# Push Past Green: Learning to Look Behind Plant Foliage by Moving It

**Anonymous Author(s)**
Affiliation
Address
email

**Abstract:** Autonomous agriculture applications (*e.g.*, inspection, phenotyping, plucking fruits) require manipulating the plant foliage to look behind the leaves and the branches. Partial visibility, extreme clutter, thin structures, and unknown geometry and dynamics for plants make such manipulation challenging. We tackle these challenges through data-driven methods. We use self-supervision to train SRPNet, a neural network that predicts what space is revealed on execution of a candidate action on a given plant. We use SRPNet with the cross-entropy method to predict actions that are effective at revealing space beneath plant foliage. Furthermore, as SRPNet does not just predict how much space is revealed but also where it is revealed, we can execute a sequence of actions that incrementally reveal more and more space beneath the plant foliage. We experiment with a synthetic (vines) and a real plant (Dracaena) on a physical test-bed across 5 settings including 2 settings that test generalization to novel plant configurations. Our experiments reveal the effectiveness of our overall method, PPG, over a competitive hand-crafted exploration method, and the effectiveness of SRPNet over a hand-crafted dynamics model and relevant ablations. Project website with robot execution videos and paper overview: https://sites.google.com/view/pushingfoliage/.

**Keywords:** Deformable Object Manipulation, Model-building, Self-supervision

## 1 Introduction

The ability to autonomously manipulate plants is crucial in the pursuit of sustainable agricultural practices [1, 2, 3, 4]. Central to autonomous plant manipulation is the *plant self-occlusion problem*. Plants self-occlude themselves (Figure 1 (left)). Plant leaves and branches have to be carefully moved aside for the simplest of agriculture problems: plant inspection, phenotyping, precision herbicide application, or finding and plucking fruits. This papers tackles this plant self-occlusion problem. We develop methods that learn to manipulate plants so as to look beneath their external foliage. Figure 1 (middle and right) shows steps from a sample execution from our method. We believe our work will serve as a building block that enables many different applications that require manipulation of plants in unstructured settings.

Manipulating external plant foliage to reveal occluded space is hard. Sensing is difficult because of dense foliage, thin structures and partial observability. Control and planning is challenging because of unknown dynamics of the plant leaves and branches, and the difficulty of building a full articulable plant model. These sensing and control challenges motivate the need for learning. However, use of typical learning paradigms is also not straight-forward. Model-free RL (*e.g.* PPO [5]) requires interaction data at a scale that is difficult to collect in the real world. Model-based RL is more sample-efficient, but is quite challenging here as precisely predicting the next observation (or state) is hard. Imitation learning is more promising but for the exploration task we tackle, the next best action depends on what has already been explored. This increases the amount of demonstration data required to train models. Lack of high-fidelity plant simulators preclude simulated training.

Our proposal is to tackle this problem through self-supervision [6, 7]. We collect a dataset of action outcomes (amount of space revealed) by letting the robot randomly interact with plants. We use this data to train a model to predict space revealed by an input action. However, in order to derive

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

**Figure 1: (left)** Plants self-occlude themselves. Two examples of leaves and branches being pushed aside for inspection and picking fruits. This paper develops learning algorithms that enable robots to tackle this plant self-occlusion problem. We show actions executed by the robot to expose the space behind vines **(middle)** and Dracaena plant **(right)**.

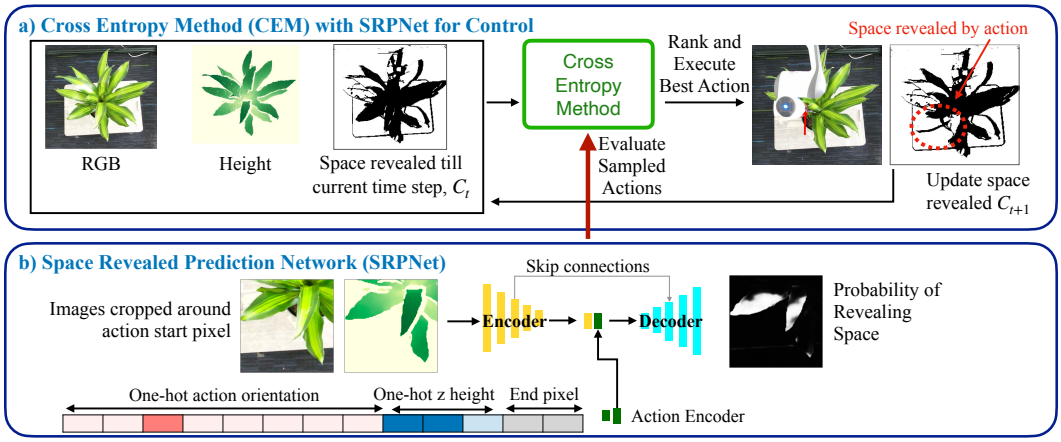


**Figure 2: Overview of PushPastGreen.** PushPastGreen learns to manipulate plants to reveal the space behind them thus tackling the plant self-occlusion problem. PushPastGreen includes Space-Revealed Prediction Network (SRPNet) that predicts where space is revealed upon execution of a pushing action, as shown in **(b)** and described in Sec. 4.1. SRPNet can not only rank actions based on how much space they will reveal, but because it can also predict *where* space gets revealed, it can also be used for executing multi-step trajectories that explore all the space behind the vines as shown in **(a)** and described in Sec. 4.3. SRPNet is trained using self-supervision as described in Sec. 4.2.

a long-term strategy for exploring all of the space beneath the plant, the model has to predict not only how much space would get revealed, but also *where* (Figure 2 (b), Section 4.1). In this way, the model output lets us reason about what *additional* space each action would reveal, allowing us to execute multi-step action sequences that explore all of the area behind the plant using a simple greedy control loop implemented via the cross-entropy method (CEM) (Figure 2 (a), Section 4.3).

This paper implements and tests these ideas on a physical platform that is tasked with revealing space behind decorative vines and a real Dracaena plant. We collect 48 hours of plant interaction data and use it to train a neural network that we call *Space-Revealed Prediction Network (SRPNet)*. SRPNet, when used with CEM, leads to effective control strategies to reveal all (or user-specified) space beneath the plant foliage. We call our overall framework *PushPastGreen* (PPG).

Experiments show that SRPNet outperforms a hand-crafted dynamics model and ablated versions of SRPNet. In physical experiments, PPG outperforms an hand-crafted exploration strategy and versions of PPG that replace SRPNet with alternative choices for modeling space revealed. In all 5 settings across vines and Dracaena, including 2 that explicitly test for generalization, we observe relative improvements ranging from 4% to 34% over the next best method. This establishes the benefits of PPG and the use of learning to manipulate plants. We will release code, data, and models.

## 2 Related Work

**Autonomous Agriculture.** Motivated by the need for adopting sustainable agricultural practices [2, 3, 1], researchers have sought to introduce and expand the use of autonomy for agricultural tasks [8,

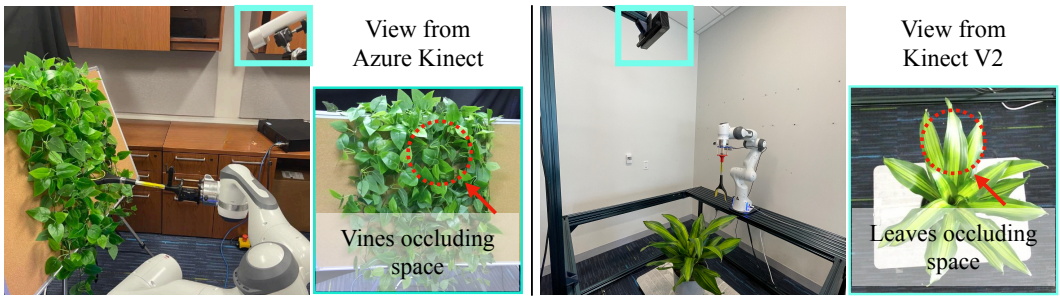


**Figure 3: Hardware setup for vines (left) and real Dracaena plant (right).** We use a grabber as the end-effector [52, 53]. View from the RGB-D camera is in the inset. The task is to move the vines and the Dracaena leaves aside to reveal the space occluded by them.

9]. While a full review is beyond our scope, major trends include a) development of specialized robotic hardware [10, 11, 12], b) development of algorithms for perception in cluttered agricultural settings [13, 14, 15], c) design of control algorithms for navigation [16, 17] and manipulation [18], and d) full autonomous farming systems [19, 18, 20, 21].

**Plant Manipulation.** For manipulation oriented tasks (*e.g.* fruit picking): [22] compute 3D grasp pose for largely unoccluded fruits, [23] design a visual servoing approach to get partially occluded fruits into full view, [24, 25, 26] output trajectories for reaching fruits while avoiding collisions with plant leaves and branches, and [10, 27] develop soft arms / end-effectors that can maneuver around plant structures. Much less research actually interacts with the plant structure to accomplish tasks. [18] hand-design strategies for pushing fruits out of the way. [28] show simulated results using probabilistic motion primitives for pushing fruits out of the way. We instead study the task of looking behind plant foliage, and hand-crafted strategies proposed in [28, 18] are not directly applicable to our setting. [29, 30] tackle reaching in plants while treating leaves as permeable obstacles, while [31] develops efficient MPC to minimize contact forces when interacting with plants. [32] learn to model object's resistance to movement by estimating stiffness distribution from torque measurements. We instead directly model the effect of actions executed on the plant.

**Manipulation of Deformable Objects.** Past works have considered manipulation of other deformable objects such as cloth [33, 34, 35, 36, 37], ropes [38, 39], elasto-plastics [40, 41], fluids [37, 42, 43], and granular media [44]. [33, 34, 38] design dynamic primitive actions to tackle cloth and rope manipulation. [45, 42, 40, 35] learn particle-based forward models for deformable material and use model-based RL for control. [46, 41] compose skills for deformable object manipulation to solve long-horizon tasks. Our study explores plant manipulation. Lack of high-fidelity plant simulators limits the applicability of past methods that rely on large amount of data in simulation [42, 36, 38]. At the same time, building dynamics models [40, 35] for plants is hard due to dense foliage, thin branch structure, and unknown heterogeneous dynamics.

**Self-supervised Learning in Robotics.** We adopt a self-supervised approach for training our models. Self-supervision techniques typically predict scalar quantities (*e.g.* grasp outcomes [6, 7], delta cloth coverage on workspace [33], pushing+grasping success [47], *etc.*). Past work has also used self-supervision to build forward models for model-predictive control [48, 49, 50, 51] in pixel or feature spaces. Our work finds a middle ground. We predict not just how much space is revealed (insufficient for executing a sequence of actions), but also where it is revealed. This lets us execute sequences of actions that incrementally expose more and more space.

## 3 Problem Setup

Figure 3 shows the 2 different plants that we tackle, a) (decorative) vines vertically hanging across a board, and b) a real Dracaena plant. The vines involve a 2D exploration problem and present challenges due to entanglement, thin structures, and extensive clutter. The real Dracaena plant exhibits a large variation in scene depth leading to a 3D problem. The Dracaena plant has big leaves that bend only in specific ways. Thus it requires careful action selection. Both test cases exhibit unknown and heterogeneous dynamics which makes it hard to manipulate them.

As one can notice in Figures 1 and 3, vines occlude the surface behind the vines. Similarly, the Dracaena leaves occlude the plant. We refer to this occlusion as the *plant self-occlusion problem*. The task is to have manipulation policies that can use the non-prehensile pushing actions (as described below) to reveal the space beneath the plant surface.

We use the Franka Emika robot and change the end effector to a grabber (as also done in past work [52, 53]). We use RGB-D cameras pointed at the plant for sensing. Our action space consists of non-prehensile planar pushing actions (also used in past work *e.g.* [51]). We sample a 3D location and push in a plane parallel to the board for the vines and to the ground for the Dracaena plant. As vines have limited depth variation, we use a fixed $z$ for the vines, but actions are sampled at varying $z$ for the Dracaena plant. Sections A.1 and B.1 provide more experimental details for the vines and the Dracaena plant.

# 4 Proposed Approach: Push Past Green

PPG adopts a greedy approach. We keep track of space that has not yet been revealed, and execute actions that would reveal the most *new* space. Doing this requires a model that predicts what space a candidate action would reveal. As plants are complex to model, such a model is hard to hand-craft. Furthermore, it is difficult to estimate the precise state and physical parameters for plants from a single RGB-D image (*e.g.* placement of leaves and branches with respect to one another, location and connectivity of all the leaves with the stems, stiffness parameters). This precludes the use of physical simulation for such prediction. Thus, we design Space-Revealed Prediction Network (SRPNet), that uses learning to directly predict space revealed on execution of a given action on a given plant configuration (Section 4.1). Learning to directly make this prediction sidesteps the complexity of precise state estimation and physical simulation necessary to build a full dynamics model for the plant. To obtain the data to train SRPNet, we adopt a self-supervised approach and execute random actions from the robot action space. We automatically compute the space revealed after an action using the RGB-D image (Section 4.2). Together with SRPNet, we design PPG, a greedy algorithm that uses the cross-entropy method (CEM) [54] to sample the action that promises to reveal the most *new* space on top of space already revealed (Section 4.3).

## 4.1 Space-Revealed Prediction Network

**Input Representation.** The input to our model is a $200 \times 200$ patch cropped out at the action start location. We crop both the RGB image and an image denoting the height relative to the surface beneath the vines (or relative to the ground beneath the Dracaena). The height image is computed using the point cloud from the RGB-D cameras. As the model sees crops around the site of interaction, each action starts at the center of the image. We only need to represent the $z$ coordinate of the push start location, the push direction ($\theta$), and the push distance ($d$). We represent these using a) a one-hot vector depicting the push direction, b) a one-hot $z$ height, and c) push distance via the location of the action end-point *i.e.* $[d\cos(\theta), d\sin(\theta)]$.

**Output.** The model produces an output that is the same size as the input. Each value in this spatial map represents the probability that space will get revealed at that location in the image upon execution of input action on the input plant configuration.

**Model Architecture and Loss.** We adopt the UNet structure [55] used in image segmentation. The encoder has 5 convolution layers. The action features are processed through 2 transposed convolution layers before being concatenated with the visual features and passed to the decoder with 5 transposed convolution layers. We add a skip connection between each corresponding convolutional and transposed convolution layer. SRPNet is trained using cross-entropy loss.

## 4.2 Data Collection and Preparation

Our self-supervised data collection procedure executes random actions from the robot action space. We divide the robot's reachable space into a grid of 2cm $\times$ 2cm cells. Action starting locations $(x, y, z)$ are sampled at the centers of these cells. We sample push directions and push by 15cm clipping to the feasible space as necessary. Each interaction takes about 30s of execution time. We record RGB-D videos and robot joint states over the entire duration of the interaction. We collected

3529 interactions for vines over 30 hours, and 2175 interactions for Dracaena over 18 hours. We split the dataset into train, val, and test splits in a 8:1:1 ratio and train one model for each plant.

We automatically compute ground truth for training the model on the collected data. This involves processing the RGB and depth image before and after the interaction. For vines, we found a simple decision rule using the color value and change in depth to work well. For Dracaena, very often the entire plant wobbles upon interaction. This leads to erroneous estimates. Thus, we first align the point clouds before and after interaction and then look for depth increase to obtain ground truth. More details are provided in Supplementary Sections A.3 and B.3.

### 4.3 Looking Behind Leaves Using SRPNet

Algorithm 1 describes our control algorithm that uses the trained SRPNet to predict actions to reveal space behind vines. At each timestep $t$ of the trajectory, we use the cross-entropy method (CEM) [54] to pick out the best action to execute (line 4). We maintain the revealed space so far ($C_t$). $C_0$ is initialized to be the space visible before any actions (line 1). Action parameters are sampled from Gaussian distributions. For each candidate action, SRPNet predicts where space would be revealed. We determine *new* space revealed by subtracting the area that has already been revealed ($C_t$) from SRPNet's output. Samples that are predicted to reveal the most *new space* are selected as elite which are used to fit a Gaussian distribution to sample actions for the next CEM iteration. After all iterations, CEM outputs $a_t$, the action found to reveal the most new space (line 4). Upon executing $a_t$, we observe the space that is actually revealed and update $C_t$ (line 6 and 7). The process is repeated for the length of the trial.

**Algorithm 1**: PPG: Revealing space beneath plants.

**Require:** Model $f$ that predicts space revealed after action
1: Current revealed space, $C_0 \leftarrow$ space visible at start
2: **for** $t \leftarrow 0$ to $T-1$ **do**
3: Receive rectified image $I_t^{\text{rectified}}$
4: $a_t \leftarrow \text{CEM}(C_t, f, I_t^{\text{rectified}})$
5: Execute action $a_t$
6: Calculate additional space revealed $c_t$
7: Update current revealed space: $C_{t+1} \leftarrow C_t \cup c_t$
8: **end for**

## 5 Experiments and Results

We test our proposed framework through a combination of offline evaluations of SRPNet on our collected dataset (Section 5.1), and online execution on our physical platform for the task of revealing space behind plants (Section 5.2). Our experiments test the effectiveness of a) learning at making predictions for revealed space, b) our neural network input representation, c) SRPNet at selecting effective actions, and d) SRPNet's spatial predictions for long-horizon and targeted exploration tasks.

### 5.1 Offline Evaluation of SRPNet

We train and evaluate SRPNet on data gathered on our physical setup as described in Section 4.2. We measure the average precision (AP) for the pixels labeled as revealed-space. We train on the train split, select model checkpoints on the validation set, and report performance on the test set in Table 1. For vines, we report performance in two settings: Vines [All] *i.e.* seeing the board (behind the vines) counts as revealed space, and Vines [5cm] *i.e.* height decrease of 5cm counts as revealed space. For Dracaena, only seeing past the leaf (as determined by our automated processing from Section 4.2) counts as revealed space.

| Methods | Vines [All] | Vines [5cm] | Dracaena |
|---|---|---|---|
| Full SRPNet (Our) | 46.3 | **54.4** | **44.2** |
| *Input Representation Ablations* | | | |
| No action | 30.2 | 43.5 | 28.4 |
| No height map | **46.9** | 49.1 | 40.6 |
| No RGB | 33.4 | 46.4 | 28.7 |
| No RGB and no height map | 28.4 | 35.2 | 10.5 |
| *Data Augmentation Ablations* | | | |
| No left/right flips | 44.1 | 52.7 | 34.6 |
| No color jitter | 41.0 | 51.4 | 30.7 |

**Table 1: Average precision for different models at predicting space revealed.** Higher is better. Our proposed input representation outperforms simpler alternatives and data augmentation boosts performance.

**Results.** Experiments presented in Table 1 reveal the effectiveness of our method and provide insights about the underlying data. **First**, across all three settings, the full model is able to extract

information from visual observations to produce higher quality output than just basing the predictions on the action information alone (*Full model vs. No RGB and no height map*). This also suggests that plant configuration (as depicted in the visual observations) is important in predicting the action outcome. **Second**, use of action information leads to better predictions (*Full model vs. No action*). This suggests that different actions at the same site produce different outcomes, and SRPNet is able to make use of the action information to model these differences. **Third**, looking at the performance across the three settings, both height map and RGB information are useful for accurate predictions. There are no trivial solutions of the form 'space gets revealed where height is high'. **Fourth**, as we only have a limited number of training samples, data augmentation strategies are effective. Supplementary Figure S3 visualizes the predictions from different versions of our model on samples from the test set. Note the nuances that our model is able to capture in contrast to the ablated versions.

### 5.2 Online Evaluation for Looking Behind Plants Task

We next measure the effectiveness of our proposed framework (PPG w/ SRPNet) for the task of looking behind plant surface (as introduced in Section 3). We measure the space revealed in units of $cm^2$ for vines and number of pixels for Dracaena.[1]

We start out by demonstrating that PPG w/ SRPNet is able to differentiate between good and bad actions. We then demonstrate our method on the task of looking behind plants over a 10 time-step horizon. Finally, we tackle the task of revealing space behind a user-specified spatial target. We conduct experiments on both the vines and the Dracaena plant. As plants can't be exactly reset, all experiments test generalization to some extent. To further test generalization, we also test explicitly on plant configurations different from those seen during training.

Inexact resets also pose a challenge when comparing methods. We randomly reset the plant (such as rotating the Dracaena) between trials and expect the variance due to inexact resets to average out over multiple trials. To prevent experimenter or other unknown environmental bias, we a) randomly interleave trials for different methods, and b) reset *before* revealing which method runs next.

#### 5.2.1 Baselines

**Tiling Baseline.** For the long-horizon exploration tasks, this hand-crafted baseline randomly samples from action candidates that are spread out across the workspace as shown in Figure 4. We aid this baseline by limiting the action candidates to be horizontal pushes for the vines and tangential actions for the Dracaena. We found these actions to be more effective than actions in other orientations, see Table S1 for vines and Table S3 for Dracaena.

**PPG w/ Other Dynamics Models.** To disentangle if the improvement is coming from our learned SRPNet or simply from keeping track of space that has already been revealed ($C_t$) in PPG, we swap SRPNet for other models in PPG. Specifically, we compare to a hand-crafted dynamics model (described below) and the SRPNet No Image (*i.e.* no RGB and no height map) model from Table 1.

**Hand-crafted Dynamics Model.** We construct hand-crafted forward models for vines and the Dracaena plant. These baseline models represent vines as vertically hanging spaghetti, and the Dracaena plant as 2D radially emanating spaghetti. Figure 5 shows the dynamics for this baseline model, and the induced free space upon action execution.

#### 5.2.2 Results

**Single Action Selection Performance.** Table 2 compares the effectiveness of PPG w/ SRPNet at picking an action that reveals the most occluded space, against random actions from the robot's action space. For the strongest comparison, we limit the random sampling to the most effective actions, horizontal pushes for the vines and tangential

| Method | Area Revealed | |
|---|---|---|
| | Vines ($cm^2$) | Dracaena (pixels) |
| Random Horizontal / Tangential Action | $211.7 \pm 35.6$ | $5125.6 \pm 2042.5$ |
| PPG w/ SRPNet (Our) | $\mathbf{344.1} \pm \mathbf{27.3}$ | $\mathbf{7644.4} \pm \mathbf{1494.3}$ |

**Table 2: PPG selects actions that are more effective at revealing space.**

[1]The criterion to automatically determine revealed space occasionally fails. We manually inspected test runs to confirm and fix the output of the automated method. Note, that this manual inspection is done during evaluation only. No method has access to such manual inspection, neither during training nor during execution.

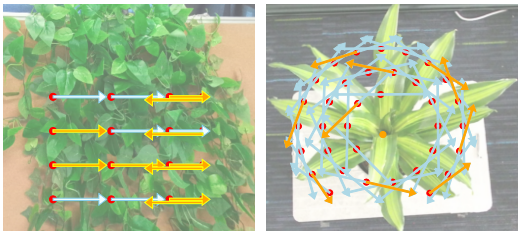

**Figure 4: Tiling baseline.** Cyan arrows show all action candidates considered, and orange arrows show 10 actions selected during an execution. We aid the baseline by limiting the candidates to the most effective actions (horizontal pushes for vines, tangential pushes for Dracaena).

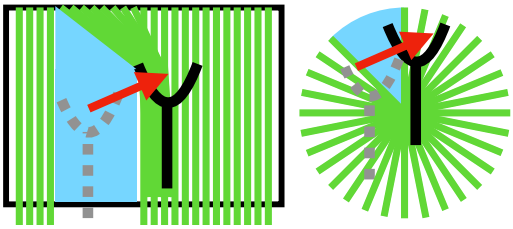

**Figure 5: Hand-crafted dynamics model** that represents vines as vertically hanging spaghetti (left), and the Dracaena plant as 2D radially emanating spaghetti (right). Cyan area represents the space revealed by the action (red arrow) under this hand-crafted model.

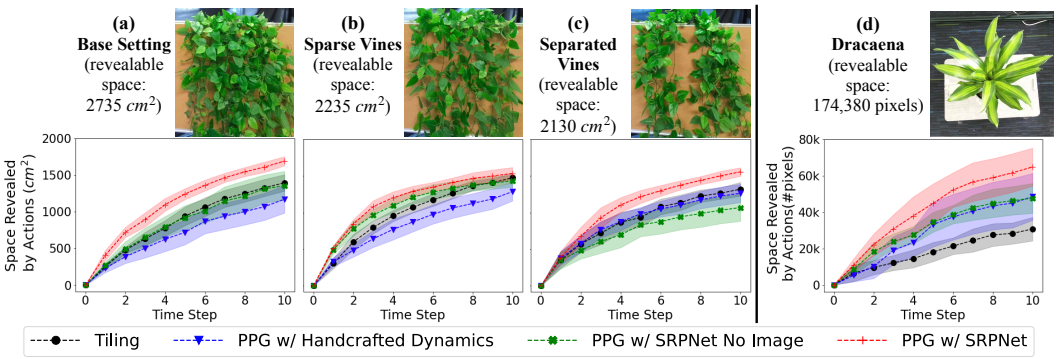


**Figure 6: Comparison of different methods for multi-step exploration of space behind plant foliage.** We show results in four settings across vines (left) and Dracaena (right). The line plots on the right show average cumulative space revealed by actions up to time step $t$ (along with 95% confidence intervals) across 10 trials. Training data for training SRPNet was collected in the base setting shown in (a) and (d) for vines and Dracaena respectively. (b) and (c) are novel settings that test generalization capabilities of our model. Our method (PPG w/ SRPNet) outperforms all baselines (a strong hand-crafted policy, and PPG with other dynamics models) across all settings.

pushes for the Dracaena, as shown in Figure 4). Table 2 reports the average space revealed (along with 95% confidence interval) over at least 20 trials for each method. Our approach leads to a relative improvement of 62% for vines and 49% for Dracaena over this strong baseline. This suggests that our model is able to interpret visual observations to identify good interaction sites.

**Long-horizon Exploration Performance.** Next, we study if SRPNet can be used for situations that require multiple sequential interactions to reveal space behind plants. The task is to maximize the cumulative space revealed over a 10 time-step episode. This further tests the quality of SRPNet which now also needs to accurately predict *where* it thinks space will be revealed.

We conduct 4 experiments, one on Dracaena and 3 on vines. For the vines, we considered 3 settings: a) Base Setting: vine setting as used for collecting training data, and 2 novel settings to test generalization: b) Sparse Vines, and c) Separated Vines. While the last two settings explicitly test generalization, we note that the first setting also tests models on novel vine configurations not exactly seen in training. For Dracaena, we only conducted experiments in the Base Setting.

Figure 6 plots the average space revealed (in $cm^2$ for vines and in pixels for Dracaena) as a function of the number of time-steps. We report the mean over 10 trials and also show the 95% confidence interval. Across all three experiments our proposed method achieves the strongest performance. Supplementary Figure S4 and videos on the website show some sample executions.

Results suggests that SRPNet is quite effective at predicting where space will get revealed (PPG w/ SRPNet *vs.* Tiling). Learning and planning via CEM lets us model complex behavior which is hard to hand-craft. Improvements over the tiling baseline increase as the action space becomes

larger (Dracaena *vs.* vines). Moreover, benefits don't just come because of keeping track of revealed space ($C_t$), but also from the use of SRPNet (PPG w/ SRPNet *vs.* PPG w/ Handcrafted Dynamics). Furthermore, our model is able to interpret the nuances depicted in the visual information to predict good actions (PPG w/ SRPNet *vs.* PPG w/ SRPNet No Image). SRPNet also leads to benefits in novel vine configurations. Benefits are larger in the separated vines case than for the sparse vines. This is possibly because the separated vines are still locally dense and SRPNet processes local patches.

**Targeted Revealing Performance.** Our final experiment tackles the task of targeted exploration. The task here is to reveal space at a user-defined region, $m$. Figure 7 (left) shows a sample user-selected region. We tackle this task by setting $C_0$ to be $\bar{m}$, the complement of the user-defined region. Figure 7 (right) presents the results (same legend as for Figure 6, but without Tiling Baseline). Again, as SRPNet reliably models the effect of actions, PPG with SRPNet outperforms PPG with other dynamics models.

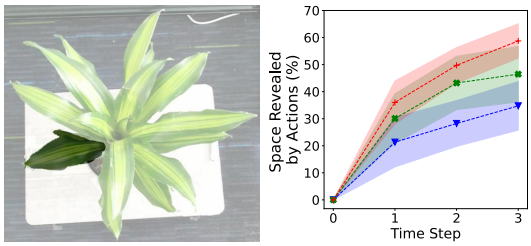


**Figure 7: PPG is also effective at revealing space behind a specific spatial target.**

## 6 Discussion

In this paper, we introduced PPG and SRPNet to tackle the problem of manipulating the external plant foliage to look within the plant (the *plant-self occlusion problem*). SRPNet uses self-supervised learning to model what space is revealed upon execution of different actions on plants. This sidesteps the difficulty in perception arising from dense foliage, thin structures, and partial information. PPG derives control strategies using SRPNet via CEM to output sequence of actions that can incrementally explore space occluded by plants. Experiments on a physical platform demonstrate the benefits of our proposed learning framework for tackling the plant self-occlusion problem. We believe our solution to the plant self-occlusion problem will serve as a building block for downstream plant manipulation applications in unstructured settings.

## 7 Limitations

We believe this is a unique and first-of-its-kind study, but it has its own limitations. We note two failure modes. First, PPG sometimes resamples an overly optimistic action (that doesn't actually reveal much space, so nothing changes and CEM returns a very similar next action) many times over without making progress. Second, as each individual push action doesn't use visual feedback it can't recover from say when a leaf slips from below the gripper. These may be mitigated by incorporating spatial diversity while selecting actions and by learning closed loop leaf manipulation policies through imitation. More generally, our overall approach relies on input from RGB-D cameras that are known to performance poorly in the wild. This may be mitigated through use of specialized stereo cameras built for farm settings [56]. Our techniques for automatic estimation of revealed space can be improved further using recent point tracking models [57], and it may be useful to build models that can predict and keep track of full 3D space. Experiments should be conducted with more diverse real plants. Future work should also rank actions from the perspective of the damage they cause to the plant, perhaps via some tactile sensing [30]. Lastly, while autonomous agriculture provides a path towards sustainable agricultural practices, societal impact of such automation should be studied before deployment.

## References

[1] A. J. Capellesso, A. A. Cazella, A. L. Schmitt Filho, J. Farley, and D. A. Martins. Economic and environmental impacts of production intensification in agriculture: comparing transgenic, conventional, and agroecological maize crops. *Agroecology and Sustainable Food Systems*, 40 (3):215–236, 2016.

[2] H. C. J. Godfray and T. Garnett. Food security and sustainable intensification. *Philosophical transactions of the Royal Society B: biological sciences*, 369(1639):20120273, 2014.

[3] J. A. Foley, N. Ramankutty, K. A. Brauman, E. S. Cassidy, J. S. Gerber, M. Johnston, N. D. Mueller, C. O'Connell, D. K. Ray, P. C. West, et al. Solutions for a cultivated planet. *Nature*, 478(7369):337–342, 2011.

[4] A. M. Davis and J. Pradolin. Precision herbicide application technologies to decrease herbicide losses in furrow irrigation outflows in a northeastern australian cropping system. *Journal of agricultural and food chemistry*, 64(20):4021–4028, 2016.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[6] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[7] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

[8] J. J. Roldán, J. del Cerro, D. Garzón-Ramos, P. Garcia-Aunon, M. Garzón, J. De León, and A. Barrientos. Robots in agriculture: State of art and practical experiences. *Service robots*, pages 67–90, 2018.

[9] C. W. Bac, E. J. Van Henten, J. Hemming, and Y. Edan. Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, 31(6):888–911, 2014.

[10] N. K. Uppalapati, B. Walt, A. J. Havens, A. Mahdian, G. Chowdhary, and G. Krishnan. A berry picking robot with a hybrid soft-rigid arm: Design and task space control. In *Robotics: Science and Systems*, page 95, 2020.

[11] W. McAllister, D. Osipychev, A. Davis, and G. Chowdhary. Agbots: Weeding a field with a team of autonomous robots. *Computers and Electronics in Agriculture*, 163:104827, 2019.

[12] A. Silwal, F. Yandun, A. Nellithimaru, T. Bates, and G. Kantor. Bumblebee: A path towards fully autonomous robotic vine pruning. *arXiv preprint arXiv:2112.00291*, 2021.

[13] H. Freeman, M. Qadri, A. Silwal, P. O'Connor, Z. Rubinstein, D. Cooley, and G. Kantor. Autonomous apple fruitlet sizing and growth rate tracking using computer vision. *arXiv preprint arXiv:2212.01506*, 2022.

[14] A. Silwal, T. Parhar, F. Yandun, H. Baweja, and G. Kantor. A robust illumination-invariant camera system for agricultural applications. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3292–3298. IEEE, 2021.

[15] F. Yandun, A. Silwal, and G. Kantor. Visual 3d reconstruction and dynamic simulation of fruit trees for robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 54–55, 2020.

[16] A. N. Sivakumar, S. Modi, M. V. Gasparino, C. Ellis, A. B. Velasquez, G. Chowdhary, and S. Gupta. Learned visual navigation for under-canopy agricultural robots. In *Robotics: Science and Systems*, 2021.

[17] A. E. B. Velasquez, V. A. H. Higuti, M. V. Gasparino, A. N. Sivakumar, M. Becker, and G. Chowdhary. Multi-sensor fusion based robust row following for compact agricultural robots. *arXiv preprint arXiv:2106.15029*, 2021.

[18] Y. Xiong, Y. Ge, L. Grimstad, and P. J. From. An autonomous strawberry-harvesting robot: Design, development, integration, and field evaluation. *Journal of Field Robotics*, 37(2):202–224, 2020.

[19] N. Strisciuglio, R. Tylecek, M. Blaich, N. Petkov, P. Biber, J. Hemming, E. van Henten, T. Sattler, M. Pollefeys, T. Gevers, et al. Trimbot2020: an outdoor robot for automatic gardening. In *ISR 2018; 50th International Symposium on Robotics*, pages 1–6. VDE, 2018.

[20] M. Presten, Y. Avigal, M. Theis, S. Sharma, R. Parikh, S. Aeron, S. Mukherjee, S. Oehme, S. Adebola, W. Teitelbaum, et al. Alphagarden: Learning to autonomously tend a polyculture garden. *arXiv preprint arXiv:2111.06014*, 2021.

[21] S. Parsa, B. Debnath, M. A. Khan, et al. Autonomous strawberry picking robotic system (robofruit). *arXiv preprint arXiv:2301.03947*, 2023.

[22] H. Kang, H. Zhou, and C. Chen. Visual perception and modeling for autonomous apple harvesting. *IEEE Access*, 8:62151–62163, 2020.

[23] C. Lehnert, D. Tsai, A. Eriksson, and C. McCool. 3d move to see: Multi-perspective visual servoing for improving object views with semantic segmentation. *arXiv preprint arXiv:1809.07896*, 2018.

[24] L. Luo, H. Wen, Q. Lu, H. Huang, W. Chen, X. Zou, C. Wang, et al. Collision-free path-planning for six-dof serial harvesting robot based on energy optimal and artificial potential field. *Complexity*, 2018, 2018.

[25] C. Schuetz, J. Baur, J. Pfaff, T. Buschmann, and H. Ulbrich. Evaluation of a direct optimization method for trajectory planning of a 9-dof redundant fruit-picking manipulator. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2660–2666. IEEE, 2015.

[26] A. Tafuro, B. Debnath, A. M. Zanchettin, and E. A. Ghalamzan. dpmp-deep probabilistic motion planning: A use case in strawberry picking robot. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8675–8681. IEEE, 2022.

[27] J. F. Elfferich, D. Dodou, and C. D. Santina. Soft robotic grippers for crop handling or harvesting: A review. *IEEE Access*, 10:75428–75443, 2022. doi:10.1109/ACCESS.2022.3190863.

[28] S. Mghames, M. Hanheide, and A. Ghalamzan. Interactive movement primitives: Planning to push occluding pieces for fruit picking. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2616–2623. IEEE, 2020.

[29] H. Nemlekar, Z. Liu, S. Kothawade, S. Niyaz, B. Raghavan, and S. Nikolaidis. Robotic lime picking by considering leaves as permeable obstacles. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3278–3284. IEEE, 2021.

[30] T. Bhattacharjee, P. M. Grice, A. Kapusta, M. D. Killpack, D. Park, and C. C. Kemp. A robotic system for reaching in dense clutter that integrates model predictive control, learning, haptic mapping, and planning. Georgia Institute of Technology, 2014.

[31] M. D. Killpack, A. Kapusta, and C. C. Kemp. Model predictive control for fast reaching in clutter. *Autonomous Robots*, 40:537–560, 2016.

[32] S. Yao and K. Hauser. Online estimation of point-based volumetric stiffness model using joint torque sensors. In *2nd Workshop on Representing and Manipulating Deformable Objects at ICRA2022*, 2022.

[33] H. Ha and S. Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. In *Conference on Robot Learning*, pages 24–33. PMLR, 2022.

[34] Z. Xu, C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song. Dextairity: Deformable manipulation can be a breeze. In *Robotics: Science and Systems*, 2022.

[35] Z. Huang, X. Lin, and D. Held. Mesh-based dynamics with occlusion reasoning for cloth manipulation. In *Robotics: Science and Systems*, 2022.

[36] T. Weng, S. M. Bajracharya, Y. Wang, K. Agrawal, and D. Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. In *Conference on Robot Learning*, pages 192–202. PMLR, 2022.

[37] X. Lin, Y. Wang, J. Olkin, and D. Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 432–448. PMLR, 2021.

[38] C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song. Iterative residual policy: for goal-conditioned dynamic manipulation of deformable objects. In *Robotics: Science and Systems*, 2022.

[39] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2146–2153. IEEE, 2017.

[40] H. Shi, H. Xu, Z. Huang, Y. Li, and J. Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks. In *Robotics: Science and Systems*, 2022.

[41] X. Lin, Z. Huang, Y. Li, J. B. Tenenbaum, D. Held, and C. Gan. Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[42] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[43] C. Do, C. Gordillo, and W. Burgard. Learning to pour using deep deterministic policy gradients. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3074–3079. IEEE, 2018.

[44] C. Schenck, J. Tompson, S. Levine, and D. Fox. Learning robotic manipulation of granular media. In *Conference on Robot Learning*, pages 239–248. PMLR, 2017.

[45] Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, and R. Tedrake. Propagation networks for model-based control under partial observation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1205–1211. IEEE, 2019.

[46] X. Lin, C. Qi, Y. Zhang, Z. Huang, K. Fragkiadaki, Y. Li, C. Gan, and D. Held. Planning with spatial-temporal abstraction from point clouds for deformable object manipulation. In *6th Annual Conference on Robot Learning*, 2022.

[47] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4238–4245. IEEE, 2018.

[48] M. I. Jordan and D. E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive science*, 16(3):307–354, 1992.

[49] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[50] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems*, 29, 2016.

[51] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Advances in neural information processing systems*, 29, 2016.

[52] S. Song, A. Zeng, J. Lee, and T. Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3): 4978–4985, 2020.

[53] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto. Visual imitation made easy. *CoRL*, 2020.

[54] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134:19–67, 2005.

[55] 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: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[56] A. Silwal, T. Parhar, F. Yandun, H. Baweja, and G. Kantor. A robust illumination-invariant camera system for agricultural applications. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3292–3298, 2021. doi:10.1109/IROS51168.2021.9636542.

[57] A. W. Harley, Z. Fang, and K. Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 59–75. Springer, 2022.

[58] K. Zhang, M. Sharma, J. Liang, and O. Kroemer. A modular robotic arm control stack for research: Franka-interface and frankapy. *arXiv preprint arXiv:2011.02398*, 2020.