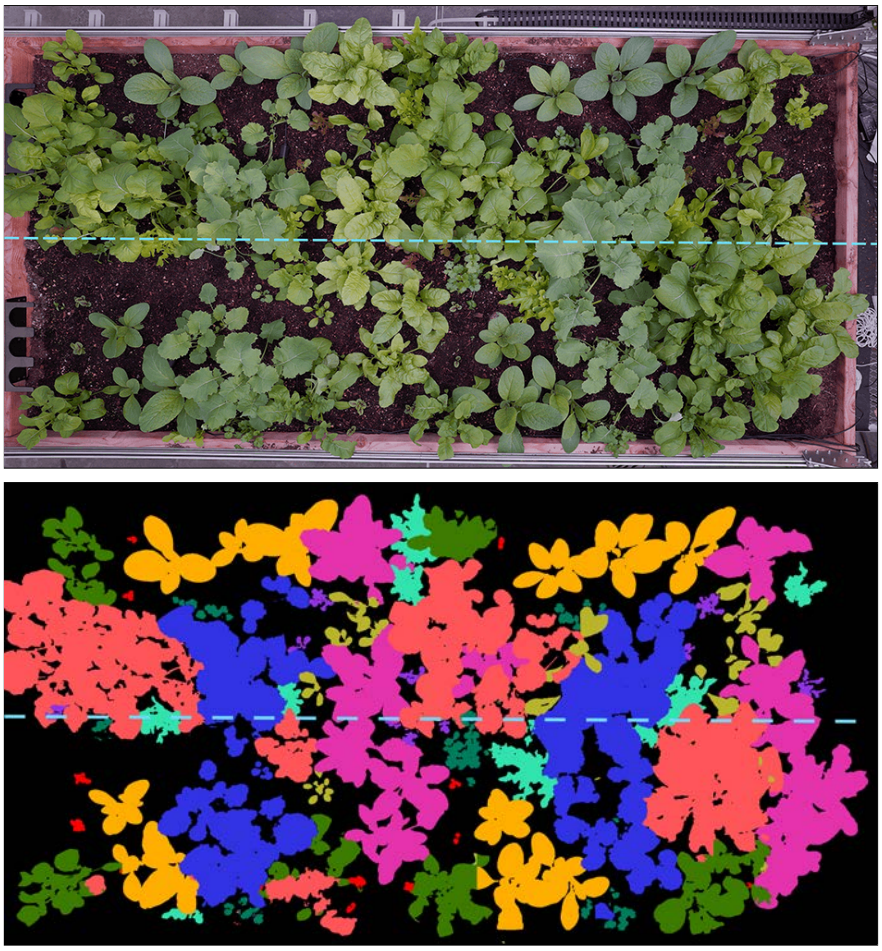


Figure 5.1: **Learned Plant Segmentation Model**. The figures above (from top to bottom) show an overhead image from October 6, 2020, and the classifier output from the network with augmented data. The overhead image is split in half as shown by the blue line. The top half is for training while the bottom half is for testing. Below, the table shows how much of the garden is covered by each plant and its respective IoU score based on the bottom half only. By adding augmented data, the model was able to more accurately classify unseen leaves when compared to the baseline with no augmented data. Low IoU for radicchio and red lettuce is consistent with a low percent of coverage. This model was later improved on by adding inductive biases such as location based priors and contour smoothening.

Figure 5.2: **Phenotyping and Bounding Disk Tracking.** 3 images from days 20, 30, and 40 of garden cycle 2. **Top row:** overhead images overlayed with the estimated bounding disks from the Bounding Disk Tracking algorithm. **Bottom row:** the masks created by the Plant Phenotyping network as well as the estimated bounding disks (same as above).

and occupancy grid, O, $L'(x, y, i) = L(x, y, i) * O(x, y, i)$, and output $\max_i L'(x, y, i)$ as the predicted label for (x, y).

We define mean IoU as $\sum_{i=0}^{i_{total}} IoU(\text{label}_i)/(i_{total}+1)$. The baseline model [1] had a mean IoU of 0.71 when compared to the ground truth at day 30. The new network, with data aggregation techniques and location based segmentation added, had a mean IoU 0.83 across the 9 labels on day 30. We saw the highest IoU of 0.97 in borage, which is one of the larger plants. Radicchio, which previously had the lowest IoU, had the largest increase from 0.23 to 0.59.

Adding location priors offers more robustness to the distribution shift in plants towards the end of the garden cycle and marginal improvements in the early stages of the garden. During day 50 and 60, mean IoU improved from 0.38 and 0.33 to 0.42 and 0.36 respectively with location based segmentation. The largest jump in IoU was for green lettuce, from 0.31 to 0.40 on day 60, while plants like kale saw little change with an IoU of 0.54 on both networks.

5.2 Bounded Disk Tracking

We define a plant's bounding disk (see Fig. 5.2) as the circle with the smallest radius such that all pixels corresponding to that plant are enclosed. This definition helps account for plants moving over time due to phototrophy [57] and irrigation [58]. We present two methods for finding circular representations of the garden's state and two metrics for comparison, and

evaluate each method against a hand-labeled benchmark for selected days using a circle IoU loss [59].

To estimate the garden state, defined by plant centers and radii $((cx_k, cy_k), r_k)$ indexed by plant type $p_k = i$, we convert the plant segmentation mask into estimates of each plant's center and radius. It is necessary to phenotype the overhead image before converting from real-life (real) to simulation (sim) to ensure pixels with the highest likelihood for that plant type affect its bounding disk representation.

We use a breadth-first-search (BFS) based algorithm and K-Means clustering to track each plant's center and radius. Both algorithms help address the issues with tracking plants over the duration of the garden lifecycle. BFS helps with irregular plant shapes and slight occlusions by continually searching outwards using a radial search heuristic, and K-Means helps address occlusion because it clusters non-contingent groups of pixels into a single bounding disk.

The BFS algorithm is initialized with seed locations and all plant radii at $0cm$. At each timestep, we use AlphaGardenSim [16] and the prior plant radius to calculate a maximum possible radius by simulating a day of plant growth. Given the prior radius, maximum radius, and minimum radius, the algorithm traverses outwards from the minimum radius. The algorithm stops when less than 10% of the newly traversed pixels are of the correct type or the maximum radius has been achieved. This process repeats each day for each plant. Even when a plant becomes fully occluded, the algorithm handles radial decrease using AlphaGardenSim's tuned wilting parameters.

The second method is a K-Means clustering based algorithm. K-Means clustering has two main assumptions – that the clusters (1) have roughly the same number of points and (2) are circularly distributed. The first assumption is true near the beginning of the garden, because plants of the same type grow similarly. However, later in the cycle, competitive relationships in the garden and occlusion start to create asymmetries, complicating this assumption. The second assumption follows from the circular model we use to track plants.

In order to benchmark the performance of these methods, we introduce two metrics: average circle utility (ACU) and percentage of pixels included (PPI). Each of these metrics is computed per plant type per timestep. Let P_i be the number of pixels in the segmentation mask of the inputted plant type that fall within at least one bounding disk, P_t be the number of pixels of the given plant type present in the segmentation mask, and P_c be the area of the union of the fitted bounding disks. We define the average circle utility as $\text{ACU} = \frac{P_i}{P_c}$ and percentage of pixels included as $\text{PPI} = \frac{P_i}{P_t}$.

We want to maximize both of these metrics, ACU and PPI, to compute the optimal bounding disks. On the extremes, these algorithms are adversarially related; smaller bounding disks tend to have higher ACUs because they will likely be centered around denser, less occluded portions of the plants. However, larger bounding disks will tend to have higher PPIs because a larger bounding disk will naturally have a larger portion of a plant k's pixels.

To judge the efficacy of these methods we compare them to hand-labeled bounding disks at various time steps. As Fig. 5.3 (left) shows, initial K-Means clustering performs well as

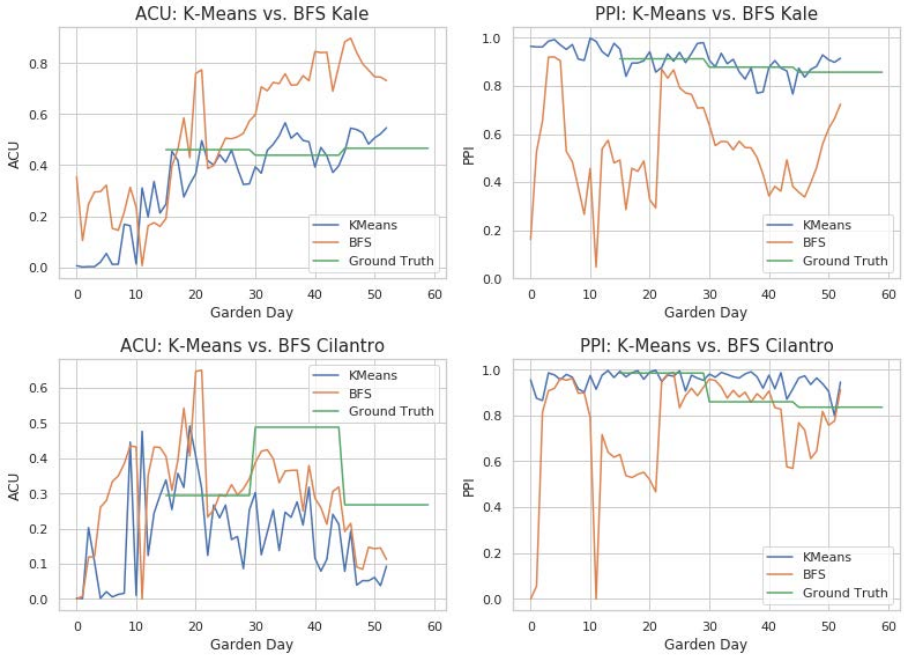


Figure 5.3: **Garden Metrics of Garden Cycle 2R for Kale and Cilantro**. We evaluate average circle utility (ACU) and percentage of pixels included (PPI) of the Breadth-First-Search (BFS) versus the K-Means bounding disk algorithms for Kale, a larger plant type, and Cilantro, a smaller plant type. **Kale:** BFS tends to have higher ACU, but lower PPI. For the days which ground truth circles exist (manually annotated), they are closer to the K-Means algorithm in both metrics. **Cilantro:** Similarly, BFS has a higher ACU and K-Means has a higher PPI. However, Cilantro generally benefits from the more conservative BFS. We adopt a mixed approach: the K-Means approach for larger plants and less occluded timesteps, and the BFS approach for denser, smaller plants.

its assumptions are easily met and the segmentation is highly effective. It also performs well on larger, less occluded plants. However, later in the cycle, this method's efficacy decreases as it overfits to segmentation errors and irregular plant shapes. As Fig. 5.3 (right) shows, BFS lags early on, but then becomes increasingly effective as plants are occluded mid-garden cycle.

5.3 Data Aggregation and Self Supervision

Hand labeling accurate ground truth masks is a tedious process. We developed a data aggregation based approach, allowing a human to make corrections to a predicted mask when the algorithm fails. This approach identifies plant sub regions using the contours of the prediction mask, and queries a human to generate the correct label. This method allowed us to quickly generate training data from multiple garden cycles to improve overall

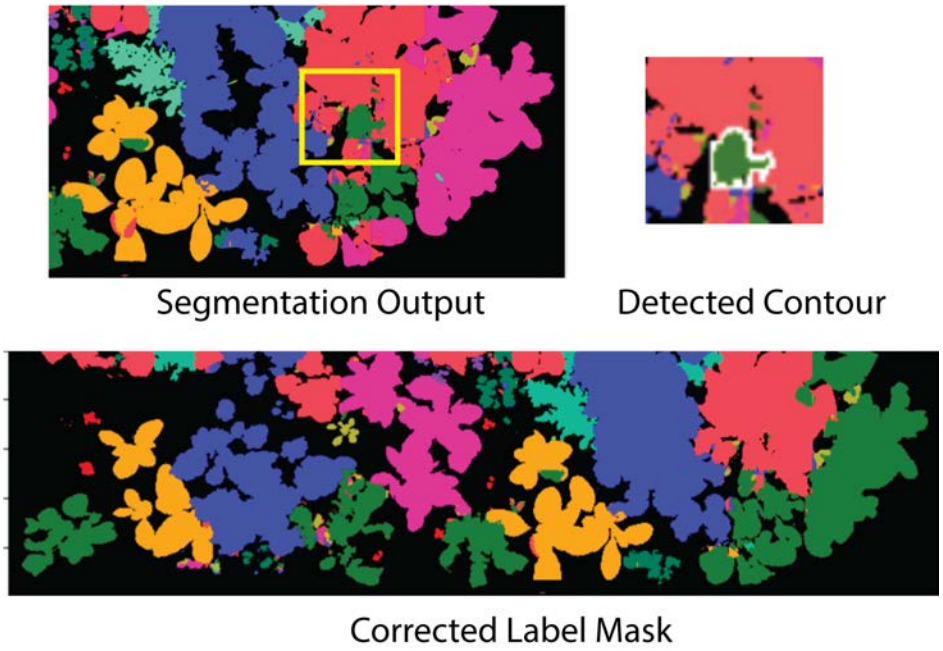


Figure 5.4: **Hand Labeling Data** Hand labeling data is a tedious and difficult task. A python script and UI was used to quickly correct labels from a model predicted output. The model is able to accurately identify contours, which allows for easy correction. However, there is a tradeoff between the time spent correcting and final IoU. In the example above, it took 15 minutes to reach over 90% accuracy in a task that previously took 3 hours to complete.

performance. This is shown in figure 5.4.

Furthermore, plants have a temporal consistency that suggest training on data from timestep t can aid in segmentation at timestep $t+1$. This is because plant features such as edges and leaf shape remain consistent as the plant grows. Training on online data is useful in improving model performance overtime and developing a zero shot learning pipeline that can adapt to unseen plants.

To do so, we leverage general purpose segmentation models such as Meta's segment anything model [60]. By correlating the segmentation data with the original seed placement, ground truth masks can be created efficiently and with high accuracy during early stages of the garden cycle.

In preliminary results, we note that off the shelf models may be somewhat inaccurate and miss plant leaves. Instead, we experimented with using the current model to distinguish plant vs other categories. This form of self supervision was successfully used to label overhead data from potted plant experiments. In future work, this principle can be extended to online training and zero shot learning.

5.4 Pruning Planner

Once a garden state for day t is estimated with the Bounding Disk Tracking algorithm, the analytic policy within AlphaGardenSim decides which plants to prune. For autonomous pruning, the system must identify and select specific target leaves to prune, be able to navigate and position the FarmBot above the chosen leaf using visual servoing, and execute the pruning action with custom hardware.

A Prune Point Identification neural network followed by a selection process is used to identify specific leaves to prune. Then, visual servoing positions the FarmBot above the proper leaf. Finally, the robot uses pruning algorithms, specific to the designated pruning tool, to cut the leaf.

5.5 Prune Point Identification

The system must identify the best leaf to cut after a plant is chosen to be pruned by AlphaGardenSim. Our baseline approach found the average point between an extrema of the plant, a point near the tip of a leaf as dictated by the bounding disk, and the plant center to find a theoretical leaf center. However, this was constrained by the reality of plants' physical makeup which often includes bending, occlusion, or oddly shaped leaves. The algorithm would frequently return points which were not on a plant or too close to an edge. We therefore explore a learned approach.

We trained a Prune Point Identification neural network based on the unsupervised domain adaptation network for plant organ counting by Ayalew et al. [61]. In the training process, our images are transformed to match the input network characteristics, allowing for a more seamless domain adaptation. The architecture consists of a Domain Adversarial Neural Network with a Gradient Reversal Layer to backpropagate between the source and target domains and classification is performed using a U-Net [61].

To evaluate this network's success in a polyculture setting, rather than its original monoculture domain, we trained it on all plant types, different sets of plant types that appeared to have distinct leaves, and on individual plant types from our domain. We found that training on all plant types led to the worst overall performance. Borage, a plant that has high success in being identified by our phenotyping network along with distinct, well-shaped round leaves, led to a network that was best able to predict leaf centers for all plant types. The final model was trained for 150 epochs with a 80/20 train/validation split for the source (CVPPP) and target datasets, 201 overhead images and masks of the Borage plant type, and evaluated visually on a random sampling of overhead images of all plant types.

The model generates a heatmap with all possible plant leaf centers. A clustering and thresholding technique is used to identify leaf centers with the highest model confidence. These points are then removed and the heatmap is re-normalized to identify less certain points. The algorithm is able to recover lower confidence leaf centers, compared to the initial normalized threshold of 0.3, while accounting for over-classification. The algorithm

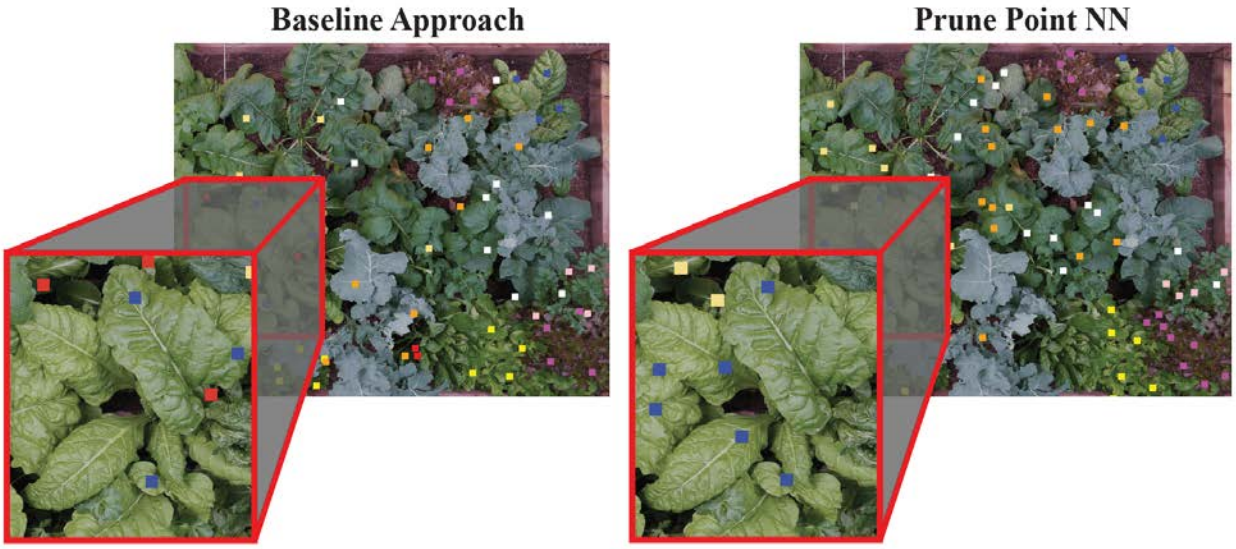


Figure 5.5: **Prune Point Identification.** Example of all plant leaf centers that were identified by the baseline algorithm (left) and the learned model (right) applied to an overhead image. Each prune point color corresponds to a different plant type. The learned model identifies more usable points with fewer misclassifications. When looking at the Swiss Chard plant (zoomed in), we see that the learned model finds 3 more prune points than the baseline approach and also does not missclassify the red prune point, which is meant for a neighboring plant type.

ensures that prune points do not land on other plants or the soil through the use of the phenotyping mask. Together, the model and recursive algorithm identify 32% more leaf centers than the baseline methodology (see Fig. 5.5). The center of mass for the identified points is an average of 38% closer to the center compared to the baseline. Pruning closer to the center of the plant is beneficial because it allows for pruning actions to cut off a greater portion of the leaf. Furthermore, as seen in Fig. 5.5, the learned method has far fewer points that lie on different plants or too close to a leaf edge.

For prune point selection, the network first identifies all possible prune points. The algorithm then eliminates all points within $3cm$ of the edge of the bounding disk, and calculates the rate of change of the radii of all neighboring plants over the last five days. The prune point that is closest to the neighboring plant that has the largest rate of decay of radii is selected in order to foster growth of the struggling neighboring plant as this plant is likely being occluded by the plant the system targets to prune.

5.6 Visual Servoing

The autonomous system must then physically arrive at the chosen prune point by translating from overhead image pixel coordinates to FarmBot (x, y) coordinates. Due to the variable

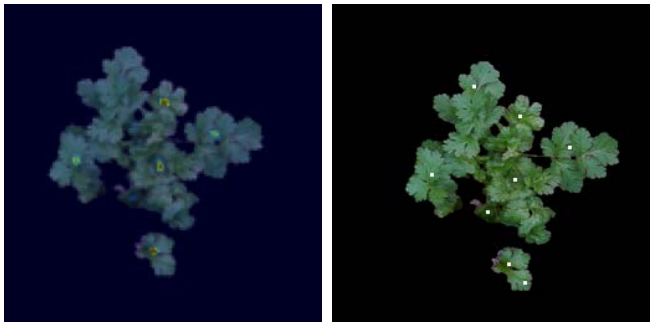


Figure 5.6: **Prune Point Heatmap** Conversion from a raw heatmap of Cilantro leaf centers to points, calculated with the recursive clustering algorithm.

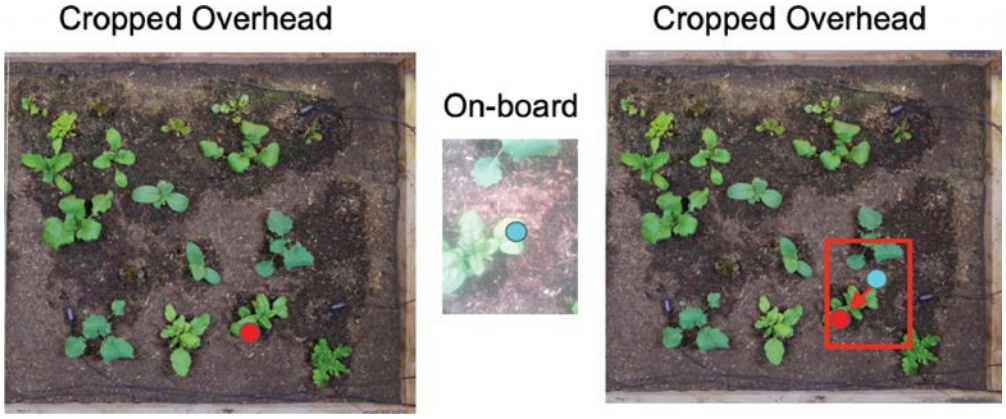


Figure 5.7: **Visual Servoing Left** An overhead image taken from the Sony Sensor marking the target farmbot coordinate in red. **Center** The onboard camera identifies the current location, and cross corelation is used to locate the current location with respect to the overhead image. **Right** An adjustment action is taken until the farmbot reaches it's destination. This is repeated until the distance between the two points are less than a threshold.

height of plants, it is not possible to create a 1-to-1 mapping of pixel coordinates to FarmBot coordinates.

The visual servoing algorithm works using an on-board snake inspection camera located adjacent to the tool end effector on the FarmBot Z-axis [40]. It allows for close-up images of plants and soil. Given plant k was chosen to be pruned, the FarmBot moves to its original seed location and takes a photo using the on-board camera. This image is then localized within the overhead 'global' image by calculating a normalized correlation coefficient between the images. Instead of exhaustively searching the entire garden bed to localize the image, the servoing algorithm constrains the search to a max area around the prune plant's center within the global image, dictated by the FOV of the on-board camera. The algorithm also iteratively tests different scales of the on-board image, which accounts for the variable height

of the canopy, and finds the scale and position that has the highest coefficient.

After finding the best match in the overhead global image, the FarmBot is instructed to move along the vector from the current location to the prune point. Then an iterative process begins, in which a 'local' image is taken at the new point and is localized within the global image. Once localized, the FarmBot moves in the vector direction a max distance of $4cm$ to prevent erroneous movement if a local image is miss-classified within the global image. The iterative cycle continues until the FarmBot reaches within $1cm$ of the prune point or reaches an iteration limit of six.

The location of the maximum correlation coefficient within a constrained region surrounding the approximate coordinates determines the current pixel location of the FarmBot in the overhead global image. Since the scale of the overhead image and the local image are not the same, the normalized correlation coefficient is calculated with the local image resized with various scaling factors. A fixed scaling factor cannot be used between the overhead image and the image because as the plants grow taller the scaling factor changes.

While this servoing method works well near the center of the garden bed, below the overhead camera, we see failure cases around the borders. These failure cases are due to the perspective of the overhead camera and the height of plants. The overhead camera and the on-board cameras view different scenes as the overhead camera does not have the benefit of being directly over every plant. Thus, the algorithm cannot find where the image is located in the overhead image with high certainty.

Chapter 6

Irrigation

6.1 Irrigation Model

AlphaGardenSim uses a discrete-time linear approximation of Richards equation proposed by Tseng et al. [62] to model irrigation actions and soil moisture dynamics. The soil moisture model is defined as follows:

$$w(x, y, t) = \max(w(x, y, t-1) - f + a_w(x, y, t) - u(x, y, t), 0)$$

AlphaGardenSim uses the previous soil moisture content $w(x, y, t-1)$, the amount of irrigation applied $a_w(x, y, t)$, plant water uptake $u(x, y, t)$, and local water loss f to calculate the current soil moisture value for each discrete grid point $p(x, y)$ at time t.

To more accurately model water dynamics in AlphaGardenSim, we conducted physical test bed experiments using soil moisture sensors as shown in fig 6.1. These experiments were used to refine the parameters $a_w(x, y, t)$, $w(x, y, t-1)$, and f in the soil moisture model.

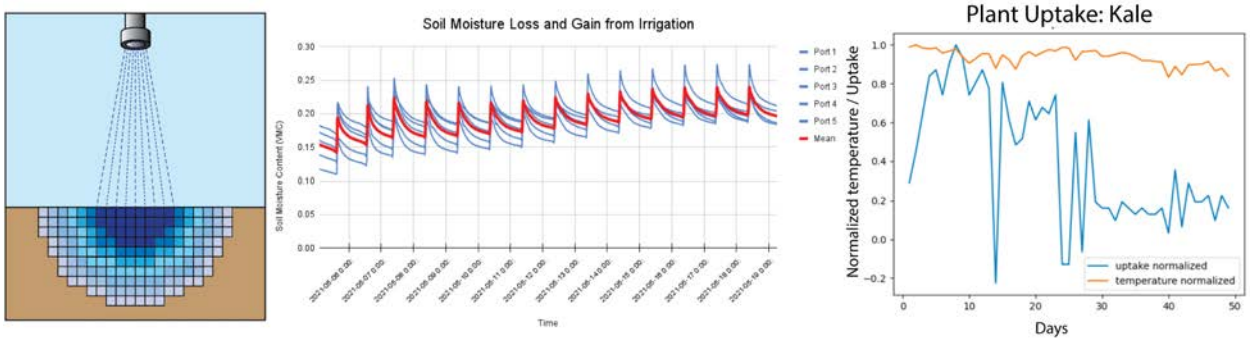


Figure 6.1: **Irrigation Experiments: Left** Water flows radially from the location of irrigation (x, y) The amount of water gain is halved every 0.01m. **Center** Water loss was averaged across 5 sensors. to have a mean of 0.042 m^3/m^3 and a standard deviation of 0.0048 m^3/m^3. **Right** Plant water Uptake shows a relationship between temperature, but conclusive results were not found.

Water applied, $a_w(x, y, t)$ can be directly found using the soil moisture sensors to measure the starting and ending VWC before and after irrigation. Further experiments were conducted to correlate the number of turns of each shrubbler drip emmiter to the outputted irrigation amount. Lastly, human irrigation use was measured using a water flow sensor.

Water Loss, f, By watering at varying frequencies over the TEROS-10 sensors and with a set volume, we were able to plot water loss over time curves. As the simulator operates on a day to day time-scale, the water loss we care to discover is that over one or more days after watering. In an experiment conducted in the physical garden bed, we directly watered $0.200L$ over five independent sensors at the same time every day. Water loss and gain is sampled from a univariate Gaussian calculated from experimental data.

Through the use of the TEROS-10 moisture sensors, we were then able to determine a model for radial flow, or spread, of water once in the soil. To discover this radial flow model, we conducted a set of experiments in which the FarmBot watered at incremental distances from the center of a soil moisture sensor, beginning directly overhead, and ending at 0.10m away. Once outside of the 0.04m radius in which water is applied, the moisture gain is roughly halved at each subsequent 0.01m when compared to the water gain within the radius. Beyond 0.09m, we found no substantial gain. Thus, we found $\Delta w(x_r, y_r) = (1/2)^r * gain$ where r is distance measured in 0.01m outside of the 0.04m radius, (x_r, y_r) is a point $r + 0.04$m away from (x, y), $w(x, y)$ is the soil moisture at point (x, y), and $gain$ is the moisture gain for soil directly under the nozzle.

Plant water uptake, $u(x, y, t)$, was more difficult to evaluate. A set of experiments were run to understand a relationship between plant type, radial size, and plant water uptake. 6 potted plant experiments were conducted, each with soil moisture sensors placed below the surface of the soil. In each pot, one plant was seeded either Kale or Turnip. Two pots were left with no seeds as control. Every 30 minutes the water content and temperature was recorded. Plants were watered for 1 minute everyday at 7am using the shrubbler drip emitters. To compute plant radius we tested using the self supervised data collection as described in Section 5.3. Each day, water uptake was computed as the difference between $w(x, y, t_u)$ - $w(x, y, t_l) - f_t$, where t_u and t_l is the maximum and minimum water content recorded on day t respectively. Water loss f was computed by averaging water loss of the control pots. Ultimately, however, we note that 2 samples of data per plant type were not enough to confidently model plant water uptake, as there are many factors at play including plant stress, temperature, and root mass.

6.2 Irrigation Policy

The analytic policy used in AlphaGardenSim decides on policy actions (watering and pruning) for each plant at every timestep t (in days) as a function of the global variables: soil moisture grid, garden diversity, and plant health. For irrigation, the policy decides on a binary amount of water (0 mL or 200 mL) and in simulation, this is done for every plant, every day. In the original physical AlphaGarden testbed, this was represented by watering

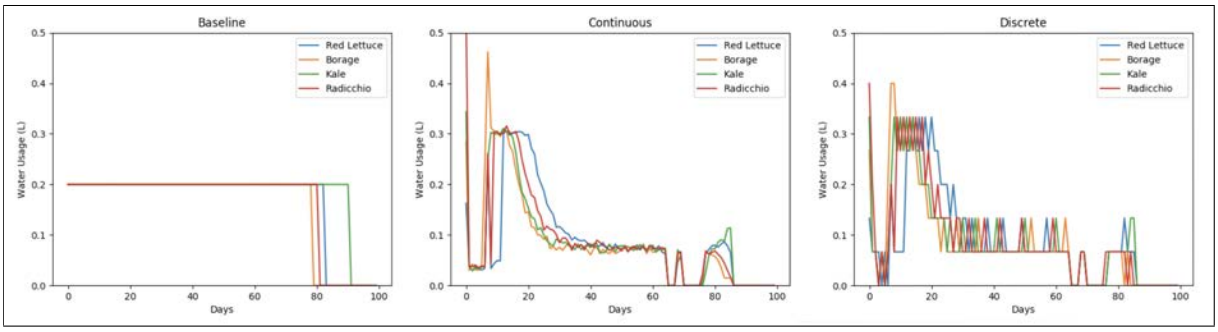


Figure 6.2: **Variable Irrigation Simulation Experiment Results.** In AlphaGardenSim, we compare three irrigation techniques. For each method we compare the average diversity and coverage across day 20 to 70. *Left* The baseline irrigation method waters 0.2L to each plant everyday, using a total of 272.4 L of water and reaching a coverage and diversity of 0.60 and 0.95. *Center* Continuous Variable Irrigation only uses 143.1 L of water and has a coverage and diversity of 0.58 and 0.95. *Right* Discrete Variable Irrigation waters in increments of [0, 66, 132, 200, 266, 332, 400] mL and achieves a coverage and diversity of 0.58 and 0.95, using 143.6L of water.

Life Cycle Stage	**VWC**
Germination	0.2
Vegetative (growth)	0.2
Reproductive (stagnant radii)	0.3
Senescence (decaying radii)	0.2
Death	0.1

Table 6.1: **Assigned Irrigation VWC** (Volumetric Water Content) for the 5 stages of the life cycle in AlphaGardenSim based on maximum possible VWC found in [2].

each plant a set amount each day regardless of state. The AlphaGarden also uses a planting mix, which has different hydraulic properties than typical field soil. Thus, for N plants over L days,

$$\textbf{Max water usage} \approx N \cdot (200 \text{ mL}) \cdot L$$

Instead of the default irrigation type where all plants receive either the same volume or no water at all, variable irrigation allows the amount of water received by a plant to be catered to the plant's water demand, growth stage, water uptake, and growth. This potentially allows the overall reduction of water usage. We modify the analytic policy in AlphaGardenSim to better optimize water usage through variable irrigation. For each plant life stage, we assign a desired water amount. These amounts can be roughly estimated from literature and tuned. We assign values as shown in Table 6.1. We compare the default (binary) implementation of the analytic policy with the variable implementation of it and evaluate water usage. This is done on a simulated 150 cm x 150 cm garden bed with 9 plants placed uniformly in a

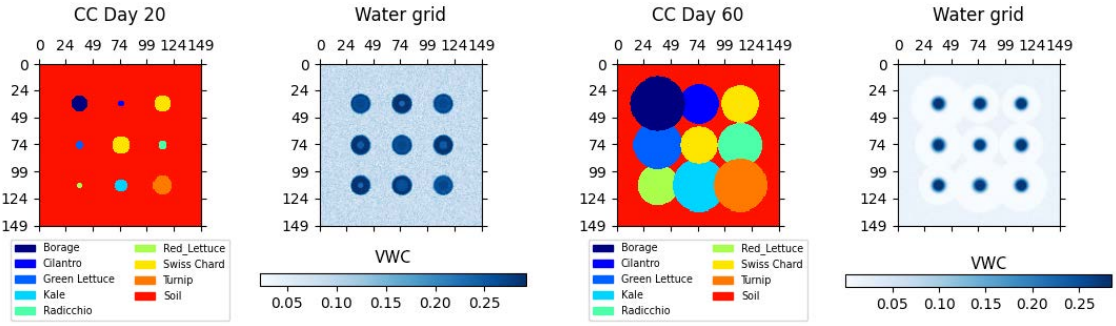


Figure 6.3: **Variable Irrigation Simulation Experiment Overview.** In AlphaGardenSim, we set up experiments to compare the binary/default implementation of the Analytic Policy with this variable implementation of Analytic Policy. The experiment setting is a 150 cm x 150 cm garden bed, 9 plants placed uniformly and a 100 day cycle.

100-day cycle. An overview of this is shown in Figure 6.3. We create two versions of the variable irrigation policy. The first version is allowed to decide any water amount from 0 mL upwards (continuous). The second version chooses from a fixed list of irrigation values (discrete). We model our real world setup with 16 plants in the simulation. Our results show that variable irrigation reduces mean water usage per day and total water usage by over 47% in the continuous case. Similarly, in the discrete case, compared with the baseline analytic irrigation policy, mean water usage per day and total water usage reduces over 47%. Results are shown in Figure 6.2

Variable irrigation can be actuated in a number of ways. One way is through the use of drip emitters. Drip irrigation is a more efficient method of irrigation than hose/nozzle irrigation as it delivers water to a much more precise location - directly to the plant root zone - than humans can. While a human could apply water more forcefully, with drip emitters water infiltrates into the planting mix more evenly as compared to human hand watering.

The slow application process of drip emitters allows the adhesive forces in the planting mix to attract water better leading to less erosion. Moreover, with nozzle irrigation, water can evaporate from the planting mix surface, and when large amounts of water are applied, canals can be formed within the planting mix, leading to non-uniform and lower retention.

Shrubbler drip emitters can be used to vary the water supply to a zone of the garden or to individual plants. Shrubbler drip emitters allow the adjustment of how much water each plant receives by varying the number of turns on the emitter head and for how long water flows through the emitter. More turns are equivalent to a higher flow rate and running for a longer time means each zone or plant receives more water. In moving from sim to real, we take advantage of this unique property of Shrubbler drip emitters and scale the water requirement for each plant from our experiments in simulation to ensure plant growth.

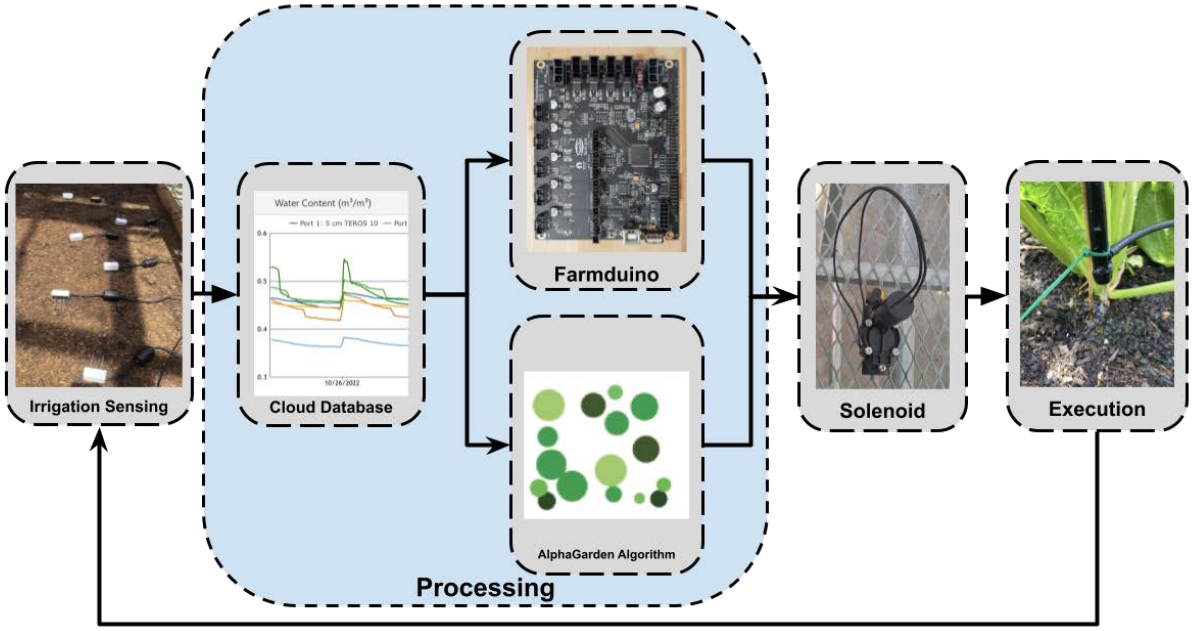


Figure 6.4: **Close Loop Irrigation** The close loop irrigation pipeline consists of three main components: sensing, processing, and execution. The output of the soil moisture sensors are averaged and uploaded to cloud. The arduino loads this data every 30 minutes, and uses an analytical policy to determine how much water to irrigate. Based on this, the ardiuno can open the solenoid to allow for irrigation.

Closed-Loop Irrigation

The goal of variable irrigation is to maintain a range of soil moisture levels required for optimal plant growth. However, this assumes consistent environmental variables such as amount of sunlight, temperature, and humidity. To account for this, we introduce closed-loop irrigation tuned by real-world data collected from a previous cycle. By contrast, in open-loop irrigation, the garden space is watered without any feedback from the garden itself.

The closed-loop system takes input from six TEROS-10 soil moisture sensors [41] which automatically upload soil moisture, measured in terms of volumetric water content (VWC), humidity, and temperature data to a cloud storage every thirty minutes. A server queries these sensors periodically and determines whether or not irrigation is required. If required, the server sends information to the Farmduino system on the Farmbot to turn on irrigation. We connect a solenoid valve to the Farmduino and to the drip emitters.

This solenoid valve turns on when it receives a signal from the Farmduino thus allowing water flow. Currently, this action is determined when the average VWC reading of the 6 soil

moisture sensors is below a threshold. We experimented with two threshold and time pairs: a normal and low metric defined as 0.25 and $0.18L/m^2$; and a normal and low metric defined as 0.25 and $0.21L/m^2$ respectively. In the first scenario, Farmbot only waters once every 24 hours: for 60 seconds for the normal threshold and for 120 seconds for the lower threshold. In the second scenario, Farmbot waters for 343 seconds every 6 hours if the lower threshold is crossed.

Chapter 7

Real World Experiments

7.1 Overview

To evaluate the entire system holistically, we ran eight autonomous cycles over two 60 day periods. We split the garden into two halves and planted identical seed placements ($1.5m \times 1.5m$) on each with different pruning regiments.

- **Cycle 1L and 1R** Cycle 1 analyzed the effects of pruning. The left half was the control (no pruning actions) and the right half was pruned using the rotary pruner.
- **Cycle 2L and 2R** Cycle 2 analyzed the effectiveness of the pruning shears. Both sides were pruned using the shears.
- **Cycle 3L and 3R** Cycle 3 compares Human versus Machine, and introduces an open loop variable irrigation policy.
- **Cycle 4L and 4R** Cycle 4 compares Humans vs Machine, using closed loop irrigation and staggered planting.

Human Intervention Our test compares the pruning and irrigation decisions of Alpha-Garden versus humans. However, on the robot side, at times, human intervention was required in the form of manually cutting prune points, adjusting selected prune points, and adjusting the orientation and depth of the pruning tool as needed.

7.2 Automated Pruning

Each half was treated as an independent garden cycle. Irrigation took place at 9:00 AM daily and every plant was watered $200mL$. After day 30, and every five days after, the autonomous system executed pruning actions. An overhead image taken at 7:00 PM was processed through the Plant Phenotyping and Bounding Disk tracking algorithm to determine the

Figure 7.1: **Top** An image of human pruning and irrigation. **Bottom** An image of the garden testbed from the side showing the farmduino and XYZ gantry system.

garden state. AlphaGardenSim would use this garden state to decide which plants to prune. The image was subsequently used for prune point identification and selection. Visual servoing and pruning algorithms were then executed on the chosen leaves.

Garden Cycles 1L and 1R In Cycles 1L and 1R, the identical seed placements ($1.5m \times 1.5m$) included 20 plants from 10 different plant types (two of each type). In Cycle 1L (the left half of the garden bed) there were no pruning actions and the garden was allowed to grow freely. In Cycle 1R (the right half) pruning actions were executed with the Rotary Pruner.

Over 6 pruning sessions for Cycle 1R, 42 plants were chosen to be pruned across 6 plant types. The system autonomously selected the turnip and kale plants on all pruning occasions, most likely due to the fact that they grew much faster than the other plants and have large radii. Due to the numerous prunings and the Rotary Pruner's nature of completing a cut and leaving a leaf vulnerable, we see both turnip plants approach their wilting stage by day 60. This could also be a sign of overpruning.

Plant Type	r_{max}	**Cycle 1L**	**Cycle 1R**	**% Change**
Kale	37	0.158	0.102	-35.44%
Turnip	33	0.085	0.043	-49.41%
Borage	32	0.122	0.076	-37.70%
Swiss Chard	28	0.105	0.102	-2.86%
Arugula	25	0.098	0.121	23.47%
Radichhio	23	0.034	0.059	73.53%
Red Lettuce	20	0.000	0.057	N/A
Cilantro	19	0.062	0.078	25.81%
Green Lettuce	16	0.028	0.095	239.29%
Sorrel	10	0.002	0.031	1450%
Diversity		0.856	**0.970**	13.32%
Coverage		**0.924**	0.784	-15.15%

Table 7.1: **Plant Type Metrics for Garden Cycles 1L & 1R.** This table shows diversity and coverage for plant types on day 60. The values for Cycle 1L (not pruned) and Cycle 1R (pruned with Rotary Pruner) are calculated via $[c_i(60) * (R/R_i)^2]$ for each plant type (Section III). The goal of pruning is to foster a diverse garden while maintaining a high coverage.

Final canopy coverage and diversity are reported in Table 7.1 for each individual plant type. The metrics were found through creating a manually labeled ground truth mask on day 60 of the garden cycle. As seen by comparing the results with and without pruning, it is clear that pruning increases diversity by creating space for smaller plants to develop. The larger plants coverage decreased while the smaller plants coverage increased, leading to a more diverse garden overall (13.32% increase). This increase in diversity did come at the cost of losing some overall coverage (15.15% decrease).

Garden Cycles 2L and 2R For Garden Cycles 2L (left) and 2R (right), we planted two identical seed placements ($1.5m \times 1.5m$). Cycles 2L and 2R included only 16 total plants from 8 plant types. Sorrel and arugula were omitted as sorrel was relatively much smaller than other plants in the garden and arugula had the tendency to grow too tall, impeding movement of the FarmBot gantry system.

For Cycles 2L and 2R, all pruning actions were performed using the Pruning Shears, and, as before, the two halves were treated independently. During Cycle 2L, 35 plants were chosen for pruning across 6 plant types, while during Cycle 2R, 38 plants were chosen across 7 plant types. We see a decline in the total number of prunings compared to Cycle 1R because of the fewer number of plants in the garden. Kale and borage (two of the largest plants in the garden) were most commonly selected in both garden cycles. No plants exhibited signs of wilting or overpruning by day 60.

To evaluate Garden Cycles 2L and 2R relative to Cycles 1L and 1R, we manually created

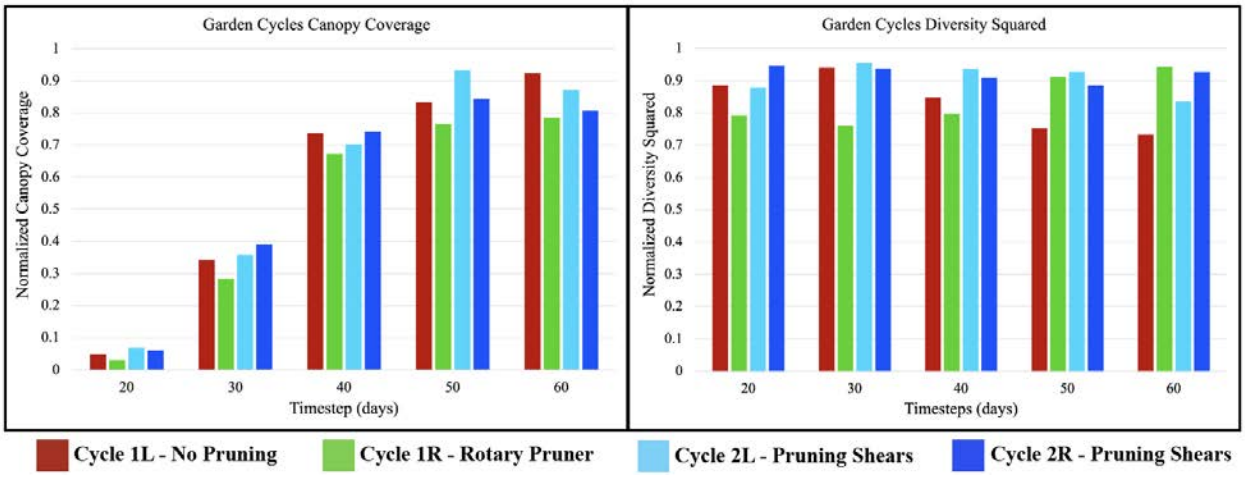


Figure 7.2: **Garden Cycle Comparison.** Data points were recorded for days 20, 30, 40, 50, and 60 through hand labeled phenotyping masks. **Left:** Comparison of the coverage of the 4 Garden Cycles. The non-pruned garden has the highest value by day 60, with Cycle 2L (pruning shears) not far behind. **Right:** Comparison of the diversity squared of the 4 Garden Cycles. The non-pruned garden had lowest diversity by day 60, and Cycles 1R (rotary pruner) and 2R (pruning shears) achieved the highest diversity.

segmentation masks for days 20, 30, 40, 50, and 60. Fig. 7.2 shows coverage and diversity graphs for all four garden cycles. We found the autonomous system to achieve an average of 0.94 normalized diversity with the Pruning Shears for Cycles 2L and 2R on day 60, and an average canopy coverage of 0.84. While the Rotary Pruner exhibited a higher diversity metric (0.97), the Pruning Shears outperformed the non-pruned garden, Cycle 1L, in terms of diversity (0.85) while sacrificing much less coverage than the Rotary Pruner, which had a final coverage of 0.78. Cycle 2L achieved significantly more coverage (10.7% more) during day 50 than Cycle 2R, which could be in part due to the greater number of prunes of Cycle 2R.

In general, the Pruning Shears executed much cleaner cuts than the Rotary Pruner and sacrificed less total canopy coverage. To try to match the effectiveness of the Rotary Pruner in terms of diversity for future gardens, the Pruning Shears could make multiple cuts per plant or could prune more frequently than every 5 days.

7.3 Robot Vs Human Comparison

Experimental Setup

We ran two 60-day cycles. Cycle 3 ran from April 15 to June 14, and cycle 4 ran from July 2 to August 31. For each garden space of 1.5m x 1.5m, we plant sixteen plants (two each for eight types): kale, borage, swiss chard, turnip, radicchio, green lettuce, cilantro, and red lettuce, using the same seed placement arrangement mirrored. Thus there are thirty-two

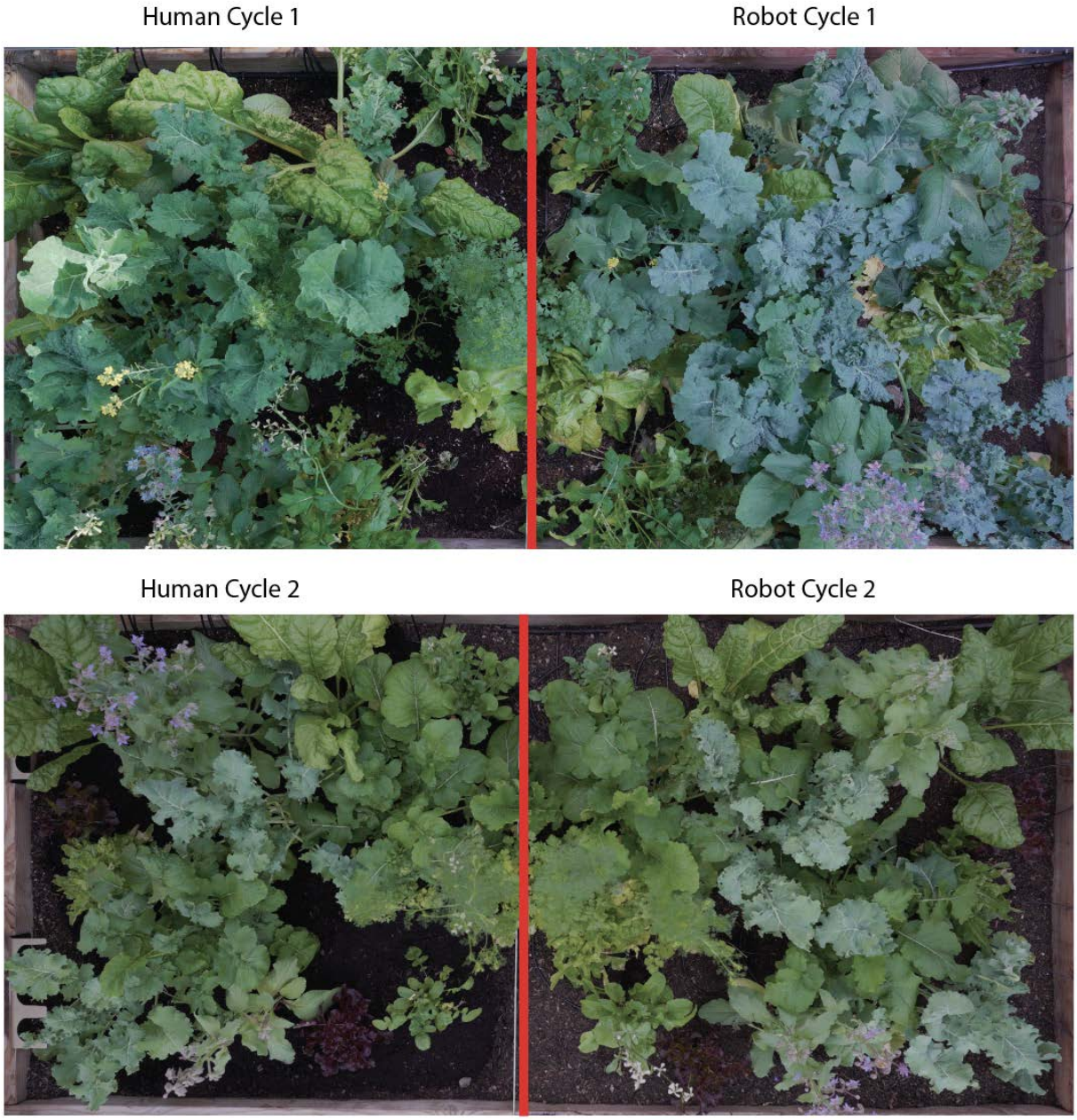


Figure 7.3: **Four gardens at day 60.** A side-by-side comparison of four 1.5m by 1.5m real gardens planted with mirrored identical seed arrangements (mirrored across the white string in the middle). In both 60-day cycles, the left half was tended by human experts, while the right half was tended by the AlphaGarden robot system. Coverage and Diversity on Day 60 are comparable. The AlphaGarden consumed 37% and 44% less water, respectively.

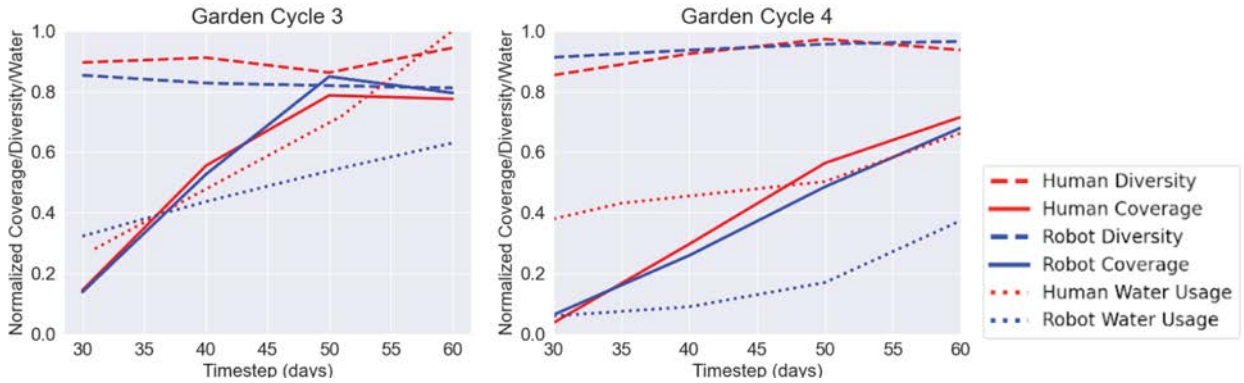


Figure 7.4: **Performance of Four Physical Gardens.** The graphs show the diversity, coverage, and water usage of four 60 day gardens with the same initial seed placement. We compute metrics starting at day 30. In each garden cycle, one side of the plant bed was tended by the robot and the other by expert gardeners. Irrigation is normalized to be between $[0, 1]$ by dividing by maximum total water usage of $413.5L$.

plants in total. After germination, there is the opportunity to thin, transplant, or replace plants depending on germination success.

The human side is watered at the discretion of the horticulturalists using a nozzle hose fitted with a water flow meter and pruned based on their knowledge. Water usage, observations, and pruning actions are logged. The robot side is watered using drip emitters of the Shrubbler type installed above each plant. As discussed in Section V, Shrubbler emitters can be adjusted to supply different amounts of water based on the turns on their heads. We tested the drip emitters to measure their flow rates, zones of influence, and VWC gain/loss as measured by the TEROS-10 sensors. We adjusted the emitter settings accordingly allocating one drip emitter to every plant on the robot side. Specifics are described under each cycle below.

A water flow meter is also attached to the pipe supplying water to the drip emitters. Water added to both sides of the garden during germination, transplanting, or thinning of the plants is excluded from the total water usage comparison.

Pruning starts anytime after Day 30 and runs till the end of the cycle. Peak growth periods are from Days 40 - 50.

To analyze the garden state, we label each plant type and compute diversity and coverage metrics every 10 days. Figure 7.1 shows an expert horticulturalist pruning and watering one side of the garden and the emitters irrigating the robot side.

Results

Our test compares the pruning and irrigation decisions of AlphaGarden versus humans. However, on the robot side, at times, human intervention was required in the form of manu-

ally cutting prune points, adjusting selected prune points, and adjusting the orientation and depth of the pruning tool as needed.

The 'human' side of the experiment was tended by UC Berkeley Oxford Tract Greenhouse staff who have on average, 10 years of professional horticultural experience.

Cycle 3

Germination Of the thirty-two plants locations, no radicchio, green lettuce, and red lettuce germinated on either side. Green lettuce and red lettuce were replaced from local nurseries with seedlings of the same plant types in an early growth stage. Radicchio was difficult to find and was replaced with arugula seedlings, which have a similar growth pattern. One turnip plant germinated on the robot side but none germinated on the human side. Turnip was replaced with mustard green seedlings on both sides of the garden. Cilantro germinated on the human side but not on the robot side. Two excess cilantro sprouts were moved from the human side to the robot side. All other plants germinated. However, one swiss chard on the robot side germinated but was stunted throughout the entire cycle. It was later found out that some of the seeds used in this cycle had been in storage for more than a year and this may have affected their viability. No additional thinning was carried out.

Irrigation On the human side, the garden was watered roughly every 1-2 days. On the robot side, open-loop irrigation took place at 7:00 AM daily for 60 seconds. The entire cycle was divided into three periods depending on the plants' estimated lifecycle: germination (16 days), early growth (19 days), and mature growth (25 days). Plants are divided into two groups based on their perceived water needs. Group 1 consisted of kale, borage, swiss chard, and turnip while group 2 consisted of radicchio, green lettuce, cilantro, and red lettuce. During germination, the emitters above group 1 plants were set to seven (7) turns which gave about 284 mL while the emitters above group 2 plants were set to six (6) turns which gave about 191 mL totaling about 3.8L per day. During early growth, the emitters above group 1 plants were set to eight (8) turns which gave about 383 mL while the emitters above group 2 plants were set to seven (7) turns which gave about 284 mL totaling about 5.33L per day. During mature growth, the emitters were returned to the same setting as in germination.

Pruning Automated pruning actions on the robot side started on day 47 and were executed with an interval of 3 days between pruning sessions. AlphaGardenSim autonomously determines what plants and what leaves to prune, utilizing the pruning pipeline as described in Chapter 5. Pruning actions were then executed by a human and/or the Farmbot. There were a total of four pruning sessions, nine plants were selected by AlphaGardenSim and were pruned.

Cycle 3 Results					
	Irrigation	Pruning	Coverage	Diversity	Water Use (L)
Human	Every 1-2 days	Every 1-2 days	0.78	0.94	413.5
Robot	Open-loop	4 pruning sessions	0.81	0.90	260.6

Table 7.2: **Cycle 3: Human Vs. Robot** Irrigation, Pruning, Coverage, Diversity and Water Use for the human and robot side for cycles 3.

Cycle 4 Results					
	Irrigation	Pruning	Coverage	Diversity	Water Use (L)
Human	Every 1-2 days	Every 1-2 days	0.72	0.94	274.1
Robot	Close-loop at 7am	4 pruning sessions	0.67	0.97	154.8

Table 7.3: **Cycle 4: Human Vs. Robot** Irrigation, Pruning, Coverage, Diversity and Water Use for the human and robot side for cycles 4.

Coverage & Diversity We found final coverage and diversity of 0.81 and 0.80, respectively, while using 260.6 liters of water on the robot side while the human side attained a coverage of 0.78 and a diversity of 0.94 and water usage of 413.5 liters. Figure 7.4 shows this. AlphaGarden achieved comparable diversity and coverage with 37% less water.

Cycle 4

Germination In cycle 2, we implemented staggered planting. Four plant types known to be slower-growing, cilantro, green lettuce, red lettuce, and radicchio, were planted from Day 1. Four plant types known to be faster-growing, kale, borage, swiss chard, and turnip, were planted from Day 11. Of the thirty-two plants locations, plants germinated in twenty-four locations. Similar to cycle 1, we transplanted and purchased plants for the locations without germination. As in Cycle 1, it was difficult to find radicchio seedlings of comparable growth and so it was replaced completely with arugula which has similar growth tendencies. All other plants including all of the faster-growing plants germinated. Manual thinning was carried out on Day 28 leaving all locations with 1-3 plants each.

Irrigation The irrigation schedule on the human side was similar to Cycle 1 with the garden watered roughly every 1-2 days. However, on the robot side, a number of changes were made. First, the same drip emitter turn settings were used throughout the plants' lifecycle. Group 1 plants, i.e. kale, borage, swiss chard, and turnip had their emitters set to seven (7) turns while group 2 plants i.e. radicchio, green lettuce, cilantro, and red lettuce had their emitters set to six (6) turns. The emphasis was to regulate how much water the

garden received by how long the flow was. For 45 days in the cycle, open-loop irrigation took place at 7:00 AM daily for 60 seconds. From day 46 to day 60, we tested out closed-loop irrigation with two different threshold and time pairs: from day 46 to day 50, we used a normal and low metric 0.25 and $0.18L/m^2$, watered for 60 seconds for the normal threshold and 120 seconds for the lower threshold and watered once every 24 hours. From day 50 to day 60, we used a normal and low metric of 0.25 and $0.21L/m^2$, respectively, and watered for 60 seconds every 24 hours for the normal threshold and for 343 seconds every six hours for the lower threshold.

In Cycle 2, the flow meter attached to the robot side malfunctioned and did not accurately record water use. Therefore we only have water usage estimates based on the irrigation schedule. Similarly, on the human side, the flow meter was erroneously reset on Day 47 and some water usage data was lost. As a result, we can only provide a lower bound on how much the human side used in Cycle 2. We estimate the human side used at least 274.1 Liters while the robot side used a total of 154.8 Liters: 42 Liters (Day 1-46), 9 Liters (Day 46-50), and 103.8 Liters (Day 51-60).

Pruning Automated pruning actions on the robot side were started on day 30 and executed with an average interval of 5 days between pruning sessions. Similarly, AlphaGardenSim autonomously determined what plants and what leaves to prune, utilizing the pruning pipeline as described above. Pruning actions were then executed by a human and/or the Farmbot. There were a total of six pruning sessions during which 45 plants were selected by AlphaGardenSim and 30 were pruned. The remaining 15 were either missing prune points or there was another error in the pipeline.

Coverage & Diversity We found final coverage and diversity of 0.67 and 0.97 respectively while using 154.8 liters of water on the robot side while the human side had a coverage of 0.72 and a diversity of 0.94 with at least 274.1 Liters. In this case, AlphaGardenSim achieved comparable diversity and coverage while using at least 44% less water. Tables 7.2 and 7.3 presents a summary of the physical experiment cycles.

Chapter 8

Limitations

This thesis proposes an autonomous system for polyculture gardening using simulation and real world experiments. The results in Chapter 7 show that in a controlled environment, this system can achieve high yield and diversity, while minimizing water use. However, in order to do so, this thesis made several assumptions and simplifications.

One of the assumptions made was that AlphaGardenSim could allow for fast and rapid learning of policies based on garden dynamics. While this proved true, the modeling of plants as a center and radius only provides a first-order approximation of natural plant growth, which is rarely a perfect circle. Moreover, AlphaGardenSim only models some dynamics, while other factors such as temperature, seasonality, soil nutrient quality, and more play a role in plant growth, giving us only part of the picture.

Another limitation of the proposed system is the laborious hand labeling of data required for the computer vision pipeline for each plant type. While a self-supervised system proposed in Chapter 5 could potentially reduce this overhead, it has yet to be tested and relies on the performance of off-the-shelf object detection models like Meta's Segment Anything Model. Additionally, this algorithm is only effective in the early stages of the plant cycle before occlusion, and it may not scale to many plant types.

Although the goal of this project was to create a scalable and sustainable automation system for polyculture gardening, the real-world experiments were conducted using custom physical hardware within a greenhouse, which may not be feasible for all users due to cost. The use of cameras, soil moisture sensors, and a farmbot gantry system also adds to the expense. Additionally, it has not been tested how this system and machine learning models will generalize to other hardware.

In the AlphaGarden system, while the irrigation actions could be carried out autonomously, pruning actions required human supervision. This means that a member of the team would need to be present at every pruning event. As well as this actions such as seed planting and soil replacement were done manually before each garden cycle. However, it is important to note that the system still offers significant benefits in terms of reducing labor requirements and increasing water use efficiency.

Chapter 9

Conclusion

This thesis answers the question: "Can machines garden?" - the answer is yes.

AlphaGarden is a real-sim-real system that automates polyculture farming. We have created an autonomous system which can image, interpret, model and act to support farming. To do so, AlphaGarden breaks this problem into three key stages: intepreting and evaluating a real world state (Real to Sim), learning policies in simulation, and enacting these policies in the real world (Sim to Real). Researched dynamics models are tuned with real world data collected from plant growth and irrigation experiments. This simulator is used to experiment and test various policies such as learned pruning, dynamic planting, and variable irrigation. A computer vision pipeline was created to translate sensor information to a simulator representation, using computer vision, optimization, and data aggregation techniques. Last, a recursive neural network, visual servoing, and cloud computing are used to enact irrigation and pruning policies. These policies were then evaluated across 8 garden cycles achieving high coverage and diversity, while reducing water usage by 44%. However, we are still far from automating polyculture farming at scale.

This thesis shows that the systems that comprise AlphaGarden can be used in parts or as a whole to aid in transforming agriculture. Simulators like AlphaGardenSim can be used to learn optimal policies beyond those presented in this paper such as: optimizing dynamic planting for companionship and soil nutrients, or learning harvesting policies to optimize plant yield. Ongoing work looks at taking a closer look at plant growth and modeling in 3D to learn better prune points.

While AlphaGarden looks at the feasibility of end to end automation, I am excited about using individual components to aid farmers at scale in the near future. Drone imagery and neural network models can be used to identify plant health, when to harvest and more. Soil moisture sensors and companion planting can be utilized to reduce water consumption. Taking a look at the big picture, I am excited to see how the current system can be generalized to new domains such as improving global forestry through tree planting and monitoring.

For code, videos, and datasets for the AlphaGarden project, see:
https://github.com/BerkeleyAutomation/AlphaGarden.

Closing Remarks

As I write this final chapter of my thesis, I am simultaneously closing the chapter of my time at Berkeley. Being a part of the 5th Year MS program has been an incredible experience, and I am grateful for the invaluable lessons and memories I have made here. Over the past three years, as a member of AutoLab, I have had the opportunity to delve into various fields such as computer vision, reinforcement learning, and automation. These experiences will stay with me as I prepare to leave Berkeley and enter the real world. I see it only fit to end my thesis the same way Alan Turing ended his paper: "We can only see a short distance ahead, but we can see plenty there that needs to be done." Although I can only see what's immediately in front of me, I am excited for the journey ahead.

Bibliography

[1] Yahav Avigal et al. "Learning Seed Placements and Automation Policies for Polyculture Farming with Companion Plants". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 902–908. DOI: 10.1109/ICRA48506.2021.9561431.

[2] Yahav Avigal et al. "Simulating Polyculture Farming to Learn Automation Policies for Plant Diversity and Precision Irrigation". In: *IEEE Transactions on Automation Science and Engineering* 19.3 (2022), pp. 1352–1364. DOI: 10.1109/TASE.2021.3138995.

[3] Mark Presten et al. *Automated Pruning of Polyculture Plants*. 2022. arXiv: 2208.10472 [cs.RO].

[4] Simeon* Adebola et al. "Can Machines Garden? Systematically Comparing the AlphaGarden vs. Professional Horticulturalists". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023.

[5] Stephen J Risch. "Intercropping as cultural pest control: prospects and limitations". In: *Environmental Management* 7.1 (1983), pp. 9–14.

[6] Timothy E Crews, Wim Carton, and Lennart Olsson. "Is the future of agriculture perennial? Imperatives and opportunities to reinvent agriculture by shifting from annual monocultures to perennial polycultures". In: *Global Sustainability* 1 (2018).

[7] S Gliessman, M Altieri, et al. "Polyculture cropping has advantages". In: *California Agriculture* 36.7 (1982), pp. 14–16.

[8] Matt Liebman. "Polyculture cropping systems". In: *Agroecology*. CRC Press, 2018, pp. 205–218.

[9] Sven Erik Jorgensen and Brian D Fath. *Encyclopedia of ecology*. Newnes, 2014.

[10] Fusuo Zhang and Long Li. "Using competitive and facilitative interactions in intercropping systems enhances crop productivity and nutrient-use efficiency". In: *Plant and soil* 248.1-2 (2003), pp. 305–312.

[11] NA Bogie et al. "Intercropping with two native woody shrubs improves water status and development of interplanted groundnut and pearl millet in the Sahel". In: *Plant and soil* 435.1-2 (2019), pp. 143–159.

[12] Teja Tscharntke et al. "Multifunctional shade-tree management in tropical agroforestry landscapes–a review". In: *Journal of Applied Ecology* 48.3 (2011), pp. 619–629.

[13] Todd S Rosenstock et al. "Agriculture's contribution to nitrate contamination of Californian groundwater (1945–2005)". In: *Journal of Environmental Quality* 43.3 (2014), pp. 895–907.

[14] Alberto Mantovani. *Pesticide risk assessment: European framework shows need for safer alternatives*. 2019. URL: https://www.openaccessgovernment.org/pesticide-risk-assessment/79226/ (visited on 12/11/2019).

[15] David Tilman et al. "Agricultural sustainability and intensive production practices". In: *Nature* 418.6898 (2002), pp. 671–677.

[16] Yahav Avigal et al. "Simulating Polyculture Farming to Tune Automation Policies for Plant Diversity and Precision Irrigation". In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2020, pp. 238–245.

[17] Jonathan Minchin. *ROMI: Robotics for Microfarming*. 2020. URL: https://www.openaccessgovernment.org/romi-robotics-for-microfarming/80533/ (visited on 01/14/2020).

[18] BOWERY FARMING INC. *Bowery farming*. 2020. URL: https://boweryfarming.com/ (visited on 10/15/2020).

[19] Florian Thomas Payen et al. "How Much Food Can We Grow in Urban Areas? Food Production and Crop Yields of Urban Agriculture: A Meta-Analysis". In: *Earth's Future* 10.8 (2022). e2022EF002748 2022EF002748, e2022EF002748. DOI: https://doi.org/10.1029/2022EF002748. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2022EF002748. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2022EF002748.

[20] Erica Dorr et al. "Environmental impacts and resource use of urban agriculture: a systematic review and meta-analysis". In: *Environmental Research Letters* 16.9 (Aug. 2021). Publisher: IOP Publishing, p. 093002. ISSN: 1748-9326. DOI: 10.1088/1748-9326/ac1a39. URL: https://doi.org/10.1088/1748-9326/ac1a39 (visited on 09/01/2022).

[21] M. Verbeek and B. Hardeweg. "From consumer to prosumer: Are small-scale home indoor farms economically viable?" In: *European Journal of Horticultural Science* 87.3 (June 2, 2022). Publisher: International Society for Horticultural Science (ISHS), Leuven, Belgium, pp. 1–12. ISSN: 1611-4434. DOI: 10.17660/eJHS.2022/031. URL: https://doi.org/10.17660/eJHS.2022/031.

[22] TjeerdJan Stomph et al. "Designing intercrops for high yield, yield stability and efficient use of resources: Are there principles?" In: *Advances in Agronomy*. Vol. 160. 1. Elsevier, 2020, pp. 1–50.

[23] James W Jones et al. "The DSSAT cropping system model". In: *European journal of agronomy* 18.3-4 (2003), pp. 235–265.

[24] Pasquale Steduto et al. "AquaCrop—The FAO crop model to simulate yield response to water: I. Concepts and underlying principles". In: *Agronomy Journal* 101.3 (2009), pp. 426–437.

[25] B Murdyantoro, D Sukma Eka Atmaja, and H Rachmat. "Application Design of Farmbot based on Internet of Things (IoT)". In: *IJASEIT* 9 (2019).

[26] Nikolaus Correll et al. "Building a distributed robot garden". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2009, pp. 1509–1516.

[27] Tom Botterill et al. "A robot system for pruning grape vines". In: *Journal of Field Robotics* 34.6 (2017), pp. 1100–1122.

[28] Francisco Ángel Luna Hernández, Mayra Hernández Oramas, and Victor Manuel Arias Peregrino. "Autonomous Urban Garden". In: *International Journal of Advanced Networking and Applications* 11.3 (2019), pp. 4277–4282.

[29] E.J. Van Henten et al. "Field Test of an Autonomous Cucumber Picking Robot". In: *Biosystems Engineering* 86.3 (2003), pp. 305–313. ISSN: 1537-5110. DOI: https://doi.org/10.1016/j.biosystemseng.2003.08.002. URL: https://www.sciencedirect.com/science/article/pii/S1537511003001557.

[30] S Hayashi et al. "Robotic harvesting system for eggplants [Solanum melongena] trained in V-shape, 2: Harvesting experiment for eggplants". In: *Journal of Society of High Technology in Agriculture (Japan)* (2003).

[31] Shigehiko Hayashi et al. "Evaluation of a strawberry-harvesting robot in a field test". In: *Biosystems Engineering* 105.2 (Feb. 1, 2010), pp. 160–171. ISSN: 1537-5110. DOI: 10.1016/j.biosystemseng.2009.09.011. URL: https://www.sciencedirect.com/science/article/pii/S1537511009002797 (visited on 09/08/2022).

[32] Sam Corbett-Davies et al. "An Expert System for Automatically Pruning Vines". In: New York, NY, USA: Association for Computing Machinery, 2012. ISBN: 9781450314732. DOI: 10.1145/2425836.2425849. URL: https://doi.org/10.1145/2425836.2425849.

[33] Sagar Maitra et al. "Intercropping—A Low Input Agricultural Strategy for Food and Environmental Security". In: *Agronomy* 11.2 (Feb. 2021), p. 343. ISSN: 2073-4395. DOI: 10.3390/agronomy11020343. URL: http://dx.doi.org/10.3390/agronomy11020343.

[34] *The State of Food and Agriculture 2020*. 2020. FAO, 2020. 210 pp. ISBN: 978-92-5-133441-6. DOI: 10.4060/cb1447en. URL: http://www.fao.org/documents/card/en/c/cb1447en (visited on 05/04/2021).

[35] Juan F. Velasco-Muñoz et al. "Sustainable Irrigation in Agriculture: An Analysis of Global Research". In: *Water* 11.9 (2019). ISSN: 2073-4441. URL: https://www.mdpi.com/2073-4441/11/9/1758.

[36] Zhen Li et al. "Closed-loop drip irrigation control using a hybrid wireless sensor and actuator network". In: *SCIENCE CHINA Information Sciences* 54 (Mar. 2011), pp. 577–588. DOI: 10.1007/s11432-010-4086-6.

[37] FarmBot. *FarmBot*. 2020. URL: https://farm.bot/ (visited on 10/26/2020).

[38] Sony. *SNC-VB770 Ultra High Sensitivity 4K Network Camera*. 2021. URL: https://pro.sony/enEE/products/specialised-cameras/snc-vb770 (visited on 09/03/2021).

[39] Sony. *FE 20mm F1.8 G Full-frame Large-aperture Ultra-wide Angle G Lens*. 2021. URL: https://electronics.sony.com/imaging/lenses/all-e-mount/p/sel20f18g (visited on 09/03/2021).

[40] FarmBot. *Electronics and Wiring*. 2021. URL: https://genesis.farm.bot/v1.5/Extras/bom (visited on 09/03/2021).

[41] Meter Group. *TEROS 10 Simple Soil Moisture Sensing*. 2021. URL: https://www.metergroup.com/environment/products/teros-10/ (visited on 09/08/2021).

[42] "Meter Group". *ZL6 advanced cloud data logger*. 2021. URL: https://www.metergroup.com/environment/products/zl6-data-logger/ (visited on 09/08/2021).

[43] Pololu. *Pololu Carrier with Sharp GP2Y0A60SZLF*. 2021. URL: https://www.pololu.com/product/2474 (visited on 09/03/2021).

[44] Niwaki. *Sentei Topiary Clippers*. 2021. URL: https://www.niwaki.com/sentei-topiary-clippers (visited on 09/08/2021).

[45] William J Price, Bahman Shafii, and Donald C Thill. "An individual-plant growth simulation model for quantifying plant competition". In: (1994).

[46] Cornelis Teunis de Wit. *Photosynthesis of leaf canopies*. Tech. rep. Pudoc, 1965.

[47] Uta Berger et al. "Competition among plants: concepts, individual-based modelling approaches, and a proposal for a future research strategy". In: *Perspectives in Plant Ecology, Evolution and Systematics* 9.3-4 (2008), pp. 121–135.

[48] Theodore C Hsiao. "Effects of drought and elevated CO 2 on plant water use efficiency and productivity". In: *Interacting stresses on plants in a changing climate*. Springer, 1993, pp. 435–465.

[49] Andrew Keller. "Evapotranspiration and crop water productivity: making sense of the yield-ET relationship". In: *Impacts of Global Climate Change*. 2005, pp. 1–11.

[50] Babette Dellen, Hanno Scharr, and Carme Torras. "Growth signatures of rosette plants from time-lapse video". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 12.6 (2015), pp. 1470–1478.

[51] Argyris Zardilis, Alastair Hume, and Andrew J Millar. "A multi-model framework for the Arabidopsis life cycle". In: *Journal of experimental botany* 70.9 (2019), pp. 2463–2477.

[52] Tadaki Hirose, Toshihiko Kinugasa, and Yukinori Shitaka. "Time of flowering, costs of reproduction, and reproductive output in annuals". In: *Reproductive allocation in plants*. Elsevier, 2005, pp. 159–188.

[53] Peter John Lumsden and Andrew J Millar. *Biological rhythms and photoperiodism in plants*. Bios Scientific Publishers, 1998.

[54] Pinetree Garden Seeds. *Pinetree Garden Seeds - Vegetable Collections*. 2020. URL: `https://www.superseeds.com/` (visited on 10/15/2020).

[55] Thomas Brox Olaf Ronneberger Philipp Fischer. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv preprint arXiv:1505.04597* (2015).

[56] He Kaiming et al. "Deep Residual Learning for Image Recognition". In: *arXiv preprint arXiv:1512.03385* (2015).

[57] Craig W Whippo and Roger P Hangarter. "Phototropism: bending towards enlightenment". In: *The Plant Cell* 18.5 (2006), pp. 1110–1119.

[58] Daniela Dietrich. "Hydrotropism: how roots search for water". In: *Journal of experimental botany* 69.11 (2018), pp. 2759–2771.

[59] Haichun Yang et al. "CircleNet: Anchor-free Detection with Circle Representation". In: *CoRR* abs/2006.02474 (2020). arXiv: `2006.02474`. URL: `https://arxiv.org/abs/2006.02474`.

[60] Alexander Kirillov et al. *Segment Anything*. 2023. arXiv: `2304.02643 [cs.CV]`.

[61] Tewodros W. Ayalew, Jordan R. Ubbens, and Ian Stavness. "Unsupervised Domain Adaptation For Plant Organ Counting". In: *CoRR* abs/2009.01081 (2020). arXiv: `2009.01081`. URL: `https://arxiv.org/abs/2009.01081`.

[62] David Tseng et al. "Towards automating precision irrigation: Deep learning to infer local soil moisture conditions from synthetic aerial agricultural images". In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2018, pp. 284–291.