

Industrial Applications of Remote Monitoring, Learning, and Actuation

Shrey Aeron



Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-91

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-91.html>

May 10, 2024

Copyright © 2024, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Acknowledgement

I extend my deepest thanks to my advisor, Professor Ken Goldberg. Your guidance and support have been crucial for my growth as a person and researcher over my last several at AUTOLab.

I am also grateful to Simeon Adebola, Ryan Hoque, Mark Presten, and Rishi Parikh, who have been wonderful mentors throughout my time at the lab, and our collaborators at Google and Siemens, who were instrumental in the research conducted and their suggestions and deep involvement. Many labmates have been crucial to my work in AUTOLab, including Sandeep Mukherjee, Ethan Qui, Kaushik Shivakumar, and Kishore Srinivas.

Industrial Applications of Remote Monitoring, Learning, and Actuation

by

Shrey Aeron

A thesis submitted in partial satisfaction of the
requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ken Goldberg, Chair
Professor S. Shankar Sastry

Spring 2024

Industrial Applications of Remote Monitoring, Learning, and Actuation

Copyright 2024
by
Shrey Aeron

Industrial Applications of Remote Monitoring, Learning, and Actuation


by Shrey Aeron

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:

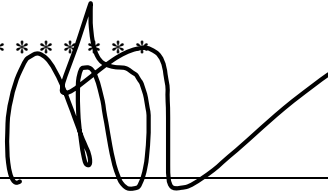
Committee:

DocuSigned by:

32B42AC2D1E3498

Professor Ken Goldberg
Research Advisor

5/10/2024

* * * * *



Professor S. Shankar Sastry
Second Reader

May 2, 2024

Abstract

Industrial Applications of Remote Monitoring, Learning, and Actuation

by

Shrey Aeron

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Ken Goldberg, Chair

0.1 AlphaGarden

Remote Sensing applies in a variety of situations, ranging from monitoring the environment, generating actionable insights, and object classification and detection techniques.

AlphaGarden is an automated test-bed for indoor polyculture farming with a gantry robot, sensors, plant and leaf phenotyping, growth cycle simulator, custom tools. Image data from an overhead camera along with soil sensors from the testbed are analyzed using a Tracking and Phenotyping network, estimating the garden’s state at each time step and being fed into the simulator to decide on the next set of pruning and irrigation actions to take to maximize coverage and diversity of the garden. Results suggest the system can autonomously achieve 94% normalized plant diversity with pruning shears while maintaining an average canopy coverage of 84% by the end of the cycles.

0.2 Push-MOG

Recently, robots have seen rapidly increasing use in homes and warehouses to declutter by collecting objects from a planar surface and placing them into a container. While current techniques grasp objects individually, Multi-Object Grasping (MOG) can improve efficiency by increasing the average number of objects grasped per trip (OpT). However, grasping multiple objects requires the objects to be aligned and in close proximity. In this work, we propose *Push-MOG*, an algorithm that computes “fork pushing” actions using a parallel-jaw gripper to create graspable object clusters. In physical decluttering experiments, we find that Push-MOG enables multi-object grasps, increasing the average OpT by 34%.

This thesis explores the nexus of these various Remote Actuation and Sensing tasks, from macro to micro scale.

To my parents and sister,

Contents

0.1	AlphaGarden	1
0.2	Push-MOG	1
	Contents	ii
	List of Figures	iv
	List of Tables	vii
I	AlphaGarden	1
1	Introduction - AlphaGarden	2
2	Related Work	4
2.1	Automated Polyculture Gardening	4
2.2	Plant Growth Simulators	4
2.3	Perception Systems	4
2.4	Automated Pruning	5
2.5	Irrigation	5
2.6	Hardware	5
3	Problem Definition	6
4	State Estimation	8
4.1	Plant Phenotyping (species tracking)	8
4.2	Plant Leaf Tracking	9
4.3	Pruning Software	11
5	Simulation	15
5.1	Pruning, Irrigation, and Planting Policies	17
6	Actuation	21
6.1	System	21

7 Experiments	27
7.1 Isolated Pruning Experiments	27
7.2 4 Garden Cycles	27
8 Limitations and Future Work	31
9 Advanced Segmentation	32
9.1 Introduction	32
9.2 Related Work	32
9.3 Qualitative Results	33
II Push-MOG	37
10 Introduction - Push-MOG	38
11 Related Work	41
11.1 Multi-object grasps	41
11.2 Pushing	41
11.3 Point Clustering	42
12 Problem Statement	43
12.1 State, Action, and Objective	43
12.2 The Clustering Problem	44
12.3 The Pushing and Grasping Problems	45
13 Push-MOG Algorithm	46
13.1 Clustering Details (Alg 2)	46
13.2 Push Planning (Alg 3)	46
13.3 MOG-Net Integration	47
14 Experiments and Results	50
14.1 Experimental Setup	50
14.2 Baselines	51
14.3 Results	51
15 Limitations and Future Work	53
Bibliography	55

List of Figures

- 1.1 **Phenotyping and Bounding Disk Tracking.** 3 images from days 20, 30, and 40 of garden cycle 4. **Top row:** overhead images overlaid with the estimated bounding disks from the Bounding Disk Tracking algorithm. **Bottom row:** the masks created by the Plant Phenotyping network 3
- 4.1 **Circle IoU for days 15, 30, 45 of Cycle 2R.** Each of these figures **K-Means (top)**, **BFS (bottom)** show average IoU Score by plant type of the method given the ground truth circles. Note that with a highly occluded plant like Raddichio, neither method is able to match ground truth effectively, because of the sparsity of the segmentation mask. 10
- 4.2 **Plant Tracking Metrics.** Here we demonstrate the adversarial nature of precision and recall on a single Borage plant. Consider three circles. **Left:** This circle fully optimizes for precision, but neglects recall and has a 100% precision and 6.9% recall. **Center:** This circle fully optimizes for recall and has a 19.8% precision and 100% recall. **Right:** This circle balances both precision and recall achieving a 32.2% precision and 98.8% recall. It is often necessary to sacrifice more recall to achieve a qualitatively better bounding disk because encompassing all pixels of a plant leaves the algorithm brittle to minor segmentation errors. 11
- 4.3 **Metrics of Garden Cycle 2R for Kale and Cilantro.** We evaluate precision and recall of the Breadth-First-Search (BFS) versus the K-Means bounding disk algorithms for Kale, a larger plant type, and Cilantro, a smaller plant type. **Kale:** BFS tends to have higher precision, but lower recall. For the days which ground truth circles exist (manually annotated), they are closer to the K-Means algorithm in both metrics. **Cilantro:** Similarly, BFS has a higher precision and K-Means has a higher recall. However, Cilantro generally benefits from the more conservative BFS. We adopt a mixed approach: the K-Means from timesteps 0 to 30 when there is qualitatively less occlusion, and the BFS approach for subsequent timesteps, when the prior helps improves robustness. 12
- 4.4 **Domain Adversarial Neural Network (DANN):** As presented in [24], The Prune Point Network utilizes a DANN with two networks that have shared weights. 12

4.5	Prune Point Identification. Example of all plant leaf centers that were identified by the baseline analytic algorithm (left) and the learned model (right) applied to an overhead image. Each point’s color corresponds to the plant type of leaf for which it was intended. The learned model identifies 3 more prune points than the baseline approach in the highlighted example.	14
4.6	Visual Servoing Algorithm	14
5.1	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.	16
5.2	Learned Pruning network architecture. A deep convolutional neural network 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.	19
5.3	Adaptive Sector Sampling. In [22], we introduced a sector observation method which sampled sectors centered around each seed location $s(x, y)$. However, this can lead to overlap in which multiple irrigation and pruning actions are taken when instead only one is needed. Left: Two plants are close to each other, so sectors centered around each of their $s(x, y)$ overlap. Right: With adaptive sector sampling, if the plants are both germinating or are both growing, their seed locations are clustered and a sector centered at the center of the cluster is instead observed.	20
6.1	Custom Irrigation Tool: For the watering of plants while reducing erosion.	22
6.2	Overhead Camera: The Sony SNC-VB77 camera is used to monitor the AlphaGarden. It’s characteristics include a 35mm sensor, a low aberration lens, and higher than 4K resolution to allow for crisp and clear images for monitoring.	23
6.3	FarmBot Gantry: This is the FarmBot gantry robot of the experiments, which can automatically mount tools to itself and move along the X,Y,Z axes.	23
6.4	Moisture Sensors: The TEROS 10 sensors are laid throughout the garden bet to measure and model soil metrics and dynamics.	24
6.5	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.	24

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. 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.	30
9.1	Extracted meshes from SuGaR. Two growth cycle days of January 8th, 2024, and February 2nd, 2024 were converted to a mesh using SuGar and then segmented using Blender’s 3D editing tools [69].	34
9.2	Output of 3D Prompted Segmentation on Plants. We use GARField [14] to manually define a prompt point to segment.	35
10.1	An example of Push-MOG pushing and grasping objects into a cluster. (1) A cluster is identified containing three objects; (2) two objects are moved toward the center object using fork pushing. (3) Once the cluster fits within the robot’s gripper, (4) the gripper closes to create a tight fit and grasps the objects that will be transported to the bin.	39
12.1	As illustrated in Fig. 10.1, Push-MOG uses “fork pushes” to consolidate objects for multi-object grasping as shown on the right.	45
13.1	Examples of fork pushing with a parallel-jaw gripper, both against an edge ((a) and (d)) and around a vertex ((b) and (c)) noticing that the direction the object will move to is roughly perpendicular to the parallel-jaw gripper.	49
15.1	This figure shows the clustering formed by Push-MOG of an initial scene. The numbers on each cluster identify the order that the algorithm will perform the pushing and grasping actions.	54

List of Tables

6.1	Isolated Pruning Experiments for the Rotary Pruner and Pruning Shears. Key: <i>Completeness</i> - 3: complete cut, 2: partial cut, 1: missed cut. <i>Precision</i> - 1: no damage to other leaves, 0: damage to other leaves. <i>Error Type</i> - A: No error, B: location, C: depth, D: other (including pruner).	26
7.1	Plant Type Metrics for Garden Cycles 1L & 1R. This table shows diversity and coverage for plant types on day 60. The values for Cycle 1L (not pruned) and Cycle 1R (pruned with Rotary Pruner) are calculated via $[c_i(60) * (R/R_i)^2]$ for each plant type (Section III). The goal of pruning is to foster a diverse garden while maintaining a high coverage.	29
14.1	Physical decluttering experimental results for random scenes, comparing baselines Frictional SOG , MOG-Net , and the method Push-MOG . Experiments were conducted with a Robotiq 2F-85 parallel-jaw gripper mounted on a UR5 robot arm, on 5 random scenes of 34 objects, with the goal placed directly adjacent (see Figs 12.1 and 15.1).	52

Acknowledgments

I extend my deepest thanks to my advisor, Professor Ken Goldberg. Your guidance and support have been crucial for my growth as a person and researcher over my last several at AUTOLab.

I am also grateful to Simeon Adebola, Ryan Hoque, Mark Presten, and Rishi Parikh, who have been wonderful mentors throughout my time at the lab, and our collaborators at Google and Siemens, who were instrumental in the research conducted and their suggestions and deep involvement. Many labmates have been crucial to my work in AUTOLab, including Sandeep Mukherjee, Ethan Qui, Kaushik Shivakumar, and Kishore Srinivas.

Part I
AlphaGarden

Chapter 1

Introduction - AlphaGarden

The idea of a remotely controlled garden started in 1995 with the inception of the telegarden by Goldberg et al. [1], [2], where the public could interact with a remote garden through the internet by planting and watering plants. Since then, the advancement of the idea and technology available led to the next natural step of a fully automated garden, helping to tackle the space and labor issues in addition to the cons of monoculture farming [3], [4]. Precision polyculture at large scales could revolutionize how society interacts with food and reduce our dependence on pesticides, water usage, and fertilizers through the natural benefits of polyculture [5]–[8].

Polyculture involves the planting of multiple crops in one area, similar to the “three sisters” crops of corn, beans, and squash. This contrasts with monoculture, which refers to a single species of vegetation grown in an area. Polyculture is not only more labor intensive, but it is more difficult to automate due to the heterogeneous composition of the crop, requiring more processing to understand the scene. However, it also provides many benefits, such as natural pest protection, better yields, and reduced depletion of soil nutrients.

The use of a robot could increase output of crops per unit area while preserving already scarce water supplies and reducing the negative health effects of intensive labor in harsh environments.

AlphaGarden’s contributions fall into the following categories:

1. Simulation
2. Advanced State Estimation
3. Actuation in Real

The AlphaGarden section presents work from the following papers with various contributions from me:

1. Simulating Polyculture Farming to Learn Automation Policies for Plant Diversity and Precision Irrigation [9]



Figure 1.1: **Phenotyping and Bounding Disk Tracking.** 3 images from days 20, 30, and 40 of garden cycle 4. **Top row:** overhead images overlaid with the estimated bounding disks from the Bounding Disk Tracking algorithm. **Bottom row:** the masks created by the Plant Phenotyping network .

2. Can machines garden? systematically comparing the AlphaGarden vs. Professional horticulturalists [10]
3. Automated Pruning and Irrigation of Polyculture Plants [11]
4. Automated pruning of polyculture plants [12]

My primary contributions to this project has been the development of a new simulator policy to reduce water and pruning actions [9] and plant leaf masking and detection [10]. I will also discuss [10], which pits the AlphaGarden against a team of seasoned horticulturalists to understand its performance and readiness for future work. As an addendum, I will share techniques for advanced state estimation and segmentation using Gaussian Splatting [13] and GARField [14] to understand the complex garden bed in a three-dimensional view, as compared to an overhead one, and extract more detailed and ground truth information. Future work could build on this to design more robust growth models.

Chapter 2

Related Work

2.1 Automated Polyculture Gardening

Automated polyculture gardening is difficult, and there exists much prior work to address this. Wiggert et al. [15] created a testbed for monitoring plant growth and water stress in real time. Correll et al. [16] designed a distributed autonomous gardening system with mobile manipulators that detect plants, irrigate, and grasp fruits. Agostini et al. [17] created a robotic system that used AI to decide specific treatments (irrigation and nutrient intake) for each individual plant. Similarly, Shirley et al. [18], introduce a robotic system to tend cherry tomatoes where the system uses sensors to monitor and maintain a database of plant nutrients, environmental conditions, and fruit locations. This paper focuses on polyculture pruning and irrigation.

2.2 Plant Growth Simulators

Many existing plant simulators model growth of single species [19]. Examples such as DSSAT [20] and AquaCrop [21] simulate a growth period in large-scale monoculture farms. In prior work, we presented AlphaGardenSim [9], [22], [23] - a polyculture garden simulator with first order models of plant growth, inter-plant dynamics, and competition for water and light. The model parameters used in AlphaGardenSim were tuned using real-world measurements from a physical garden testbed as described in [9], [23]. AlphaGardenSim was used to automate seed placement for plant diversity and coverage.

2.3 Perception Systems

Identifying plant species-phenotyping- is an important task for plant monitoring. Ayalew et al. [24] present a method to use an unsupervised domain adaptation network to adapt the meticulously pre-labeled Computer Vision Problems in Plant Phenotyping (CVPPP)

dataset [25], [26] to other plant and image domains. The data consists of single plants, their leaves, and a point map of leaf centers. We build on this work by transferring the results to a polyculture setting, as discussed in Section 4.3.

2.4 Automated Pruning

Pruning is a necessary capability to cultivate a polyculture garden. Prior work in autonomous pruning includes rose and bush trimming with a robot arm [27], [28]. Habibie et al. [29] trained a Simultaneous Localization and Mapping (SLAM) algorithm to enable automated fruit harvesting in a red apple tree field. Cuevas-Velasquez et al. [27] demonstrated success using visual servoing to account for changes in stem poses to determine cutting points.

In a controlled greenhouse, Van Henten et al. [30] 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 system for autonomously pruning a polyculture garden.

2.5 Irrigation

Automation of irrigation is being explored and deployed in various ways ranging from simple Internet-of-Things(IoT)-based solutions in low-income and middle-income countries[31], [32] to robotics solutions in place of manual labor in high-income countries[31]. In contrast to rainfed agriculture [33], irrigation is essential when considering semi-indoor and indoor farming. Irrigation in agriculture contributes to more than 70% of global water use, and yet increasing demand as world population grows results in increasing water scarcity [33], [34]. Sustainable irrigation practices in agriculture are therefore important[34]. In Alpha-GardenSim, we work on improving irrigation to optimize coverage and diversity further while reducing water usage. We present some of our ongoing work with irrigation in this paper.

2.6 Hardware

For robot hardware, we use FarmBot, an open source gantry robot commercially available since 2016 [9]. Prior work with this system has examined kinematic modeling to enhance trajectory planning [35]. A team from Telkom University used FarmBot to create a web application to help human users with seed planting, watering, and plant monitoring routines [36]. More recently, researchers have proposed a FarmBot simulator “to support the development of a control software able to implement different [precision agriculture] strategies” [37]. We combine a more detailed simulator with the FarmBot hardware and custom pruning and irrigation tools to tend a polyculture garden from planting, to germination, growth, reproduction and decay.

Chapter 3

Problem Definition

A Garden Cycle consists of planting an arrangement of selected plant types, then irrigating and pruning until growth is completed. The objective is to optimize coverage, plant diversity, and water usage. In this paper, we focus on maximizing coverage and diversity through pruning and irrigation actions.

A growing zone is a rectangular planter bed of size (w, h) in cm . A seed placement algorithm is used to generate seed placements [38]. There are M plant species σ_1 to σ_m . Each species has an associated germination time g , maturation time m and maximum radius R . The lifecycle of each plant k is defined by four stages: germination, growth $l_i := m_i - g_i$ (the time between a plant type i 's germination and maturation), waiting, and wilting, as defined in [22]. A growth cycle starts with a seed placement for k seeds: $p_1(x, y, \sigma)$ to $p_k(x, y, \sigma)$ - and some fraction germinate to produce up to k plants. We track the plants each day t for $t = 1$ to t_{max} . For each plant $k \in [0, n)$, the plant has its center coordinates (cx_k, cy_k) and current radius r_k , both in cm . Each plant k also has a corresponding plant type σ_i . A garden state on day t includes all information described above for every plant $k \in [0, n)$ at day t . Thus, a garden state is defined as follows:

$$s(t) = \{p_k : ((cx_k, cy_k), r_k), \dots\}, k \in [0, n)$$

We define coverage, $c(t)$, as the sum of all plant canopy coverage, $\sum_i c_i(t)$, over total area $w \cdot h$ at day t [39]. For each plant type i with maximum radius R_i , we define garden diversity as follows; let

$$\begin{aligned} \mathbf{v}(t) &= [c_i(t) \cdot (R/R_i)^2, \dots], \forall i \\ d(t) &= H(\mathbf{v}(t)) \end{aligned}$$

where $H(\cdot)$ is a function that calculates entropy defined as:

$$H(X) := - \sum_{x \in X} p(x) \log p(x) [40],$$

$\mathbf{v}(t)$ is a vector of normalized plant type coverage, and R is the average maximum radius over all plant types. Multiplying $c_i(t)$ by $(R/R_i)^2$ normalizes each plant type's canopy coverage.

We normalize because it is unrealistic to assume that a smaller, less dominant species will have the same coverage as a much larger, faster growing species. We aim to maximize both $c(t)$ and $d(t)$ at peak growth point through pruning actions. Coverage, $c(t)$, quantifies how well the plants are using the available space and Diversity, $d(t)$, tells us the rate we want all plant types to grow at approximately.

We assume:

1. The robot can reach all points on all plants,
2. only the planted seeds sprout in the garden (there are no extraneous plants).

Chapter 4

State Estimation

4.1 Plant Phenotyping (species tracking)

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 (Section 4.2), and provides information on plant growth, diversity, and coverage.

For each day, the latest image from the Sony camera is pulled from the server to be used for phenotyping. The images used are taken in the evening (usually around 6pm/7pm) when there is uniform lighting without shadows.

We trained a model using UNet architecture [41] with a ResNet34 [42] 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 image patches from six hand-labeled overhead images from previous garden cycles.

Each image is split into 512×512 RGB patches and augmented via shifting and rotating. We extract leaf masks from various stages in the garden and overlay these leaves on top of the existing patches to augment the data set.

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. By doing this, we are not bottlenecked by hand labelling and can generate training data from multiple garden cycles to improve overall performance.

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

To address the distribution shift in later days, we introduce a prior probability distribution pixel level occupancy grid based on seed placement and plant maximum radius given

from our tuned simulator [22]. We define a variable R_t^k , and c_t^k as the maximum radius and center of plant k at timestep t , and a $1630 \times 3478 \times (i_{total} + 1)$ occupancy grid, O defined as:

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

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

We use this occupancy grid as a prior probability distribution, and compute a new likelihood grid L' as an element-wise multiplication of the original segmentation output, L , and occupancy grid, O , $L'(x, y, i) = L(x, y, i) * O(x, y, i)$, and output $\max_i L'(x, y, i)$ as the predicted label for (x, y) .

4.2 Plant Leaf Tracking

Tracking plant leaves over the lifespan of a garden is challenging because plant shape frequently changes day-to-day due to occlusion, biasing an overhead view of the data. A tracking method should therefore be robust to these temporary occlusions and slight changes. However, the algorithm should be sensitive to real changes in plant location due to phototrophy [43] and irrigation [44].

We define the plant bounding disk (see Fig. 1.1) as the circle with the smallest radius such that the canopy corresponding to that plant is enclosed. We present two methods for finding circular representations of plants state and two metrics for comparison, and evaluate each method against a hand-labeled benchmark for selected days using a circle Intersection over Union (IoU) loss [45]. Recall that we know the seed location for each plant.

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

We use a breadth-first-search (BFS) based algorithm and K-Means clustering to track each plant center and radius. Both algorithms help address the issues with tracking plant 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. We can see this in Figure 4.1 by comparing each algorithm against a human-labeled circle. K-Means out performs BFS early and late in the cycle when plants are not occluded or are qualitatively highly occluded, but underperforms BFS in the middle of the cycle when plants are only qualitatively moderately occluded.

The BFS algorithm is initialized with seed locations and all plant radii at $0cm$. At each timestep, we use AlphaGardenSim [22] 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.

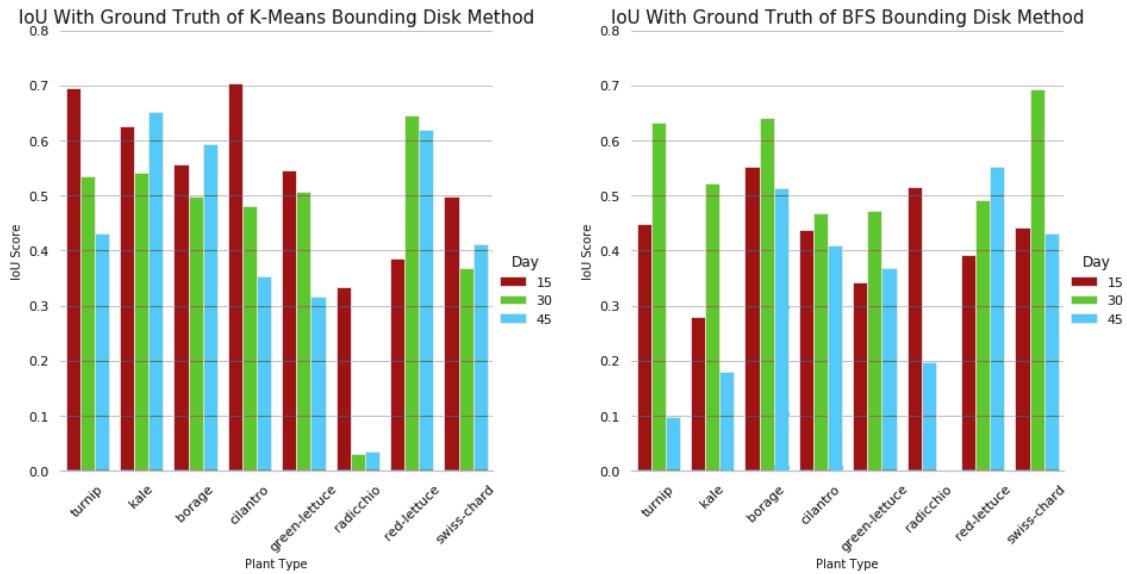


Figure 4.1: **Circle IoU for days 15, 30, 45 of Cycle 2R.** Each of these figures **K-Means (top)**, **BFS (bottom)** show average IoU Score by plant type of the method given the ground truth circles. Note that with a highly occluded plant like Raddichio, neither method is able to match ground truth effectively, because of the sparsity of the segmentation mask.

The algorithm terminates when it has explored no new pixels of the target type in the past two circumferences (representing the end of the plant) or the maximum radius has been achieved. This process repeats each day for each plant. Even when a plant becomes fully occluded, the algorithm handles radial decrease using AlphaGardenSim’s tuned wilting parameters.

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

In order to benchmark the performance of these methods we use statistical precision and recall. Both precision and recall are computed per plant type per timestep. Let P_i be the number of pixels in the segmentation mask of the inputted plant type that fall within at least one bounding disk—the true positives. P_t be the number of pixels of the given plant type present in the segmentation mask—the true positives and false negatives, and P_c be the area of the union of the fitted bounding disks—the true and false positives. We then have that Precision = $\frac{P_i}{P_c}$ and Recall = $\frac{P_i}{P_t}$.

Just as in classification, we wish to maximize both precision and recall to compute the optimal bounding disks. We, however, need to consider the tradeoffs between precision and

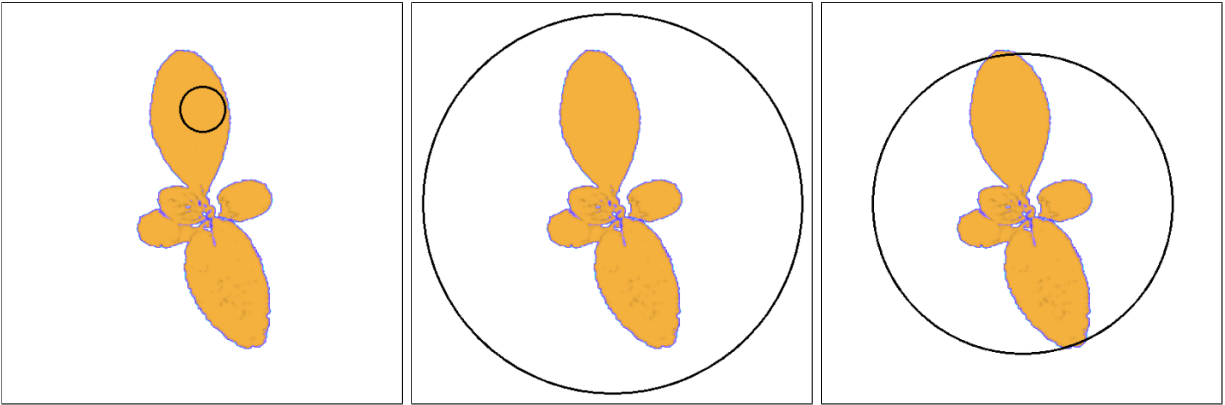


Figure 4.2: **Plant Tracking Metrics.** Here we demonstrate the adversarial nature of precision and recall on a single Boraginaceae plant. Consider three circles. **Left:** This circle fully optimizes for precision, but neglects recall and has a 100% precision and 6.9% recall. **Center:** This circle fully optimizes for recall and has a 19.8% precision and 100% recall. **Right:** This circle balances both precision and recall achieving a 32.2% precision and 98.8% recall. It is often necessary to sacrifice more recall to achieve a qualitatively better bounding disk because encompassing all pixels of a plant leaves the algorithm brittle to minor segmentation errors.

recall in our setting; smaller bounding disks tend to have higher precision because they will likely be centered around denser, less occluded portions of the plant. However, larger bounding disks will tend to have higher recall because a larger bounding disk will naturally have a larger portion of plant type k 's pixels (see Fig. 4.2).

To judge the efficacy of these methods we compare them to hand-labeled bounding disks at various time steps. As Fig. 4.3 (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, K-Means' efficacy decreases as it overfits to segmentation errors and irregular plant shapes. As Fig. 4.3 (right) shows, BFS lags early on, but then becomes increasingly effective as plants are occluded mid-garden cycle. We adopt a mixed approach: the K-Means from timesteps 0 to 30 when there is qualitatively less occlusion, and the BFS approach for subsequent timesteps, when the prior helps improves robustness.

4.3 Pruning Software

Once a garden state for day t is estimated with the Bounding Disk Tracking algorithm, the analytic policy within AlphaGardenSim decides which plant to prune[9]. For autonomous pruning, the system must identify and select specific target leaves to prune, be able to navigate and position the FarmBot above the chosen leaf using visual servoing, and execute the pruning action with custom hardware.

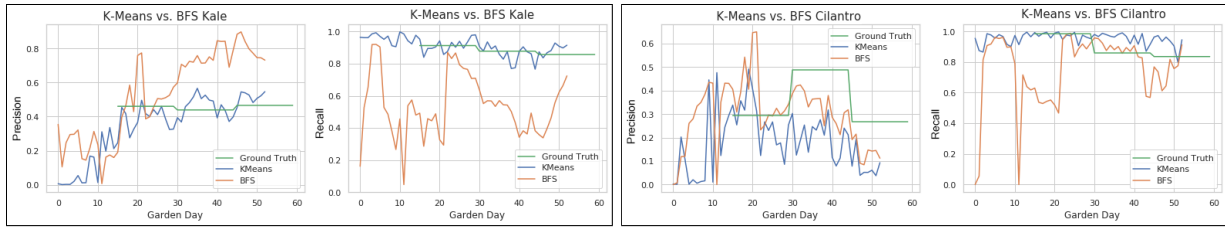


Figure 4.3: **Metrics of Garden Cycle 2R for Kale and Cilantro.** We evaluate precision and recall of the Breadth-First-Search (BFS) versus the K-Means bounding disk algorithms for Kale, a larger plant type, and Cilantro, a smaller plant type. **Kale:** BFS tends to have higher precision, but lower recall. For the days which ground truth circles exist (manually annotated), they are closer to the K-Means algorithm in both metrics. **Cilantro:** Similarly, BFS has a higher precision and K-Means has a higher recall. However, Cilantro generally benefits from the more conservative BFS. We adopt a mixed approach: the K-Means from timesteps 0 to 30 when there is qualitatively less occlusion, and the BFS approach for subsequent timesteps, when the prior helps improves robustness.

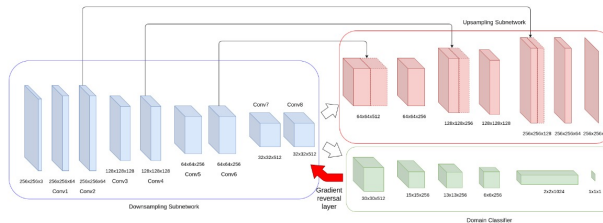


Figure 4.4: **Domain Adversarial Neural Network (DANN):** As presented in [24], The Prune Point Network utilizes a DANN with two networks that have shared weights.

Prune Point Identification

We next present a prune point identification algorithm that takes in an overhead image, segmentation mask, and list of plants to prune, and outputs (x, y) prune point coordinates. A good prune point can help promote growth and diversity in the garden. Pruning closer to the center of the bounding circle of the plant is beneficial because it allows for the FarmBot to cut off a greater portion of the leaf.

Initially, an analytic baseline algorithm finds the average point between the extrema of the plant bounding disk and its’ center to find a theoretical leaf center(see Fig. 4.5). However, plants often include bending, occlusion, or oddly shaped leaves. The baseline algorithm would frequently return points that were not on a plant or too close to an edge. We therefore explore a learned approach.

We train a Prune Point Identification neural network based on the unsupervised domain adaptation network for plant organ counting by Ayalew et al. [24], whose architecture is seen in Figure 4.4. In the training process, singular plant input images are transformed

to a size of 256x256, and masked using the phenotyping network output. The architecture consists of a Domain Adversarial Neural Network (DANN) with a Gradient Reversal Layer to backpropagate between the source and target domains. Classification is performed using a U-Net [24].

DANNs are unique as they allow for adaptation to a new domain that does not contain any ground truth data. The Komatsuna Dataset [46] is a pre-labeled and masked set of plants with their leaves over time, very similar to the masked images in AlphaGarden. The dataset is used as the base of the network, with masked and cropped images of plants from the Phenotyping Network as inputs.

To evaluate this network’s success in a polyculture setting, rather than its original monoculture domain, we trained one model, ignoring plant type. We found that training on all plant types led to lower overall performance. However, training on 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 able to predict leaf centers for all the plant types. The final model was trained for 150 epochs with a 80/20 train/validation split for the source (CVPPP) [25], [26] and target datasets, 201 overhead images and masks of the Borage plant type, and evaluated qualitatively on a random sampling of overhead images of all plant types.

The model outputs a probability map with all possible plant leaf centers. To extract viable points from the map, a clustering and thresholding technique is used to identify leaf centers with the highest model confidence. These point clusters are then removed and the heatmap is re-normalized. In this way, the algorithm is able to recover lower confidence leaf centers. Additionally, prune points are filtered to remove any points on other plants or the soil. This is done by correlating the prune point to the mask provided by the phenotyping network.

Together, the learned method identifies 32% more prune points on plant leaves than the baseline (see Fig. 4.5). The identified prune points are an average of 38% closer to the center of the plant compared to the baseline. Quantitatively, as shown in Fig. 4.5, for Swiss Chard as an example, the learned method identifies six prune points while the baseline approach identifies only three prune points.

To select the optimal prune point, the network first identifies all possible prune points. The algorithm then eliminates all points within 3cm of the edge of the bounding disk as these lie on the edge of the plant, and calculates the rate of change of the radii of all neighboring plants over the last five days. The prune point closest to the neighboring plant that has the largest rate of decay of radii is selected to foster growth of the struggling neighboring plant.

Visual Servoing for Pruning Tool

The autonomous pruning tool 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.

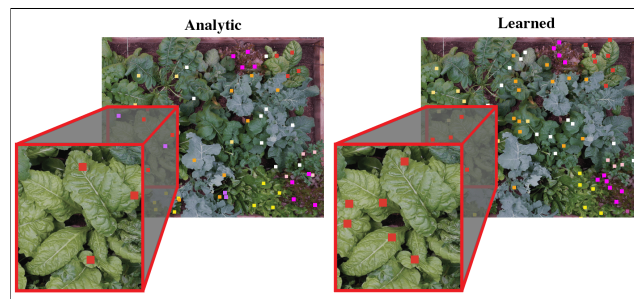


Figure 4.5: **Prune Point Identification.** Example of all plant leaf centers that were identified by the baseline analytic algorithm (left) and the learned model (right) applied to an overhead image. Each point’s color corresponds to the plant type of leaf for which it was intended. The learned model identifies 3 more prune points than the baseline approach in the highlighted example.



Figure 4.6: **Visual Servoing Algorithm**

We use a closed-loop hybrid visual servoing technique that combines elements of both image-based visual servoing and position-based visual servoing [47].

Fig. 4.6 shows the overhead image and the image taken by the on-board camera at its final location after visual servoing to reach selected points in two instances.

The visual servoing algorithm can fail when plants grew too high and close to the on-board camera (approximately $0.4m$ above the soil surface).

To remedy this, we moved the on-board camera to approximately $0.7m$ above the soil surface, allowing it to capture unobstructed images of the garden and better localize them within the global image.

Chapter 5

Simulation

In AlphaGardenSim 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 $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O})$.

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 [9]. 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 [22] with a new planting action that seeds a plant at point (x, y) at timestep t . A plant can be planted only in locations labeled as *soil*.

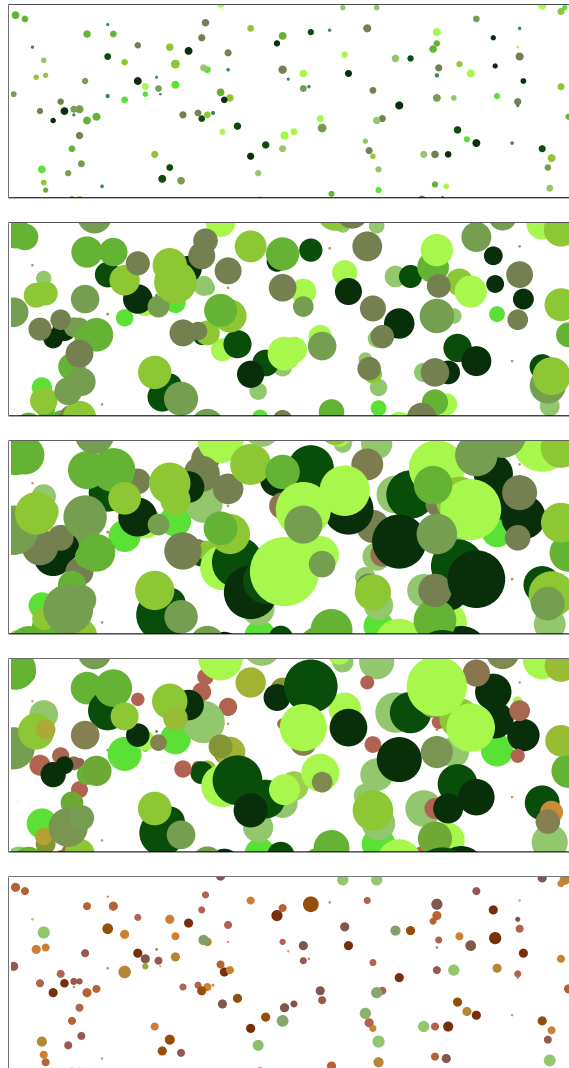


Figure 5.1: **Plant Life Stages.** Each plant is modeled with a life cycle trajectory, consisting of five stages (from top to bottom image): germination, vegetative, reproductive, senescence, and death. When plants get underwatered or overwatered, their radius decays exponentially and their color turns 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.

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 [9].

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 .

5.1 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 [22], 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 [22], 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 described in [9] 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 [23] and those in Section [9] 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 [9]) that 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 $\mathbf{o}(x, y, t)$.

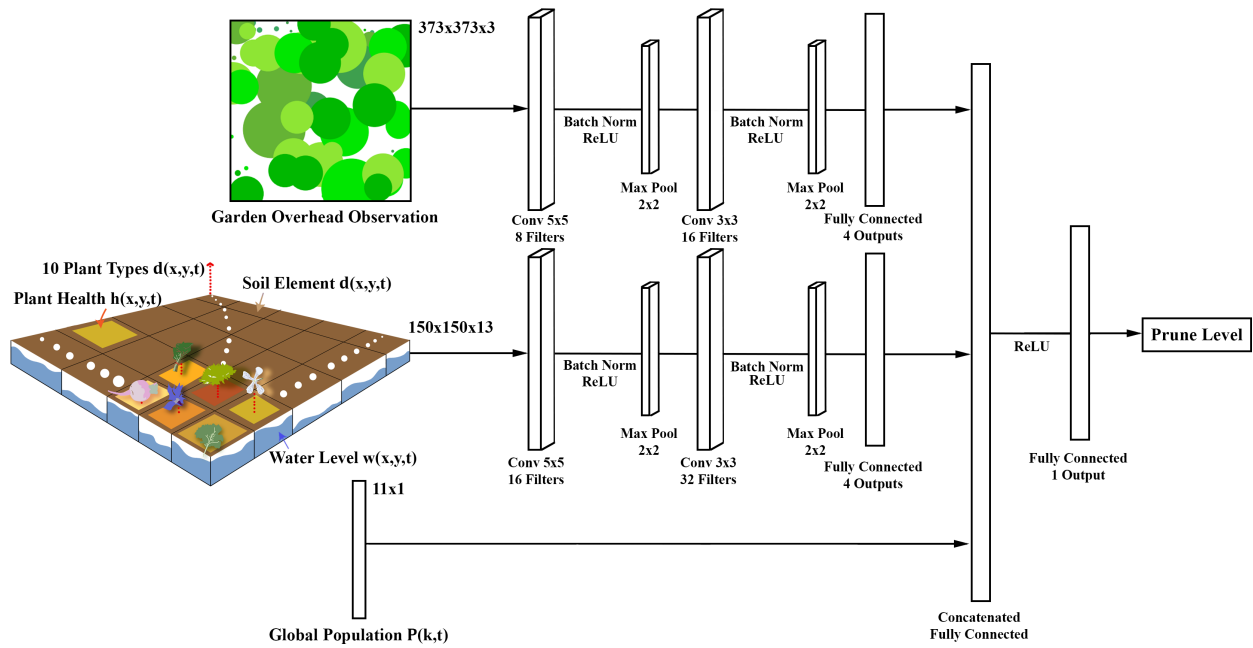


Figure 5.2: Learned Pruning network architecture. A deep convolutional neural network with 18,244 parameters. The network takes three inputs: 1) an RGB image of the full garden; 2) a matrix of $h(x, y, t)$, $w(x, y, t)$ and $\mathbf{d}(x, y, t)$ for all (x, y) in the garden; 3) the global population distribution $\mathbf{P}(k, t)$ including soil coverage. The network predicts a prune level for each observation using demonstrations from Variable Pruning.

Learned Pruning

We introduced Learned Pruning in [23], 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 Fig. 5.2. 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 the new planting action defined in Section 5 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

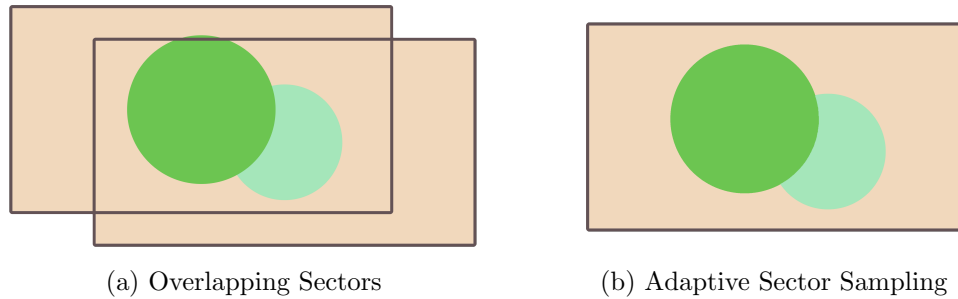


Figure 5.3: **Adaptive Sector Sampling.** In [22], we introduced a sector observation method which sampled sectors centered around each seed location $s(x, y)$. However, this can lead to overlap in which multiple irrigation and pruning actions are taken when instead only one is needed. **Left:** Two plants are close to each other, so sectors centered around each of their $s(x, y)$ overlap. **Right:** With adaptive sector sampling, if the plants are both germinating or are both growing, their seed locations are clustered and a sector centered at the center of the cluster is instead observed.

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 [22], 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, as seen in Fig. 5.3, 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 6

Actuation

6.1 System

Hardware

FarmBot Gantry Robot: AlphaGarden is a $3.0m \times 1.5m$ raised planter bed located in the UC Berkeley greenhouse, seen in Figure 6.3. A commercial FarmBot [48] gantry robot is installed over the planter bed frame. This CNC robot may travel to any location in the garden from the soil level to $0.4m$ above. 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.

Water Nozzle To build a functional autonomous garden, we designed and fabricated custom tools for watering and pruning, as seen in Figures 6.1 and 6.5, respectively. The watering nozzle creates a wide and uniform output stream [23], for soil water retention. The watering nozzle attaches to the UTM.

Overhead Camera We mounted a Sony SNC-VB770 digital camera [49] with a $20mm$ Sony lens [50] $2m$ above the garden bed to monitor AlphaGarden, as seen in Figure 6.2. The camera's major requirements include (1) image resolution, (2) image distortion, (3) power delivery, (4) remote data accessibility, and (5) reliability. It has a DSLM $35mm$ sensor with a maximum 4240×2832 resolution (1.4x higher than 4K) image mode. The $20mm$ lens minimizes distortion and allows us to capture the entire garden. We vetted the $20mm$ lens for characteristics including barrel distortion and lateral and axial chromatic aberrations. Together, the lens and sensor capture the entire garden bed with high clarity. This minimizes two problems that normal lenses introduce: occlusion and changes in relative size. Using a normal lens, plants at the edges of the field of view would often be occluded by plants in the center and distortion would affect how a plant's center and radius is computed. The combined camera and lens automatically records photos every hour. We then use the



Figure 6.1: **Custom Irrigation Tool:** For the watering of plants while reducing erosion. homography transformation based on the corners of the garden bed to correct the image scale and orientation.

Onboard Camera A custom borescope inspection camera [51] is located adjacent to the UTM on the Z-axis. It allows for close-up images of plant and soil. The onboard camera integrates with the FarmBot OS.

Soil Moisture Sensors To measure soil moisture and model soil dynamics, we distributed six TEROS-10 soil moisture sensors [52] throughout AlphaGarden, shown in Figure 6.4. These measure the Volumetric Water Content (VWC) of the soil with a $430mL$ volume of influence. The sensors connect to a ZL6 Data Logger [53], which publishes readings every 30 minutes.

Primary Workstation This is to collect measured data and upload it to the FarmBot Cloud solution. This allows for data to be reviewed online and for the FarmBot to be manipulated remotely.

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 .



Figure 6.2: **Overhead Camera:** The Sony SNC-VB77 camera is used to monitor the AlphaGarden. It's characteristics include a $35mm$ sensor, a low aberration lens, and higher than 4K resolution to allow for crisp and clear images for monitoring.



Figure 6.3: **FarmBot Gantry:** This is the FarmBot gantry robot of the experiments, which can automatically mount tools to itself and move along the X,Y,Z axes.

The tools we designed are operated through the FarmBot system 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 design, implement, and evaluate two options.

Rotary Pruner 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 uses 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



Figure 6.4: **Moisture Sensors:** The TEROS 10 sensors are laid throughout the garden bet to measure and model soil metrics and dynamics.



Figure 6.5: **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.

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 [54] adjacent to the FarmBot UTM pointing towards the soil surface. After arriving at the prune coordinates and measuring d , the Rotary Pruner is then toggled on, and the FarmBot is lowered to $d + 5cm$; the system overestimates the depth of the leaf in order to ensure a cut. The Rotary Pruner is then toggled off and returned to its home position.

Although the Rotary Pruner proved useful for many of the initial pruning actions, it spotlighted a few shortcomings that we wished to fix with a redesigned pruning attachment. Firstly, since the Rotary Pruner uses two separate attachments, the autonomous system had to regularly switch these attachments, adding unwanted power consumption and increasing the likelihood of mechanical failure. Secondly, due to the Rotary Pruner's relatively aggressive method of operation, it caused debris to fly, and 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.

Pruning Shears For a quieter, more precise and delicate pruning tool, we motorized a pair of Japanese topiary shears. A pair of Niwaki Topiary Shears [55] were fastened directly to the FarmBot's gantry rails. A YANSHON Digital 360° servo motor is able to close the shears by winding a high strength steel cable attached to one handle of the shears onto a spool; the shears reopen with a spring mechanism when the cable is unwound. This assembly is mounted to a 2-axis servo gimbal (using BETU Digital 270° servo motors). The gimbal is able to position the shears vertically, horizontally, or at any intermediate angle as well as rotate the shears a full 180° to account for any leaf direction, allowing the FarmBot to trim with greater precision as well as reach the tops of plants. The servos connect to the FarmBot PWM header and integrate with the FarmBot OS.

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

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

Table 6.1: **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 (including pruner).

Chapter 7

Experiments

7.1 Isolated Pruning Experiments

To evaluate the two pruning tools, we ran pruning experiments using two plant types in pots. 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. 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 included tears and fragments. The Pruning 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. Table 6.1 shows these results as discussed. 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 while the robot descends to cut the leaf. As noted previously, in these isolated pruning experiments, prune points were manually annotated since the goal was to test the pruning tool(s). So a bad prune point being manually selected, in some cases caused a failure as reported in Table 6.1.

7.2 4 Garden Cycles

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

Intersection over Union (IoU) is $\sum_{i=0}^{i_{total}} \text{label}_i / (i_{total} + 1)$. The baseline phenotyping model which is the same as [23] as explained in Section 4.1 had a mean IoU of 0.71 when compared to the ground truth at day 30. The approach presented, incorporating data aggregation techniques and with location based segmentation added, had a mean IoU of 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 when using the baseline model, had the largest increase from 0.23 to 0.59.

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

Human Intervention Although the goal is a fully automated polyculture garden pruning system, some human intervention was required. Seed planting was performed with human labor. A member of the project team (Mark) 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, as outlined in Section 4.3. This occurred in 45% of pruning operations. However, as also discussed in Section 4.3, by adjusting the on-board camera height, we were able to reduce the failure rate of servo positioning for future garden cycles. No other human intervention was performed in terms of weeding or irrigation.

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

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

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

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

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

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

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

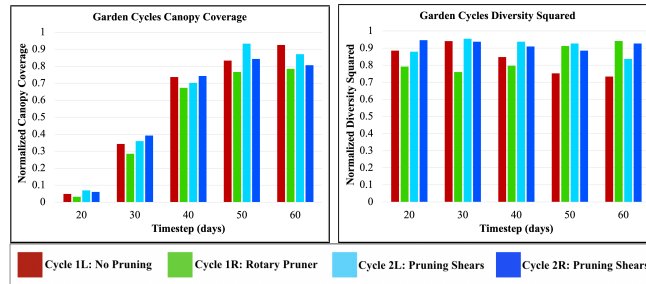


Figure 7.1: **Garden Cycle Comparison.** Data points were recorded for days 20, 30, 40, 50, and 60 through hand labeled phenotyping masks. **Left:** Comparison of the coverage of the 4 Garden Cycles. The non-pruned garden has the highest value by day 60, with Cycle 2L (pruning shears) not far behind.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

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. Fig. 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) during day 50 than Cycle 2R, which could be in part due to the greater number of prunes of Cycle 2R.

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

The reduction of overall coverage as diversity increases raises a very important consideration. While plant coverage is important for yield and productivity, the advantages of diversity including soil and neighboring plant support, and the survival of smaller plants remain pertinent. Results suggest that the choice of pruning tool affects the tradeoff as the Pruning Shears have a smaller tradeoff gap compared to the Rotary Pruner. In future work, we will explore this tradeoff and what approaches exist to mitigate the decrease in coverage while still promoting plant diversity.

Chapter 8

Limitations and Future Work

One limitation of this study is cost, as the cost for cameras and the robot far exceeds the value of plants produced in such a small garden. Also, these results are for ten plant types. While we believe that these results are generalizable, we have not performed experiments beyond these ten. We cannot claim that these results can be generalized or scaled to be cost effective.

We present algorithms for the automated pruning of polyculture plants to encourage plant diversity while maintaining canopy coverage. We also present a learned Plant Phenotyping Network and Bounding Disk Tracking algorithm to estimate the state of a polyculture garden. For automated pruning, we present a Prune Point Identification Network and visual servoing to transfer from pixel coordinates to real world FarmBot coordinates. We designed and evaluated two custom pruning tools. We evaluated the automated pruning system in a real world polyculture setting over the course of 4 Growth Cycles.

In [10] we introduce a closed loop irrigation policy that uses the TEROS soil moisture sensors to analytically determine the amount of water to irrigate. Nevertheless, this policy does not account for plant models, such as the plant water uptake model. Literature points to plant uptake being a factor of root mass and root length density [56]. However, root mass and root length density are hard to measure or calculate so future research will explore learning these variables using other known features of the system.

One issue is that plant diversity and coverage can be inversely related. In practical settings, farmers may choose to sacrifice diversity for coverage. In future work, we plan to integrate policies to optimize water usage, and explore using seed placement as an inductive bias to our phenotyping model. We also plan to explore closed-loop visual servoing for the Pruning Shears.

Chapter 9

Advanced Segmentation

9.1 Introduction

As covered in Section 4, State Estimation with overhead images poses challenges with occlusion of leaves by future growth, which reduces the amount of information the model has to predict the future plant states. Not only that, bounded circle tracking loses important per-leaf information about the plant in concern. Bounded leaf tracking does not account for different leaf sizes and is simplistic in its approach to monitoring the plant’s state. Other methods of segmentation and state estimation may provide more actionable data to quantify the plant’s state and create a model for its growth over time. As discussed in Figure 4.2, precision and recall are two crucial metrics for evaluating the performance of segmentation models. The goal is to improve both metrics by providing the model with higher-fidelity data aided by more informed segmentation techniques.

9.2 Related Work

Neural Radiance Fields (NeRF) and Gaussian Splatting are recent 3D representation methods that allows the synthesis of multiple viewpoints, including novel views, from images and corresponding poses [57], [58]. NeRF and Gaussian Splatting have advanced and influenced state-of-the-art methods in many fields such as robotics for localization [59] and grasping [60] by extending or improving parts of the pipeline [61].

In considering agriculture and plant modeling, NeRF has the potential to enhance our current simulations by providing volumetric representations that are closer to what we see in nature, as the result works below have explored. [62] recently extended a NeRF-derived method for use in implicit mapping for crop inspection. They present Target-Aware Implicit Mapping system (TAIM), a SLAM-based pose mapping system for pose estimation to create detailed 3D maps of strawberry and sweet pepper crop rows from RGB images taken by a robot in a greenhouse. Jignasu et al [63] uses a NeRF-derived method, Mip-NeRF, to reconstruct the geometry for a maize plant using a video collected using a mobile phone

with the goal of using the NeRF for creating a digital twin. Additionally, [64] applied NeRFs to peanut plants and showed that the resultant PeanutNeRF was effective both for plants with foliage and without foliage. Most recently, [65] present their work extending NeRF for 3D panoptic scene representation that is end-to-end. Trained using noisy robot odometry poses and inconsistent panoptic predictions, PAg-NeRF successfully produces accurate scene geometry, photo-realistic renders, and consistent 3D panoptic representations. In contrast to all of these, we are interested in NeRFs for early growth monitoring and growth model augmentation for indoor farms. NeRFs and Splats only generate point clouds, which can be visualized later, which does not give much affinity to manipulate the model after the fact. SuGaR [66] proposes a method to use a Network to convert a Radiance Field to a tessellated mesh by using the Poisson Reconstruction method, as compared to the standard marching cubes. The refinement step links each respective Gaussian to the surface of the generated mesh.

While a 3D Model of the garden is important, it is crucial to segment out the plants from this representation for better tracking. This includes segmenting out the plant itself, as well as its individual parts like leaves. GARField [14] proposes a method to using Segment Anything (SAM) [67] combined with Radiance Fields to extract groups of objects from the scene. By optimizing 2D SAM masks and using extra depth information, GARField segments separate groups of objects. By recursively segmenting the objects, GARField creates a hierarchy of groups. Building on this, Roggiolani et. al [68] propose a method for robust 3D instance leaf segmentation in an unsupervised fashion. Through the use of 3D point clouds outputted from NeRF or Gaussian Splatting, the authors account for occlusion and distortion before generating the instance segmentation of the leaves.

9.3 Qualitative Results

Gaussian Splatting and Meshing

Using SuGAR [66], a Gaussian Splat and Mesh model is trained after running a Structure for Motion algorithm. Once the algorithm is run for 7000 iterations on an nVidia A100 GPU, a mesh model is generated, which was imported into Blender and manually cleaned up to segment out the plant mesh from the total area mesh. The results from two different runs for the same plant's growth cycles can be seen in Figure 9.1.

GARField

While creating meshes generates useful visual features to evaluate qualitatively, it is difficult to automatically segment out and track changes in the plant body. By using SAM, prompting a point on the plant can help to segment it out. Since SAM requires a point to be prompted, this is not still fully autonomous. The prompted segmentation results can be seen in Figure 9.2. Future work includes fine tuning the SAM model to solely segment plants,

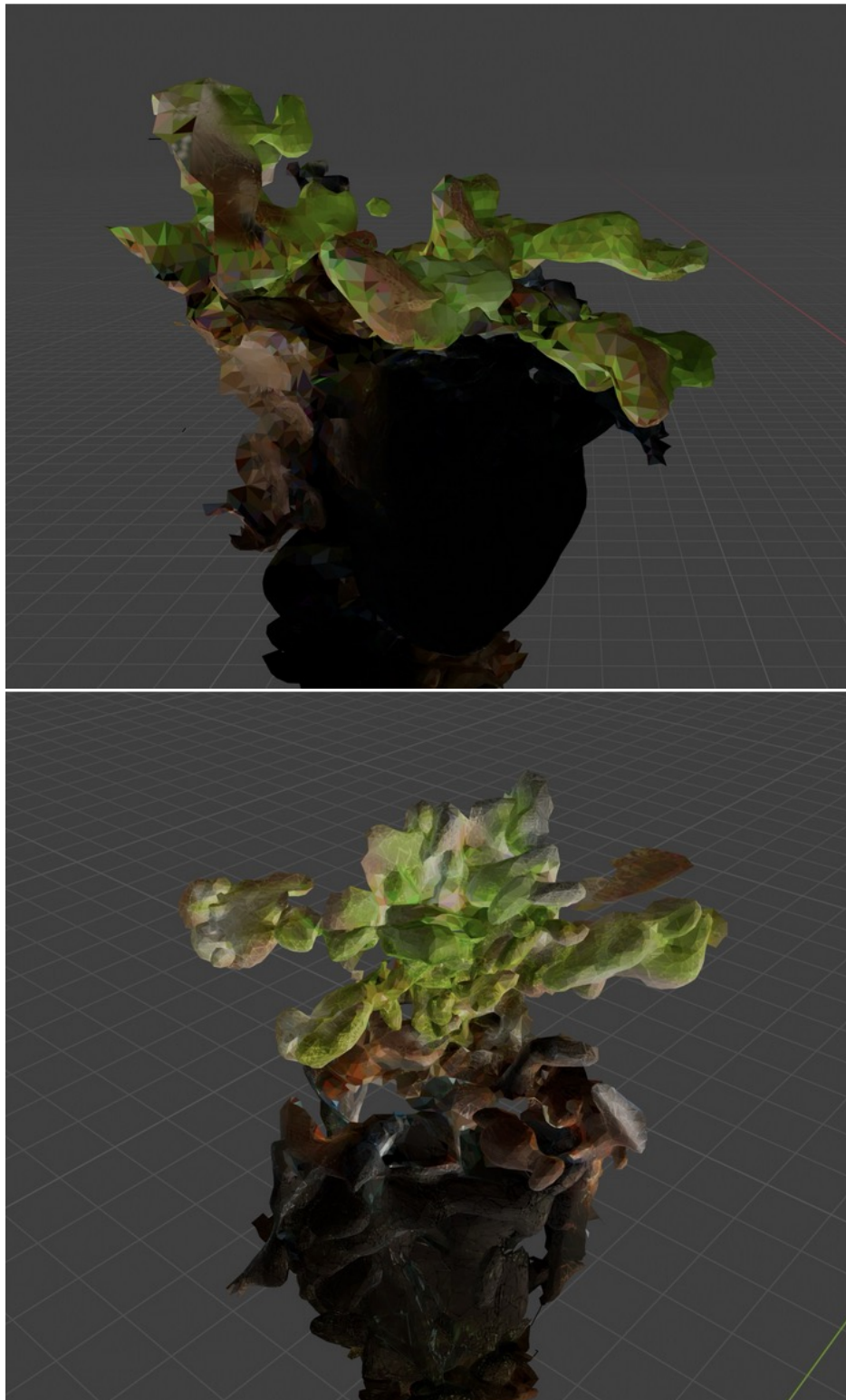


Figure 9.1: **Extracted meshes from SuGaR.** Two growth cycle days of January 8th, 2024, and February 2nd, 2024 were converted to a mesh using SuGar and then segmented using Blender's 3D editing tools [69].

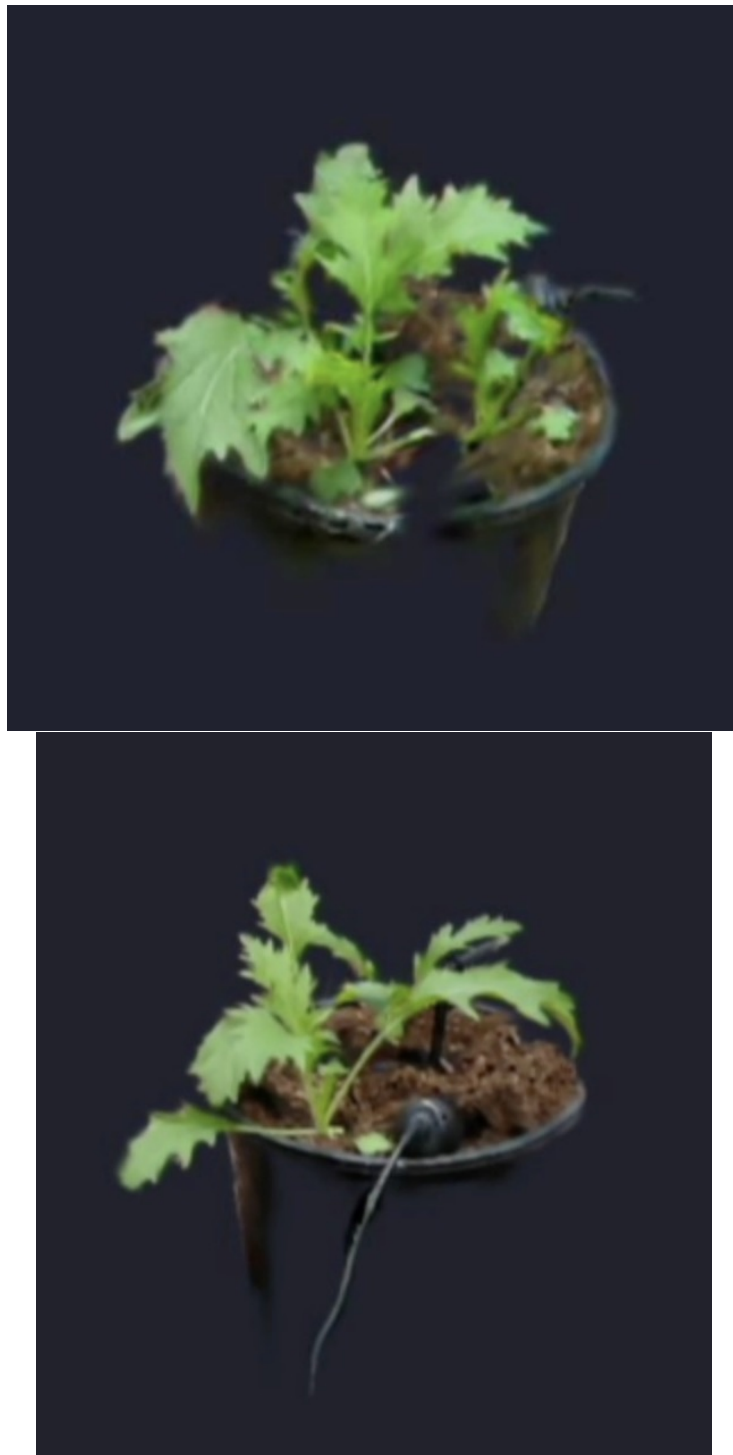


Figure 9.2: **Output of 3D Prompted Segmentation on Plants.** We use GARField [14] to manually define a prompt point to segment.

making the system fully autonomous in plant mask generation. This provides a promising start for instance segmentation on the leaves and to achieve more favorable performance.

Part II
Push-MOG

Chapter 10

Introduction - Push-MOG

Efficient object manipulation is a key task in industrial, commercial, and domestic robotics, incorporating elements from motion planning, grasping, and task planning [70]–[73]. In particular, many applications focus on the task of transferring a collection of objects from a surface into a bin or basket, for the purpose of either clearing the surface or preparing the objects for transportation to another location. This task is typically solved by picking one object at a time. However, an alternative approach of removing multiple objects at once with *multi-object grasping* (MOG) has received relatively little attention. MOG can provide better efficiency than traditional single-object grasping [74], [75], especially when the bin is relatively far away. Prior work on MOG mainly focused on developing techniques (such as MOG-Net [75]) for detecting and executing effective multi-object grasps in a scene. However, grasping multiple objects at once requires all of them to be close enough to fit within the gripper width, which may be rare when the objects are distributed randomly. Thus, to create graspable object clusters, pushing [76], [77] can be useful.

In this work, we propose Push-MOG, an algorithm that uses pushes to increase the average number of objects transported per trip to the bin. Push-MOG uses hierarchical clustering to identify clusters of objects which could potentially be grasped together. To execute stable pushes on polygonal objects with a parallel-jaw gripper, Push-MOG utilizes *fork pushing*, in which the jaw is opened to an appropriate width and the object is pushed perpendicular to the jaw, thus enabling a variety of stable pushes (see Fig. 13.1).

We also propose (mean) *objects per trip* (OpT) as a metric for evaluating the effectiveness of multi-object grasps. Thus, in single-object grasping, the OpT is 1 (or slightly less than 1 due to failed grasp attempts), whereas MOG improves efficiency by increasing OpT. We evaluate Push-MOG in a physical environment consisting of randomly-distributed polygonal objects on a flat surface, in which the robot executes both grasps and pushes using a parallel-jaw gripper. We consider two baselines: (i) Single-Object Grasping (SOG), which removes objects at random, one at a time; (ii) and MOG-Net, which searches for multi-object grasps but does not apply rearrange objects with pushes. In physical robot experiments, we find that prior work (MOG-Net) achieves an OpT of ≈ 1 , suggesting that naturally occurring multi-object grasps are rare, while Push-MOG achieves an OpT of 1.339, though at a cost

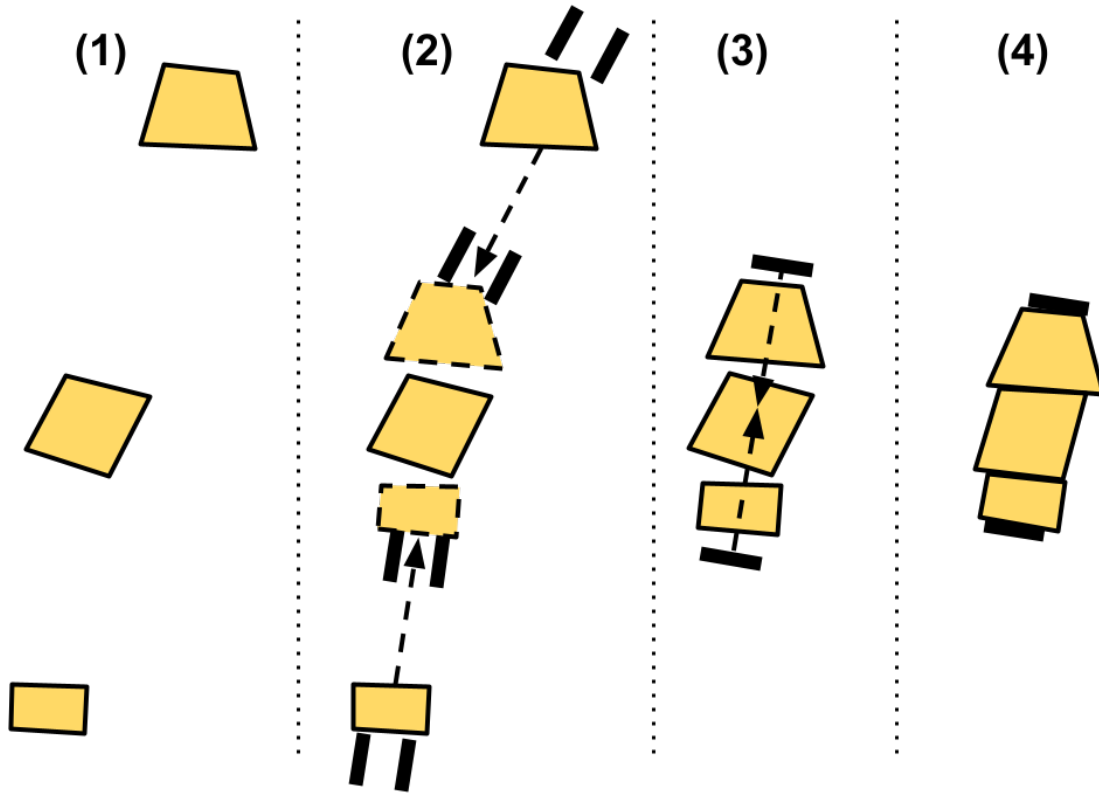


Figure 10.1: An example of Push-MOG pushing and grasping objects into a cluster. (1) A cluster is identified containing three objects; (2) two objects are moved toward the center object using fork pushing. (3) Once the cluster fits within the robot’s gripper, (4) the gripper closes to create a tight fit and grasps the objects that will be transported to the bin.

of extra time for executing pushes. Since increasing OpT is more valuable when the bin is further away, Push-MOG can improve picking efficiency when trips are long. We make the same assumptions as Agboh et al. [74]: that the objects are extruded convex polygons, and have roughly uniform mass so that their centers of mass coincide with their geometric centers. However, unlike [74], we also assume the objects are stable and do not topple when pushed.

In this work, we make the following contributions:

- We formulate a new variant of the decluttering problem, in which a robot uses pushing actions to increase the number of objects it can move simultaneously to the bin. To solve this problem, we propose and implement Push-MOG. We also propose objects per trip (OpT) as a metric for evaluating the effectiveness of MOG algorithms.
- We propose and implement *fork pushing* as a way to quickly and stably move objects

on a work surface using a parallel-jaw gripper.

- We evaluate the performance of Push-MOG and two baselines (one using only single-object grasping and one using multi-object grasping without pushes) and find that Push-MOG increases OpT by 34%.

This section presents work from the following papers with various contributions from me:

1. Algorithm Design
2. Physical Experiments

Chapter 11

Related Work

11.1 Multi-object grasps

Prior work on multi-object grasping [78]–[81] used multi-fingered robot hands and proposed conditions for a stable grasp of objects, focusing on numerical simulations without physical robot multi-object grasps or a focus on automation. Chen et al. [82] propose a method for a robot to dip its gripper inside a pile of identical spherical objects, close it, and estimate the number of grasped objects. Shenoy et al. [83] focus on the same problem but with the goal of transporting the picked spherical objects to another bin.

Another line of work has focused on designing appropriate grippers for picking up multiple objects at once. Jiang et al. [84] proposed a multiple suction cup vacuum gripper, while Nguyen et al. [85] proposed a soft gripper based on elastic wires for multi-object grasping.

Our work is focused on efficiently rearranging scenes with pushes [86]–[89] before multi-object grasping. Sakamoto et al. [90] proposed such a picking system where the robot first uses pushing [91]–[93], to move one cuboid to the other and thereafter grasp both cuboids. Our approach handles any multi-sided polygonal object and focuses on large-scale problems where clustering is required to decide which objects to push and where.

11.2 Pushing

Pushing is a fundamental primitive in robot manipulation [94]. Robotic pushing has been used to move a target object to a goal location [77], [95], rearrange a working surface [96], [97], and retrieve items from cluttered shelves [88], [89], [98].

During robotic pushing, uncertainty in the state, control, and model can result in failures [99], [100]. Prior work [76], [101] has trained networks to generate robust pushing actions. Others have taken an open-loop approach to generate these robust pushes [102], [103]. For example, Johnson et al. [104] propose robustness metrics and use them to generate robust open-loop pushes. They exploit gripper and object geometries to generate

these robust pushes. We take a similar open-loop pushing approach and leverage object and gripper geometries to generate robust pushes.

11.3 Point Clustering

Prior work in point clustering is well-established [105], [106], specifically in the hierarchical variant [107], [108], which aims to cluster data in an unsupervised fashion. This method constructs a distance-based tree that forms clusters of points from the bottom up which is then split at a ‘height’ to determine the final clusters. Additional work by Dao et al. [109] explores clustering with human-defined constraints. It introduces the idea of simplifying and solving NP-Hard problems by constraining them through a limit on inter-cluster distance metrics, such as maximum cluster diameter, or within-cluster variation [109]. In this work, we use a distance-based hierarchical clustering algorithm but split object clusters to satisfy gripper width constraints.

Chapter 12

Problem Statement

Agboh et al [74] studied the problem of using an overhead camera and a parallel-jaw gripper to transport a collection O of extruded polygonal convex objects (prisms) from a flat work surface to a bin or box adjacent to the workspace by taking advantage of *multi-object grasping*, in which multiple objects are grasped simultaneously and transported together to the bin.

In this work, we extend this problem by adding the *fork push* action, in which the gripper pushes an object. By doing so, the robot can arrange the objects to create more efficient grasps. Since pushing is faster to execute than grasping and the objects are typically closer to each other than to the bin, using pushes to facilitate multi-object grasps can reduce the number of trips needed to clear the objects. As in Agboh et al. [75], we consider a frictional model of planar grasping, which not only corresponds better to real grasping problems but also permits a larger set of stable grasps, thus allowing the algorithm to consider a wider range of potential solutions.

12.1 State, Action, and Objective

Let the set of objects on the work surface be $O = \{o_0, o_1, \dots, o_{N_o-1}\}$ where each o_i is a convex polygonal object and N_o is the number of objects on the work surface. The robot executes a sequence of *pushes* and transports (which we call *trips*); each push translates and/or rotates a single object in the workspace without colliding with others, while each trip grasps a set of closely clustered objects and transfers them to the bin.

We divide this problem into three sub-problems:

1. *Clustering*: Divide the objects into clusters. No cluster should contain more objects than can be stably grasped by the gripper.
2. *Pushing*: Push the objects in each cluster close together.
3. *Trip*: Perform grasps (containing as many objects as possible each time) and transport the grasped objects to the bin until no clusters remain.

Due to the inherent uncertainty of working with real objects, a grasp might not get every object in a cluster, so a cluster may need multiple grasps to clear completely. The clustering step is purely computational and does not involve any physical action.

Since we don't optimize an explicit objective in the clustering or pushing problems, we instead aim to guide the choice of algorithm to maximize overall OpT.

12.2 The Clustering Problem

In the clustering stage, the goal is to partition the set of objects O into clusters in such a way that (i) the objects in any cluster are close together and (ii) each cluster consists of objects that can be stably grasped together (provided they are pushed into an appropriate configuration). Finally, since the goal of clustering the objects is to reduce the number of grasps necessary to clear the workspace, a good overall clustering will partition the objects into as few clusters as possible, as each cluster roughly corresponds to one grasp. We denote the output of the clustering algorithm as a list of clusters C , and each individual cluster as $c \subseteq O$ (since a cluster is a set of objects).

In this problem, each object o 's position is represented by its geometric centroid, which we denote M_o . This allows the application of an appropriately modified version of the hierarchical clustering algorithm (which clusters points).

An important characteristic of each object o is its *minimum final grasp diameter* d_o^* , corresponding to the minimum width of a stable single-object grasp on o . Then, given a cluster c , we denote the *minimum grasp diameter of c* as $d_c^* = \sum_{o \in c} d_o^*$. The gripper also has a width $d^{(\text{gr})}$, and if $d_c^* > d^{(\text{gr})}$ we regard cluster c as ungraspable (thus requiring division into smaller clusters)¹.

Also, it is important for the clusters to not interfere with each other during the pushing step, which might happen if clusters are spatially intermingled. While point clustering usually avoids this (such clusters are locally non-optimal), the grasp diameter constraint may make such a solution to be 'optimal' unless it is specifically excluded. We thus add a constraint that for any $c_1, c_2 \in C$, the centroids of the objects in c_1 (i.e. the set $\{M_o : o \in c_1\}$) are linearly separable from the centroids of the objects in c_2 .

The clustering problem is then defined to partition O into a set of clusters C such that: (1) $d_c^* \leq d^{(\text{gr})}$ for all $c \in C$; (2) any $c_1, c_2 \in C$ have a line which separates the centers of mass of the objects they contain; (3) each cluster has small pairwise distances between its objects; and (4) there are few clusters overall. It is solved using a modified hierarchical clustering algorithm.

¹It is possible that even if $d_c^* \leq d^{(\text{gr})}$ the cluster does not have a stable grasp. For example, a cluster of three identical equilateral triangles cannot be stably grasped at all.

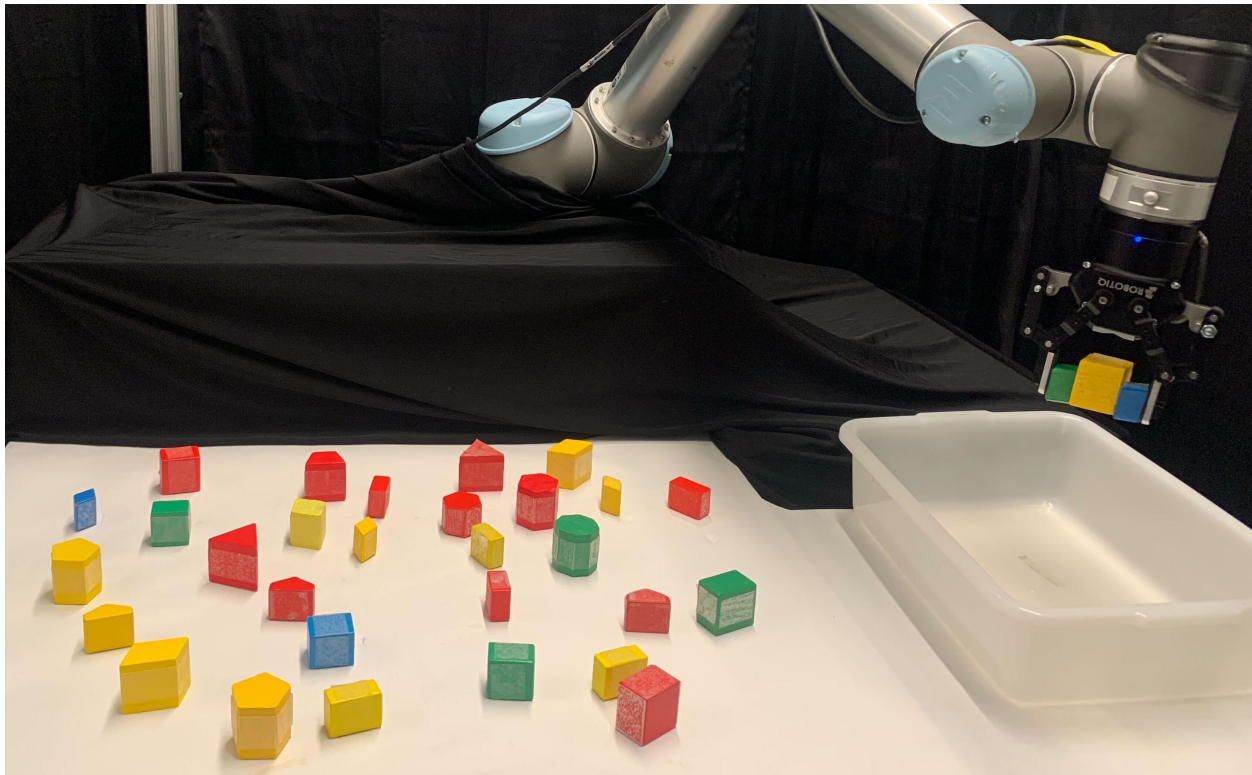


Figure 12.1: As illustrated in Fig. 10.1, Push-MOG uses “fork pushes” to consolidate objects for multi-object grasping as shown on the right.

12.3 The Pushing and Grasping Problems

After the clusters have been defined, the next step is to push each cluster closer together to make multi-object grasping more effective (see Fig. 10.1 for an illustration). We assume that (due to the separating line constraint in the clustering problem) each cluster can be treated as a separate instance of the pushing problem.

Finally, once the pushing is done, we move the objects to the bin via multi-object grasping. As in [75], the objective is to move the objects to the bin with as few grasps as possible (without pushing, as that has already been done).

Chapter 13

Push-MOG Algorithm

We present the Push-MOG algorithm in Alg. 1. After perceiving an initial scene, it computes a cluster of objects. We plan and execute a series of pushing actions to bring together the clustered objects so that they are touching each other. Finally, we query the MOG-Net algorithm [75] to transport that cluster through Multi-Object grasps. Below, we outline how the Push-MOG algorithm plans and executes these steps.

13.1 Clustering Details (Alg 2)

As described in Section 12.2, the goal of this step is to cluster objects into graspable subsets. In order to accomplish this we compute the centroids of each object and use a bottom-up hierarchical clustering. We find the smallest bounding box that encloses each object and cluster them based on their proximity to each other. Once these clusters are formed, we check that the sum of the widths of all the objects in the cluster does not exceed the width of the gripper. If we encounter a non-valid cluster c (i.e. such that $d_c^* > d^{(\text{gr})}$), then we split c into two clusters c_1, c_2 using a separating line (objects are divided based on their centroids), minimizing the difference between the cluster grasp diameters $|d_{c_1}^* - d_{c_2}^*|$.

13.2 Push Planning (Alg 3)

Push planning considers how to push objects in a cluster closer together in order to facilitate grasping. This can be broken into two steps: (i) determining the desired locations of the cluster’s objects; (ii) computing the parameters of the pushes to get them there. For object o , we denote the desired location of its centroid as M'_o , which is chosen both to make pushing easier and to position the objects for grasping. For simplicity, we only specify a desired location for the centroid (and not a desired orientation). All pushes are executed in one motion in a straight line; the direction and distance that o is pushed is given by the vector $M'_o - M_o$ (to move its center of mass from M_o to M'_o).

Algorithm 1. Push-MOG

```

1 do
2    $I \leftarrow$  Image of the workspace
3   Parse  $I$  into a 2D object map of all objects  $O$ 
4   Partition  $O$  into clusters  $C$  (Alg. 2, see 13.1)
5   Get a cluster  $c$  from  $C$  at random
6   Plan the pushes,  $P$  (Alg. 3, see 13.2)
7   for push  $p \in P$  do
8     | Predict post-push location of pushed object
9     | Run MOG-Net (13.3), using predicted post-push locations, to identify grasp  $g$ 
10    | Execute  $g$  to remove  $c$ 
11 while objects remain on the workspace;

```

Algorithm 2. Clustering Algorithm

```

Input       :  $O$ : List of objects
Output     :  $C$ : List of clusters
1 Compute centroids  $M_o$  for all  $o \in O$ 
2 Hierarchical clustering on  $\{M_o : o \in O\}$  to generate clusters  $C$ 
3 Check the validity of each cluster  $c \in C$  based on max gripper width
4 for non-valid cluster  $c \in C$  do
5   | Minimize weight  $W$  over  $\theta$  using the following: for  $\theta \in [-\frac{\pi}{4}, \frac{\pi}{4}]$  do
6   |   |  $v = \langle \cos \theta, \sin \theta \rangle$ 
7   |   | Split the objects into two clusters  $c_1, c_2$  on an infinite line defined by  $v$ ,
8   |   |   | through the centroid of  $c$ 
9   |   |   | For this  $\theta$ , calculate weight difference  $W = |\sum_{o \in c_1} d_o^* - \sum_{o \in c_2} d_o^*|$ 
9   | Split the cluster on this  $\theta$ 

```

For executing a push with a parallel-jaw gripper, we propose the technique of *fork pushing*: we angle the gripper so the line between the jaws is perpendicular to the desired push direction. This allows the gripper to push both along an edge or around a vertex of an object (see Fig. 13.1), whereas a flat-edged pusher could not push against a vertex without the object undergoing a major rotation, deviating from the desired path.

13.3 MOG-Net Integration

Following the pushing step, we use MOG-Net [75] on the clusters to clear the workspace. To improve efficiency, we use the simulated coordinates of the push action to plan grasps instead of taking an image of the workspace again.

However, this process is not fully robust, because pushes may cause other blocks in the way to deviate from their estimated location, leading to a faulty grasp.

Algorithm 3. Push Planning Algorithm

Input : C : List of Clusters
Output : v_{start}, v_{end} : Start and end push points for each object

- 1 **for** *valid cluster* $c \in C$ **do**
- 2 $M_c = \frac{\sum_{\text{object } o \in c} M_o}{|c|}$
- 3 $m = \arg \min_{o \in c} \|M_o - M_c\|$, the central object in the cluster
- 4 **for** *object* $o \in c \setminus m$ **do**
- 5 $v = M_o - M_m, v_l$ line with endpoints M_o, M_m
- 6 Calculate center of furthest edge e_o and corner d_o of object o from m , which intersects with v_l
- 7 Calculate closest edge e_m on o to o_m on v_l
- 8 Scale $v \rightarrow v_p$ to capture the length of the push vector $\|v\|$
- 9 Ensure no collisions by shortening v_p , such that $v_p + o$ does not collide with o_m

To remedy this, whenever a trip is performed (thus moving the gripper out of the camera's view of the workspace), a new image of the workspace is taken, allowing these errors to be corrected; the algorithm also uses this image to re-plan the clustering (13.1) and pushing (13.2) on the remaining blocks (which may have been pushed aside or missed during an earlier grasp).

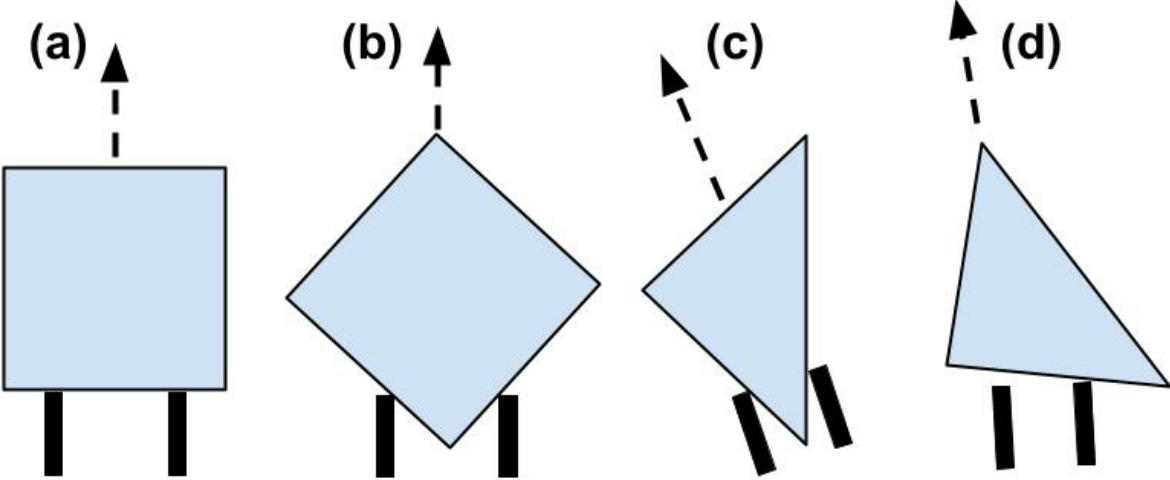


Figure 13.1: Examples of fork pushing with a parallel-jaw gripper, both against an edge ((a) and (d)) and around a vertex ((b) and (c)) noticing that the direction the object will move to is roughly perpendicular to the parallel-jaw gripper.

Chapter 14

Experiments and Results

To evaluate the performance of the policies developed in the paper, we run experiments in a real workspace.

14.1 Experimental Setup

An example input is shown in Fig. 15.1. We use a Universal Robotics (UR) 5 Robot with a Robotiq 2F-85 gripper mounted to its wrist. We perform experiments on 34 convex objects of various sizes, ranging from 3-sided to 8-sided. All experiments assume a random placement of the objects in the workspace, and one such configuration is shown in Fig. 15.1. To generate a random placement, we uniformly sample the workspace and create circles that encompass the max radius of any object we are using. Before creating each circle we ensure that they do not overlap, and continue sampling the object set without replacement until all objects are placed. Then a human sets up the configuration in real, mirroring the randomly generated image. We generated 5 random scenes, on which we will perform all experiments. We add friction on the graspable faces of the objects by wrapping them in frictional tape.

To perceive the environment, we use an Intel RealSense L515 Camera, which outputs RGB-D images, mounted directly above the workspace. To calibrate the Camera-Robot transform, use an ArUco marker [110] which is manually calibrated with an accuracy of $\sim 1cm$.

The pushing action consists of 5 repeated steps:

1. Open the jaws to 30 percent width
2. Move the gripper above the desired push location
3. Move down to the object's Z-height
4. Move the gripper and push it to its planned location
5. Move the gripper above its planned location

The Multi-Object Grasping action is executed with the MOG-Net algorithm [75]. Here we repeat the following:

1. Plan a grasp for each cluster
2. Move the gripper above the desired grasp location
3. Open the jaws fully
4. Move down to the work surface
5. Close the jaws to complete the grasp
6. Move the gripper outside the workspace and open the jaws fully to drop the objects into the basket

14.2 Baselines

We compare Push-MOG against two baselines: Frictional SOG, which uses single-object grasps, and MOG-Net [75], which uses multi-object grasps but does not use additional actions to help group the objects together. See Section 10 for detailed information on baselines.

14.3 Results

Results are given in Table 14.1. We find that randomly-placed objects are not generally well-positioned for multi-object grasps, as OpT is almost identical between MOG-Net and Frictional SOG, thus requiring the use of additional actions such as pushes to assist MOG. Indeed, the similarity in the performance of Frictional SOG and MOG-Net suggests that there are few if any, ‘naturally occurring’ multi-object grasps in a typical scene of randomly-scattered objects. Push-MOG successfully uses the pushing action to generate graspable object clusters, increasing OpT by roughly 34%. This comparatively modest increase in OpT (as compared to the experimental results on the original MOG-Net [75]) is because objects that are stable under pushing tend to have larger minimum grasp diameters and the gripper we used has a relatively small max opening width, so it can grasp at most 2 or 3 objects at a time.

Table 14.1: Physical decluttering experimental results for random scenes, comparing baselines **Frictional SOG**, **MOG-Net**, and the method **Push-MOG**. Experiments were conducted with a Robotiq 2F-85 parallel-jaw gripper mounted on a UR5 robot arm, on 5 random scenes of 34 objects, with the goal placed directly adjacent (see Figs 12.1 and 15.1).

Methods	Objects per hour (OPH)
Frictional SOG	331.70 ± 4.33
MOG-Net	325.23 ± 6.81
Push-MOG	250.16 ± 17.06

Chapter 15

Limitations and Future Work

This work proposes Push-MOG , a novel algorithm that consolidates polygonal objects into optimal clusters which increase the number of objects per grasp. This work has the following limitations: 1) pushing can cause misalignments where grasps fail 2) Push-MOG does not always avoid collisions 3) total time is slow, leading to fewer picks per hour, even though Push-MOG gets more objects per trip. In future work, we can incorporate *vertical stacking* to create larger multi-object clusters.

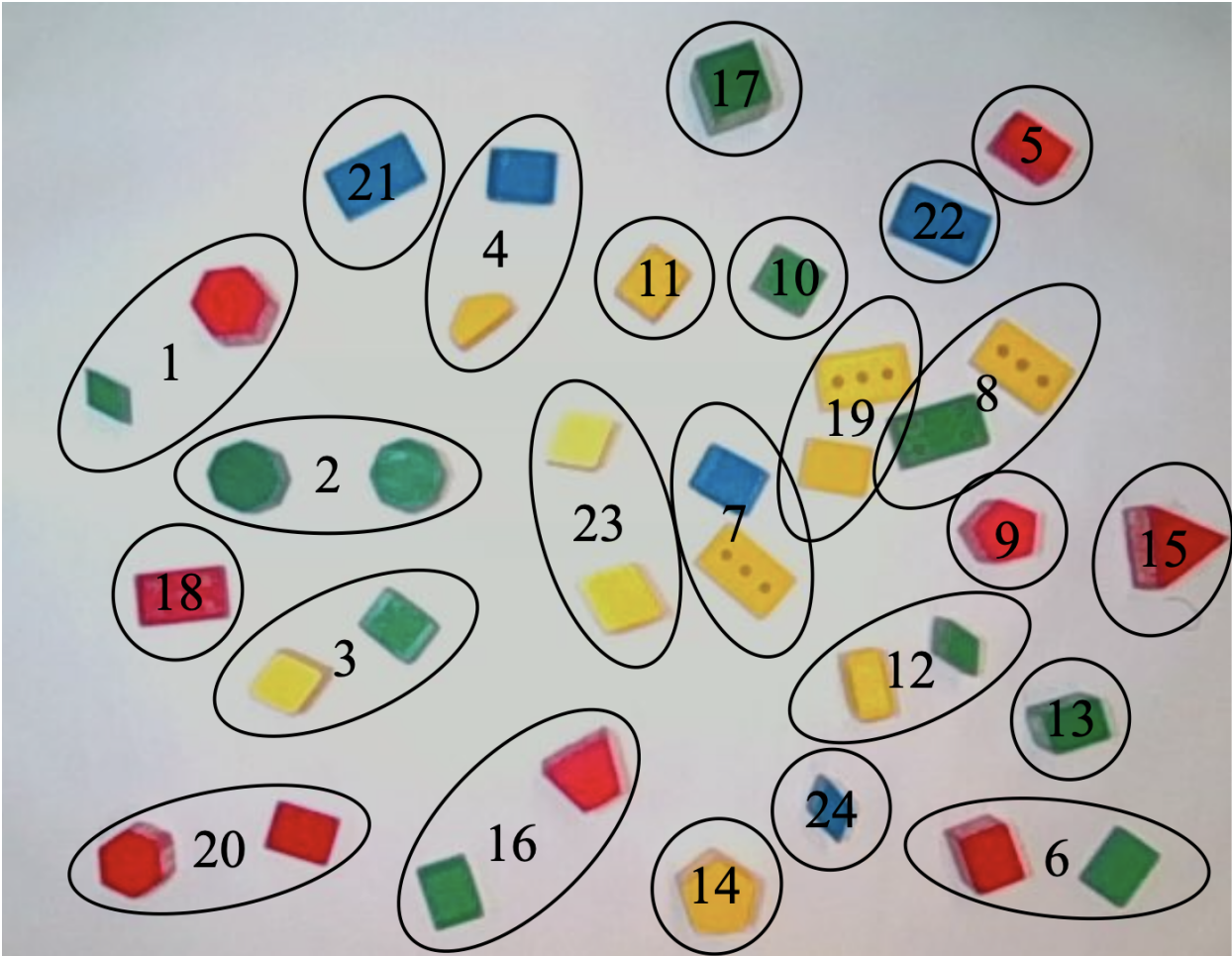


Figure 15.1: This figure shows the clustering formed by Push-MOG of an initial scene. The numbers on each cluster identify the order that the algorithm will perform the pushing and grasping actions.

Bibliography

- [1] K. Goldberg and J. Santarromana. “The telegarden.” (1995), [Online]. Available: <https://goldberg.berkeley.edu/garden/Ars/> (visited on 12/11/2019).
- [2] K. Goldberg, *The Robot in the Garden: Telerobotics and Telepistemology in the Age of the Internet*. MIT Press, 2001.
- [3] T. S. Rosenstock, D. Liptzin, K. Dzurella, A. Fryjoff-Hung, A. Hollander, V. Jensen, A. King, G. Kourakos, A. McNally, G. S. Pettygrove, *et al.*, “Agriculture’s contribution to nitrate contamination of californian groundwater (1945–2005),” *Journal of Environmental Quality*, vol. 43, no. 3, pp. 895–907, 2014.
- [4] A. Mantovani. “Pesticide risk assessment: European framework shows need for safer alternatives.” (2019), [Online]. Available: <https://www.openaccessgovernment.org/pesticide-risk-assessment/79226/> (visited on 12/11/2019).
- [5] S. J. Risch, “Intercropping as cultural pest control: Prospects and limitations,” *Environmental Management*, vol. 7, no. 1, pp. 9–14, 1983.
- [6] T. E. Crews, W. Carton, and L. Olsson, “Is the future of agriculture perennial? imperatives and opportunities to reinvent agriculture by shifting from annual monocultures to perennial polycultures,” *Global Sustainability*, vol. 1, 2018.
- [7] S. Gliessman and M. Altieri, “Polyculture cropping has advantages,” *California Agriculture*, vol. 36, no. 7, pp. 14–16, 1982.
- [8] M. Liebman, “Polyculture cropping systems,” in *Agroecology*, CRC Press, 2018, pp. 205–218.
- [9] Y. Avigal, W. Wong, M. Presten, M. Theis, S. Aeron, A. Deza, S. Sharma, R. Parikh, S. Oehme, S. Carpin, J. H. Viers, S. Vougioukas, and K. Goldberg, “Simulating polyculture farming to learn automation policies for plant diversity and precision irrigation,” *IEEE Transactions on Automation Science and Engineering*, pp. 1–13, 2022. DOI: 10.1109/TASE.2021.3138995.
- [10] S. Adebola, R. Parikh, M. Presten, S. Sharma, S. Aeron, A. Rao, S. Mukherjee, T. Qu, C. Wistrom, E. Solowjow, and K. Goldberg, “Can machines garden? systematically comparing the alphagarden vs. professional horticulturalists,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 779–11 785. DOI: 10.1109/ICRA48891.2023.10161497.

- [11] S. Adebola, M. Presten, R. Parikh, S. Aeron, S. Mukherjee, S. Sharma, M. Theis, W. Teitelbaum, E. Solowjow, and K. Goldberg, “Automated pruning and irrigation of polyculture plants,” *IEEE Transactions on Automation Science and Engineering*, 2024.
- [12] M. Presten, R. Parikh, S. Aeron, S. Mukherjee, S. Adebola, S. Sharma, M. Theis, W. Teitelbaum, and K. Goldberg, “Automated pruning of polyculture plants,” in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, 2022, pp. 242–249. DOI: 10.1109/CASE49997.2022.9926632.
- [13] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3d gaussian splatting for real-time radiance field rendering,” *ACM Transactions on Graphics*, vol. 42, no. 4, Jul. 2023. [Online]. Available: <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>.
- [14] C. M. Kim, M. Wu, J. Kerr, M. Tancik, K. Goldberg, and A. Kanazawa, “Garfield: Group anything with radiance fields,” in *arXiv*, 2024.
- [15] M. Wiggert, L. Amladi, R. Berenstein, S. Carpin, J. Viers, S. Vougioukas, and K. Goldberg, “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.
- [16] N. Correll, N. Arechiga, A. Bolger, M. Bollini, B. Charrow, A. Clayton, F. Dominguez, K. Donahue, S. Dyar, L. Johnson, *et al.*, “Building a distributed robot garden,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 1509–1516.
- [17] A. Agostini, G. Alenyà, A. Fischbach, H. Scharr, F. Wörgötter, and C. Torras, “A cognitive architecture for automatic gardening,” *Computers and Electronics in Agriculture*, vol. 138, pp. 69–79, 2017, ISSN: 0168-1699. DOI: <https://doi.org/10.1016/j.compag.2017.04.015>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0168169916304768>.
- [18] D. Ruth Anita Shirley, K. Ranjani, G. Arunachalam, and D. A. Janeera, “Automatic distributed gardening system using object recognition and visual servoing,” in *Inventive Communication and Computational Technologies*, G. Ranganathan, J. Chen, and Á. Rocha, Eds., Singapore: Springer Singapore, 2021, pp. 359–369, ISBN: 978-981-15-7345-3.
- [19] T. Stomph, C. Dordas, A. Baranger, J. de Rijk, B. Dong, J. Evers, C. Gu, L. Li, J. Simon, E. S. Jensen, *et al.*, “Designing intercrops for high yield, yield stability and efficient use of resources: Are there principles?” In *Advances in Agronomy*, 1, vol. 160, Elsevier, 2020, pp. 1–50.
- [20] J. W. Jones, G. Hoogenboom, C. H. Porter, K. J. Boote, W. D. Batchelor, L. Hunt, P. W. Wilkens, U. Singh, A. J. Gijssman, and J. T. Ritchie, “The dssat cropping system model,” *European journal of agronomy*, vol. 18, no. 3-4, pp. 235–265, 2003.

- [21] P. Steduto, T. C. Hsiao, D. Raes, and E. Fereres, “Aquacrop—the fao crop model to simulate yield response to water: I. concepts and underlying principles,” *Agronomy Journal*, vol. 101, no. 3, pp. 426–437, 2009.
- [22] Y. Avigal, J. Gao, W. Wong, K. Li, G. Pierroz, F. S. Deng, M. Theis, M. Presten, and K. Goldberg, “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.
- [23] Y. Avigal, A. Deza, W. Wong, S. Oehme, M. Presten, M. Theis, J. Chui, P. Shao, H. Huang, A. Kotani, S. Sharma, R. Parikh, M. Luo, S. Mukherjee, S. Carpin, J. H. Viers, S. Vougioukas, and K. Goldberg, “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.
- [24] T. W. Ayalew, J. R. Ubbens, and I. Stavness, “Unsupervised Domain Adaptation for Plant Organ Counting,” in *Computer Vision – ECCV 2020 Workshops*, A. Bartoli and A. Fusiello, Eds., Cham: Springer International Publishing, 2020, pp. 330–346, ISBN: 978-3-030-65414-6.
- [25] M. Minervini, A. Fischbach, H. Scharr, and S. A. Tsiftaris, “Finely-grained annotated datasets for image-based plant phenotyping,” *Pattern Recognition Letters*, vol. 81, pp. 80–89, 2016, ISSN: 0167-8655. DOI: <https://doi.org/10.1016/j.patrec.2015.10.013>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167865515003645>.
- [26] M. Minervini, A. Fischbach, H. Scharr, and S. Tsiftaris, *Plant phenotyping datasets*, 2015. [Online]. Available: <http://www.plant-phenotyping.org/datasets>.
- [27] H. Cuevas-Velasquez, A.-J. Gallego, R. Tylecek, J. Hemming, B. van Tuijl, A. Mencarelli, and R. B. Fisher, “Real-time stereo visual servoing for rose pruning with robotic arm,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 7050–7056. DOI: 10.1109/ICRA40945.2020.9197272.
- [28] N. Strisciuglio, R. Tylecek, M. Blaich, N. Petkov, P. Biber, J. Hemming, E. v. Henten, T. Sattler, M. Pollefeys, T. Gevers, T. Brox, and R. B. Fisher, “Trimbot2020: An outdoor robot for automatic gardening,” in *ISR 2018; 50th International Symposium on Robotics*, 2018, pp. 1–6.
- [29] N. Habibie, A. M. Nugraha, A. Z. Anshori, M. A. Ma’sum, and W. Jatmiko, “Fruit mapping mobile robot on simulated agricultural area in gazebo simulator using simultaneous localization and mapping (slam),” in *2017 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, 2017, pp. 1–7. DOI: 10.1109/MHS.2017.8305235.

- [30] E. Van Henten, J. Hemming, B. Tuijl, J. Kornet, J. Meuleman, J. Bontsema, and E. van Os, “An autonomous robot for harvesting cucumbers in greenhouses.,” *Auton. Robots*, vol. 13, pp. 241–258, Nov. 2002. DOI: 10.1023/A:1020568125418.
- [31] FAO, 2022, *The State of Food and Agriculture 2022. Leveraging automation in agriculture for transforming agrifood systems*, en. FAO, Nov. 2022, ISBN: 978-92-5-136043-9. DOI: 10.4060/cb9479en. [Online]. Available: <http://www.fao.org/documents/card/en/c/cb9479en> (visited on 12/16/2023).
- [32] E. Nigussie, T. Olwal, G. Musumba, T. Tegegne, A. Lemma, and F. Mekuria, “IoT-based Irrigation Management for Smallholder Farmers in Rural Sub-Saharan Africa,” *Procedia Computer Science*, The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2020) / The 10th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH 2020) / Affiliated Workshops, vol. 177, pp. 86–93, Jan. 2020, ISSN: 1877-0509. DOI: 10.1016/j.procs.2020.10.015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050920322808> (visited on 12/16/2023).
- [33] FAO. 2020, *The State of Food and Agriculture 2020. Overcoming water challenges in agriculture*. FAO, 2020, 210 pp., ISBN: 978-92-5-133441-6. DOI: 10.4060/cb1447en. [Online]. Available: <http://www.fao.org/documents/card/en/c/cb1447en> (visited on 05/04/2021).
- [34] J. F. Velasco-Muñoz, J. A. Aznar-Sánchez, A. Batlles-delaFuente, and M. D. Fidelibus, “Sustainable irrigation in agriculture: An analysis of global research,” *Water*, vol. 11, no. 9, 2019, ISSN: 2073-4441. DOI: 10.3390/w11091758. [Online]. Available: <https://www.mdpi.com/2073-4441/11/9/1758>.
- [35] M. Erick, S. Fiestas, R. Sixto, and G. Prado, “Modeling and simulation of kinematics and trajectory planning of a farmbot cartesian robot,” *INTERCON*, vol. 1, 2018.
- [36] B. Murdyantoro, D. Sukma Eka Atmaja, and H. Rachmat, “Application design of farmbot based on internet of things (iot),” *IJASEIT*, vol. 9, 2019.
- [37] V. A. Murcia, J. F. Palacios, and G. Barbieri, “Farmbot simulator: Towards a virtual environment for scaled precision agriculture,” in *Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future*, D. Trentesaux, T. Borangiu, P. Leitão, J.-F. Jimenez, and J. R. Montoya-Torres, Eds., Cham: Springer International Publishing, 2021, pp. 234–246, ISBN: 978-3-030-80906-5.
- [38] V. Kamat, S. Aeron, A. Gu, H. Jalan, S. Adebola, and K. Goldberg, “Polypod: An algorithm for polyculture seed placement,” in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, 2023, pp. 1–6. DOI: 10.1109/CASE56687.2023.10260454.

- [39] F. Liu, Q. Song, J. Zhao, L. Mao, H. Bu, Y. Hu, and X.-G. Zhu, “Canopy occupation volume as an indicator of canopy photosynthetic capacity,” *New Phytologist*, vol. 232, no. 2, pp. 941–956, 2021. DOI: <https://doi.org/10.1111/nph.17611>. eprint: <https://nph.onlinelibrary.wiley.com/doi/pdf/10.1111/nph.17611>. [Online]. Available: <https://nph.onlinelibrary.wiley.com/doi/abs/10.1111/nph.17611>.
- [40] T. M. Cover and J. A. Thomas, “Entropy, relative entropy, and mutual information,” in *Elements of Information Theory*. John Wiley & Sons, Ltd, 2005, ch. 2, pp. 13–55, ISBN: 9780471748823. DOI: <https://doi.org/10.1002/047174882X.ch2>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/047174882X.ch2>. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/047174882X.ch2>.
- [41] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., Cham: Springer International Publishing, 2015, pp. 234–241, ISBN: 978-3-319-24574-4.
- [42] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *arXiv preprint arXiv:1512.03385*, 2015.
- [43] C. W. Whippo and R. P. Hangarter, “Phototropism: Bending towards enlightenment,” *The Plant Cell*, vol. 18, no. 5, pp. 1110–1119, 2006.
- [44] D. Dietrich, “Hydrotropism: How roots search for water,” *Journal of experimental botany*, vol. 69, no. 11, pp. 2759–2771, 2018.
- [45] H. Yang, R. Deng, Y. Lu, Z. Zhu, Y. Chen, J. T. Roland, L. Lu, B. A. Landman, A. B. Fogo, and Y. Huo, “Circlenet: Anchor-free glomerulus detection with circle representation,” in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, A. L. Martel, P. Abolmaesumi, D. Stoyanov, D. Mateus, M. A. Zuluaga, S. K. Zhou, D. Racoceanu, and L. Joskowicz, Eds., Cham: Springer International Publishing, 2020, pp. 35–44, ISBN: 978-3-030-59719-1.
- [46] H. Uchiyama, S. Sakurai, M. Mishima, D. Arita, T. Okayasu, A. Shimada, and R.-i. Taniguchi, “An easy-to-setup 3d phenotyping platform for komatsuna dataset,” in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 2038–2045. DOI: [10.1109/ICCVW.2017.239](https://doi.org/10.1109/ICCVW.2017.239).
- [47] F. Chaumette and S. Hutchinson, “Visual servo control. i. basic approaches,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006. DOI: [10.1109/MRA.2006.250573](https://doi.org/10.1109/MRA.2006.250573).
- [48] FarmBot. “Farmbot.” (2021), [Online]. Available: <https://farm.bot/> (visited on 09/03/2021).

- [49] Sony. “Snc-vb770 ultra high sensitivity 4k network camera.” (2021), [Online]. Available: <https://pro.sony/enEE/products/specialised-cameras/snc-vb770> (visited on 09/03/2021).
- [50] Sony. “Fe 20mm f1.8 g full-frame large-aperture ultra-wide angle g lens.” (2021), [Online]. Available: <https://electronics.sony.com/imaging/lenses/all-e-mount/p/sel20f18g> (visited on 09/03/2021).
- [51] FarmBot. “Electronics and wiring - borescope camera.” (2024), [Online]. Available: <https://genesis.farm.bot/v1.4/Extras/bom/electronics-and-wiring#borescope-camera> (visited on 01/20/2024).
- [52] Meter Group. “Teros 10 simple soil moisture sensing.” (2021), [Online]. Available: <https://www.metergroup.com/environment/products/teros-10/> (visited on 09/08/2021).
- [53] ”Meter Group”. “Zl6 advanced cloud data logger.” (2023), [Online]. Available: <https://www.metergroup.com/environment/products/zl6-data-logger/> (visited on 09/08/2023).
- [54] Pololu. “Pololu carrier with sharp gp2y0a60szlf.” (2021), [Online]. Available: <https://www.pololu.com/product/2474> (visited on 09/03/2021).
- [55] Niwaki. “Sentei topiary clippers.” (2021), [Online]. Available: <https://www.niwaki.com/sentei-topiary-clippers> (visited on 12/06/2021).
- [56] L. Faria, M. Rocha, Q. de Jong van Lier, and D. Casaroli, “A split-pot experiment with sorghum to test a root water uptake partitioning model,” *Plant and Soil*, vol. 331, pp. 299–311, Jun. 2010. DOI: 10.1007/s11104-009-0254-0.
- [57] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “Nerf: Representing scenes as neural radiance fields for view synthesis,” in *ECCV*, 2020.
- [58] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3d gaussian splatting for real-time radiance field rendering,” *ACM Transactions on Graphics*, vol. 42, no. 4, Jul. 2023. [Online]. Available: <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>.
- [59] D. Maggio, M. Abate, J. Shi, C. Mario, and L. Carlone, *Loc-NeRF: Monte Carlo Localization using Neural Radiance Fields*, arXiv:2209.09050 [cs], Sep. 2022. DOI: 10.48550/arXiv.2209.09050. [Online]. Available: <http://arxiv.org/abs/2209.09050> (visited on 08/26/2023).
- [60] J. Kerr, L. Fu, H. Huang, Y. Avigal, M. Tancik, J. Ichnowski, A. Kanazawa, and K. Goldberg, “Evo-NeRF: Evolving NeRF for Sequential Robot Grasping of Transparent Objects,” en, in *Proceedings of The 6th Conference on Robot Learning*, ISSN: 2640-3498, PMLR, Mar. 2023, pp. 353–367. [Online]. Available: <https://proceedings.mlr.press/v205/kerr23a.html> (visited on 08/26/2023).

- [61] F. Dellaert, *NeRF at CVPR 2022*, en, Jun. 2022. [Online]. Available: <https://dellaert.github.io/NeRF22/> (visited on 08/26/2023).
- [62] S. Kelly, A. Riccardi, E. Marks, F. Magistri, T. Guadagnino, M. Chli, and C. Stachniss, “Target-Aware Implicit Mapping for Agricultural Crop Inspection,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 9608–9614. DOI: 10.1109/ICRA48891.2023.10160487.
- [63] A. Jignasu, E. Herron, T. Z. Jubery, J. Afful, A. Balu, B. Ganapathysubramanian, S. Sarkar, and A. Krishnamurthy, “Plant geometry reconstruction from field data using neural radiance fields,” in *2nd AAAI Workshop on AI for Agriculture and Food Systems*, 2023. [Online]. Available: https://openreview.net/forum?id=TvKKqWn_6.
- [64] F. Saeed, J. Sun, P. Ozias-Akins, Y. J. Chu, and C. C. Li, “PeanutNeRF: 3D Radiance Field for Peanuts,” en, in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Vancouver, BC, Canada: IEEE, Jun. 2023, pp. 6254–6263, ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00665. [Online]. Available: <https://ieeexplore.ieee.org/document/10209004/> (visited on 08/26/2023).
- [65] C. Smitt, M. Halstead, P. Zimmer, T. Läbe, E. Guclu, C. Stachniss, and C. McCool, *Pag-nerf: Towards fast and efficient end-to-end panoptic 3d representations for agricultural robotics*, 2023. arXiv: 2309.05339 [cs.R0].
- [66] A. Guedon and V. Lepetit, “Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering,” *arXiv preprint arXiv:2311.12775*, 2023.
- [67] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, *Segment anything*, 2023. arXiv: 2304.02643 [cs.CV].
- [68] G. Roggiolani, F. Magistri, T. Guadagnino, J. Behley, and C. Stachniss, “Unsupervised pre-training for 3d leaf instance segmentation,” *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7448–7455, Nov. 2023, ISSN: 2377-3774. DOI: 10.1109/lra.2023.3320018. [Online]. Available: <http://dx.doi.org/10.1109/LRA.2023.3320018>.
- [69] Blender Online Community, *Blender - a 3d modelling and rendering package*, Blender Foundation, Blender Institute, Amsterdam, 2024. [Online]. Available: <http://www.blender.org>.
- [70] J. Ichnowski, M. Danielczuk, J. Xu, V. Satish, and K. Goldberg, “Gomp: Grasp-optimized motion planning for bin picking,” in *ICRA*, 2020.
- [71] J. Ichnowski, Y. Avigal, V. Satish, and K. Goldberg, “Deep learning can accelerate grasp-optimized motion planning,” *Science Robotics*, vol. 5, no. 48, 2020.

- [72] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, “Fastrack: A modular framework for fast and guaranteed safe motion planning,” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 1517–1522. DOI: 10.1109/CDC.2017.8263867.
- [73] J. Ichnowski, Y. Avigal, Y. Liu, and K. Goldberg, “Gomp-fit: Grasp-optimized motion planning for fast inertial transport,” in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 5255–5261. DOI: 10.1109/ICRA46639.2022.9812387.
- [74] W. C. Agboh, J. Ichnowski, K. Goldberg, and M. R. Dogar, *Multi-object grasping in the plane*, 2022. DOI: 10.48550/ARXIV.2206.00229. [Online]. Available: <https://arxiv.org/abs/2206.00229>.
- [75] W. C. Agboh, S. Sharma, K. Srinivas, M. Parulekar, G. Datta, T. Qiu, J. Ichnowski, E. Solowjow, M. Dogar, and K. Goldberg, *Learning to efficiently plan robust frictional multi-object grasps*, 2022. DOI: 10.48550/ARXIV.2210.07420. [Online]. Available: <https://arxiv.org/abs/2210.07420>.
- [76] E. Arruda, M. J. Mathew, M. Kopicki, M. N. Mistry, M. Azad, and J. L. Wyatt, “Uncertainty averse pushing with model predictive path integral control,” *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 497–502, 2017.
- [77] F. R. Hogan and A. Rodriguez, “Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics,” in *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, K. Goldberg, P. Abbeel, K. Bekris, and L. Miller, Eds. Cham: Springer International Publishing, 2020, pp. 800–815, ISBN: 978-3-030-43089-4. DOI: 10.1007/978-3-030-43089-4_51. [Online]. Available: https://doi.org/10.1007/978-3-030-43089-4_51.
- [78] K. Harada and M. Kaneko, “Enveloping grasp for multiple objects,” in *ICRA*, 1998.
- [79] K. Harada, M. Kaneko, and T. Tsujii, “Rolling-based manipulation for multiple objects,” *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 457–468, 2000.
- [80] T. Yamada, T. Ooba, T. Yamamoto, N. Mimura, and Y. Funahashi, “Grasp stability analysis of two objects in two dimensions,” in *ICRA*, 2005.
- [81] T. Yamada and H. Yamamoto, “Static grasp stability analysis of multiple spatial objects,” *Journal of Control Science and Engineering*, vol. 3, 2015.
- [82] T. Chen, A. Shenoy, A. Kolinko, S. Shah, and Y. Sun, “Multi-object grasping - estimating the number of objects in a robotic grasp,” in *IROS*, 2021.
- [83] A. Shenoy, T. Chen, and Y. Sun, “Multi-object grasping - efficient robotic picking and transferring policy for batch picking,” *IEEE IROS*, 2022.

- [84] P. Jiang, J. Oaki, Y. Ishihara, and J. Ooga, *Multiple-object grasping using a multiple-suction-cup vacuum gripper in cluttered scenes*, 2023. arXiv: 2304.10693 [cs.R0].
- [85] V. P. Nguyen and W. T. Chow, “Wiring-claw gripper for soft-stable picking up multiple objects,” *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 3972–3979, 2023. DOI: 10.1109/LRA.2023.3273512.
- [86] M. Dogar and S. S. Srinivasa, “A framework for push-grasping in clutter,” in *RSS*, 2011.
- [87] M. Danielczuk, J. Mahler, C. Correa, and K. Goldberg, “Linear Push Policies to Increase Grasp Access for Robot Bin Picking,” in *CASE*, 2018.
- [88] H. Huang, M. Dominguez-Kuhne, V. Satish, M. Danielczuk, K. Sanders, J. Ichnowski, A. Lee, A. Angelova, V. Vanhoucke, and K. Goldberg, “Mechanical search on shelves using lax-ray: Lateral access x-ray,” in *IEEE IROS*, 2021.
- [89] W. Agboh and M. Dogar, “Real-time online re-planning for grasping under clutter and uncertainty,” in *IEEE Humanoids*, 2018.
- [90] T. Sakamoto, W. Wan, T. Nishi, and K. Harada, “Efficient picking by considering simultaneous two-object grasping,” 2021.
- [91] E. Huang, Z. Jia, and M. T. Mason, “Large-scale multi-object rearrangement,” in *ICRA*, 2019.
- [92] W. Bejjani, W. Agboh, M. Dogar, and M. Leonetti, “Occlusion-aware search for object retrieval in clutter,” in *IEEE IROS*, 2021.
- [93] M. Hasan, M. Warburton, W. Agboh, M. Dogar, M. Leonetti, H. Wang, F. Mush-taq, M. Mon-Williams, and A. Cohn, “Human-like planning for reaching in cluttered environments,” in *ICRA*, 2020.
- [94] M. Mason, *Mechanics of Robotic Manipulation* (Intelligent Robotics and Autonomous Agents series). MIT Press, 2001, ISBN: 9780262263740. [Online]. Available: <https://books.google.com/books?id=Ngdeu3go014C>.
- [95] W. C. Agboh, D. Ruprecht, and M. R. Dogar, “Combining coarse and fine physics for manipulation using parallel-in-time integration,” *ISRR*, 2019.
- [96] J. E. King, M. Cagnetti, and S. S. Srinivasa, “Rearrangement planning using object-centric and robot-centric action spaces,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3940–3947. DOI: 10.1109/ICRA.2016.7487583.
- [97] H. Song, J. A. Haustein, W. Yuan, K. Hang, M. Y. Wang, D. Kragic, and J. A. Stork, “Multi-object rearrangement with monte carlo tree search: A case study on planar nonprehensile sorting,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 9433–9440. DOI: 10.1109/IROS45743.2020.9341532.

- [98] Muhayyuddin, M. Moll, L. Kavraki, and J. Rosell, “Randomized physics-based motion planning for grasping in cluttered and uncertain environments,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 712–719, 2018. DOI: 10.1109/LRA.2017.2783445.
- [99] W. C. Agboh and M. R. Dogar, “Pushing fast and slow: Task-adaptive planning for non-prehensile manipulation under uncertainty,” in *WAFR*, 2018.
- [100] W. Agboh and M. Dogar, “Robust physics-based manipulation by interleaving open and closed-loop execution,” *CoRR*, vol. abs/2105.08325, 2021.
- [101] W. Yuan, J. A. Stork, D. Kragic, M. Y. Wang, and K. Hang, “Rearrangement with nonprehensile manipulation using deep reinforcement learning,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 270–277. DOI: 10.1109/ICRA.2018.8462863.
- [102] K. Y. Goldberg, “Orienting polygonal parts without sensors,” *Algorithmica*, vol. 10, pp. 201–225, 1993.
- [103] M. Erdmann and M. Mason, “An exploration of sensorless manipulation,” *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 369–379, 1988. DOI: 10.1109/56.800.
- [104] A. M. Johnson, J. E. King, and S. Srinivasa, “Convergent planning,” *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1044–1051, 2016. DOI: 10.1109/LRA.2016.2530864.
- [105] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. [Online]. Available: <https://faculty.marshall.usc.edu/gareth-james/ISL/>.
- [106] B. Mirkin and B. Mirkin, *Clustering for Data Mining: A Data Recovery Approach*, 1st. Chapman and Hall/CRC, 2005. DOI: 10.1201/9781420034912.
- [107] D. Müllner, *Modern hierarchical, agglomerative clustering algorithms*, 2011. DOI: 10.48550/ARXIV.1109.2378. [Online]. Available: <https://arxiv.org/abs/1109.2378>.
- [108] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola, *Fast optimal leaf ordering for hierarchical clustering*, Jun. 2001. DOI: 10.1093/bioinformatics/17.suppl_1.S22. [Online]. Available: https://doi.org/10.1093/bioinformatics/17.suppl%5C_1.S22.
- [109] T.-B.-H. Dao, K.-C. Duong, and C. Vrain, *Constrained clustering by constraint programming*, Combining Constraint Solving with Mining and Learning, 2017. DOI: <https://doi.org/10.1016/j.artint.2015.05.006>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0004370215000806>.

- [110] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, *Automatic generation and detection of highly reliable fiducial markers under occlusion*, 2014. DOI: <https://doi.org/10.1016/j.patcog.2014.01.005>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0031320314000235>.