Design and Implementation of Physical Experiments for Evaluation of the AlphaGarden: an Autonomous Polyculture Garden

Mark Presten
Ken Goldberg, Ed.

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-95
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-95.html

May 13, 2022

Design and Implementation of Physical Experiments for Evaluation of the AlphaGarden: an Autonomous Polyculture Garden

by Mark Presten

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 2022

* * * * * * *

Professor Lisa Yan
Second Reader

13 May 2022

Abstract

Design and Implementation of Physical Experiments for Evaluation of the AlphaGarden: an Autonomous Polyculture Garden

by

Mark Presten

Masters of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Ken Goldberg

Polyculture farming — where multiple crop species are grown simultaneously — has potential to reduce pesticide and water usage while improving the utilization of soil nutrients. However, it is much harder to automate polyculture than monoculture. This report presents two contributions to the research and development of polyculture farming, with the first being AlphaGardenSim [6, 4, 5]: a fast, first order, open-access polyculture farming simulator with single plant growth and irrigation models tuned using real world measurements. AlphaGardenSim can be used for policy learning as it simulates inter-plant dynamics, including light and water competition between plants in close proximity and approximates growth in a real greenhouse garden at 25,000× the speed of natural growth. We discuss the development of the simulator, the models used for growth, light, and irrigation, real-to-sim model tuning methods, policies trained in simulator, and metrics and results for simulated garden cycles.

The latter half of this report presents AlphaGarden: an automated system for pruning and irrigating living plants in a physical testbed that uses policies in AlphaGardenSim to decide real-time actions. This system utilizes novel hardware and algorithms for automated pruning. Using an overhead camera to collect data from a physical 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 prune. A trained neural network detects and targets specific prune points on the plant. Two custom-designed pruning tools, compatible with a FarmBot [19] gantry system, are experimentally evaluated and execute autonomous cuts through controlled algorithms. We show results for four 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 by the end of the cycles. In ongoing work, we optimize water usage and also compare the AlphaGarden system to a human gardener.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank first and foremost my research advisor Professor Ken Goldberg for his guidance, support, and for giving me the opportunity and resources to research the AlphaGarden project within UC Berkeley's AUTOLAB. I would also like to thank Yahav Avigal for his mentorship within the AlphaGarden project. To my co-authors and research colleagues within AUTOLAB, thank you for making this work possible.

The work presented in this paper comes from multiple publications, a paper under review, and on-going research projects. Namely, this paper combines contributions from [6, 4, 5] alongside the paper under review for IEEE International Conference on Automation Science and Engineering (CASE) 2022 titled "Automated Pruning of Polyculture Plants" by Presten et al. Ongoing projects are discussed at the end of the report. I worked directly on the growth and irrigation models in the simulator, the tuning of said models, and the experimentation and evaluation of several policies. I led the effort in implementing the autonomous system and carrying out real-world experiments.

Chapter 1

Introduction

For over 10,000 years, the practice of plant cultivation has been integral to human civilization. Many factors influence the quality and quantity of plant growth, such as irrigation, pesticide use, weather conditions, and plant disease. Industrial agriculture aims to maximize yield by growing a single plant species in isolation, also known as monoculture farming. Polyculture farming, on the other hand, imitates the diversity of natural ecosystems by simultaneously growing different crops, and is a sustainable alternative that uses biodiversity to reduce pesticides, disease, and weeds [21, 33, 11]. Polyculture is also more practical for confined urban spaces and essential for aesthetic gardens.

However, polyculture farming is more laborious than monoculture, requiring maintenance to ensure that larger, more dominant plant types do not overwhelm smaller, slow-growing plants. Furthermore, the inherent layout of polyculture farming makes non-invasive autonomous cultivation difficult due to the close proximity of each plant. A robot with a reliable and sustainable control policy has the potential to increase yield and diversity and reduce water consumption.

Finding an optimal policy is a challenging task. First, the long time constants for real-world experiments motivates the use of a simulated environment. Second, it is difficult to simulate inter-plant dynamics, including competition for light, water and nutrients. Third, in order to enact a learned policy in a real world environment, an autonomous robotic system must be created that is able to recognize and parse the state of a garden as well as accurately complete pruning and irrigation actions.

This thesis presents the research conducted for the AlphaGarden project out of AUTOLAB [3]. It presents the methods and results that were showcased in the publications [6, 4, 5] centered around AlphaGardenSim: a fast, first order, open-access polyculture garden simulator. To the best of our knowledge, this is the first polyculture garden simulator that simulates inter-plant dynamics and competition for resources for policy learning. Chapters 4 and 5 explore the development of the simulator, the tuning of growth and irrigation models using real world measurements, modeling companionship relationships that affect inter-plant dynamics, and learning automation policies.

This report also presents the work described in the paper titled “Automated Pruning of

Figure 1.1: **AlphaGarden. Top:** Physical testbed with the FarmBot gantry system. AlphaGarden includes a custom Rotary Pruner, custom Pruning Shears, a depth sensor, an on-board snake inspection camera, an overhead camera, and soil moisture sensors. **Bottom:** Overhead image of growth based on two mirrored seed placements for Garden Cycles 2L and 2R (See 7.2).

Polyculture Plants" by Presten et al., which is currently under review for IEEE CASE 2022. In Chapters 6 and 7, we describe a physically and algorithmically implemented autonomous system that is capable of interpreting the state of the garden and enacting actions chosen by AlphaGardenSim, thus bridging the gap between the real world and simulation.

The AlphaGarden system predicts individual plant centers and radii over time using a Plant Phenotyping network and a Bounding Disk Tracking algorithm [4, 5]. A pruning policy trained in AlphaGardenSim [6, 4, 5] identifies pruning actions to optimize plant diversity and coverage using the corresponding center and radii data. Two novel, custom-designed pruning tools and algorithms autonomously prune plants while a learned Prune Point Identification network identifies and selects specific leaves to prune. Real world experiments, as seen in Figure 1.1, suggest that the autonomous system is capable of pruning plants to facilitate plant diversity while maintaining high canopy coverage. To the best of our knowledge, this is the first system in a polyculture farming setting capable of autonomously deciding and pruning plants. We present results for four 60-day autonomous garden cycles where AlphaGarden decides and executes all actions.

Finally, Chapter 8 gives a discussion on the limitations of the overall system and how this research may be translated to other applications. In Chapter 9, we discuss future work and ongoing projects. We plan to focus on comparing AlphaGarden to a human gardener, creating irrigation policies to limit total water usage, improving pruning capabilities, and implementing a robust seed placement algorithm that is adaptable to different environments.

Chapter 2

Related Work

2.1 Plant Simulators and Growth Modeling

Past work in plant growth simulation has predominantly focused on monoculture agriculture, with the exception of a few simulators that include the option to model growth of multiple species in a garden [50]. The most widely used simulation models, DSSAT [30] and AquaCrop [49], are intended for simulating large scale, monoculture agricultural operations. Furthermore, these point-based models make the assumption that plants are grown homogeneously. Therefore, these models are not well-suited for a polyculture setting, where gardens are heterogeneous.

There exist individual plant models that model inter-plant competition, but to the best of our knowledge, there does not exist one (other than ours) for a polyculture setting. For example, Damgaard et al. [14] proposed modeling competition between individual plants based on density and size differences, but their work does not explicitly model resource competition, which is important for tuning a policy that affects the distribution of resources in a garden. Price et al. [44] introduced a simulator for individual plant growth and competition with promising results, but their work only modeled plant radii and did not take into account competition for resources other than water. According to Berger et al. [8], a review on individual-based approaches for modeling plant competition, existing models lack consideration for the effect of plants on resource levels in an environment. Thus, we were motivated to develop our own first order simulator for tuning a polyculture gardening policy as seen in [6, 4, 5].

Czárán and Bartha [13] proposed a broad classification of individual-based plant competition models as either grid-based models or individual-based neighborhood models. Grid-based models discretize a region into a grid of cells that may be occupied by plants, while individual-based neighborhood models represent plants in a continuous space. Furthermore, grid-based models typically use empirical rules to define plant competition, whereas individual-based neighborhood models define explicit mechanisms that regulate competition. One such individual-based neighborhood model is the zone-of-influence model [8], where a

plant acquires resources from a circular zone proportional to the plant's size. Plants with overlapping zones are in competition with each other, and the growth rate of a plant decreases as overlaps increase. While these models allow for greater modeling complexity, grid-based models make simplifying assumptions and reduce computational cost.

Gou et al. [23] propose a model to simulate the growth of two species in a strip-relay intercropping system by calibrating the plant-specific parameters given observed field data. However, this model only takes into account light competition, assuming that irrigation is sufficient. The model allows for analysis of different seed placements on plant growth but restricts to the strip intercropping environment. Tan et al. [52] builds on Gou et al. and include the effects of water acquisition suggesting that plants use land and water more efficiently in intercropping. However, their model does not allow for exploring spatial patterns beyond the strip-relay setting and limits to two species.

Both Gou et al. [23] and Tan et al.[52] do not make explicit use of plant characteristics to define plant inter-relations. On the other hand, Yu [60] uses a simulated functional-structural plant model to investigate which plant traits contribute to complementary relationships and the effects of different plant placements, assuming irrigation provides sufficient water for all plants.

CoppeliaSim [45] comes closer to simulating a polyculture garden, as plants are able to be controlled separately. This simulator was used to train a crop monitoring green house robot to navigate a greenhouse and identify diseased crops [2]. Even though each plant had unique parameters, they did not model inter-plant interactions.

While these simulators consider the polycultural setting, they either do not model the light and water competition simultaneously and/or are limited by the placement geometry that they consider. In Chapters 4 and 5, we present extensions to AlphaGardenSim to incorporate plant relationships and consider inter-plant cooperation.

2.2 Plant Monitoring

GeoSim [53] is a tool that adds spatial functionality to point-based agricultural models by leveraging data from a geographic information system (GIS) to run independent simulations at different geospatial points, allowing for heterogeneous simulation. However, tuning a policy for managing a small-scale polyculture garden necessitates simulation at the individual plant level. Fernando et al. [20] use a greenhouse to evaluate mobile robot monitoring of plant health and soil moisture. Recently, Chebrolu et al. [9] developed a point cloud registration algorithm that enables plant monitoring to analyze growth at the single-plant level. It can be used to tune a single-plant growth model, but does not reveal inter-plant interactions, which are required to provide higher granularity data for polyculture modeling.

Phenotyping is an important task for monitoring plants, similar to object tracking and identification. Ayalew et al. [7] present a method to use an unsupervised domain adaptation network to adapt the meticulously pre-labeled Computer Vision Problems in Plant Phenotyping (CVPPP) dataset [37, 36] to other plant and image domains. The data consists of

single plants, their leaves, and a point map of leaf centers. This reduces the human effort required to track, count, and identify leaf centers. Our work builds on this by transferring the results to a polyculture setting, as discussed in Chapter 6. We extend phenotyping further by converting segmentation masks to plant specific formats characterized by the plants' size and type (see 6).

2.3 Agricultural Automation

Humans have continuously improved farming techniques, and in recent years, have introduced methods for agricultural automation. In 1995, the Telegarden, an art installation by Goldberg et al. [32, 22], allowed internet visitors to interact with a remote garden by planting and watering plants. Wiggert et al. [57] developed a testbed that enables real-time data collection of plant water stress to automate and optimize plant-level irrigation. Our most recent work differs from these as we focus on autonomously pruning a diverse garden bed. Correll et al. [10] designed a distributed autonomous gardening system with mobile manipulators that detect plants, irrigate, and grasp fruits. While related, our work focuses on tools that would enable a fully automated polyculture pruning system.

Pruning is a necessary capability to tend a polyculture garden. Prior work in autonomous pruning includes rose and bush trimming with a robot arm [12, 51]. Habibie et al. [24] trained a Simultaneous Localization and Mapping (SLAM) algorithm to enable automated fruit harvesting in a red apple tree field. Cuevas-Velasquez et al. [12] demonstrated success using visual servoing to account for changes in stem poses to determine cutting points. In a controlled greenhouse, Van Henten et al. [55] used a robot with a thermal cutting tool to harvest cucumbers. We extend prior work by developing an autonomous pruning pipeline for trimming leaves in a controlled environment. To the best of our knowledge, this is the first case of autonomously pruning a polyculture garden.

FarmBot is an open source gantry robot commercially available since 2016 that is used in our autonomous system. Prior work with this system has examined kinematic modeling to enhance FarmBot trajectory planning [17]. A team from Telkom University used FarmBot to create a web application to help human users with seed planting, watering, and plant monitoring routines [39]. More recently, researchers have proposed a FarmBot simulator "to support the development of a control software able to implement different [precision agriculture] strategies" [38]. We use the FarmBot together with custom pruning and irrigation tools to tend a polyculture garden from planting, through germination, growth, reproduction and decay.

Chapter 3

The Polyculture Growing Problem

The goal of AlphaGarden is to use policies learned in simulation to autonomously and physically tend to real-world plants over a Garden Cycle period. A Garden Cycle consists of planting an arrangement of selected plant types, then completing growth through two actions: irrigating and pruning. Pruning, in particular, is required to ensure all plants reach their potential because growth tendencies vary by type. Garden *quality* is a function of coverage, plant diversity, and water usage. An optimal autonomous system aims to maximize coverage and diversity through pruning actions, while minimizing water usage.

Each garden has a total of N plants, placed within a planter bed of size (H, W) in centimeters (cm). For each plant $i \in [0, n)$, the plant has its center coordinates (cx_i, cy_i) and current radius r_i, both in cm. Each plant i also has a corresponding plant type k (equivalent to p_i), which dictates the estimated germination time g_k, maturation time m_k, and maximum radius R_k. The lifecycle of each plant i is defined by the duration of its five stages of growth: germination, vegetative, reproductive, and senescence, as explained in Chapter 4. An interpreted garden state includes all information described above for every plant $k \in [0, n)$ on day t, and is a simplified version of the simulator garden state $\mathbf{s}(t)$ to be defined in 4. Thus, an interpreted garden state is defined as follows:

$$s(t) = \{p_i : ((cx_i, cy_i), r_i), ...\}, i \in [0, N)$$

Given an interpreted garden state, a policy trained in a simulation that is tuned with real world data can accurately decide which, if any, actions to take on each plant, as seen in Chapter 5. These actions must then be transferred into the real world and properly executed on the physical plants as seen in Chapter 6.

For this project, the physical plants are grown in a 3.0m×1.5m raised planter bed located in the UC Berkeley greenhouse. For autonomous cycles, we split the planter bed into two halves and grow identical seed placements (1.5m×1.5m) in each with different pruning regiments (see 7.2). The cycles last 60 days. In Chapter 7, we showcase the results for these Garden Cycles by analyzing coverage and diversity metrics correlated to pruning actions.

Chapter 4

AlphaGardenSim

4.1 Overview

In AlphaGardenSim [6, 4, 5] the goal is to grow a lush and diverse polyculture garden, represented as a discrete $H \times W$ grid containing N plants uniformly sampled from a set of k plant types, as well as types *soil* and *unknown*, within a growing period T while minimizing irrigation. We can frame the general problem as a Partially Observable Markov Decision Process (POMDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O})$, as shown in Table 4.1.

States ($\mathcal{S}$). A state $\mathbf{s}(t)$ includes the following quantities at timestep t for every point (x, y) in the garden: the seed locations $\mathbf{c}(x, y)$, the health of each plant $h(x, y, t)$ and the soil moisture levels $w(x, y, t)$ in the garden. The timestep t is in days for AlphaGardenSim. We also introduce a vacancy score $e(x, y, t)$ as the minimum distance from point (x, y) to any plant. We define $\mathbf{d}(x, y, t)$ be a vector of length $k + 1$ representing one of the k plant types (or soil) type that is visible overhead at point (x, y). With full state knowledge this is a 1-hot vector, however in a physical garden this induces a distribution over the plant types.

Actions ($\mathcal{A}$). The agent can execute any combination of the following actions, or none, per observation:

- Watering, $a_w(x, y, t)$, applies a fixed amount of water to a circle of radius 9 centered at the center point of the observation (x, y), following the irrigation model described in Section 4.2. The amount of water applied to each grid cell decays exponentially as it approaches the edges of the watering circle.

- Pruning, $a_p(x, y, t)$ reduces the radius of a plant. We define a pruning window of size 5×5, centered at the center point of the observation (x, y). A pruning action will reduce the radii of all plants visible within the pruning window by a pruning level p, which is set by default to $p = 5\%$. This is to simulate the inaccuracy of an automated pruner that is likely to prune plants in the neighborhood of the target leaf.

- Planting, $a_s(x, y, t)$. We extend the action space presented in [6] with a new planting action that seeds a plant at point (x, y) at timestep t. A plant can be planted only in

State Variables				
$\mathbf{d}(x,y,t)$	$h(x,y,t)$	$w(x,y,t)$	$e(x,y,t)$	$\mathbf{c}(x,y)$
$[H,W,k+1]$	$[H,W]$	$[H,W]$	$[H,W]$	$[H,W,k]$
Plant Type	Plant Health	Water Amount	Vacancy	Seed Locations
Action Variables				
$a_p(x,y,t)$		$a_w(x,y,t)$		$a_s(x,y,t)$
$[2,N]$		$[2,N]$		$[H,W,k]$
Pruning		Watering		Planting
Reward Variables				
$r_d(t)$		$r_w(t)$		$r_c(t)$
$[1,]$		$[1,]$		$[1,]$
Plant Diversity		Water Efficiency		Canopy Coverage

Table 4.1: Simulator state, action and reward variables. Dimensionality is shown in the second row of each variable section.

locations labeled as *soil*.

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 4.2.

Rewards ($\mathcal{R}$). As the objective is to achieve a diverse garden with maximal yield and water efficiency, we define $\mathbf{P}(k,t)$, the global population in the garden as a distribution over the k plant types, and the following rewards:

- $r_d(t)$, the garden diversity at timestep t is defined as the normalized entropy of the global population in the garden:

$$r_d(t) = \frac{H(\mathbf{P}(k,t))}{\log k} = \frac{-\sum_{i=1}^{k} \mathbf{P}(i,t) \log \mathbf{P}(i,t)}{\log k}$$

- $r_c(t)$, the garden canopy coverage is defined as the total percent coverage at timestep t, taking into account only the coverage of the plants, ignoring the uncovered space labeled as *soil*:

$$r_c(t) = \frac{\sum_{i=1}^{k} \mathbf{P}(i,t)}{H \cdot W}$$

- $r_w(t)$, the garden water efficiency is defined as the negative water use at day t:

$$r_w(t) = -\sum_{x,y} w(x,y,t)$$

Observations ($\mathcal{O}$). 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.

4.2 Modeling

Plant Representation

We extend the model proposed by Price et al. [44] which represents a plant using a seed location and a radius by adding the height attribute, allowing competition for light in addition to water competition. This abstraction is both efficient and expressive, as it allows to simulate inter-plant occlusions, implicitly leading to competition for resources and complex interactions.

Garden Dynamics

Our process-based crop model [6] simulates plant growth according to endogenous plant parameters and environmental conditions. AlphaGardenSim executes a sequence of updates at each timestep: lighting, water use and plant growth.

Lighting Update

We assume a fixed light source directly above the garden. To simulate photosynthesis [58] as a part of the plant growth model, plants allocate light based on the size of their leaf area. When a plant is occluded by taller plants, light is distributed in an exponentially decaying fashion, where the i^{th} tallest plant at point (x, y) in the grid receives $(\frac{1}{2})^i$ amount of light from point (x, y), where $i \in \{0, ..., n_{xy}\}$ and n_{xy} defines the number of plants for which the distance between their seed location and point (x, y) is smaller than their radius r, as demonstrated in Figure 4.1. For plant j with radius r_j, AlphaGardenSim estimates the total amount of light the plant accumulates l_u by a summation over the light allocated from all garden points that are less than distance r_j from the plant's seed location.

Water Use

Water uptake is defined by a zone-of-influence model [8], allowing access to soil moisture concurrent to the plant's circular size. In general, allocation of water depends on a plant's allocated light, l_u, and water competition in intersecting coordinates. The allocated light defines the maximal amount of water required by a plant

$$w_{max} = \frac{c_2}{c_1}\sqrt{l_u},$$

where c_1 and c_2 are plant-specific parameters that control a plant's resource efficiency - c_1 corresponds to water use efficiency and c_2 corresponds to light use efficiency. Larger values

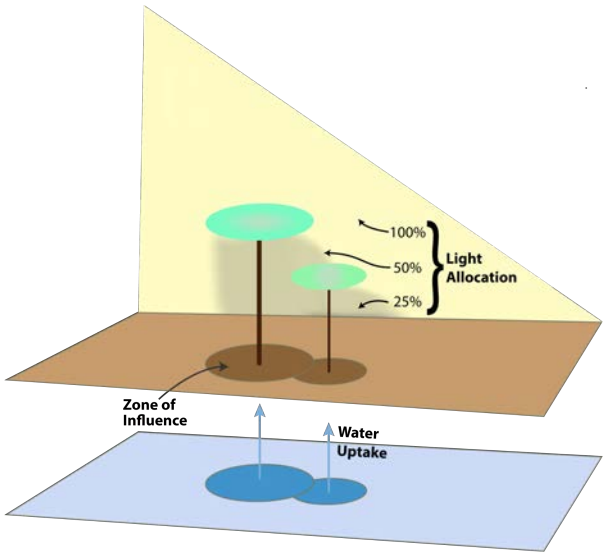


Figure 4.1: **Light and Irrigation Models.** Each plant receives light based on the size of its unoccluded leaf area in the grid, i.e., the number of grid points visible overhead, while occluded points allocate light in an exponentially decaying fashion. The plant's water uptake is then drawn from its 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.

for c_1 or c_2 represent higher biomass accumulation per unit of resource at each timestep [27]. In AlphaGardenSim, c_2 is held constant for all plants, thus c_1 can be seen as the biomass accumulation parameter.

Competition for resources occur for each coordinate in the overlapping zones of influence of plants. Available water is randomly distributed among the plants present in the overlapping regions. For each such plant, we allow it to use the maximum amount of water it desires from this coordinate. In intermediate growth stages, this value is defined on a per plant basis as

$$w_d = \frac{w_{max}}{l_a},$$

where w_d is a plant's desired water amount and l_a is a plant's total leaf area. This approach causes a plant to grow slower in expectation as more of its zone overlaps with the zones of other plants, in accordance with the zone-of-influence model. Furthermore, the water uptake is limited by a soil-specific permanent wilting point that represents a lower bound from which a plant can extract soil moisture [31].

Plant Growth

In AlphaGardenSim, we assign each plant the following growth parameters: germination time, maturation time, growth rate, and growth potential. These values were tuned by

monitoring and analyzing the growth of one hundred and twenty real world plants, and averaging the growth of a plant with others of its same species. Growth parameters from our experiments can be found in Table 4.2. Germination time and maturation time determine a plant's growth stage, and are sampled from a normal distribution with a calculated variance. Growth rate, which is defined as the biomass accumulation variable c_1, and growth potential are parameters that directly determine a plant's size.

The amount of allocated light and water resources impact the biomass pool that is available for growth. A plant's growth is modeled as a logistic curve [15]

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

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, which controls how large the plant will grow.

For each plant, $\tilde{g}$ is then strategically distributed to vertical and radial growth to ensure maximum unoccluded leaf area. Therefore, we define $l_{o,i}$ and $l_{u,i}$, the number of points where plant i is occluded and unoccluded respectively, and model this dynamic as follows: Here, k_1 and k_2 are plant-specific parameters that control the ratio of $\tilde{g}$ a plant apportions to radial growth - k_1 is the lower bound and k_2 is the upper bound. This is reflective of the genetically ingrained habit and morphology of the individual species [28].

After executing the three update steps, the radius and height are incremented according to the computed ratio.

Inter-Plant Dynamics

Originally, plants were treated independently of one another in AlphaGardenSim and relationships between different plant types were not accounted for [6]. To address this, we add companionship relationships between plant types which dictate growth patterns of individual plants dependent on their placement relative to others.

Both above and below ground interactions influence the companionship relationship factor between two plant types, which can be positive or negative [1, 42, 29]. Example of above ground interactions include changes to the physical environment such as providing shade, protecting against weather damage, and supplying structural support. Below ground interactions include providing nitrogen which fertilizes the soil, root-root activity and allelopathy, which occurs when a plant releases toxic chemicals that inhibit growth of other plants [4].

In AlphaGardenSim, we use the model described in [4] to account for companionship relations. 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

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 4.2: 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 [46]. Growth time is found by subtracting g_1 from m_1.

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 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 companionship factor is used to update the daily growth parameter, $\tilde{g}$, which is determined by water and light resource allocation. The new daily radial growth parameter is defined to be $g = \tilde{g} \cdot c$.

Plant Life Cycle

The plant life cycle consists of five non-overlapping stages: germination, vegetative, reproductive, senescence and death [61, 26]. 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 [34].

Germination. Germination starts when the seed is planted. In this stage the plant occupies a single point in the garden and has 0 radius and height. It allocates resources according to the model described previously, however it does not grow, maintaining 0 radius and height until it transitions to the next stage. The initial non-zero radius and height are random variables sampled from a plant-specific Gaussian distribution.

Vegetative. During the vegetative stage, the plant allocates resources and grows according to the model specified previously, unless it experiences stress from over or underwatering.

Reproductive. During the reproductive stage, the plant behaves similarly to the vegetative stage, except it does not change in radius or height, unless it experiences stress from over or underwatering.

Senescence. During the senescence stage, the plant does not change in height, however it allocates less water than before and its radius decays exponentially as it is wilting. If w_d is the plant's desired water amount, throughout the senescence stage it is multiplied by a coefficient, so that the adjusted desired water amount $\widetilde{w}_d$ decreases linearly to 0 over time:

$$\widetilde{w}_d = \frac{1-t}{t_s} w_d$$

where t is the amount of time the plant has spent in the senescence stage, and t_s is the total duration of the senescence stage.

Death. When the plant dies, it stops allocating resources and does not change in radius or height. However, it continues to occupy space in the garden, potentially occluding plants.

Water Stress

AlphaGardenSim models the response of plants to suboptimal irrigation, namely over and underwatering, during the two life stages in which the plant accumulates biomass: the vegetative and the reproductive stages. A plant receives sufficient irrigation if the following conditions are met:

$$w(t) \geq T_o \cdot w_d$$
$$\tilde{w}(t) \leq T_u \cdot w_d,$$

where $w(t)$ and $\tilde{w}(t)$ are the total amount of soil moisture within the plant's radius and its water uptake, respectively, T_o and T_u are over and underwatering plant-specific threshold parameters, and w_d is the plant's desired water amount.

Otherwise, the plant enters into water stress, and its radius decays exponentially until it reaches a fraction of its radius and transitions to the death stage or it receives sufficient irrigation. In addition, the effects of water stress are visualized via the plant's color, becoming progressively more brown as it continues to be stressed.

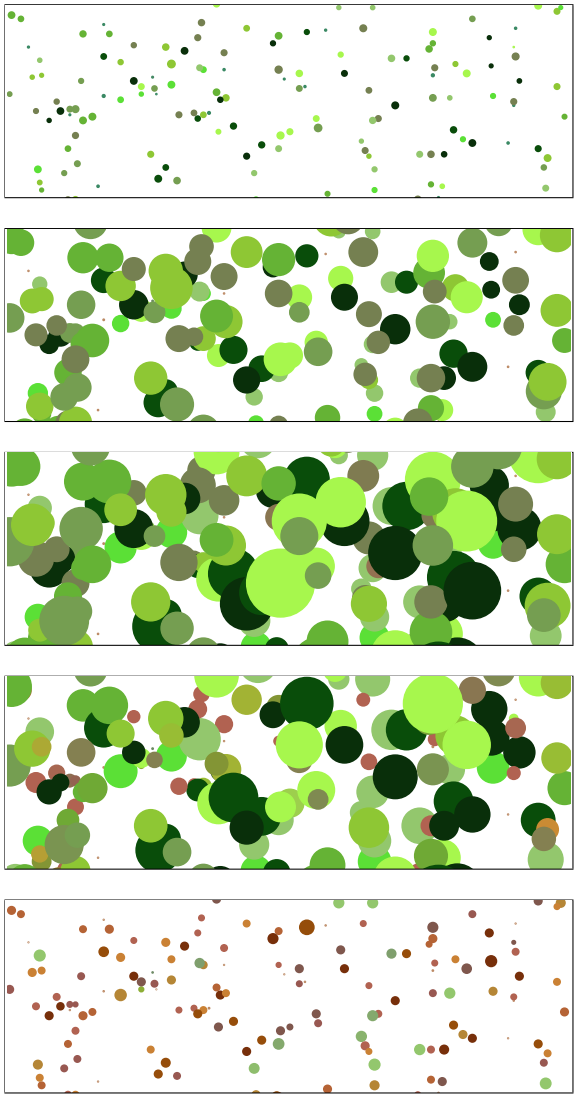


Figure 4.2: **Plant Life Stages**. Each plant is modeled with a life cycle trajectory, consisting of five stages (from top to bottom image): germination, vegetative, reproductive, senescence, and death. When plants get underwatered or overwatered, their radius decays exponentially and their color turns brown, and after a short period they move to the death stage. However, if they receive their desired water amount prior to that, they can return to their original stage.

Irrigation

AlphaGardenSim uses a discrete-time linear approximation of Richards equation proposed by Tseng et al. [54] to model irrigation actions and soil moisture dynamics. As described in [6, 5, 4], 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 six TEROS-10 [35] volumetric water content soil moisture sensors connected to a ZL6 Data Logger [35]. 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. We intend to tune plant uptake, $u(x, y, t)$, in future real world experiments.

We first made modifications to the irrigation application parameter, $a_w(x, y, t)$, by carrying out experiments with the FarmBot watering nozzle. Using a compartmentalized container placed beneath the nozzle, we discovered the area of influence of a watering action to be concentrated within a circle of 0.04m radius. Furthermore, we identified the FarmBot nozzle to have a flow rate of 0.083 L/s. In AlphaGardenSim, $a_w(x, y, t)$ is set to 0.200 L.

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.

Next, we used the moisture sensors to tune the local water loss parameter, f, and build upon our findings in [4]. 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. One such water loss curve can be seen in Figure 4.3. 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. We model water loss and gain over each square grid point, with side length of 0.01m, in AlphaGardenSim by sampling from a univariate Gaussian calculated from experimental data.

We calculated the daily loss by averaging the loss of all five sensors. For each sensor, the daily loss was the difference between the highest value recorded by the sensor within three hours after watering, in which soil moisture gain occurred, and the lowest value recorded

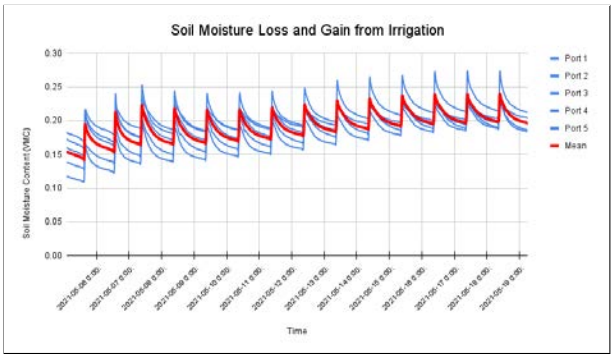


Figure 4.3: Soil moisture curve generated from TEROS-10 soil moisture sensors connected to a data logger to determine water loss and gain rates. Irrigation was applied every 24 hours. Soil moisture readings were recorded every 30 minutes. The five blue curves represent five different sensors that were each watered independently. The red curve is the average of the readings of all five sensors.

directly before the next watering action. The Gaussian for water loss over a single day has a mean of 0.042 m^3/m^3 and a standard deviation of 0.0048 m^3/m^3, and f is sampled from such in AlphaGardenSim. For more than one day after watering, the Gaussian for loss has a mean of 0.01 m^3/m^3 and a standard deviation of 0.0001 m^3/m^3. Similarly, we modeled the *gain* as a univariate Gaussian calculated from daily gains over a two week span, where the daily gain was calculated by taking the difference between the value right before and within three hours after watering on the same day. The Gaussian for gain has a mean of 0.046 m^3/m^3 and a standard deviation of 0.0054 m^3/m^3 and was set accordingly in simulation.

Moreover, we tuned the prior soil moisture content parameter, $w(x, y, t-1)$, using another set of experiments with the TEROS-10 sensors. To do so, we identified the maximal volumetric water content of our specific soil, which indicates the water storage capacity of the medium. In the experiments, we saturated five different samples of soil using varying watering techniques and discovered the max volumetric water content to be around 0.3. Thus, we capped both $w(x, y, t-1)$ and $w(x, y, t)$ at this value.

Through the execution of physical test bed experiments and the utilization of soil moisture sensors and the FarmBot watering nozzle, we were able to tune parameters of the AlphaGardenSim irrigation model to more realistically simulate the characteristics of the real world garden.

4.3 Pruning, Irrigation, and Planting Policies

We evaluate the performance of different polyculture pruning, irrigation and planting policies by assessing their robustness in varying garden settings to achieve high plant yield and reduce water use in AlphaGardenSim.

Policies

We implement five policies:

1. Uniform Policy, a policy that irrigates according to a fixed schedule and prunes all plants uniformly.
2. Fixed Pruning, a policy that irrigates and prunes plants with a fixed pruning level based on water availability, plant health and garden diversity.
3. Variable Pruning, 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 Pruning, a deep supervised learned policy that learns from Variable Pruning prune level demonstrations to predict prune levels over 1500X faster than Variable Pruning.
5. Dynamic Planting, a policy that seeds plants throughout the lifespan of the garden to achieve indefinite garden growth.

Uniform Policy. Introduced in [6], Uniform Policy irrigates all plants every other day similar to an array of drippers or sprinklers in farms and greenhouses. To limit overcrowding, every 5 days, the policy prunes all plants that grew beyond a threshold with $p = 5\%$.

Fixed Pruning. In [6], we presented Fixed Pruning, which utilizes soil moisture, plant health and global diversity to dynamically prune and irrigate each sector it observes. For every $\mathbf{o}(x, y, t)$, Fixed Pruning applies one of four actions: irrigate, prune, irrigate and prune, or none.

If any of the plant health values $h(x, y, t)$ in $\mathbf{o}(x, y, t)$ within the radial distribution of the water nozzle indicates underwatered, the policy irrigates the sector. If the sector does not contain any plants or only dead plants, Fixed Pruning does not irrigate. To avoid irrigating plants that are overwatered, the policy sums all $w(x, y, t)$ in the sector and doubles $w(x, y, t)$ wherever $h(x, y, t)$ contains an overwatered plant. If the total sum is less than a threshold, the sector is irrigated.

Fixed Pruning selects a pruning action if the proportion of any plant type, calculated by $r_d(t)$, in the pruning window is greater than a uniform threshold.

Variable Pruning. Experiments in prior work [4] and those in Section 5.4 suggest that, due to a fixed pruning level, Fixed Pruning struggles to manage plants with significant differences in germination times, maturation times and max radii. To address this limitation, we introduced Variable Pruning, a policy that selects a pruning level $p \in \mathcal{P}$ for each day t from a discrete set of six pruning levels $\mathcal{P}$. Every timestep, Variable Pruning takes a 1-step lookahead to simulate the potential multi-modal entropy *mme* (see Section 5.2) that

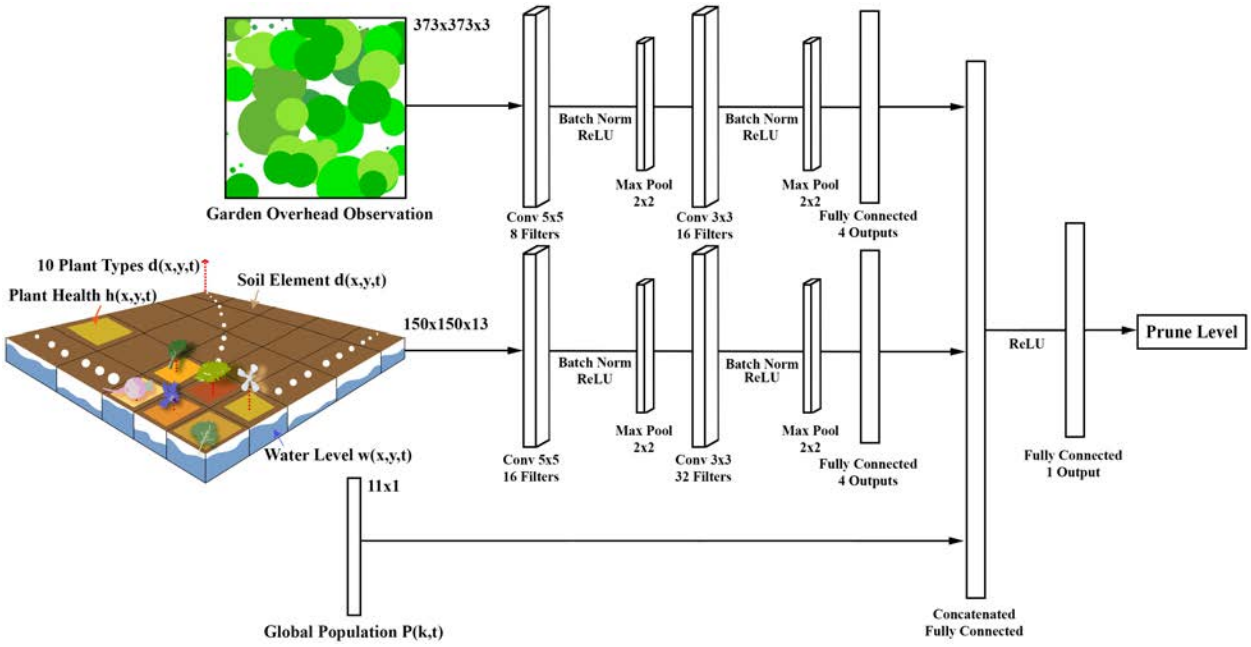


Figure 4.4: Learned Pruning network architecture. A deep convolutional neural network with with 18,244 parameters. The network takes three inputs: 1) an RGB image of the full garden; 2) a matrix of $h(x, y, t)$, $w(x, y, t)$ and $\mathbf{d}(x, y, t)$ for all (x, y) in the garden; 3) the global population distribution $\mathbf{P}(k, t)$ including soil coverage. The network predicts a prune level for each observation using demonstrations from Variable Pruning.

would result from choosing pruning level $p_i \in \mathcal{P}$ on the current garden state. After selecting p, Variable Pruning uses Fixed Pruning to collect pruning and irrigation actions for every $\mathbf{o}(x, y, t)$.

Learned Pruning. We introduced Learned Pruning in [4], as a way to speed-up 1-step lookahead with Variable Pruning by over 1500X. We train a deep supervised learned policy, mapping prune level p demonstrations from Variable Pruning to full garden states as illustrated in Figure 4.4. A deep CNN with 18,244 parameters takes in an RGB garden overhead observation, a matrix of plant health, plant types, and water availability, and the global population distribution to determine a prune level for a plant.

Dynamic Planting. Dynamic Planting is an extension of Variable Pruning that uses a 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 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.

Adaptive Sector Sampling

In prior work [6], we introduced 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, 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

Simulator Experiments

5.1 Experimental Overview and Setup

To evaluate the functionality and policies within AlphaGardenSim as described in Chapter 4, we conduct experiments on 150cm×150cm simulated gardens with 100 plants sampled with replacement from k plant types. Plants are seeded at random locations $s(x, y) = \mathbf{d}(x, y, 0)$. To promote plant germination and early growth, we set the overwatering threshold to $T_o = 100$ as it represents the maximum amount of water in a 10×10cm square around a plant. The underwatering threshold is set to $T_u = 0.1$.

5.2 Evaluation

We evaluate the policies described in Section 4.3 on randomly seeded experiments, using the following metrics:

1. **Average Total Plant Coverage** - We average the total percent coverage in a single experiment over days 20 to 70 of the growing period, taking into account only the coverage of the plants, ignoring the uncovered space labeled as *soil*:

$$AC = \frac{\sum_t r_c(t)}{T}.$$

2. **Average Diversity** - We average the diversity in a single experiment between days 20 and 70 of the growing period, $T = 50$:

$$AD = \frac{\sum_t r_d(t)}{T}.$$

   During the beginning and end of the growing period, the diversity is always high, since all plants are very small (germinating or dying). Therefore, these diversity measurements are not reflective of the policy's performance.

3. **Water Usage (liters)** - We sum the water used in a single experiment over the entire growing period (100 days):
$$WU = \sum_t -r_w(t).$$

4. **Multi-Modal Entropy** - We model diversity as the normalized entropy of $\mathbf{P}(k,t)$ for all k plant types. Maximum diversity, hence, equates to a uniform distribution within $\mathbf{P}(k,t)$. However, prior work [4] and Section 5.4 suggests that high diversity can be achieved with low plant coverage and consequently, high soil exposure. Thus, we define $\tilde{k}$ be the union of the k plant types and an additional type representing the amount of unoccluded soil, so that $\mathbf{P}(\tilde{k},t)$ will include soil coverage. We define multi-modal entropy (mme) as:
$$mme(t) = \frac{H\left(\mathbf{P}\left(\tilde{k},t\right)\right)}{\log \tilde{k}} = \frac{-\sum_{i=1}^{\tilde{k}} \mathbf{P}(i,t) \log \mathbf{P}(i,t)}{\log \tilde{k}}.$$

5.3 Adaptive Sector Sampling Experiments

We compare the evaluation metrics achieved with adaptive sector sampling versus the sector observation approach from [6]. Gardens contain 100 plants, 10 of each plant type from Table 4.2. As described in Section 4.2, the amount of water a plant receives while growing after germination corresponds to how much it grows. A smaller $c_{d,grow}$ results in more sectors observed for growing plants as less plants are within the threshold for clustering. A $c_{d,grow}$ = 1cm resembles the sector sampling approach from [6]. We experiment with two $c_{d,grow}$ clustering thresholds: 2cm and 8cm. We fix $c_{d,germ}$ to 8cm as germination duration does not depend on the amount of water provided. Results averaged across 20 gardens over 100 days are summarized in Table 5.1. With a $c_{d,grow}$ = 2cm, Fixed Pruning observes more sectors and provides more water to each plant than $c_{d,grow}$ = 8cm. However, both thresholds achieve comparable coverage, diversity and mme to the sector observation approach from [6]. By clustering germinating and growing plants that are $c_{d,germ}$ = 8cm and $c_{d,grow}$ = 2cm of each other, Fixed Pruning uses 37% less water over the entire garden simulation period, 38% less irrigation actions, and 35% less pruning actions. With a $c_{d,grow}$ = 8cm threshold, Fixed Pruning is able to use 50% less water, irrigation actions and pruning actions.

5.4 Pruning and Irrigation Experiments

Plant Types from Table 4.2

We evaluate Uniform Policy, Fixed Pruning, and Variable Pruning on gardens with 100 edible plants, 10 plants from each of the 10 plant types with growth parameters in Table 4.2. Observations to policies are determined through adaptive sector sampling described in Section 4.3 with $c_{d,germ}$ = 8cm and $c_{d,grow}$ = 8cm.

Metric	Without	$c_{d,grow} = 2$cm	$c_{d,grow} = 8$cm
Avg coverage	0.71	**0.74**	0.71
Avg diversity	0.91	0.91	0.91
Avg multi-modal entropy	0.83	**0.84**	0.82
Avg water use (liters)	15.73	9.80	**7.36**
Num. of irrigation actions	7862.6	4900.5	**3680.9**
Num. of pruning actions	732.3	474.0	**292.5**

Table 5.1: Policy evaluations of Fixed Pruning averaged across 20 test gardens during days 20 to 70 with and without adaptive sector sampling. We observe germinating plants within 8cm of each other in the same observation sector and experiment with 2cm and 8cm cluster radii for growing plants. Both a 2cm and 8cm cluster radii for growing plants are able to achieve comparable coverage, diversity and multi-modal entropy to the sector observation approach from [6]. While $c_{d,grow} = 2$cm uses over 35% less water, irrigation actions and pruning actions than without adaptive sectoring, $c_{d,grow} = 8$cm uses over 50% less water and actions.

Through experiments, we found that a fixed prune level of 15% for Fixed Pruning and the set of pruning levels $\mathcal{P} \in (5\%, 10\%, 16\%, 20\%, 30\%, 40\%)$ for Variable Pruning, leads to high coverage, diversity and *mme* on the plant set. Results averaged across 20 different random garden seed placements over 100 garden days are summarized in Table 5.2. Compared to a baseline no pruning policy which irrigates every other day, Uniform Policy achieves higher diversity but overprunes plants due to only being able to prune every 5 days. Both Variable Pruning and Fixed Pruning achieve higher coverage, diversity, multi-modal entropy than Uniform Policy by dynamically selecting which plants to irrigate and prune each day based on plant health, garden diversity, and multi-modal entropy for Variable Pruning.

Plant Types from Table 5.3

As suggested in prior work [6, 4], Fixed Pruning struggles to achieve both high coverage and diversity, and consequently high *mme* on plants with different germination times, maturation times and max radii. To illustrate this, we conduct experiments on gardens with 100 plants, 10 plants from each of the 10 plant types in Table 5.3 where faster growing plants grow five to eight times faster than slower ones. We use the observation sampling method described in prior work [6], sampling m sectors centered at each $s(x, y)$ and $\frac{m}{10}$ sectors centered at non-seed points.

We simulate Fixed Pruning with 15% and 1% pruning levels, and Variable Pruning with prune levels $\mathcal{P} \in (5\%, 10\%, 16\%, 20\%, 30\%, 40\%)$. To achieve higher plant diversity, Fixed Pruning with a 15% prune level aggressively prunes the faster growing plants during their growing period to match the size of the slower growing plants. The policy achieves high

Metric	Uniform	Fixed	Variable	Learned
Avg coverage	0.59	0.71	**0.74**	-
Avg diversity	0.88	**0.91**	0.88	-
Avg multi-modal entropy	0.75	0.82	0.82	-
Avg water use (liters)	7.53	7.36	**7.34**	-
Avg coverage	-	0.23	**0.46**	0.42
Avg diversity	-	**0.76**	0.67	0.67
Avg multi-modal entropy	-	0.37	**0.55**	0.53
Avg water use (liters)	-	**18.67**	19.81	19.80
Computation time (seconds)	-	-	336.18	**0.22**

Table 5.2: Policy evaluations of Uniform Policy, Fixed Pruning, Variable Pruning, and Learned Pruning averaged across 20 test gardens each with 100 plants. **Top 4 rows:** experiments use the 10 plant types from Table 4.2. Metrics are averaged between days 20 to 70 as policies do not prune prior to day 20 and plants begin to die after day 70. **Bottom 5 rows:** experiments use the 10 plant types from Table 5.3. The faster growing plants begin to die after day 50, so we instead average metrics for these gardens between days 20 to 50. The computation time represents the time it takes a policy to compute an action given an observation. The Variable Pruning is computational intensive as it evaluates different pruning levels, while Learned Pruning performs similarly but has a significantly lower computation time.

diversity at the cost of low coverage, resulting in low multi-modal entropy. Fixed Pruning with a 1% prune level prunes the fast-growing plants less, but fails to prune enough to achieve uniform plant diversity. Thus, a 1% prune level also results in low multi-modal entropy.

We compare Fixed Pruning with a 15% prune level against Variable Pruning on 20 randomly seeded gardens. Results are averaged in Table 5.2. The faster growing plants from Table 5.3 begin to die after day 50 so results are averaged between days 20 and 50. Variable Pruning selects a prune levels that would result in the highest *mme* 1-day into the future and initially uses a small prune level of 5% to allow faster growing plants to grow. As the faster growing plants begin to die, Variable Pruning uses higher prune levels to increase diversity and maintain high multi-modal entropy. As a result, Variable Pruning achieves high coverage, diversity and multi-modal entropy.

Due to the long runtime of simulating 1-step lookahead with Variable Pruning, we train Learned Pruning to learn prune levels for each day given full garden states of gardens with slow and fast growing plants from Table 5.3. We simulate Variable Pruning on 6,500 gardens with randomized seed locations to collect prune level demonstrations between days 20 to 100 as policies do not prune before day 20. Variable Pruning selects a 5% prune level 95% of the time and levels 10% and greater 5% of the time to maximize multi-modal entropy for the slow and fast plant types. We increase the number of demonstrations for prune levels 10% and greater by 8× by rotating and flipping observations. The network is trained with

Plant Type	Germination (days)	Maturation (days)	Max Radius (cm)
Fast growing	9.8	20.2	93.3
Slow growing	25.6	90.4	31.95

Table 5.3: **Fast and Slow Plant Types.** Average germination time, maturation time, and max radii of 5 fast and 5 slow growing plant types. We experimented with varying germination times, maturation times and max radii to create the plant set above where Uniform Policy achieves low multi-modal entropy *mme*, as illustrated in Fig. 5.1.

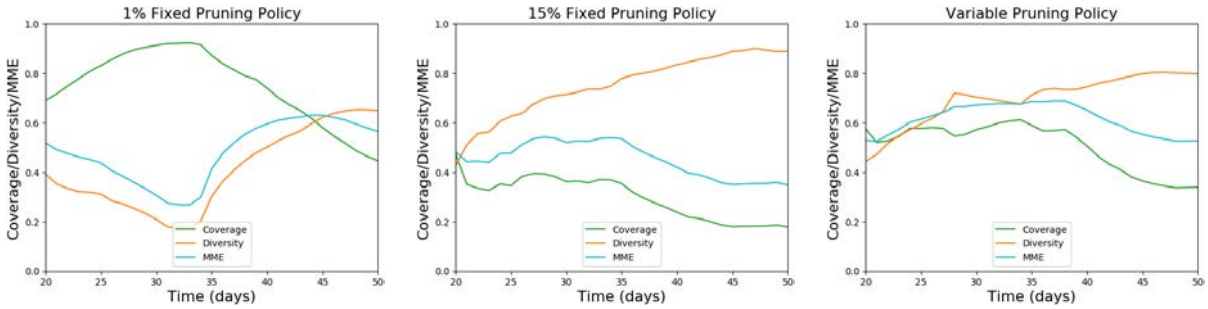


Figure 5.1: Simulation results on gardens between days 20 and 50 with the fast and slow growing plant types from Table 5.3. Metrics are shown between days 20 and 50 as the faster growing plant types begin to die after day 50. **Left:** Simulation results for Fixed Pruning with fixed prune levels of 1%. With a 1% fixed prune level, Fixed Pruning achieves high coverage but struggles to maintain diversity. As a result multi-modal entropy is low. **Middle:** With a 15% prune level, Fixed Pruning achieves high diversity but low coverage as a result of pruning the fast growing plants to match the size of the slower plants. **Right:** Variable Pruning simulation results. By optimizing for multi-modal entropy, the policy is able to manage both coverage and diversity through variable prune levels and achieve the highest multi-modal entropy. During earlier days, Variable Pruning uses smaller prune rates to allow the faster growing plants the grow. As the fast plants begin to die, to maintain high multi-modal entropy, Variable Pruning prunes more frequently.

520K demonstrations for 55 epochs with the Adadelta [62] optimizer and mean squared error loss using 4 hardware threads and 4 Tesla V100 GPUs. The network architecture and optimization framework is written in Python using PyTorch. Table 5.2 summarizes results averaged across 20 test gardens withheld from the training dataset. Learned Pruning achieves comparable coverage, diversity, *mme* and water usgae to Variable Pruning but is over 1500× faster predicting p for days 20 to 50.

5.5 Dynamic Planting Experiments

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

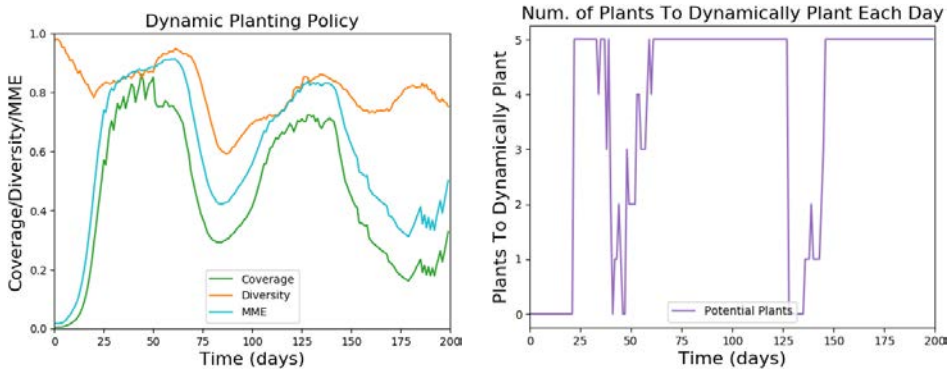


Figure 5.2: **Dynamic Planting Policy.** The policy seeds up to 5 new plants every day after day 20. During periods where coverage is high in the garden, there is little vacant space to seed new plants. As a result, the number of plants selected to be dynamically planted drops during days 35 to 61 and 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)
	0.50	0.82	0.63	158.98

Table 5.4: 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.

simulate a growing period of 200 days. We follow the observation method from [6] 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. We average results across 10 test gardens with random seed placements, as seen in Table 5.4 and Figure 5.2. 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 vacancy $e(x, y, t) \geq 8$cm. After the new plants germinate and mature, coverage rebounds.

Chapter 6

AlphaGarden Autonomous Pipeline

6.1 Overview

This chapter will discuss the implementation of the fully autonomous pipeline. To execute an AlphaGardenSim policy physically, we need to interpret the real-world garden state $s(t)$ (Chapter 3), pass it into AlphaGardenSim (Chapter 4), extract the plant specific actions, and then enact these actions in the real-world garden. Figure 6.1 shows this pipeline. For state estimation, a trained phenotyping network identifies plant types (see Section 6.2) and a Bounding Disk Tracking algorithm transforms the segmentation masks into a simulator readable format (Section 6.3). Subsequently, the Fixed Pruning Policy decides irrigation and pruning actions based on the garden state. The actions are then executed using a

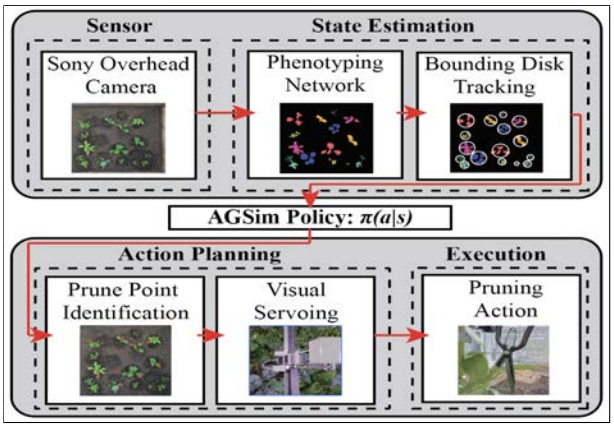
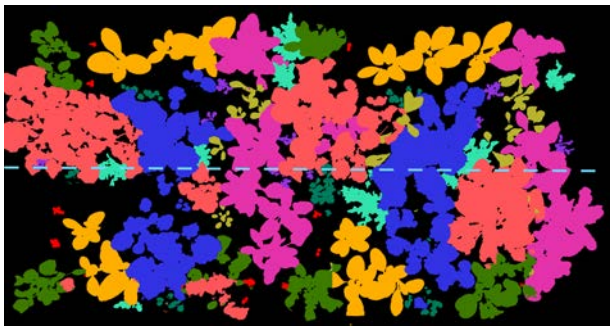


Figure 6.1: **Automated Pruning Pipeline:** The overhead Sony camera takes photos on an hourly basis. The images are processed by a Plant Phenotyping Network followed by Bounding Disk Tracking algorithm to identify the garden's state. AlphaGardenSim determines which plants to prune in real time. Given the simulator's decisions, a Prune Point Identification network identifies specific leaves to prune. This is followed by visual servoing to arrive at the leaf location in the physical garden and then execution of the prune using a custom pruning tool.



Name	unknown / soil	kale	swiss chard	turnip	arugula	borage	green lettuce	cilantro	sorrel	radicchio	red lettuce
% Coverage	48.4	11.8	10.1	9.7	7.4	7.0	1.7	1.0	0.3	0.2	0.1
Baseline IoU	0.88	0.78	0.81	0.55	0.40	0.82	0.72	0.46	0.70	0.13	0.39
IoU with augmentation	0.92	0.92	0.93	0.81	0.72	0.95	0.87	0.71	0.76	0.57	0.65

Figure 6.2: **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.

Prune Point Identification network (Section 6.4), visual servoing algorithms (Section 6.4), and custom hardware (Section 6.5).

6.2 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.

We mounted a Sony SNC-VB770 digital camera [48] with a 20mm Sony lens [47] 2m above the garden bed to monitor the garden. The VB770 satisfies our major requirements that include (1) resolution, (2) image distortion, (3) FOV, (4) power delivery, and (5) remote

data accessibility. It has a DSLM 35mm sensor with a maximum 4240×2832 resolution, and the camera publishes photos every hour. We trained a model using UNet architecture [41] and ResNet34 [25] 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 garden cycles. Training uses categorical cross-entropy loss over 100 epochs and utilizes a 75-25 train-validation split. Each image is split into 512×512 RGB patches and augmented via shifting and rotating. This additional augmented data improves network robustness as shown in Fig. 6.2. 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 [4]. Figure 6.2 shows the network's prediction on the bottom half of an overhead image, which is unseen to the network. When evaluated, the model has a mean IoU of 0.80. The model performs well in identifying plant types with high coverage, but has lower accuracy in plants that are not common in the overhead image.

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 performance.

Accurate segmentation for plants after day 30 becomes increasingly important in order 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 [6]. 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), and 1 otherwise.

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

We define mean IoU as $\sum_{i=0}^{i_{total}} IoU(label_i)/(i_{total} + 1)$. The baseline model [4] 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

Figure 6.3: **Phenotyping and Bounding Disk Tracking.** 3 images from days 20, 30, and 40 of garden cycle 4. **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).

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. On 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.

6.3 Bounding Disk Tracking

While visual occlusions present challenges to tracking plant shapes over their lifespan, the estimated bounding disk of the plant should remain relatively consistent.

We define a plant's *bounding disk* (see Figure 6.3) 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 [56] and irrigation [16]. 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) 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 [6] 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, K-Means clustering, has two main assumptions: (1) the clusters have roughly the same number of points and (2) the clusters are circularly distributed. The first assumption is true near the beginning of the garden, because plants of the same type grow similarly. However, this assumption complicates later in the cycle as competitive relationships in the garden and occlusion start to create asymmetries. The second assumption follows from the circular model we use to track plants.

In order to benchmark model performance, we introduce two metrics: average circle utility (ACU) and percentage of pixels included (PPI). 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 given bounding disks. We then define the average circle utility as $ACU = \frac{P_i}{P_c}$ and percentage of pixels included as $PPI = \frac{P_i}{P_t}$. Each metric is computed per plant type per timestep.

We want to maximize both ACU and PPI to compute the optimal bounding disks. At 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 Figure 6.4 (left) shows, initial K-Means clustering performs well as 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 Figure 6.4 (right) shows, BFS lags early on, but then becomes increasingly effective as plants are occluded mid-garden cycle.

6.4 Pruning Planner

Once a garden state on 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

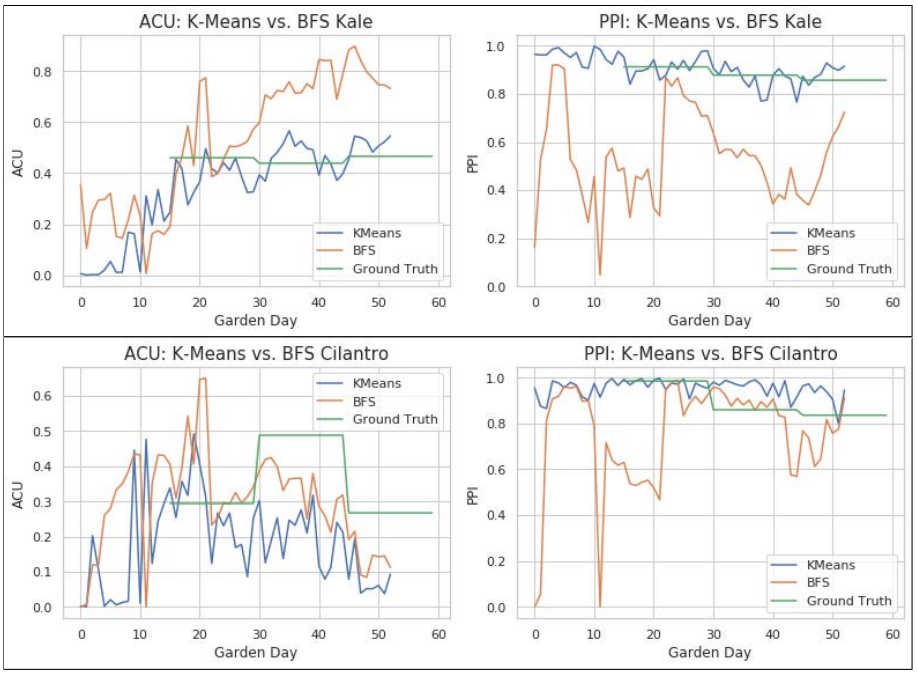


Figure 6.4: **Garden Metrics of Garden Cycle 2R for Kale and Cilantro**. Kale demonstrates the statistics for larger plants, while Cilantro demonstrates them for smaller plants. We evaluate average circle utility (ACU) and percentage of pixels included (PPI) of the Breadth-First-Search (BFS) versus the K-Means bounding disk algorithms for both plant types. **Kale:** BFS tends to have higher ACU, but lower PPI. For the days which ground truth circles exist, 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.

navigate and position the FarmBot above the chosen leaf using visual servoing, and execute the pruning action with custom hardware.

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. [7]. 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 [7].

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 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 Figure 6.5). 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 Figure 6.5, the learned method has far fewer points that lie on different plants.

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.

Visual Servoing for Pruning Tool

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 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 [18]. 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,

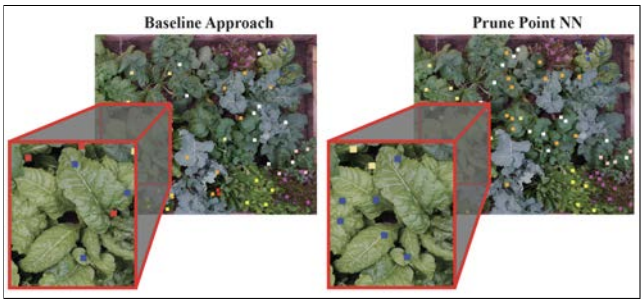


Figure 6.5: **Prune Point Identification.** Example of all plant leaf centers that were identified by the baseline algorithm (left) and the model (right) applied to an overhead image. Each prune point color corresponds to a different plant type. The learned model identifies more usable points with fewer misclassifications. When looking at the Swiss Chard plant highlighted (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.

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.

Although the visual servoing algorithm is quite robust, when plants grew too high and close to the on-board camera (approximately 0.4m above the soil surface), it was not able to take a clear image of the garden, which led to failed localization and servoing to the prune point location. To remedy this, we moved the on-board camera to approximately 0.7m above the soil surface, away from plants' reach, allowing it to capture unobstructed images of the garden and better localize them within the global image.

6.5 Pruning Hardware

The goal of pruning is to reduce the coverage of plant k centered at a point (x_k, y_k) with radius r_k. To tend to and prune plants, the autonomous system uses a commercial FarmBot [19] installed over the 3.0m×1.5m planter bed frame. This CNC robot can 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. The tools we designed are operated through the FarmBot system

Figure 6.6: **Pruning tools. Left:** CAD and physical model of Rotary Pruner with a high speed motor and trimming blades. **Right:** CAD and physical model of Pruning Shears with three servos to control closing, tilt, and orientation.

with no human intervention. Once the FarmBot moves to a prune location, the pruning tool then aims to remove all or part of the leaf structure in that neighborhood to reduce coverage. We designed, implemented, and studied two options: Rotary Pruner and Pruning Shears.

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 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 $(d + 5)$cm; 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 spotlights 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 [40] were fastened directly to the FarmBot's gantry rails. A YANSHON Digital 360° servo motor closes 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 $(d + 5)$cm and activated. Once a cut is complete, the shears return to their default positioning.

Chapter 7

Real World Experiments

7.1 Isolated Pruning Experiments

To evaluate the two pruning tools, we ran isolated pruning experiments on eggplant and bell pepper. We chose eggplant for its large leaves comparable to kale, borage, and turnip, and we chose bell pepper for its smaller abundant leaves similar to cilantro and lettuce. We placed a grown potted plant near the midline of the garden bed, took an overhead image, and then passed a manually annotated plant center and prune point into our visual servoing and pruning algorithms.

For each tool, we made 5-6 prunings on both plant types, observing completeness of the cut, precision of the cut (if any neighboring leaves were harmed in the process), and any error that may have occurred. Our results are in Table 7.1. We found the Rotary Pruner was more likely to complete a cut, but it also tended to over prune. Furthermore, due to the nature of the Rotary Pruner, the final cuts were not 'clean' and showcased tears and fragments. The Shears, on the other hand, generally caused little secondary damage and debris and made clean cuts, but were more prone to incompletely cut or miss a leaf. The main reasons for failure to execute a prune were due to bad prune point selection or the pruning tool pushing a leaf out of the way.

7.2 Four Garden Cycles

To evaluate the entire system holistically, we ran four autonomous cycles over two 60 day periods. We split the garden into two halves and planted identical seed placements (1.5m×1.5m) on each. 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 garden state. AlphaGardenSim would use this garden state to decide which plants

Plant Type	Cut	Pruning Shears Results			Rotary Pruner Results		
		Compl.	Precision	Err.	Compl.	Precision	Err.
Eggplant	1	2	0	B	2	0	B
	2	3	0	A	2	1	A
	3	2	0	B	3	1	A
	4	3	1	A	3	1	A
	5a	2	0	C,D	2	0	B
	5b	3	0	C,D	2	0	B
	6a	2	1	D	2	1	B
	6b	-	-	-	2	1	B
BellPepper	1	1	0	B	1	1	D
	2	2	0	C	1	1	B
	3	3	0	A	1	1	B
	4	1	0	B	3	0	A
	5a	1	0	C,D	3	0	A
	5b-c	1	0	C,D	-	-	-
	5d	3	1	C,D	-	-	-
	6	-	-	-	2	1	B

Table 7.1: **Isolated Pruning Experiments** for the Rotary Pruner and Pruning Shears. **Key:** *Completeness-* 3: complete cut, 2: partial cut, 1: missed cut. *Precision-* 1: no damage to other leaves, 0: damage to other leaves. *Error Type-* A: No error, B: location, C: depth, D: Other.

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.

Human Intervention

Although the goal is a fully automated polyculture garden pruning system, some human intervention was required during the Garden Cycles. Seed planting was performed with human labor. The author of this work was present during all pruning actions, which were executed in batches. While all decisions were made autonomously, human intervention was used to correct robot position when the FarmBot gantry failed to servo to the correct target location, which occurred on 45% of pruning operations. However, by moving the on-board camera higher up on the gantry system, we were able to reduce the rate of fail of visual servoing for future Garden Cycles. No other human intervention was performed in terms of weeding or irrigation.

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 7.2: **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 $[c_i(60) \cdot (R/R_i)^2]$ for each plant type (Section III). The goal of pruning is to foster a diverse garden while maintaining a high coverage.

Garden Cycles 1L and 1R

In Cycles 1L and 1R, the identical seed placements (1.5m×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.

In Table 7.2 we report the final canopy coverage and diversity for each individual plant type. The compared both metrics by manually labeling a ground truth mask on day 60 of the garden cycle. 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 came at the cost of losing some overall coverage (15.15% decrease).

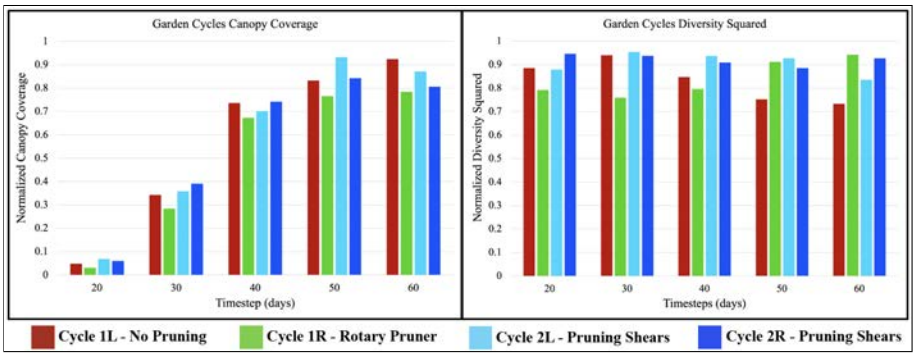


Figure 7.1: **Garden Cycle Comparison.** Data points were recorded for days 20, 30, 40, 50, and 60 through hand labeled phenotyping masks. **Left:** Comparison of the coverage of the 4 Garden Cycles. Note that the non-pruned garden has the highest value by day 60, with Cycle 2L (pruning shears) not far 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.

Garden Cycles 2L and 2R

For Garden Cycles 2L and 2R, we planted two identical seed placements (1.5m×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 segmentation masks for days 20, 30, 40, 50, and 60. Figure 7.1 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) on 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 five days.

Chapter 8

Limitations

The task of creating a fully autonomous garden is a daunting one. In many cases we had to simplify the problem statement and abstract away many complex natural phenomena. In this section, I will discuss a few of these simplifications as well as the shortcomings of current implementations.

In AlphaGardenSim, we found it extremely difficult to model both growth and water dynamics. While we implemented models that fit well in most scenarios and match the conditions we observed in our experimental testbed. In practice, the randomness of nature renders perfect prediction of growth and water dynamics impossible. For growth in particular, the seasons drastically impact the growth of plants (germination rates, growing periods, and the size of the plants themselves). However, we did not model variables like time of year, weather, temperature, or humidity in AlphaGardenSim; thus it is difficult to predict how plants will grow year-round. For consistency, we grew all cycles between the months of April and September. Because of this constraint on growing periods, the limited size of the physical testbed, and the long time constants of growing real-world plants, we were limited in our ability to run growth cycles, whether it be for tuning models in AlphaGardenSim, for testing pruning actions, or for carrying out autonomous runs.

The work presented in this report, particularly the physical AlphaGarden implementation, is largely centered around pruning and not irrigation. The motivation was that for a polyculture garden, the most essential cultivating action is pruning. For all growth and autonomous cycles presented in this work, all plants were watered with a fixed amount and thus we were not minimizing water usage. In current work, we address this issue by (1) using drip emitters to more precisely apply irrigation and (2) implementing a new plant water uptake model so we can more accurately predict how much water each plant consumes. Even with these implementations, it is still difficult to find a irrigation regiment that uses the least water and still enables plant growth — if the irrigation policy is too strict, we risk inhibiting the growth of the plants.

One key constraint of the AlphaGarden autonomous implementation is the top-down perspective. We monitor plants from above and consequently lack information regarding the internal density and the complete height map of the canopy. This abstraction was

created based on the metrics derived in our simulator (coverage and diversity) which use the overhead canopy coverage. While we are able to derive a state of the garden using this overhead perspective, the state is not fully complete. It is possible that height information and plant density would help the simulator choose more optimal pruning and irrigation actions.

Finally, the physical system is constrained by the FarmBot gantry system itself. The FarmBot allows us to execute pruning and irrigation actions by being able to travel to a set of (x, y, z) coordinates above the testbed. However, the pruning system we created was only feasible using servos. The system itself does not have the same versatility as a 6 degree of freedom (DOF) robotic manipulator, and thus cannot reach the inners of plants or have the capabilities to move branches out of the way to execute cuts. Thus, the pruning tools we created have a sole purpose of eliminating canopy coverage at the top layer.

As mentioned in Section 7.2, the Garden Cycles were not completely autonomous. All seeds had to be planted by hand and if multiple plants sprouted from one location, the excess ones were removed manually. When it came time to prune, the human bystander (myself) would upload the latest overhead photo and pass it through the pipeline. I would then monitor all pruning sequences and would reset/stop the FarmBot if it failed during motion or was in danger of colliding the pruning tool with the gantry system - a rare motion failure. If a leaf was fully cut, I would remove it from the testbed by hand. For partial or incomplete cuts, I would leave the leaf be.

Chapter 9

Conclusion

Despite recent advances in robotics and automation, automating a garden remains challenging. This report discusses the methods and results of the AlphaGarden project - a project with the goal of creating a fully autonomous polyculture garden. The development, tuning, and evaluation of AlphaGardenSim were first presented in the following publications [6, 4, 5]. We explored various automation policies that included irrigation and pruning actions and evaluated them using custom metrics. In Chapters 6 and 7, we translate the simulator to a real-world environment and explain the autonomous pipeline. The autonomous system is able to estimate the state of the garden using an overhead camera alongside a plant phenotyping and Bounding Disk Tracking algorithm. Once receiving actions from AlphaGardenSim, the FarmBot uses custom pruning tools to and pruning planners to execute the cuts. We present results for 4 60-day garden cycles in which AlphaGarden tends to a polyculture garden. The work involving the implementation and experiments of the AlphaGarden system is part of a paper that is under review for CASE 2022, by Presten et al.

As discussed in Chapter 8, creating a fully autonomous polyculture garden requires that some information be abstracted away and that methods be streamlined in order to achieve viable results. Stepping back, there are several key lessons that I learned while working on this project. The first being that this problem is solvable when executed in controlled environments; if these environments are highly controlled (weather, soil nutrients, humidity, plant species, growing area, etc.) then I anticipate the success of such an autonomous system will be even greater. With that being said, I believe that creating a versatile system that can translated to many different plant types and environments is difficult. For example, if a new plant type were to be used in the polyculture garden, its parameters in the simulator would have to be updated, which would require monitoring growth of this plant in test cycles, and the plant phenotyping network would need collected data for training. Another observation is that the system for pruning and irrigation has to be non-invasive and comprehensive enough to be able to reach all plants in an extremely compact environment. In sum, I believe systems like these are possible and have the potential to reduce water usage while increasing yield but are difficult to generalize and translate to unregulated environments.

The AlphaGarden project will continue in future work. We are currently in the process

of running a Garden Cycle in which a human gardener tends to one half the testbed, and the AlphaGarden system tends to the other half. Here, we plan to analyze water usage, canopy coverage, diversity, and overall plant health and record observations that are not included in the aforementioned metrics. We are actively working on integrating policies to optimize water usage using drip irrigation systems instead of hose irrigation. We also plan to explore closed-loop visual servoing for the Pruning Shears in order to improve the pruning success rate. The project has the potential to extend into further areas regarding seed placement algorithms that create planting arrangements to optimize water usage, plant symbiotic relationships, and yield, which could be used around the world in manual and autonomous systems alike. For code, videos, and datasets for the AlphaGarden project, see https://github.com/BerkeleyAutomation/AlphaGarden.

Bibliography

[1] Katarzyna Adamczewska-Sowińska and Józef Sowiński. "Polyculture Management: A Crucial System for Sustainable Agriculture Development". In: *Soil Health Restoration and Management*. Springer, 2020, pp. 279–319.

[2] K. R. Aravind and P. Raja. "Design and Simulation of Crop Monitoring Robot for Green House". In: (2016).

[3] AUTOLAB. *UC Berkeley Automation Laboratory*. 2022. URL: https://autolab.berkeley.edu/ (visited on 05/09/2022).

[4] 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.

[5] 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* (2022), pp. 1–13. DOI: 10.1109/TASE.2021.3138995.

[6] 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.

[7] 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.

[8] 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.

[9] Nived Chebrolu, Thomas Läbe, and Cyrill Stachniss. "Spatio-Temporal Non-Rigid Registration of 3D Point Clouds of Plants". In: ().

[10] 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.

[11] 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).

[12] Hanz Cuevas Velásquez et al. "Real-time Stereo Visual Servoing for Rose Pruning with Robotic Arm". In: May 2020, pp. 7050–7056. DOI: 10.1109/ICRA40945.2020.9197272.

[13] T Czárán and S Bartha. "The effect of spatial pattern on community dynamics; a comparison of simulated and field data". In: *Progress in theoretical vegetation science*. Springer, 1990, pp. 229–239.

[14] Christian Damgaard, Jacob Weiner, and Hisae Nagashima. "Modelling individual growth and competition in plant populations: growth curves of Chenopodium album at two densities". In: *Journal of Ecology* 90.4 (2002), pp. 666–671.

[15] 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.

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

[17] M. Erick et al. "Modeling and Simulation of Kinematics and Trajectory Planning of a Farmbot Cartesian Robot". In: *INTERCON* 1 (2018).

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

[19] FarmBot. *FarmBot*. 2021. URL: https://farm.bot/ (visited on 09/03/2021).

[20] Sandunika Fernando et al. "AI Based Greenhouse Farming Support System with Robotic Monitoring". In: *2020 IEEE REGION 10 CONFERENCE (TENCON)*. IEEE. 2020, pp. 1368–1373.

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

[22] Ken Goldberg. *The Robot in the Garden: Telerobotics and Telepistemology in the Age of the Internet*. Mit Press, 2001.

[23] Fang Gou, Martin K van Ittersum, and Wopke van der Werf. "Simulating potential growth in a relay-strip intercropping system: model description, calibration and testing". In: *Field Crops Research* 200 (2017), pp. 122–142.

[24] Novian Habibie et al. "Fruit mapping mobile robot on simulated agricultural area in Gazebo simulator using simultaneous localization and mapping (SLAM)". In: (2017).

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

[26] 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.

[27] 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.

[28] Moritoshi Iino, Chen Long, and Xiaojing Wang. "Auxin-and abscisic acid-dependent osmoregulation in protoplasts of Phaseolus vulgaris pulvini". In: *Plant and Cell Physiology* 42.11 (2001), pp. 1219–1227.

[29] Aaron L. Iverson et al. "REVIEW: Do polycultures promote win-wins or trade-offs in agricultural ecosystem services? A meta-analysis". In: *Journal of Applied Ecology* 51.6 (2014), pp. 1593–1602. DOI: 10.1111/1365-2664.12334. eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/1365-2664.12334. URL: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/1365-2664.12334.

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

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

[32] Joseph Santarromana Ken Goldberg. *The Telegarden*. 1995. URL: https://goldberg.berkeley.edu/garden/Ars/ (visited on 12/11/2019).

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

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

[35] *METER Environment*. METER. URL: https://www.metergroup.com/environment/ (visited on 10/29/2020).

[36] M. Minervini et al. *Plant Phenotyping Datasets*. 2015. URL: http://www.plant-phenotyping.org/datasets.

[37] Massimo Minervini et al. "Finely-grained annotated datasets for image-based plant phenotyping". In: *Pattern Recognition Letters* (2015), pp. -. ISSN: 0167-8655. DOI: http://dx.doi.org/10.1016/j.patrec.2015.10.013. URL: http://www.sciencedirect.com/science/article/pii/S0167865515003645.

[38] V A Murcia, J F Palacios, and G BarbieriEmail author. "FarmBot Simulator: Towards a Virtual Environment for Scaled Precision Agriculture". In: *SOHOMA* 987 (2021).

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

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

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

[42] J.E. Parker et al. "Companion planting and insect pest control". In: *Weed and Pest Control - Conventional and New Challenges* (Jan. 2013), pp. 1–30.

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

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

[45] Eric Rohmer, Surya PN Singh, and Marc Freese. "CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework". In: *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*. 2013.

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

[47] 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).

[48] 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).

[49] 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.

[50] 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.

[51] Nicola Strisciuglio et al. "TrimBot2020: an outdoor robot for automatic gardening". In: (Apr. 2018).

[52] Meixiu Tan et al. "Dynamic process-based modelling of crop growth and competitive water extraction in relay strip intercropping: Model development and application to wheat-maize intercropping". In: *Field Crops Research* 246 (2020), p. 107613.

[53] Kelly R Thorp and Kevin F Bronson. "A model-independent open-source geospatial tool for managing point-based environmental model simulations at multiple spatial locations". In: *Environmental modelling & software* 50 (2013), pp. 25–36.

[54] 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.

[55] E.J. Van Henten et al. "An Autonomous Robot for Harvesting Cucumbers in Greenhouses." In: *Auton. Robots* 13 (Nov. 2002), pp. 241–258. DOI: 10.1023/A:1020568125418.

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

[57] Marius Wiggert et al. "RAPID-MOLT: A Meso-scale, Open-source, Low-cost Testbed for Robot Assisted Precision Irrigation and Delivery". In: *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2019, pp. 1489–1496.

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

[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] Yang Yu. "Crop yields in intercropping: meta-analysis and virtual plant modelling". PhD thesis. Wageningen University, 2016.

[61] 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.

[62] Matthew D Zeiler. "Adadelta: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).