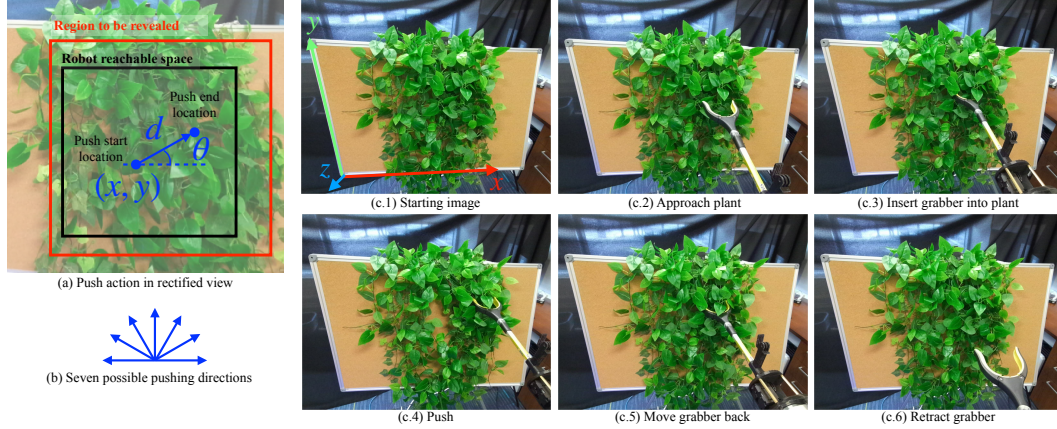


## Push Past Green: Learning to Look Behind Plant Foliage by Moving It

### Supplementary Material

## A Implementation Details for Vine Experiments

### A.1 Robot Action Space



**Figure S1: Robot’s action space for vine setup.** (a) shows the rectified image that we operate in, the region to be revealed (red box), and the region that the robot can reach (black box). The robot can execute push actions that start at a pixel  $(x, y)$  in the rectified image and push a distance of  $d$  at an angle  $\theta$ . We use 7 discrete push directions  $\{0, \pi/6, \pi/3, \pi/2, \dots, \pi\}$  as shown in (b). (c.1) through (c.6) show a sample execution of the push action.

The robot’s action space consists of non-prehensile pushing actions. As shown in Figure S1 (a), these actions are parameterized by  $(x, y, \theta, d)$ . Such parameterization for pushing actions has been used in past works, *e.g.* [51]. Here,  $(x, y)$  denotes the start location for the push interaction on the board,  $\theta$  denotes the push angle, and  $d$  denotes the push length. As shown in Figure S1 (b), we sample  $\theta$  to be one of 7 angles from  $\{0, \pi/6, \pi/3, \pi/2, 2\pi/3, 5\pi/6, \pi\}$ . We do not sample angles greater than  $\pi$  because pushing towards the bottom of the vines only drags down the vines and could pull the board over. We assume that the grabber inserts deep enough into the vines to push the vines but not too far to knock it over; therefore, the pushes are planar actions executed with the same  $z$  value. We estimate the location and orientation of the board and establish a coordinate frame that is aligned with the board. Push locations and orientations are expressed in this coordinate frame. We implement these actions by moving the grabber through 4 waypoints, as shown in Figure S1 (c.2) to Figure S1 (c.5). In Figure S1 (c.4), we can see the effect of a randomly sampled action on the state of the vines. We drive the Franka Emika robot between these waypoints using the Franka-interface and frankapy library [58].

### A.2 SRPNet

For the vine setup, we are unable to position the camera such that it is perpendicular to the board. Therefore, we design SRPNet to work on rectified images of the scene, such that the camera is looking straight at the vines. This corresponds to using a homography to transform the image such that the surface underneath the vines becomes fronto-parallel. We build the model to only reason about a  $40\text{cm} \times 40\text{cm}$  neighborhood around the action start location. Parts of the board get occluded behind the robot arm as the robot executes the action. These occluded parts and area with no depth readings are masked out for evaluation and training.

### A.3 Data Collection

The robot’s actions are in the same fronto-parallel plane used for SRPNet as described earlier. We estimate the space that can be safely reached by the robot ahead of time to make sure it is not close to its joint limits during interactions. The resulting space is roughly  $40\text{cm} \times 40\text{cm}$ . We divide the feasible space into a  $20 \times 20$  grid. Action starting locations  $(x, y)$  are sampled at the centers of these

Push Angle	0	$\pi/6$	$\pi/3$	$\pi/2$	$2\pi/3$	$5\pi/6$	$\pi$	Full Dataset
# Interactions	985	460	360	348	359	433	584	3529
Mean area revealed (cm <sup>2</sup> )	215.7	177.3	93.6	58.8	100.4	180.9	237.1	170.3

**Table S1: Statistics for the different push directions in the collected vine dataset.** Collected dataset reveals many aspects of the problem. For example, for vines, horizontal push actions (0 and  $\pi$ ) are the most effective at this task.

504 grid squares (*i.e.*, 400 possible starting locations). We sample push directions from the 7 possible  
505 angles,  $\{0, \pi/6, 2\pi/6, \dots, 6\pi/6\}$ , and push by 15cm clipping to the feasible space as necessary.  
506 Therefore, not all interactions have  $d = 15$ ; for starting locations near the boundary,  $d < 15$ .

507 Our full dataset contains 3529 interactions (summed to roughly 30 hours) collected over 11 different  
508 days (nonconsecutive). This data includes 2571 interactions done specifically for the purpose of  
509 data collection. The remaining interactions come from when we were developing control algorithms.  
510 These don't follow uniform sampling from the robot's action space and are biased towards horizontal  
511 actions since the most effective actions for the baselines are often horizontal actions.

512 We automatically compute the ground truth for training the model on the collected data. Specifically,  
513 we use color thresholding to determine when the surface beneath the vines has been fully exposed.  
514 We found this simple strategy to be reasonably robust. Note that while we train and use SRPNet  
515 to predict whether *all* vines were moved aside to reveal the board, we can process the data in other  
516 ways to also train the model for other tasks. For example, we can re-purpose the data for a task that  
517 involves only looking beneath the first layer of vines. We can re-compute ground truth to identify  
518 locations where the height decreased by (say) more than 5cm for such a task.

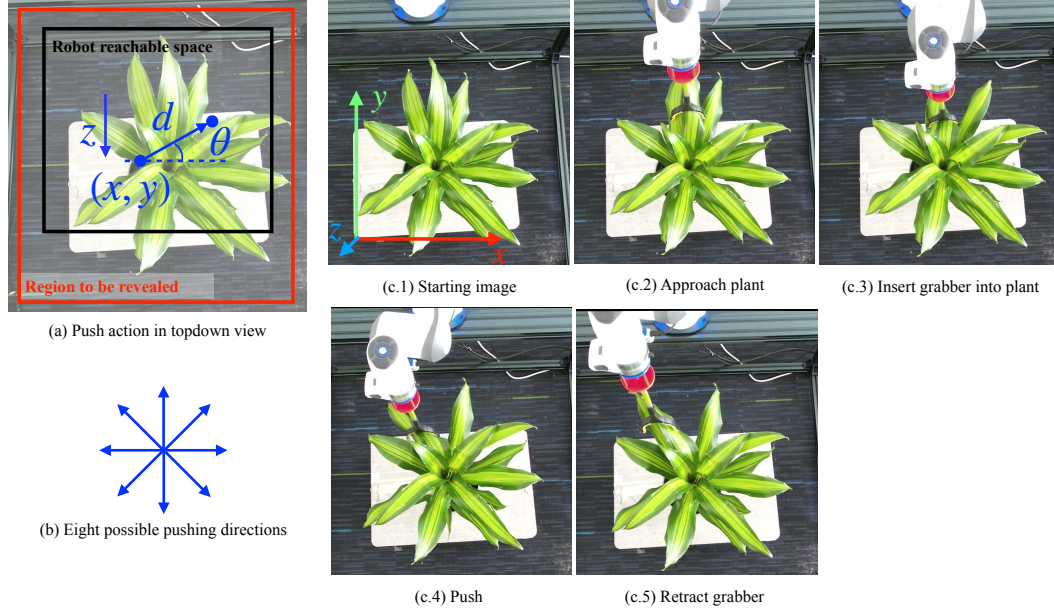
#### 519 A.4 Cross-entropy Method

520 Our CEM implementation uses 3 iterations that each evaluate 300 candidate actions. We sample  
521  $(x, y, \theta)$  from Gaussian distributions. In the first CEM iteration,  $x, y, \theta$  are sampled from Gaussians  
522 with different mean and variances, chosen to cover the whole action space. The parameters are then  
523 discretized to match the distribution from data collection. When sampling actions, we only retain  
524 action samples that are feasible (*i.e.* within the robot's reachable space as shown in Figure S1 (a)).  
525 Elite samples are the top 20% candidates that have the most amount of *new space* revealed. Running  
526 line 3 to 6 in Algorithm 1 (Section 4.3) for vines takes about 5 seconds.

## B Implementation Details for Dracaena Experiments

### B.1 Robot Action Space

The robot’s action space for Dracaena is similar to that of vines. However, since the Dracaena leaves are at different heights, we define three possible  $z$  values that the grabber can insert to. The Dracaena plant body is about 45cm tall so we defined the  $z$  values to be about 22.5, 17.5, and 12.5cm from the top of the plant. For each  $z$  value, planar pushing actions  $(x, y, \theta, d)$  are defined on a plane parallel to the ground. We sample  $\theta$  from 8 possible angles:  $\{0, \pi/4, \pi/2, 3\pi/4, \pi, 5\pi/4, 3\pi/2, 7\pi/4\}$ . The angles are 45 degrees away from one another instead of 30 degrees as used in vines because we want to keep the total number of possible actions reasonable.



**Figure S2: Dracaena robot action space.** Similar to Figure S1, (a) shows the image from the camera, (b) shows the pushing directions, and (c) shows the sample execution of a push action.

### B.2 SRPNet

Since the Kinect camera is looking down at the Dracaena plant, SRPNet does not work on rectified images as it does for vines and instead takes in images from the camera as they are. We project action start locations into their image coordinates using the camera intrinsics and crop around the locations to obtain local patches to input into the network. When training SRPNet, adding another head to predict height decrease in addition to the binary classification head helps AP performance. We use Huber loss with  $\delta = 0.1$  to provide an auxiliary loss to the network.

### B.3 Data Collection

The reachable space of the robot in the Dracaena setup is roughly  $57\text{cm} \times 53\text{cm}$  and corresponds to a  $29 \times 27$  grid of 2cm cells. Similar to the vines’ setup, the action starting point  $(x, y)$  is sampled from these 783 possible locations. Given that pushing from the center of the plant tends to displace it entirely, we aim to discourage such actions to prevent damage to areas where new leaves may sprout. We manually delineate a rectangular region around the plant center and do not sample or execute actions in this region. We also sample  $z$  from 3 possible values (22.5, 17.5, and 12.5cm from the top of the plant as mentioned before), push directions from 8 possible angles,  $\{0, \pi/4, \pi/2, 3\pi/4, \pi, 5\pi/4, 3\pi/2, 7\pi/4\}$ , and push by 15cm clipping to the feasible space as necessary. Therefore, not all interactions have  $d = 15$ ; for starting locations near the boundary,  $d < 15$ .

Since the plant wobbles during pushing, we discount the area that is revealed due to whole-plant movement. We construct plant point clouds before and after an action; then, iterative closest point (ICP) is performed to align the two point clouds. During execution, the robot body occludes parts

Push Angle	0	$\pi/4$	$\pi/2$	$3\pi/4$	$\pi$	$5\pi/4$	$3\pi/2$	$7\pi/4$	Full Dataset
# Interactions	257	262	295	273	249	297	289	253	2175
Mean area revealed (pixels)	1391.4	1138.7	990.4	802.0	1154.0	1110.8	1154.9	1495.2	1147.7

**Table S2: Statistics for the different push directions in the collected Dracaena dataset.**

of the plant, so we mount a Intel RealSense camera at the wrist to fill in these occluded regions to aid ICP. Area where the plant height has decreased in the aligned point cloud is considered to be revealed space.

#### B.4 Cross-entropy Method

We follow the same algorithm as the one outlined in Algorithm 1 (Section 4.3). The Dracaena CEM uses 3 iterations that each evaluate 300 candidate actions. We sample  $(x, y, \theta, z)$  from uniform distributions within the robot’s reachable space. The parameters are then discretized to match the data collection’s distribution. Top 20% candidates that reveal the most amount of new space are chosen as elite samples that are fitted with Gaussian distributions for the next iteration. Running one iteration takes about 7 seconds.

#### B.5 Comparing Tangential to Random Actions

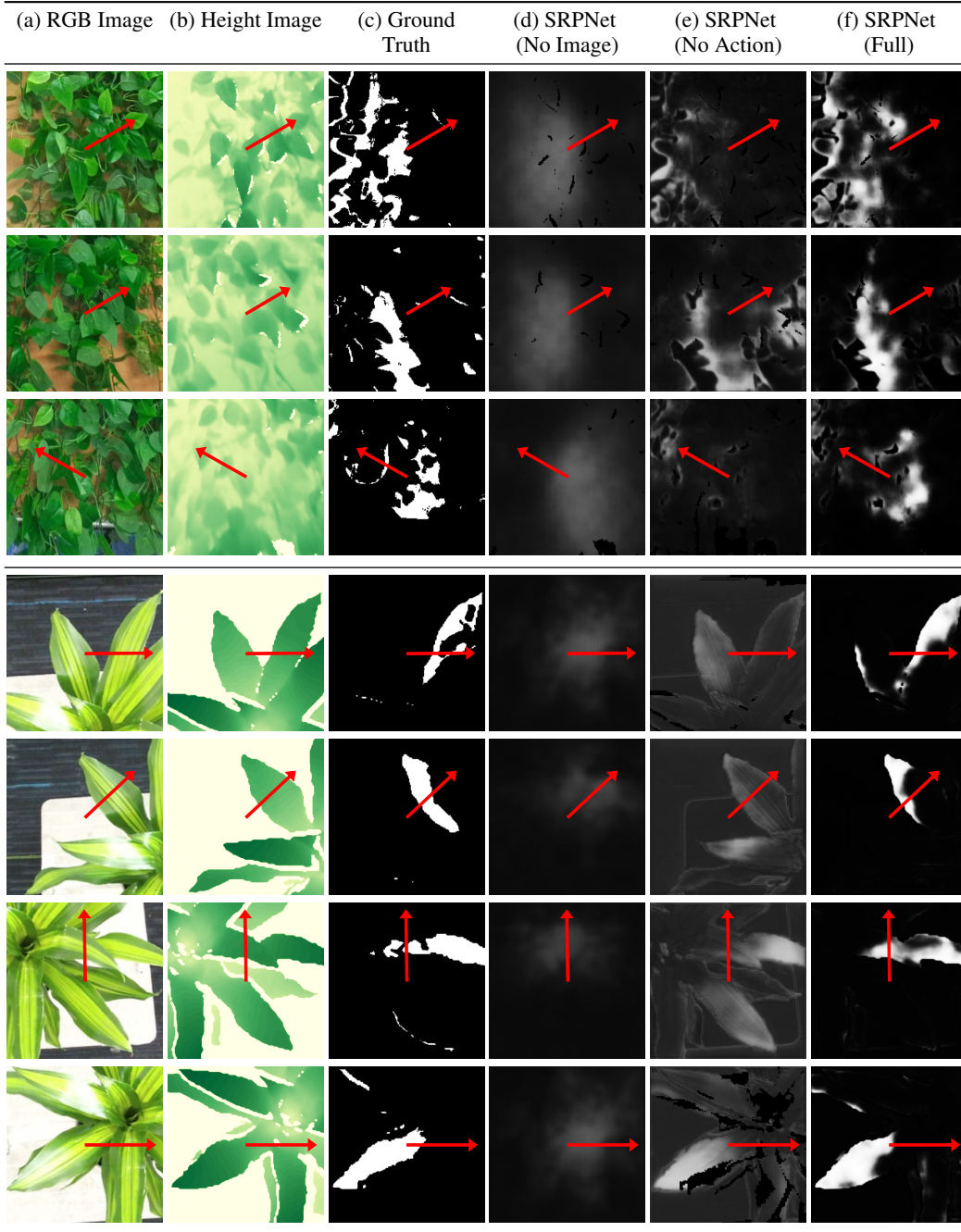
Method	Area revealed (pixels)
Random Action	$3956.1 \pm 1213.2$
Tangential Action	<b><math>5125.6 \pm 2042.5</math></b>

**Table S3: Effectiveness of tangential actions.** We execute actions tangent to Dracaena leaves in the Tiling baseline because they reveal more space on average compare to random actions.

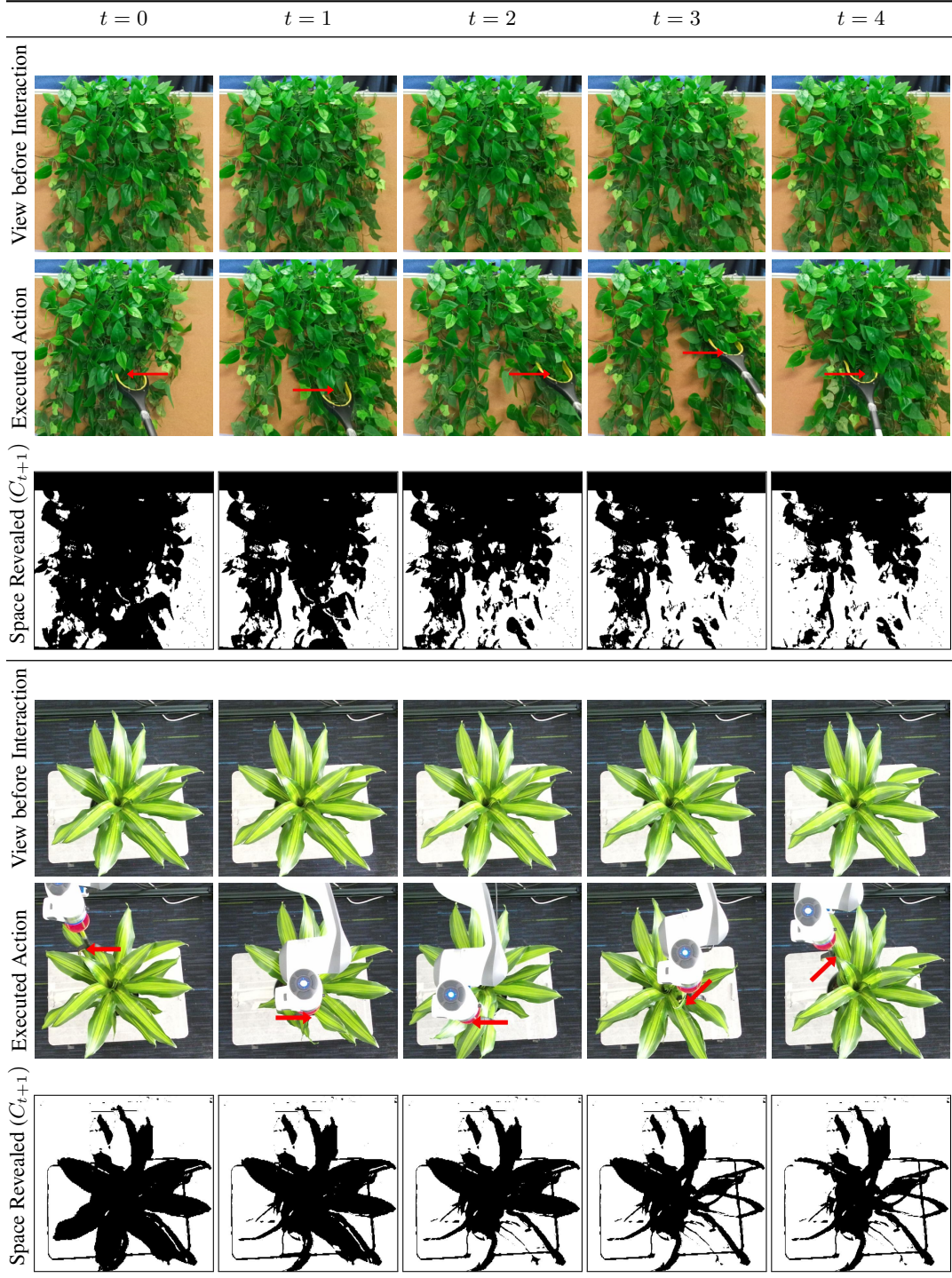
We chose horizontal actions for the Tiling baseline of vines because they on average reveal the most amount of space. In order to come up with a similar Tiling baseline for Dracaena, we observe that leaves are pushed aside more easily when the grabber moves tangent to the leaves. We verify that tangent actions are better than random actions by comparing average space revealed upon execution of actions from the two methods. As shown in Table S3, tangential actions reveal more space than random actions, so we use them in the Tiling baseline to test the effectiveness of PPG w/ SRPNet against this strong baseline.



## C Visualizations



**Figure S3: Visualizations of output from our proposed SRPNet.** We show examples from the test set. The white regions in ground truth images represent space revealed by actions drawn as red arrows. Column (d) shows prediction from SRPNet without image input (*i.e.* no RGB, no height), column (e) shows prediction from SRPNet without action input, and column (f) shows predictions from SRPNet. The brighter the region, the higher the predicted probability of revealing space. Ground truth revealed space indicates the complexity of the task and suggests why the hand-crafted dynamics model (shown in Figure 5) performs poorly at this task. SRPNet is able to effectively use the visual information to make good predictions.



**Figure S4: First five time steps of a sample execution from our method.** Top row shows the RGB image before interaction, middle row shows the push action executed, and the bottom row shows the cumulative space revealed so far. Our model picks actions that are effective at revealing space.