By Alexander You, Nidhi Parayil, Josyula Gopala Krishna, Uddhav Bhattarai, Ranjan Sapkota, Dawood Ahmed, Matthew Whiting, Manoj Karkee, Cindy M. Grimm, and Joseph R. Davidson

Semiautonomous Precision Pruning of Upright Fruiting Offshoot Orchard Systems

©SHUTTERSTOCK.COM/NET VECTOR

An Integrated Approach

Dormant pruning is an important orchard activity for maintaining tree health and producing high-quality fruit. Due to decreasing worker availability, pruning is a prime candidate for robotics. However, pruning also represents a uniquely difficult problem, requiring robust systems for perception, pruning point determination, and manipulation that must operate under variable lighting conditions and in complex, highly unstructured environments. In this article, we introduce a system for pruning modern planar orchard architectures with simple pruning rules that combines various subsystems from our previous work on perception and manipulation. The integrated system demonstrates the ability to autonomously detect and cut pruning targets with minimal control of the environment, laying the groundwork for a fully autonomous system in the future. We validate the performance of our system through field trials in a sweet cherry

Digital Object Identifier 10.1109/MRA.2023.3309098
Date of current version: 11 September 2023

orchard, ultimately achieving a cutting success rate of 58% across 10 trees. Though not fully robust and requiring improvements in throughput, our system is the first to operate on fruit trees and represents a useful base platform to be improved in the future.

INTRODUCTION

The production of high-value tree fruit crops, such as fresh market apples, pears, and cherries, requires a large seasonal workforce. After harvesting, the most labor-intensive orchard activity is dormant season pruning (i.e., after leaf drop) [1]. Pruning is a critical perennial operation required to maintain tree health and produce high yields of quality fruit. Pruning rejuvenates the tree, replacing unproductive wood with new fruiting sites. However, pruning is also a repetitive, strenuous, and sometimes dangerous task involving workers standing on ladders on uneven terrain with sharp cutting tools during the winter. Furthermore, although not as time constrained as harvesting, pruning involves more complex decision making and requires experienced, skilled workers who are increasingly difficult to find given the increased uncertainty in the general availability of agricultural labor [2]. As a result of these factors—plus rising production costs—the tree fruit industry is highly motivated to transition to robotic pruning.

For the past several years, Oregon State University and Washington State University have been developing the algorithms, methods, etc. required for robotic pruning, including foreground tree segmentation [3], semantic skeletonization of the tree geometry [4], and learning-based control for accurate cuts [5]. In this article, we present an integrated system (Figure 1) that advances the state of the art in sensing capabilities for pruning. Our system, once driven in front of a target tree, is capable of autonomously detecting pruning targets, moving toward them, and then executing precision cuts. This system represents our most complete end-to-end system to date and is the first such system to be tested on fruit trees. We validated our system by testing it in a commercial sweet cherry tree orchard in Prosser, WA, USA. In the remainder of this article, we describe the integrated system, summarize key findings from field trials, and discuss areas where significant performance improvements are needed to move this technology closer to commercial readiness.

BACKGROUND

Recent advances in deep learning, computer vision, and robotics have led to a proliferation of work on using robots to physically manipulate/interact with specialty crops (e.g., harvesting, thinning, pollinating, etc.). One active area of research closely related to robotic pruning is robotic harvesting, i.e., using manipulators and computer vision systems to autonomously detect and harvest fruits and vegetables. A recent review article [6] discusses 50 different robotic harvesting systems developed up to 2014, covering a wide variety of fresh market fruits and vegetables, such as apples, oranges, and tomatoes. For both harvesting and pruning, the main challenges lie in perception and manipulation. Outdoor agricultural environments are some of the most difficult places for robots to operate in; common factors that negatively affect performance include variable lighting conditions, unstructured environments, and suboptimal terrain that hinders mobility.

Compared to harvesting, the prior work on robotic pruning remains relatively sparse. Recently, however, there has been an increase in interest in the area, covering aspects such as tree modeling for pruning point detection [7], [8], pruning manipulator design [9], [10], path planning [11], and manipulator control [12]. Some examples of research prototype end-to-end systems for fruit-related pruning include [13], Vision Robotics [14], and the Bumblebee system [15], all of which focus on grapevines. There has also been work on automated pruning for landscapes/gardens, such as the Trimbot2020 system [16], which performs rose pruning and bush trimming. However, there are currently no commercially available robotic pruning systems for trees.

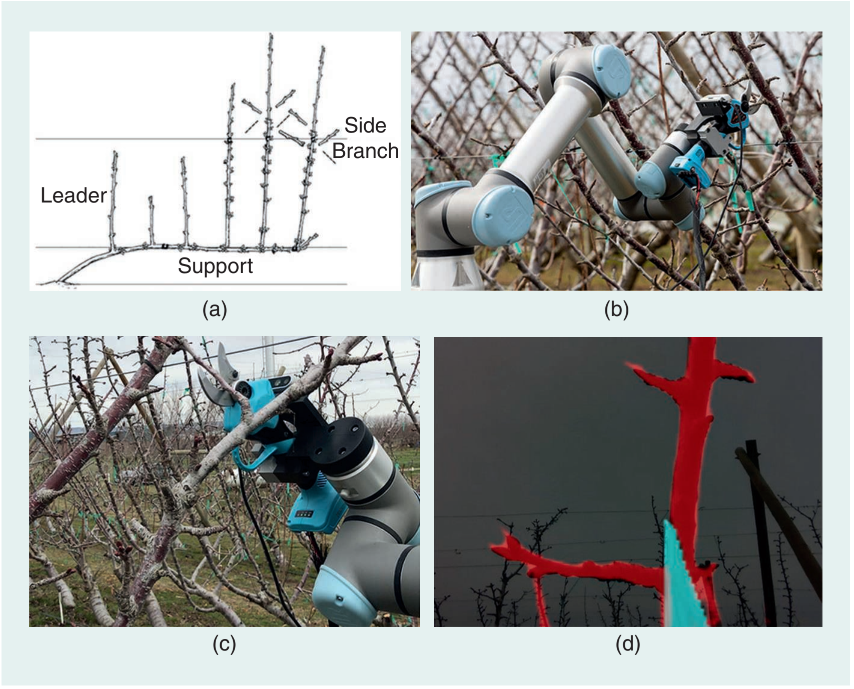


FIGURE 1. (a) A diagram of an upright fruiting offshoot cherry tree structure, including the side branches to be pruned from the vertical leaders. (b) and (c) Our pruning robot uses a manipulator mounted on a mobile base with an eye-in-hand red, green, blue plus depth sensor and electric bypass shears. (d) Segmented image taken during the approach to the pruning point. [Figure 1(b) courtesy of Kate Prengaman/Good Fruit Grower.]

SYSTEM DESIGN

In this work, our focus is on the dormant pruning of sweet cherry trees trained in an upright fruiting offshoot (UFO) [17] configuration (Figure 1). UFO trees are characterized by the presence of multiple long leader branches that grow vertically from a horizontal support branch. The trees in our test orchard are trained in a V-shaped trellis configuration, where each wall is tilted approximately 40°. UFO cherry trees typically produce the most fruit on new spurs and near the base of one-year-old shoots, and so the recommended pruning rule for UFO trees is simply to cut off sufficiently long side branches extending from the leaders with either thinning or stub cuts [18]. For our work, we assume the goal is to execute a stub cut with a stub length of 3 cm.

Our hardware setup is shown in Figure 2. The robot consists of a six-degrees-of-freedom (6-DoF) Universal Robots (Odense, Denmark) UR5e manipulator mounted on an actuated prismatic axis (for a total of 7 DoF). The prismatic axis has a travel range of 1 m and consists of a heavy duty linear slide actuated by a Nema 23 stepper motor with a lead screw transmission. Control of the prismatic axis is via a microcontroller and serial communication. For field trials, the robot is installed on the back of a utility vehicle and powered with a portable generator.

The eye-in-hand pruning end effector integrates battery-operated electric bypass pruners, controllable via serial communication, with an Intel (Santa Clara, CA, USA) RealSense D435 red, green, blue plus depth (RGB-D) camera. The camera is located above the cutters and pitched downward at an angle of 10° so that the top blade is visible when the pruners are open. The shears are rated for cutting branches up to 3.2 cm in diameter (https://salemmaster.com/). Our choice to use a single commercially available camera as the entire vision system is intended to ease system deployment and demonstrate that pruning is possible even without more complex and expensive computer vision setups.

Robot control uses the Universal Robot's Robotic Operating System (ROS) driver running on Ubuntu 18.04. The computer vision algorithms were executed on a separate Dell XPS-15 laptop equipped with an NVIDIA GeForce GTX 1050 Ti graphics card running Windows 10. These algorithms represent the main bottleneck for system execution speed. Vision-based controller commands were computed on the Windows computer and sent via serial communication to the ROS computer, which, in turn, relayed the commands to the robot arm.

"THOUGH NOT FULLY ROBUST AND REQUIRING IMPROVEMENTS IN THROUGHPUT, OUR SYSTEM IS THE FIRST TO OPERATE ON FRUIT TREES AND REPRESENTS A USEFUL BASE PLATFORM TO BE IMPROVED IN THE FUTURE."

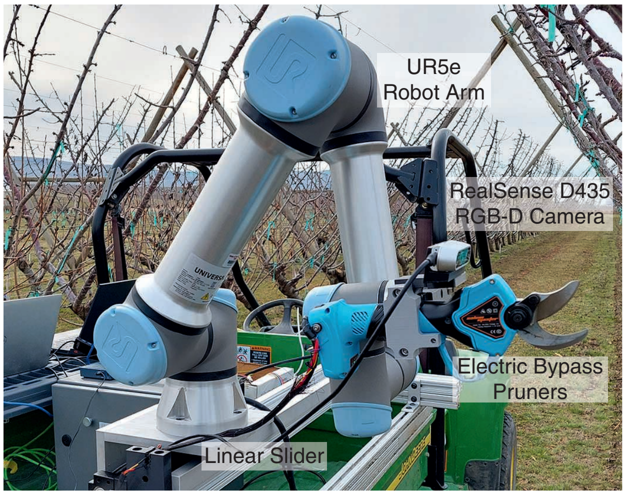


FIGURE 2. Our pruning setup, consisting of a Universal Robots UR5e robot mounted on a linear axis. The end effector consists of a set of electric bypass pruners along with a RealSense D435 red, green, blue plus depth (RGB-D) camera.

PRUNING PROCEDURE

In this section, we discuss the operation of our pruning system and the various essential subcomponents/algorithms. Our approach is motivated by the fact that, due to the manipulator's limited reach and the narrow UFO orchard rows, trees are significantly wider and taller than the field of view for a single camera positioned on the vehicle, making the acquisition of a full 3D scan of the tree impractical. These constraints necessitate a stop-and-go pruning approach in which the vehicle is driven in front of multiple locations in front of a tree to discover prunable branches. At each location, the system executes a "pan-and-scan" routine through a set of 28 fixed waypoints in a zigzag shape [Figure 3(b)] to identify potential pruning points. The lawnmower pattern is designed to avoid large joint movements between waypoints that could lead to collisions with the trellis wall, with the waypoints being spaced to cover the view of the planar tree architecture with minorly overlapping images. Planning between joint waypoints is done using the RRT-Connect algorithm [19], implemented via the Open Motion Planning Library [20] in ROS.

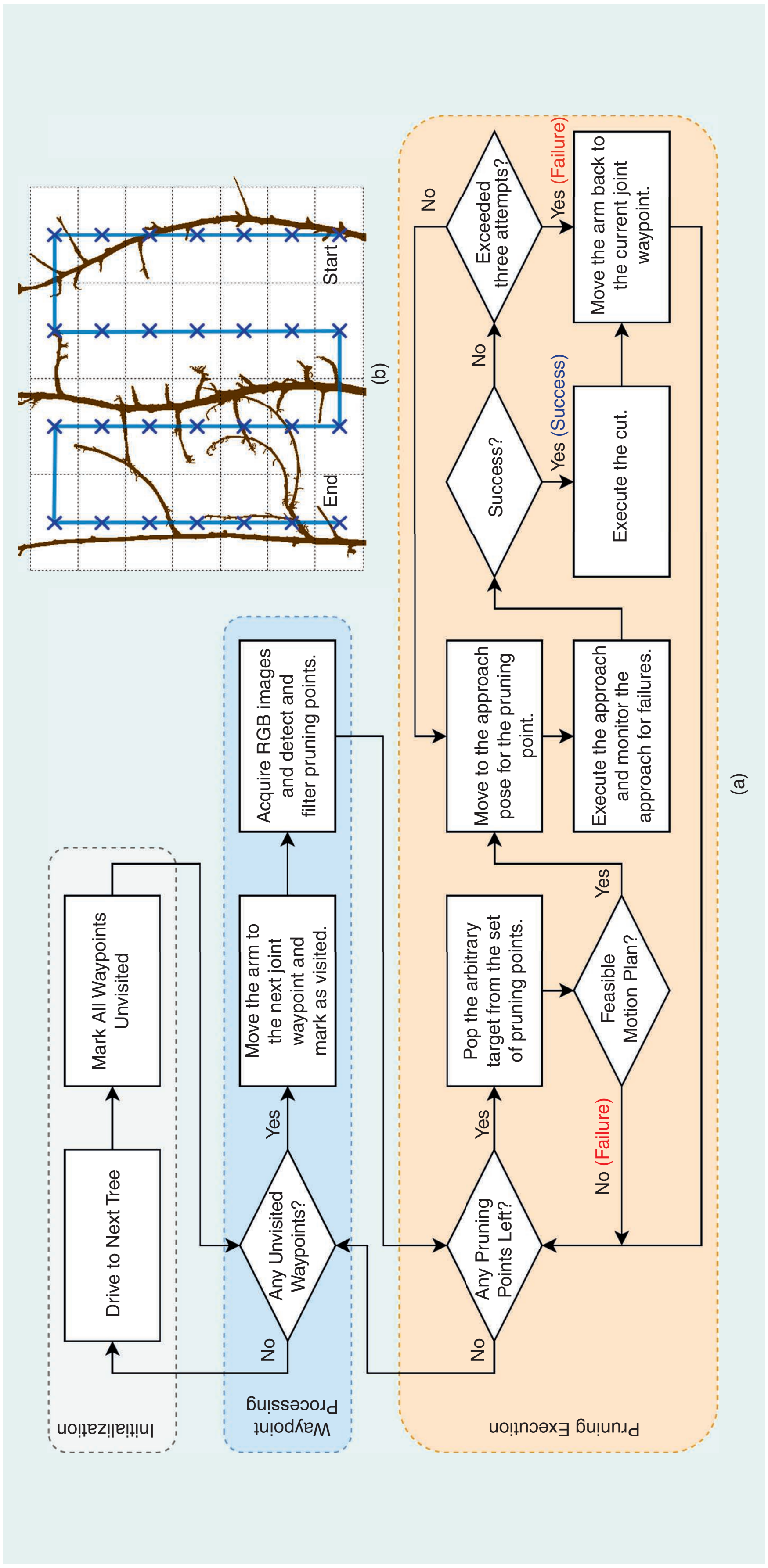
Figure 3 shows the execution flowchart for pruning. At each waypoint, the robot acquires a set of two RGB images that are used to detect potential pruning points in the environment. This pipeline is shown in Figure 4. Once pruning points are identified and converted to 3D position estimates, the robot plans an approach pose in front of the target. Once there, the robot executes a hybrid controller that uses feedback from the RGB camera and the robot's force–torque sensor to guide the cutters to enclose the target branch, at which point the system executes the cut. We include human intervention at this step to prevent unintended cuts to the tree or trellis.

In the following sections, we describe the details for each of the system components. We assume that the vehicle has already been driven to a suitable location for pruning.

BRANCH SEGMENTATION

Fundamentally, all pruning systems must be capable of differentiating branches near the camera from branches beyond the camera and other background elements, i.e., performing foreground branch segmentation. This task is made very difficult by the highly unstructured and repetitive nature of orchard environments; other trees and structural elements will be visible from behind the target tree of interest, making it difficult to differentiate foreground trees from background trees with just a single image. Past systems have used various methods, such as trailers towed by tractors that completely encompass the plant of interest [13], [14], [21] and sophisticated lighting systems for controlling the exposure of the images [22]. However, it is clear that using such setups significantly increases the complexity and unwieldiness of the systems. Other studies have also used depth information from RGB-D cameras, but depth data on these cameras is far from perfect, and networks trained using RGB-D data typically require large amounts of manually labeled data for training.

Our foreground branch segmentation framework, which is the crux of our entire pruning framework, uses a generative adversarial network (GAN) [23] to take in RGB images and output masks corresponding to two scene elements: branches in the foreground and the cutters. Notably, we train this GAN using synthetic data, using a simulated orchard along with heavy texture randomization to allow the GAN to generalize to real trees. Furthermore, the GAN augments the input



Initialization
Drive to Next Tree
Mark All Waypoints Unvisited
Waypoint Processing
Any Unvisited Waypoints?
No
Yes
Move the arm to the next joint waypoint and mark as visited.
Acquire RGB images and detect and filter pruning points.
Pruning Execution
Any Pruning Points Left?
No
Yes
Pop the arbitrary target from the set of pruning points.
Feasible Motion Plan?
No (Failure)
Yes
Move to the approach pose for the pruning point.
Execute the approach and monitor the approach for failures.
Success?
Yes (Success)
No
Execute the cut.
Exceeded three attempts?
Yes (Failure)
No
Move the arm back to the current joint waypoint.
(a)
Start
End
(b)

FIGURE 3. (a) The execution flowchart of the pruning system. At each vehicle location, the system initializes to the starting state. It then alternates between two steps: moving to the next waypoint and detecting pruning points, followed by operating on each of the detected pruning points. (b) The 28-waypoint scanning procedure used to locate pruning points.

image with colorized optical flow data, i.e., the measured pixel movement between two RGB images. Optical flow is a useful proxy for depth since, due to parallax, translational camera movement will result in more pixel movement for objects closer to the foreground. This optical flow image is computed by taking two adjacent RGB images and feeding them through the FlowNet2 deep neural network [24]. More details can be found in [3].

Our motivation for training our segmentation network this way was to completely avoid the use of manually labeled data as well as to not require a specialized system beyond an off-the-shelf consumer-grade camera. Optical flow is naturally obtained as a camera mounted on an end effector moves around, and FlowNet2 is able to produce optical flow estimates that work on both simulated and real images, allowing for the sim-to-real transfer.

“IN THIS WORK, WE MAKE THE ASSUMPTION THAT PRUNING POINTS CAN BE IDENTIFIED LOCALLY AS THE ROBOT OPERATES.”

PRUNING POINT ESTIMATION

Once the environment is perceived, the next step in the pruning process is to detect prunable branches and obtain 3D estimates of their positions. Until now, most research on automated pruning point detection has focused on grapevines, using stereo vision systems and skeletonization algorithms to form a 3D model and using the model to make decisions on which branches to cut and where. Silwal et al. [15] utilize a spur pruning method that counts the buds on each cane and leaves a fixed number of buds on each cane, while Botterill et al. [13] implement a cane pruning algorithm that considers feature vectors of the detected canes to decide which ones to prune. In our previous work, we developed a skeletonization algorithm [4] that used heuristic knowledge about the UFO tree architecture to reconstruct a global model of the tree from a dense 3D point cloud and identify side branches. However, for our particular application, global knowledge of the tree structure is not necessary, as prunable side branches are detectable from a close-up view of portions of the tree. Therefore, in this work, we make the assumption that pruning points can be identified locally as the robot operates.

The pruning point detection process (Figure 4) first begins with a one-channel foreground branch mask (see the “Branch Segmentation” section). The branch mask is then fed through the Mask R-CNN [25] instance segmentation network, provided through the Detectron2 library [26]. Mask R-CNN takes in an image and outputs a set of bounding boxes and binary masks for each individually detected item in the scene. Using the foreground branch mask as the input to Mask R-CNN rather than the RGB image enables generalization across multiple environments.

Ideally, the training of Mask R-CNN would be done with simulated data, but we did not have sufficiently accurate digital models of the trees at the time. As such, to train Mask R-CNN, we manually labeled 371 images, which were split 80%/10%/10% among training/validation/testing. We labeled five classes of foreground objects: leaders, side branches, spurs, an “other” branch category used for tree branches that either extended off the side of the image or did not belong in any of the aforementioned categories, and nonbranch objects (primarily wires, wooden posts, and ribbons). An example of a labeled image is shown in Figure 5.

Once the instance segmentation on the mask is complete, the next step is to find all pixels corresponding to prunable locations where the side branches meet the leaders. To do this, the system disregards all detections except those corresponding to leaders and side branches. The goal is to match each

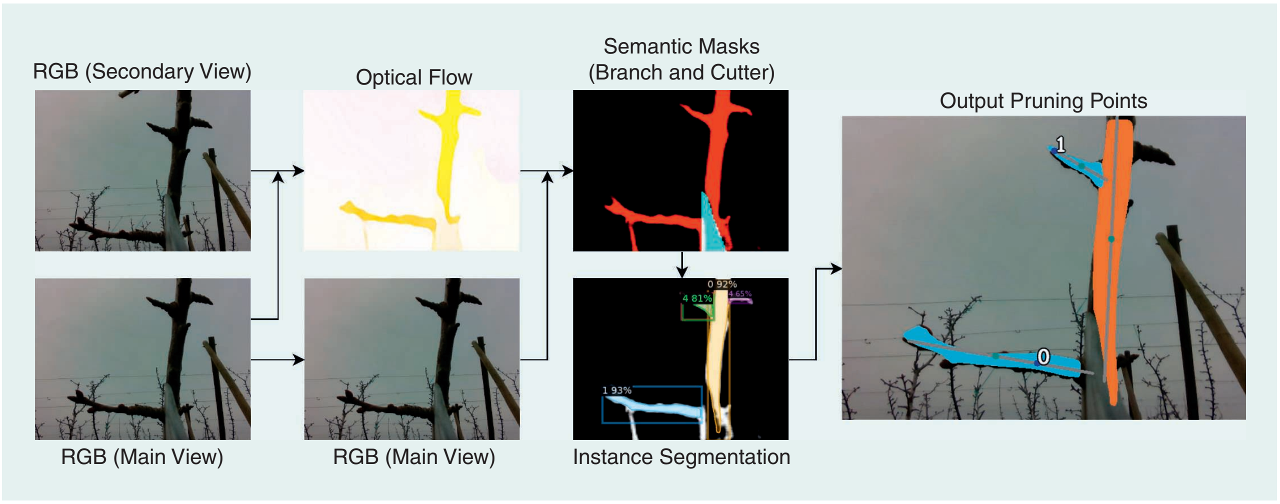


FIGURE 4. The pipeline for the segmentation system used to detect pruning points. First, the system obtains semantic masks of the scene (the branches are red, and the cutter is cyan) by utilizing optical flow data computed from two RGB images. It then feeds the branch mask into a Mask R-CNN network to identify visible branches. Each instance is then processed to obtain pixel estimates for pruning points (shown as points 0 and 1, with 1 being a false positive).

side branch detection to a leader (if possible) and determine the location of intersection. The method is illustrated in Figure 6. First, all of the detected masks are decomposed using principal component analysis to yield a primary orientation and center. Each leader branch is assigned a width w equal to the average number of pixels in the horizontal direction along the mask. Each side branch–leader pair is checked to see if their point of intersection p^* lies inside of the leader and if any part of the side branch mask is sufficiently close to the leader boundary (at width w). If so, letting $\vec{b}$ represent the unit vector for the side branch (facing away from the trunk), the pixel pruning point is computed as $p^* + \left(\frac{w}{2} + m\right)\vec{b}$, where $m = 90$ is a manually specified pixel offset from the join point between the leader and the side branch. This process is repeated for every side branch–leader pair.

Finally, each 2D pixel estimate must be converted into a 3D location to move the robot. To convert the pixel estimate into a 3D location, we make an assumption that the pixel is located on a plane 30 cm away from the camera's optical frame in the z-direction. The system uses the intrinsics of the camera to yield a 3D estimate of the point in the camera's optical frame, which is then transformed using the kinematic model of the arm to obtain a world frame estimate for the pruning point. This estimate of the pruning point in the world is not particularly precise since the 30-cm planar estimates for the pruning points will not always be true, and the 90-pixel offset is chosen arbitrarily. However, the "Closed-Loop Approach (Waypoint to Cutpoint)" section discusses the approach phase controller that corrects for this error.

Once the system identifies the pruning points, it must choose an order in which to cut them. We have explored task sequencing for pruning in our previous work [27]. However, for this pruning setup, the close-up view of the scene means there are rarely more than two eligible targets to prune. Therefore, we choose to sequence the points arbitrarily.

CLOSED-LOOP APPROACH (WAYPOINT TO CUTPOINT)

Once a pruning point is detected and its position determined, the final step of the pruning process is to move the pruning implement to the target point and execute a cut. If this pruning point estimate is perfectly accurate, then the system simply needs to plan a path that does not collide with any obstacles to move the cutters and cut. This open-loop method of planning, in which the system does not use sensor feedback as it moves the manipulator toward the goal, has been the dominant approach for previous pruning systems. However, many external factors (e.g., vehicle slippage, sensor noise, and wind) can affect the accuracy of the original estimates. Furthermore, such open-loop approaches have no way of regulating the dynamic interaction of the manipulator with the environment, which can lead to damage to the environment or to the robot. These issues motivated the development of a hybrid controller for the approach phase that uses both visual and force feedback for accurate cutting, the first of its kind to be used in an end-to-end pruning system.

The details of the hybrid controller are explained in our previous work [5]. The visual controller is a deep neural network that takes in the segmented version of the scene, as described in the "Branch Segmentation" section, i.e., a two-channel image corresponding to masks for the foreground branches and the cutter as well as the colorized optical flow image to yield a five-channel input. It outputs a control action $(v_x, v_y) \in [-1,1] \times [-1,1]$. Given a forward velocity of $s = 0.03$, the end effector is commanded to move at a Cartesian velocity of $[sv_x, sv_y, s]$ in the end effector's frame. We chose to run the visual controller at one frame per second to obtain reasonable optical flow

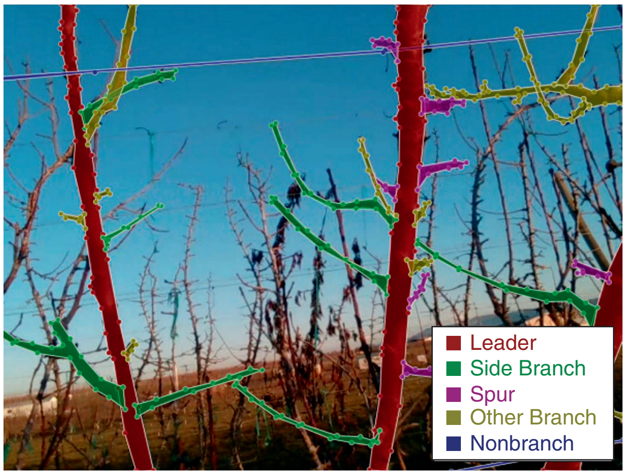


FIGURE 5. An example of a labeled image, showing five labeled classes of foreground objects: leaders, side branches, small spurs, "other" branches (usually offscreen branches that do not definitively fit in one of the former categories), and nonbranch objects.

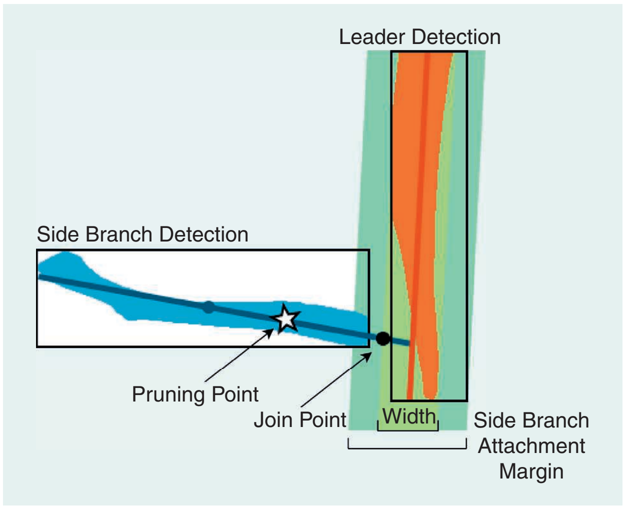


FIGURE 6. The process of matching a detected side branch with a leader. Each detection is represented as a line segment. If a side branch intersects the leader line segment and the mask is sufficiently close to the leader, the pruning point is computed a fixed offset away from the estimated join point between the side branch and leader.

estimates in between successive frames before updating the control velocity.

This controller was trained in a simulated PyBullet [28] environment by formulating a pruning episode as a Markov decision process and training it using the proximal policy optimization reinforcement learning (RL) algorithm [29]. In each pruning episode, the agent was presented with a view of a prunable branch and allowed to steer the cutter toward the target. The agent would receive a large reward for successfully enclosing the branch in the cutter mouth and a large penalty otherwise. In the intermediate time steps, the agent was also rewarded for keeping the target cutpoint in view of the camera. This was done by defining a "hit point" on the blade and computing its image-based distance to the episode target, as shown in Figure 7. So long as the robot can keep the cutter hit point aligned with the target, the branch should contact the blade of the cutter, ultimately leading to an episode success. By utilizing the segmentation network as a means of normalizing synthetic and real images, the controller is able to transfer to a real robot with no adjustments necessary.

Once a force exceeding 1.5 N is detected by the force–torque sensor on the robot's wrist, the system switches over to an admittance controller that regulates the control velocities to guide the branch inside the cutters while minimizing forces on the environment. Once the admittance controller terminates, we execute the cut using the electric cutters.

SYSTEM EVALUATION

We evaluated our system at a Washington State University UFO sweet cherry research orchard in Prosser, WA, USA, on an overcast day on 17 March 2022. To conduct our experiments, we identified 10 locations in the orchard that had a suitable number of eligible branches for pruning; i.e., there had to be at least two leaders located within the scanning range of the linear axis, each with at least one candidate branch for pruning. To avoid collisions, we manually pre-pruned any branches sticking out from the trellis wall. At each location, a human operator reset the linear axis to the zero position and positioned the vehicle so that the end effector was placed at an appropriate distance from the trellis wall (about 20–30 cm), and the first leader was centered in the view of the camera. Once the vehicle was positioned, we executed our 28-waypoint scanning procedure. We evaluated the system by assessing its accuracy in detecting and successfully cutting detected branches as well as examining the length of the stubs remaining after the cuts and the overall runtime of the system.

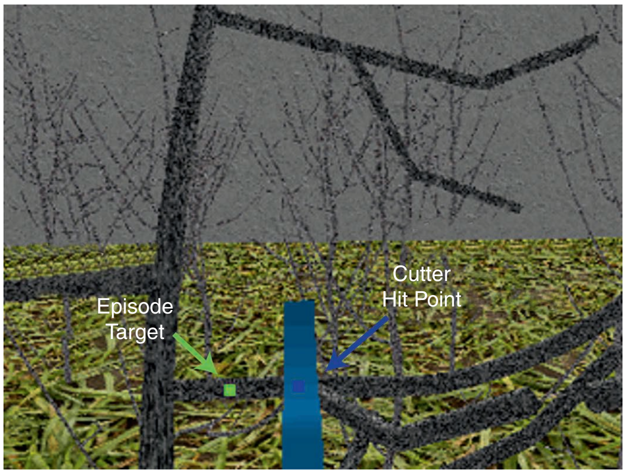


FIGURE 7. An example of an image from the simulated training environment. The agent receives an intermediate reward for bringing a predefined point on the cutter close to the episode target in the image space. (These points are not known to the agent during training.) If the agent loses sight of the episode target, the episode counts as a failure.

Ideally, the system would operate fully autonomously, as described in Figure 3. However, for this field trial, we chose to manually intervene during several steps:

- For safety, we retained manual control over executing the final cut to avoid cutting trellis wires. We also manually drove the vehicle to each cutting location to avoid collisions between the arm and orchard.
- Various components were made semiautonomous because of the difficulties associated with testing the system only in the lab prior to conducting the field trials. First, we manually filtered out false positive pruning point detections before executing the cutting process to avoid pruning fruiting spurs or nonexistent targets. (No corrections were made for false negative detections.) Second, instead of using the cutter mask that comes out of the segmentation framework, we manually created a mask of the pruners for the approach phase since the view of the cutter in the image never changes. Finally, we monitored the approach phase manually and terminated the process if the cutters missed the target branch, rewinding the robot back to the start of the approach. For each pruning target, we allowed three attempts to successfully cut the target before moving on to the next target.

RESULTS AND DISCUSSION

DETECTION AND CUTTING ACCURACY

In total, the system detected 38 branches that were long enough to cut. Out of these 38 branches, the robot was ultimately able to cut 22 of them, representing a 58% cutting success rate. The breakdown of the causes of failure is as follows:

- Six failures were due to a motion planning failure to the approach position. Motion plans with a total joint displacement above a given threshold (in our case, π radians) were also counted as failures, as they likely indicated poor-quality motion plans.
- Of the remaining 32 branches, 10 of the attempts failed due to exhausting the three-attempt limit to reach the target, representing a 69% success rate in ultimately reaching the branch.

- Of the remaining 22 branches, the average number of attempts it took to reach the target was 1.4.

For the 22 successfully cut branches, the length of the remaining stub had a mean of 3.8 cm, with a standard deviation of 1.7 cm and ranging between 1 cm and 7 cm. Our target stub length was 3 cm, demonstrating that our use of visual feedback led to accurate cuts. However, we recognize that the precision of the controller can still be improved. Given that the internodal distance on sweet cherry trees is roughly 2.5 cm, we would like to target a precision of 1 cm in the future.

Various factors accounted for the robot missing the target. In some situations, the initial estimate obtained using an assumption of a distance of 30 cm was sufficiently inaccurate that the cutting target would be barely visible or out of sight from the approach position. Other than that, most misses were caused by the end effector passing underneath the target branch; we observed that our RL-trained controller sometimes failed to move the end effector up at critical moments.

Regarding the accuracy of the branch detection, as previously noted, there were 38 true positive detections of pruning points. By far, the biggest issue with the system was the presence of false positives. In total, the system made 115 false positive detections; 96 of these were simply spurs being detected as side branches. We attribute this to our views in the training data being farther away from the trellis wall than during our trials; since the camera was closer up during the trials, the spurs appeared to be of sufficient length in the images to be classified as side branches. In practice, this issue could be addressed by using depth data to measure the length of each detection and using a length-based cutoff to filter out detected spurs. Ten of the false positives were due to horizontal trellis wires being detected as side branches, indicating a need to more robustly model the environment. The remaining nine false positives were mainly due to false detections of nonexistent leaders leading to a spurious intersection.

There were also 27 instances of false negatives in which an intersection of a side branch and a leader was not detected. Two of them were due to a leader not being detected. For all other 25 instances, though, both the leader and the side branch were properly detected, and so the lack of intersection was due to implementation issues in our intersection algorithm, which we will address for subsequent trials.

“IN ADDITION TO IMPROVING THE PERFORMANCE OF THE INDIVIDUAL SUBCOMPONENTS OF OUR SYSTEM, ONE MAJOR SYSTEM ASPECT WE WOULD LIKE TO INVESTIGATE IN THE FUTURE IS THE DESIGN OF THE MANIPULATOR.”

RUNTIMES

Figure 8 shows the breakdown of the system's cycle time. Since the number of cuts executed is variable across experiments, we choose to analyze the runtimes of the scanning process and of a single cutting attempt separately.

Overall, an entire scan of a region spanning 0.6 m horizontally and 0.6 m vertically took 284 s. The majority of the time (115 s) was actually spent moving the camera at each waypoint (to obtain alternate views). This was because we had to move the robot arm slowly to prevent the arm from vibrating when the arm reached its goal. (Vibration often led to poor optical flow estimates.) However, there was also some redundant movement because the robot would first move to the scan waypoint, then to the offset view, and then back to the scan waypoint (rather than continuing to the next scan waypoint). Eliminating this redundancy could cut down on the time spent moving the camera by half.

The other notably slow part of the scanning procedure was moving the linear axis, which took about 23 s to move 20 cm, resulting in an additional 69 s to the scanning procedure. The slow movement speed was chosen partially due to safety concerns, but the speed is also limited somewhat by the screw-based design of the system. Future iterations of our work will focus on increasing the design's travel speed. The remaining sources of time are from moving the arm between waypoints (50 s) and running the detection algorithms (49 s), which are not notably inefficient, though the speed of FlowNet2 and Mask R-CNN could be improved by using a more modern graphics card.

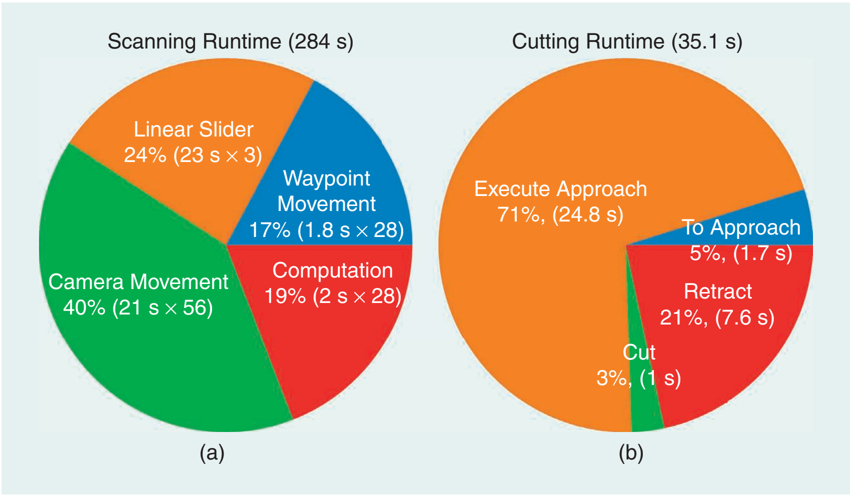


FIGURE 8. (a) The average time spent in the different stages of the scanning procedure, not including operations on detected pruning points since the number of cuts and cutting attempts varied per tree (an average total of 284 s). (b) The average time spent at each stage for a single cutting attempt assuming the approach succeeds on the first try (an average total of 35.1 s).

For the cutting process, assuming success on the first try, an average cut took 35.1 s, consisting of 1.7 s to move the arm to the approach position, 24.8 s to execute the approach, 1 s to execute the cut, and 7.6 s to retract the arm back to the starting position. The main constraint to the approach was the 3-cm/s limit on the forward velocity of the end effector, which we imposed for safety and monitoring purposes. One main contributor to the approach time was waiting for the admittance controller to terminate, as, in some instances, the desired balance of forces never occurred, leading us to wait for the 15-s timeout on that part of the controller. Increasing the velocity of the approach would also be possible. However, moving the arm faster can lead to noisy force–torque estimates that may accidentally trigger the transition to the admittance controller, which we occasionally observed even with the 3-cm/s cutoff. Additional work is required for developing a more robust criterion for switching to and terminating the admittance controller.

DISCUSSION

Overall, though the system was not fully autonomous, and the current performance is short of commercial readiness, we demonstrated that the system is capable of detecting and executing cuts in a manner that can be made fully autonomous through improvements to the individual subcomponents. In this section, we briefly discuss the improvements necessary to achieve fully automated pruning (excluding autonomous mobility of the vehicle):

- *Reducing false positives*: As noted, one notable shortcoming of the system was the high rate of false positives when detecting prunable branches, especially with spurs being detected as branches. We believe that a major reason for this was because the Mask R-CNN network was trained using image data that was farther away from the tree than during the pruning trials. Since spurs appeared larger in the image for our trials, this led to them being misidentified as side branches. While more robust training of Mask R-CNN would certainly help, we should also aim to integrate phenotype measurements into the detection system, i.e., estimating the length of each detected branch using depth data. This would allow us to set thresholds for cutting viability grounded in horticultural knowledge of the system.
- *Failure detection*: Proper failure detection is also another critical component for our system to be fully autonomous. While improved accuracy of the closed-loop controller is desirable, we cannot expect any system to ever attain 100% accuracy. In particular, our system needs to be able to check if the branch is inside of the cutters after detecting an impact, which could be done via a sensor or by executing a wiggling motion. The system should also be able to quickly detect situations in which it has missed the target branch entirely, such as when the target branch has fallen out of sight of the camera. This could be done through initializing an estimate of the target in the image and tracking it as the camera moves. Finally, the system must strictly avoid the possibility of cutting a trellis wire, which is a difficult task given how thin the wires are in the image. A conservative approach would be to model the wires in the environment using our semantic segmentation network and reject any pruning candidates whose bases come within a certain threshold of a wire.
- *Increasing the action space*: To simplify the training of the approach controller, we assumed that a planar view of the orchard would be sufficient for maneuvering the manipulator, limiting the controller to 2 DoF. However, in reality, this is not sufficient due to the presence of branches that grow outward from the orchard wall that our system is currently unable to cut. We are working on an updated RL-based pruning controller that will utilize all 6 DoF of the manipulator to be able to cut branches extending from the planar canopy wall.

“ THOUGH THE ACCURACY AND CYCLE TIME OF THE SYSTEM CAN STILL BE IMPROVED, OUR FIELD TRIALS IN A REALISTIC OUTDOOR ENVIRONMENT DEMONSTRATE THAT OUR PIPELINE IS A VIABLE ONE FOR USE IN A REAL ORCHARD. ”

CONCLUSIONS

In this article, we present an end-to-end system that can semiautonomously prune sweet cherry trees in a modern planar architecture. First, we use a novel segmentation method to reliably extract the foreground branches of the scene, avoiding the need to control the lighting of the environment. Using the foreground masks, we then use a Mask R-CNN network to detect pruning points. Finally, after projecting the detected pixel locations into 3D space and moving the robot to an approach position in front of the target, we utilize a closed-loop hybrid vision/interaction controller to accurately guide the cutters to the branch, allowing us to compensate for many sources of error in the original point estimate while regulating interaction with the environment. Though the accuracy and cycle time of the system can still be improved, our field trials in a realistic outdoor environment demonstrate that our pipeline is a viable one for use in a real orchard.

In addition to improving the performance of the individual subcomponents of our system, one major system aspect we would like to investigate in the future is the design of the manipulator. We observed that the UR5e's revolute joints were not suitable for use in a compact orchard space. If the robot was too close to the trellis wall, it had a difficult time avoiding collisions with the wall, while, if it was too far away, the arm would fail to reach the target points. Our goal will be to explore other manipulator designs with kinematics that are better posed for modern orchard systems.

ACKNOWLEDGMENT

This research is supported in part by U.S. Department of Agriculture National Institute of Food and Agriculture (USDA-NIFA) through the Agriculture and Food Research Initiative, Agricultural Engineering Program (Award 2020-67021-31958), and the Artificial Intelligence (AI) Research Institutes program supported by National Science Foundation and USDA-NIFA under the AI Institute: Agricultural AI for Transforming Workforce and Decision Support (Award 2021-67021-35344).

AUTHORS

Alexander You, Collaborative Robotics and Intelligent Systems Institute, Oregon State University, Corvallis, OR 97331 USA. E-mail: youa@oregonstate.edu.

Nidhi Parayil, Collaborative Robotics and Intelligent Systems Institute, Oregon State University, Corvallis, OR 97331 USA. E-mail: parayiln@oregonstate.edu.

Josyula Gopala Krishna, Collaborative Robotics and Intelligent Systems Institute, Oregon State University, Corvallis, OR 97331 USA. E-mail: josyulag@oregonstate.edu.

Uddhav Bhattarai, Center for Precision and Automated Agricultural Systems, Washington State University, Prosser, WA 99350 USA. E-mail: uddhav.bhattarai@wsu.edu.

Ranjan Sapkota, Center for Precision and Automated Agricultural Systems, Washington State University, Prosser, WA 99350 USA. E-mail: ranjan.sapkota@wsu.edu.

Dawood Ahmed, Center for Precision and Automated Agricultural Systems, Washington State University, Prosser, WA 99350 USA. E-mail: dawood.ahmed@wsu.edu.

Matthew Whiting, Center for Precision and Automated Agricultural Systems, Washington State University, Prosser, WA 99350 USA. E-mail: mdwhiting@wsu.edu.

Manoj Karkee, Center for Precision and Automated Agricultural Systems, Washington State University, Prosser, WA 99350 USA. E-mail: manoj.karkee@wsu.edu.

Cindy M. Grimm, Collaborative Robotics and Intelligent Systems Institute, Oregon State University, Corvallis, OR 97331 USA. E-mail: cindy.grimm@oregonstate.edu.

Joseph R. Davidson, Collaborative Robotics and Intelligent Systems Institute, Oregon State University, Corvallis, OR 97331 USA. E-mail: joseph.davidson@oregonstate.edu.

REFERENCES

[1] R. Verbiest, K. Ruysen, T. Vanwalleghem, E. Demeester, and K. Kellens, “Automation and robotics in the cultivation of pome fruit: Where do we stand today?” *J. Field Robot.*, vol. 38, no. 4, pp. 513–531, Jun. 2021, doi: 10.1002/rob.22000.

[2] L. Calvin and P. Martin, “The U.S. produce industry and labor: Facing the future in a global economy,” U.S. Dept. Agriculture, Econ. Res. Service, Washington, DC, USA, Econ. Res. Rep. No. ERR-106, Nov. 2010.

[3] A. You, C. Grimm, and J. R. Davidson, “Optical flow-based branch segmentation for complex orchard environments,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2022, pp. 9180–9186, doi: 10.1109/IROS47612.2022.9982017.

[4] A. You, C. Grimm, A. Silwal, and J. R. Davidson, “Semantics-guided skeletonization of upright fruiting offshoot trees for robotic pruning,” *Comput. Electron. Agriculture*, vol. 192, Jan. 2022, Art. no. 106622, doi: 10.1016/j.compag.2021.106622.

[5] A. You, H. Kolano, N. Parayil, C. Grimm, and J. R. Davidson, “Precision fruit tree pruning using a learned hybrid vision/interaction controller,” in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2022, pp. 2280–2286, doi: 10.1109/ICRA46639.2022.9811628.

[6] 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,” *J. Field Robot.*, vol. 31, no. 6, pp. 888–911, Jul. 2014, doi: 10.1002/rob.21525.

[7] B. Ma, J. Du, L. Wang, H. Jiang, and M. Zhou, “Automatic branch detection of jujube trees based on 3D reconstruction for dormant pruning using the deep learning-based method,” *Comput. Electron. Agriculture*, vol. 190, Nov. 2021, Art. no. 106484, doi: 10.1016/j.compag.2021.106484.

[8] M. Fernandes et al., “Grapevine winter pruning automation: On potential pruning points detection through 2D plant modeling using grapevine segmentation,” in *Proc. IEEE 11th Annu. Int. Conf. CYBER Technol. Automat., Contr., Intell. Syst. (CYBER)*, 2021, pp. 13–18, doi: 10.1109/CYBER53097.2021.9588303.

[9] A. Zahid, M. S. Mahmud, L. He, D. Choi, P. Heinemann, and J. Schupp, “Development of an integrated 3R end-effector with a cartesian manipulator for pruning apple trees,” *Comput. Electron. Agriculture*, vol. 179, Dec. 2020, Art. no. 105837, doi: 10.1016/j.compag.2020.105837.

[10] B. Zhang, X. Chen, H. Zhang, C. Shen, and W. Fu, “Design and performance test of a Jujube pruning manipulator,” *Agriculture*, vol. 12, no. 4, Apr. 2022, Art. no. 552, doi: 10.3390/agriculture12040552.

[11] A. Zahid, L. He, D. D. Choi, J. Schupp, and P. Heinemann, “Collision free path planning of a robotic manipulator for pruning apple trees,” in *Proc. ASABE Annu. Int. Virtual Meeting*, St. Joseph, MI, USA: American Society of Agricultural and Biological Engineers, 2020, Art. no. 2000439.

[12] F. Yandun et al., “Reaching pruning locations in a vine using a deep reinforcement learning policy,” in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2021, pp. 2400–2406, doi: 10.1109/ICRA48506.2021.9562075.

[13] T. Botterill et al., “A robot system for pruning grape vines,” *J. Field Robot.*, vol. 34, no. 6, pp. 1100–1122, Sep. 2017, doi: 10.1002/rob.21680.

[14] “Vision Robotics Corporation provides solutions and expertise for robotic systems and mechanization,” Vision Robotics Corp., San Diego, CA, USA, 2017. [Online]. Available: https://www.visionrobotics.com/

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

[16] N. Strisciuglio et al., “TrimBot2020: An outdoor robot for automatic gardening,” in *Proc. 50th Int. Symp. Robot.* (ISR), Frankfurt, Germany: VDE, 2018, pp. 1–6.

[17] M. Whiting, “Precision orchard systems,” in *Proc. Automat. Tree Fruit Production, Princ. Pract.*, 2018, pp. 75–93.

[18] L. E. Long, G. A. Lang, M. D. Whiting, and S. Musacchi, *Cherry Training Systems*. Corvallis, OR, USA: Oregon State Univ., 2015, pp. 50–56. [Online]. Available: https://hdl.handle.net/2376/6087

[19] J. J. Kuffner and S. M. LaValle, “RRT-connect: An efficient approach to single-query path planning,” in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2000, vol. 2, pp. 995–1001, doi: 10.1109/ROBOT.2000.844730.

[20] I. A. Sucan, M. Moll, and L. E. Kavraki, “The open motion planning library,” *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012, doi: 10.1109/MRA.2012.2205651.

[21] A. Gongal, A. Silwal, S. Amatya, M. Karkee, Q. Zhang, and K. Lewis, “Apple crop-load estimation with over-the-row machine vision system,” *Comput. Electron. Agriculture*, vol. 120, pp. 26–35, Jan. 2016, doi: 10.1016/j.compag.2015.10.022.

[22] A. Silwal, T. Parhar, F. Yandun, H. Baweja, and G. Kantor, “A robust illumination-invariant camera system for agricultural applications,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2021, pp. 3292–3298, doi: 10.1109/IROS51168.2021.9636542.

[23] I. Goodfellow et al., “Generative adversarial nets,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, vol. 27, pp. 2672–2680.

[24] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “FlowNet 2.0: Evolution of optical flow estimation with deep networks,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 2462–2470.

[25] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in *Proc. IEEE Int. Conf. Comput. Vision*, 2017, pp. 2980–2988, doi: 10.1109/ICCV.2017.322.

[26] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. “Detectron2.” GitHub. Accessed: Mar. 10, 2022. [Online]. Available: https://github.com/facebookresearch/detectron2

[27] A. You, F. Sukkar, R. Fitch, M. Karkee, and J. R. Davidson, “An efficient planning and control framework for pruning fruit trees,” in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2020, pp. 3930–3936, doi: 10.1109/ICRA40945.2020.9197551.

[28] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” Pybullet, 2016. [Online]. Available: https://pybullet.org

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