Abstract—Robotic tree pruning requires highly precise manipulator control in order to accurately align a cutting implement with the desired pruning point at the correct angle. Simultaneously, the robot must avoid applying excessive force to rigid parts of the environment such as trees, support posts, and wires. In this paper, we propose a hybrid control system that uses a learned vision-based controller to initially align the cutter with the desired pruning point, taking in images of the environment and outputting control actions. This controller is trained entirely in simulation, but transfers easily to real trees via a neural network which transforms raw images into a simplified, segmented representation. Once contact is established, the system hands over control to an interaction controller that guides the cutter pivot point to the branch while minimizing interaction forces. With this simple, yet novel, approach we demonstrate an improvement of over 30 percentage points in accuracy over a baseline controller that uses camera depth data.

I. INTRODUCTION

Pruning in the dormant season is an important component of producing high-value, fresh market tree fruit. Pruning involves selectively removing branches that are closely spaced, too old/long, diseased, or otherwise unproductive. This process rejuvenates the tree, helps maintain tree health, and provides high yields of quality fruit at harvest. However, selective pruning is a costly and labor-intensive process, comprising the second highest source of labor costs (after harvesting) for fruits such as apples [1], [2]. It is also a dangerous task, with workers standing on ladders on uneven terrain with sharp shears. Moreover, the industry at large is facing uncertainty over labor availability in the future [3]. As such, the tree fruit industry is interested in using robots to automate precision pruning operations [4].

Automated pruning requires i) sensing and modeling a tree in an outdoor environment; ii) autonomously determining the pruning points (i.e. which branches to remove); iii) efficiently planning a collision-free sequence of cuts; and (iv) executing those precise cuts. The orchard environment is highly unstructured and cluttered, with branches of varying compliance interspersed with rigid features such as trunks, posts, and trellis wires. Navigating pruners — here we assume that the cutting tool is a set of bypass pruners — to the desired cut points requires robust collision avoidance, managing the dynamic interaction between the robot and branch, and the ability to locally adapt if the tree shifts or moves. A robotic pruning system must also be robust to missed features in the model (e.g., small branches or wires that are difficult to detect from a distance) and to errors arising from sensor noise and imperfect kinematic models. Furthermore, outdoor robots operate on rough, unstable terrain which can exacerbate errors. All of these errors can lead to scenarios similar to ones we experienced during our previous work [5] where our position-based controller, which relied on depth data with no interaction control regulating contact, nearly snapped a post off of our experimental setup.

To address these problems, in this paper we propose a hybrid controller (Figure 1) which splits the pruning task into two stages. First, we utilize a learned visual controller which eschews the need for accurate point clouds and kinematics by relying solely on a 2D image to guide the cutters to a desired target, taking into account the correct length of the pruning cut (i.e. the robot cuts the branch where the branch should be pruned). This controller handles alignment of the branch and cutter in the vertical and horizontal directions of the image plane (perpendicular to the cutter). Since a visual controller alone cannot handle delicate environmental interactions, once we detect contact with the tree, we transition to an admittance controller incorporating force feedback. This controller has two main tasks: limiting the forces between the robot and the environment, and accurately guiding the branch towards the pivot point of the cutter for an optimal, precise cut.

We evaluate our system on a test setup using branches obtained from a cherry tree fixed to a rigid trellis. We...
compare our hybrid controller against a set of position-based controllers which rely directly on a point estimate produced from a depth camera point cloud. We demonstrate an improvement in cutting accuracy of over 30 percentage points compared to the best position-based controller. Furthermore, our system is able to more consistently cut branches at a pre-specified length, cutting the standard deviation of such cut lengths nearly in half compared to the baseline controllers.

II. RELATED WORK

A comprehensive review of robotic pruning is given in [6], focusing on subproblems such as branch sensing, pruning point detection, end-effector design, and path planning. While a fairly substantial amount of work has gone into these subproblems, research on controllers that bring a physical end-effector to pruning points is sparse. To our knowledge, the only work using a physical manipulator to reach pruning points was our own [5] and those from [7] and [8] (grapevines). Zahid et al. [9] presented a pipeline for collision-free planning for apple tree pruning, but the work was evaluated only in simulation. Because of the lack of work on physical pruning systems, we focus on hybrid force controllers and reinforcement learning schemes for robotic manipulation in general.

Hybrid force-control systems have been developed for a variety of manipulation problems where both visual and force feedback are required, such as door opening [10], [11], peg-in-the-hole assembly problems [12], and surface-based point/contour tracking applications [13], [14]. A common control method is to partition the degrees of freedom of control between the visual and force components. This method is similar to our own in which the vision controller handles vertical-horizontal alignment while the interaction controller handles vertical-forward alignment (though we choose to run our two controllers separately). However, these methods were often limited by the computer vision techniques employed in the visual controllers, which typically assumed highly predefined environments with known markers or models for easy target detection and pose estimation.

Modern deep learning-based, image-based approaches to grasping and manipulation have shown success in generalizing across large object sets, allowing for more robust manipulation. A comprehensive review of deep learning methodologies and applications for robotic manipulation can be found in [15]. While such algorithms have obtained success in moving the object to be manipulated inside of the end-effector, many of the vision-only controllers assume that compliance in the manipulator will prevent the end-effector from being damaged by the interaction. In contrast, pruners are very rigid, sharp objects interacting in an environment with both delicate, compliant objects (branches, buds, leaves) and stiff objects (wires, posts, large branches) which can damage the cutter. This motivates the need to utilize force feedback once contact is made instead of continuing with a vision-only approach.

III. SYSTEM DESIGN

For our tree pruning system design, we make the assumption that we have a 3D tree model with predetermined pruning points, and that these pruning points, when converted to the robot’s base frame, are roughly aligned with the corresponding points on the real tree. Furthermore, we assume that there are no notable obstructions in front of any targets that would completely obstruct a cutting implement from moving towards the goal. Finally, we assume that we are working within a planar tree environment, similar to modern high-density orchard systems, which means that all targets should be discernible from a 2D image taken with a camera pointed towards the tree. This assumption, plus the general noisiness of depth data from depth cameras, motivated us to build a control system that uses 2D image data to align the tool with the target branch.

Our robot system is shown in Figure 2. It consists of a Universal Robots UR5e (Odense, Denmark) six degree-of-freedom industrial manipulator with a replica 3D pruning implement attached to the robot’s tool flange. We mount an Intel RealSense D435 (Santa Clara, CA, USA) on top of the cutter such that the cutting implement is in the camera’s field of view, allowing for accurate vertical and horizontal positioning of the pruners with respect to the tree. The goal is to navigate the cutters so that the branch is located at the pivot point of the cutters, minimizing the amount of force required to prune the enclosed branch [16].

The control flow of our pruning process is shown in Figure 3. Our hybrid control system is divided into two main systems: A vision-based controller which focuses on moving the cutter to the general proximity of the target branch (Section IV), followed by a force-based admittance controller which aligns the branch with the cutter pivot point (Section V).

IV. VISION SYSTEM

The goal of the vision system is to guide the cutters so that the branch is enclosed in the cutter jaws. We assume the robot end-effector is initially positioned 15-20 cm from the target branch, the camera sees both the cutter and the target
Our primary challenge is building a controller that is easily trainable and robust to a wide variety of lighting, branch shape, color, texture, and background complexities. To do this, we adopt the approach in [17] and train a Generative Adversarial Network (GAN) [18], which essentially segments out the branches in the scene. Importantly, this GAN performs similarly for both simulated and real images, allowing us to train our controller entirely in simulation. We then train a deep neural network using an actor-critic reinforcement learning paradigm. The neural network takes in a segmented image and outputs a vertical and horizontal control action to move the robot, with the goal of positioning the branch inside the cutter jaws.

First, we cover the simulation environment used to collect simulated data (Section IV-A). Next, we discuss how we adapt existing work on sim-to-real transfer the aforementioned GAN (Section IV-B). Finally, we formally define the reinforcement learning problem (Section IV-C).

A. Simulation environment

We mimic our physical experimental setup (Section VI-A) in the PyBullet physics engine [19] (Figure 4). The setup consists of the UR5e arm mounted on a cart with the cutter as an end-effector, a trellis system consisting of a wooden frame, wires, and three spindle tree models randomly chosen from our 8 pre-generated models. Each of the tree spindles is annotated with several pruning targets on the side branches, located approximately 3 cm from the upright leaders. The tree models are assumed to be completely stiff.

We render RGB images of the simulated environment from the virtual camera to use as input to the GAN. We also compute the distance from the cutter to the given target cut point and use collision detection to query if the target branch has entered the cutter or not (Figure 4). Since the trees are completely stiff, the collision physics between the robot and the environment are not realistic. Therefore, we extend the success region slightly beyond the cutter mouth. The assumption is that subsequent movements of the arm will not cause the branch to leave the success region.

B. Real-to-sim transfer

Our approach to the sim-to-real transfer follows the approach of [17], which trains a GAN to transform the real image into a “synthetic” version — essentially performing a segmentation. They train the GAN with simulated images of the environment generated using a wide variety of lighting, colors, and textures. This heavy randomization allows the system to generalize to real images surprisingly well. The simple, segmented image essentially decouples the visual processing of a complex scene from the decision of what action to take given where the branch is.

We follow their methodology to create a 3-channel segmented representation of our environment. First, we identify five unique elements in the scene: The cutter, the trees, the wooden trellis frame, the trellis wires, and everything else (i.e. the walls and floor). We replace the object’s true texture with a unique uniform color texture for each element and render the scene again with the simplified colors. Next, we convert the RGB image into the HSV (hue-saturation-value) space. Finally, we replace the S channel with a segmentation mask from PyBullet specifically for the trees (255 if the pixel corresponds to a tree and 0 otherwise).

To train the GAN, we pair each segmented environment representation with a corresponding randomized version in
which each object is assigned one of 5000 random synthetic textures and the lighting’s source, color, and intensity have been randomized. We generate 10,000 randomized segmented image pairings and utilize the pix2pix architecture [20] to learn how to translate the randomized images to their segmented form. Examples of the trained GAN working on different images are shown in Figure 5.

C. Reinforcement learning

We now formally define a pruning episode, as well as the state space \( S \), the action space \( A \), the reward function \( R \), and the transition function \( T \) which define a Markov Decision Process (MDP). Each episode is given a maximum time length \( T \) s with a time interval of \( \delta t \) s, resulting in a maximum number of timesteps of \( N_t = \frac{T}{\delta t} \). At the start, we select a target cut \( p_{cut} \) on a random spindle and position the cutter 15-20 cm in front of \( p_{cut} \).

At each timestep we obtain a 424 x 240 segmented image from the GAN. Since the cutter and branch are in the right of the image, we crop the bottom-right 360 x 180 of the image and rescale it to 160 x 80. Therefore, our state space is the set of 160 x 80 3-channel images \( S = Im_{160 \times 80 \times 3} \).

The segmented image is fed through a deep neural network which produces two numeric values, \( a_x, a_y \in [-1,1] \), representing the velocity outputs \( A = [-1,1] \times [-1,1] \). By default, the cutter moves in the tool frame’s z-axis at a speed of \( s \) m/s. The chosen action values modify the \( x \) and \( y \) components of this velocity vector, such that the final velocity vector is \( \vec{v} = (a_x s, a_y s, s) \). The transition function \( T \) maps to the next state by setting the tool velocity to \( \vec{v} \) and running the simulator forward for \( \delta t \) s.

The reward function \( R \) is engineered to reward the cutter for approaching the target cut point, giving more reward as the cutter approaches the target, while heavily penalizing attempts which end up missing the cutters. The total reward per episode ranges between \( -T \) and \( T \). Each timestep’s reward depends only on the state \( s_{n+1} \) reached after applying action \( a_n \) at state \( s_n \). At each new state \( s_{n+1} \), we check for one of three terminal conditions:

- **(Success)** The branch has entered the success zone.
- **(Failure)** The branch has entered the failure zone.
- **(Failure)** The episode has reached \( N_t \) time steps.

Upon success, the agent receives a reward corresponding to the remaining time \( T - \delta t \) , while upon failure, it receives a large penalty of \( -T \). Otherwise, the reward is a value between 0 and \( \delta t \) depending on how close the cutter is to the target and a minimum distance threshold \( d_{min} = 0.10 \text{ m} \):

\[
R(s_{t+1}) = \delta t \max \left( 1 - \frac{d_{t+1}}{d_{min}}, 0 \right)
\]

(1)

For training in simulation, we set \( T = 1 \text{ s} \) and \( \delta t = 0.10 \text{ s} \) \( (N_t = 10) \). To encourage fast episode completion, we set the forward velocity equal to \( s = 0.30 \text{ m/s} \). For the real robot arm, we limited the forward velocity to \( s = 0.03 \text{ m/s} \).

We use the Proximal Policy Optimization (PPO) algorithm [21] from the stable-baselines3 package [22] to learn the desired control actions. We use the default hyperparameters and deep neural network architecture. This consists of a feature extractor with three convolutional layers and one feedforward layer with ReLU activation. The action and value networks share the output of the feature extractor and process the features through their own feedforward layers to obtain the velocity command and value estimates.

V. ADMITTANCE CONTROLLER

Once a force exceeding 0.75 N is detected, we assume that the target branch has contacted a cutting surface and transition to an interaction controller. Regulating the dynamic interaction with the branch is critical for two reasons. First, we want to minimize contact forces to prevent damage to the tool/robot. Second, to deliver maximum cutting force, the branch should be located as close to the blade pivot point as possible (see Figure 6).

The contact task is characterized by interaction between a ‘stiff’ industrial manipulator coupled with an elastically compliant branch. We choose to use admittance control: the robot senses external forces using its internal wrist force-torque sensor and controls its motion in response. An admittance controller imitates the dynamics of a spring-mass damper in the robot’s task space, as described by Equation 2:

\[
M \ddot{x}_c + B \dot{x} + K x = F_{ext}
\]

(2)

Here, \( M \), \( B \), and \( K \) are the mass, damping, and stiffness matrices to be imitated, \( (x, \dot{x}) \) are the end-effector position and velocity in task space, \( F_{ext} \) is the wrench exerted on the environment, and \( \ddot{x}_c \) is the output acceleration as determined by the controller. Since there is no desired “position” of
the end-effector, we can eliminate the stiffness term and rearrange Equation 2 as shown in Equation 3

\[ \ddot{x}_c = M^{-1}(\mathcal{F}_{\text{ext}} - B\dot{x}) \] (3)

Once the branch is in contact, we assume that \( x \)-axis (side to side) motions are unnecessary. This is captured in a square selection matrix \( \Lambda \) with a diagonal of \( \text{diag}(\Lambda) = [0 \ 0 \ 0 \ 0 \ 1 \ 1] \). Multiplying \( \Lambda \) by \( \mathcal{F}_{\text{ext}} \) ensures that the admittance controller only generates motion in the vertical and forward axes.

Because the force-torque sensor is inherently noisy, we impose both a moving average filter and a deadzone function on the raw wrench data. The sensor publishes at 500Hz, so we implemented a 51-value moving average filter to create \( \mathcal{F}'_{\text{ext}} = \text{filter}(\mathcal{F}_{\text{ext}}) \). While this results in a 0.1 s delay, we assume that the maximum speed of the cutter (currently 1 cm/s) is small enough to mitigate any significant instability.

We also impose a deadzone function in which all force errors \( F_e \) under some threshold \( F_{th} = 0.2 \text{ N} \) are set to zero so that small force variations do not produce motion. The deadzone function is defined in Equation 4

\[ dz(F_e) = \text{sgn}(F_e) \max(|F_e| - F_{th}, 0) \] (4)

We define the desired end-effector wrench to be \( \mathcal{F}_{\text{des}} \), and define the desired velocity to be all zeroes. Thus, the full admittance controller is as shown in Equation 5

\[ \ddot{x}_c = M^{-1}(dz(\mathcal{F}_{\text{des}} - \Lambda \mathcal{F}'_{\text{ext}}) - B\dot{x}) \] (5)

For our experiments, we defined the desired wrench to be

\[ \mathcal{F}_{\text{des}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -2 \end{bmatrix} \] (6)

in which a 2 N force is imposed forwards on the branch. We designed the hardware such that the cut point and sensor are located at the same height; therefore, when the branch is at the pivot point, the moment about the \( x \) axis will be 0. The controller declares the objective complete when the sensed torque in the \( x \) direction is within 0.0025 N-m of the desired torque and the tool has moved less than .5 mm upward or forward in the last 1s.

We set the controller gains to diagonal matrices with \( \text{diag}(M) = [0 \ 0 \ 0 \ 0 \ 100 \ 10] \) and \( \text{diag}(B) = [0 \ 0 \ 0 \ 0 \ 400 \ 250] \). On the real robot, controller commands were sent to the UR5e as discrete velocities: \( \ddot{x}_{n} = \ddot{x}_{n-1} + \ddot{x}_c * dt \).

VI. EXPERIMENTS

A. Experimental Setup

We set up a mock trellis system (Figure 7) to compare our controller with a baseline controller in a physical environment. This setup is based on the structure found in modern orchards, consisting of a wooden frame with two trellis wires running across the frame. We then affix three tree branches, obtained from a real unpruned sweet cherry tree, to the base of the frame and tie the branches down to the trellis wires. This gives the branches compliance properties similar to those found in an actual orchard. We identified seven viable pruning target branches on the leaders (i.e. smaller side branches growing from the main leaders).

B. Comparison controllers

In order to evaluate our hybrid controller we define three position-based controllers. These were chosen to independently evaluate our controller versus a position-based one, and the effectiveness of the force-based controller over a vision-based one. Specifically:

- (OL) A simple open-loop Cartesian-space proportional controller (with maximum velocity of 3 cm/s) which stops when the cutter pivot is within 1 mm of the target.
- (CL) A closed-loop controller which sets the target 10 cm past the target estimate and uses the open-loop proportional controller, but transitions to the admittance controller upon contact.
- (OL-) A miscalibrated open-loop position controller. This is the same as the open-loop position controller, but when estimating the target position, the pose of the camera is perturbed by 1 cm in a random direction with rotation of 5 degrees about a random axis, resulting in a different target estimate than the other methods.

C. Experimental trials and evaluation criteria

For each trial, we first selected one of the seven target pruning points, each of which was marked with a removable piece of tape 3 cm from the leader. After driving the robot back to one of four home positions, each of which provided a distinct view of the tree (from the front, left, right, and tilted up), we selected a point from the point cloud corresponding to the target, giving us a 3D estimate of the desired pruning cut point (including a perturbed estimate from the miscalibrated controller). We then planned a path so that the cutter was 15 cm away from the estimated target, oriented normal to the trellis plane, and then removed the piece of tape marking the target.

At this point, we run each of the four control systems in succession (our hybrid controller plus the three position-based controllers) and recorded the following metrics to evaluate the performance of each cut:

- Accuracy: Is the target branch in the mouth of the cutter?
Generally speaking, the depth estimates provided from the camera fell short of the actual targets, which meant that in most of the open-loop runs, the cutter simply stopped in front of the target without ever touching it. However, on the occasions that the open-loop controllers did hit the branch, because of the lack of force feedback, they would push into the branch with a very high amount of force, sometimes upwards of 20 N as a result of hitting a vertical leader branch. Meanwhile, both our hybrid controller and simple closed-loop controller kept all forces below 10 N, demonstrating the value of the interaction controller in preventing damage to the environment. Although our controller did generally incur higher forces than the simple closed-loop controller, likely due to the acceleration of the tool from responding to the neural network, at no point did it ever hit the environment with enough force to cause any notable damage.

One note is that there appeared to be some error in our “ground truth” kinematics when transforming point cloud points into the world frame. This was evidenced by two observations: the similarity of the two open loop controllers’ overall performance, and the fact that when running the experiments, left-side branch target estimates tended to be far away from the leader while right-side branch target estimates tended to be very close to the leader. While locating and fixing the calibration issues may have improved the overall performance of the simple closed-loop controller, it would also reinforce the point that our system is robust to imperfect kinematic calibration, as well as other perturbations such as cart or tree movement that a point estimate-based controller simply has no way to handle.

VIII. Conclusion

In this paper, we introduced a framework for highly accurate robotic pruning that uses 2D RGB images and force data to consistently execute accurate cuts with minimal force. Through our experiments, we demonstrated that our system is robust to a number of issues that have traditionally been associated with depth camera data, primarily kinematic miscalibrations and incorrect depth estimates. While we conduct our experiments in an indoor laboratory setting, our framework represents a step towards moving robotic precision pruning to the outdoor orchard environment. Future
work will include adding rotation as a third dimension to the velocity command (to ensure perpendicular alignment), as well as determining how to adjust the segmentation network to work in more complex outdoor settings with background noise from adjacent rows.

REFERENCES

[1] P. S. U. C. of Agricultural Sciences and P. S. U. C. Extension, *Penn State Tree Fruit Production Guide*. Penn State, College of Agricultural Sciences, 2015.

[2] K. Gallardo, M. Taylor, and H. Hinman, “Cost estimates of establishing and producing Gala apples in Washington,” *Washington State University*, 2009.

[3] L. Calvin and P. Martin, “The US produce industry and labor: Facing the future in a global economy,” Tech. Rep., 2010.

[4] 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?” *Journal of Field Robotics*, vol. 38, no. 4, pp. 513–531, 2021.

[5] A. You, F. Sukkar, R. Fitch, M. Karkee, and J. R. Davidson, “An efficient planning and control framework for pruning fruit trees,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3930–3936.

[6] A. Zahid, S. Mahmud, L. He, P. Heinemann, D. Choi, and J. Schupp, “Technological advancements towards developing a robotic pruner for apple trees: A review,” *Computers and Electronics in Agriculture*, vol. 189, no. January, 2021.

[7] V. R. Corporation. (2017) Vision Robotics Corporation. [Online]. Available: https://www.visionrobotics.com/

[8] T. Botterill, S. Paulin, R. Green, S. Williams, J. Lin, V. Saxton, S. Mills, X. Chen, and S. Corbett-Davies, “A robot system for pruning grape vines,” *Journal of Field Robotics*, vol. 34, no. 6, pp. 1100–1122, 2017.

[9] 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 *ASABE 2020 Annual International Meeting*. American Society of Agricultural and Biological Engineers, 2020, p. 1.

[10] M. Prats, P. J. Sanz, and A. P. Del Pobil, “Vision-tactile-force integration and robot physical interaction,” in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 3975–3980.

[11] A. J. Schmid, N. Gorges, D. Göger, and H. Worn, “Opening a door with a humanoid robot using multi-sensory tactile feedback,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 285–291, 2008.

[12] Y. Mezouar, M. Prats, and P. Martinet, “External hybrid vision/force control,” in *Intl. Conference on Advanced Robotics (ICAR07)*, 2007, pp. 170–175.

[13] A. C. Leite, F. Lizzarelde, and L. Hsu, “Hybrid vision-force robot control for tasks on unknown smooth surfaces,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation*, 2006. *ICRA 2006*. IEEE, 2006, pp. 2244–2249.

[14] J. Baeten and J. De Schutter, “Hybrid vision/force control at corners in planar robotic-contour following,” *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 2, pp. 143–151, 2002.

[15] M. Q. Mohammed, K. L. Chung, and C. S. Chyi, “Review of deep reinforcement learning-based object grasping: Techniques, open challenges, and recommendations,” *IEEE Access*, vol. 8, pp. 178450–178481, 2020.

[16] A. Zahid, M. S. Mahmud, and L. He, “Evaluation of Branch Cutting Torque Requirements Intended for Robotic Apple Tree Pruning,” in *2021 ASABE Annual International Virtual Meeting*. American Society of Agricultural and Biological Engineers, 2021, p. 1.