An Efficient Planning and Control Framework for Pruning Fruit Trees

Alexander You[1], Fouad Sukkar[2], Robert Fitch[2], Manoj Karkee[3], Joseph R. Davidson[1]

***Abstract*— Dormant pruning is a major cost component of fresh market tree fruit production, nearly equal in scale to harvesting the fruit. However, relatively little focus has been given to the problem of pruning trees autonomously. In this paper, we introduce a robotic system consisting of an industrial manipulator, an eye-in-hand RGB-D camera configuration, and a custom pneumatic cutter. The system is capable of planning and executing a sequence of cuts while making minimal assumptions about the environment. We leverage a novel planning framework designed for high-throughput operation which builds upon previous work to reduce motion planning time and sequence cut points intelligently. In end-to-end experiments with a set of ten different branch configurations, the system achieved a high success rate in plan execution and a 1.5x speedup in throughput versus a baseline planner, representing a significant step towards the goal of practical implementation of robotic pruning.**

I. Introduction

Robots show great promise for transforming agriculture given the demands of feeding the increasing world population and mounting pressure on the supply of agricultural labor [1]. In the context of fresh market fruit production, research effort has focused largely on automating harvesting tasks such as fruit picking and sorting [2]. In comparison, relatively few results have been demonstrated for the task of fruit tree pruning, where diseased or unproductive branches are removed to optimize growth and fruiting sites. Selective pruning is a critical yet costly process; [3] showed that pruning contributes to over 20% of variable costs for Gala apple orchards in Washington. Pruning also is physically demanding and exposes workers to hazards such as falls, cuts, and repetitive strain injuries [4]. Non-selective mechanized pruning methods exist, but have been shown to induce lower-quality yields for certain classes of fruit trees [5].

Although selective pruning is a prime target for automation by robots, only a few complete end-to-end automated pruning systems for specialty crops have been demonstrated [6], including a grapevine pruner by Vision Robotics (first shown in 2014) [7] and a grapevine pruner by Botterill et al. [8]. A robotic pruning system must be able to perceive the 3D pose of the branch to be cut, as well as move a cutting implement precisely to a cut point while avoiding obstacles such as other branches or structures. A fully autonomous system must also be able to choose cut points automatically, but here we consider the case where cut points are manually labelled in a 2D image of the area to be pruned. The main challenge that stands in the way of practical implementation of such systems is to perform these tasks with sufficient throughput to be economically viable.

This research is supported by an Australian Government Research Training Program (RTP) Scholarship, the University of Technology Sydney, Oregon State University, and the Washington Tree Fruit Research Commission. AY received support from a scholar award funded by the ARCS Foundation Oregon.

[1]Collaborative Robotics and Intelligent Systems (CoRIS) Institute, Oregon State University, Corvallis OR 97331, USA {youa, joseph.davidson}@oregonstate.edu

[2]School of Mechanical and Mechatronic Engineering, University of Technology Sydney, Ultimo NSW 2006, Australia {fouad.sukkar@student., robert.fitch@}uts.edu.au

[3]Center for Precision and Automated Agricultural Systems, Washington State University, Prosser WA 99350, USA manoj.karkee@wsu.edu

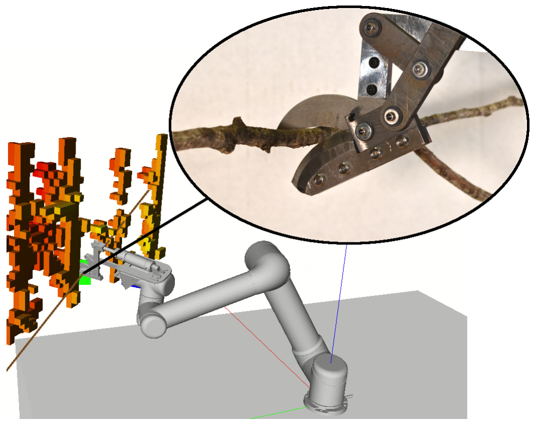
Fig. 1. Our autonomous pruning system with *(inset)* a custom cutter mechanism.

This paper introduces a general algorithmic framework for pruning. Given human input from a standard visual feed, we present an end-to-end system that is capable of executing multiple cuts efficiently, with minimal assumptions about the environment. To achieve a low average cut cycle time, it is necessary to consider the speed of motion planning, executing the plans, and of cutting the branch. Moreover, the choice of the sequence of cuts to be made can be a dominant factor in minimizing manipulator travel time between cuts. Our system uses a manipulator arm with a low-cost RGB-D camera and custom high-speed pneumatic shears, shown in Fig. 1, as the cutting implement. As a successful cut requires accurate estimates of the 3D branch poses, we utilize a novel closed-loop visual feedback system tailored for navigating branches to accurately position the manipulator. We adapt existing work to plan the manipulator trajectory and integrate it into an optimization process that uses physically-realistic time cost estimates to determine the cut sequence.

Our main contribution lies in an end-to-end demonstration of pruning with a high throughput rate. We compare our system to a baseline planner that uses Euclidean distance between cut points in determining the cut sequence, representative of state-of-the-art systems. We report average

Fig. 2. Sweet cherry trees being grown in an upright fruiting offshoot (UFO) architecture.

planning throughput from ten experiments that is higher than the baseline (1.5x speedup) with a high (92%) success rate. One technical contribution is the adaptation of an existing fast motion planning framework to pruning, requiring the definition of several critical algorithmic components specific to this application, and its use in providing accurate estimates in optimizing the cut sequence. We also contribute a demonstration of a novel perception pipeline for registering cut points. Our results indicate that this system, given modest further development and tuning, could feasibly approach performance levels that would be acceptable in practice.

II. Related Work

A majority of the effort in automating dormant pruning has focused on perception problems such as tree modelling and branch detection [6]. Modern orchards, such as the upright fruiting offshoot (UFO) sweet cherry system shown in Fig. 2, are more amenable to automation due to their relatively simple and uniform layouts. However, it is still necessary to reason about branch structure and make online decisions in order to avoid damaging the crop.

The closest work to ours is that of Botterill et al. [8], who introduced a complete grapevine-pruning system that uses Bidirectional RRT (BiRRT) for motion planning with open-loop control, and Euclidean distance between cut points for optimizing the cut sequence. In comparison, we leverage the Fast Reliable and Efficient Database Search Motion Planner (FREDS-MP) framework [9] to compute efficient plans, increase success rates, and provide more accurate estimates when sequencing cut points. Our system also incorporates sensor feedback to improve cutting accuracy.

Automated pruning is related to robotic harvesting [10], [11], [12], [13], [14]. However, the set of feasible poses for successful pruning is smaller than that of picking a fruit due to the importance of approaching the branch with a certain cutter orientation. Pruning arguably requires greater precision due to the consequences of making an incorrect cut. Furthermore, visual servoing techniques in harvesting tend to focus on discrete fruit [15], whereas pruning involves tracking visually indistinct cut points on a branch.

III. System Overview

Our robot system, shown in Fig. 3, consists of a Universal Robots (Odense, Denmark) UR5e equipped with a cutter

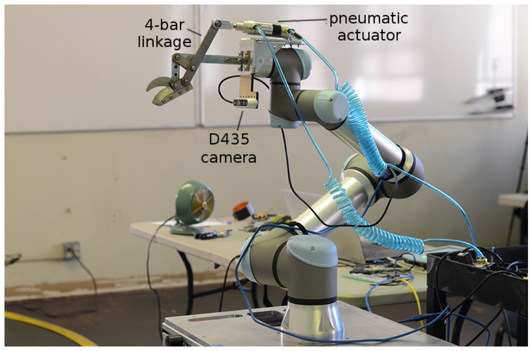

Fig. 3. Our pruning robot setup, consisting of a UR5e arm, an Intel RealSense D435 RGB-D camera, and custom pneumatically-actuated shears.

and an eye-in-hand Intel (Santa Clara, CA) RealSense D435 camera mounted beside the cutter. The UR5e is a six degree-of-freedom (DOF) industrial manipulator with a ROS control interface [16]. The cutter is a pneumatically-actuated four-bar linkage with custom-ground blades developed in-house at Oregon State University; initial tests of the cutter showed that it was able to consistently cut branches up to 10mm in diameter near the pivot point of the blades.

Figure 4 shows a flowchart illustrating the process of selecting branches and moving the cutter to the desired cut points on the tree. There are three main phases: *pre-execution*, in which a human operator manually selects the points on the tree to be cut via a human-robot collaborative interface; *planning*, in which the planner determines the cut point order and plans paths between goal poses; and *execution*, where the manipulator positions the end effector and cuts the branch. A major component of the execution system is a visual feedback loop which runs in parallel with the servoing to update the goal position.

IV. Perception

This section describes the perception pipeline of the system. In particular, we provide implementation details on image coregistration and depth refinement for robust cut point tracking.

A. Environment Representation

The tree to be pruned and surrounding trellis structure, W_{env}, are modelled using an OctoMap [17], [18], i.e. a 3D occupancy grid, constructed from point cloud data from the camera (illustrated in Fig. 5). We currently consider data from a single point cloud. In general, the framework is agnostic to obstacle representation and any convenient representation can be used.

B. Cut Point Generation

Cut point selection is currently performed by a human operator by clicking on a 2D image from the camera. In order to obtain a 3D global estimate of the clicked point, we assume the existence of a plane $\mathscr{P}$ which represents an estimate of our orchard system.

Suppose each cut point is indexed by an identifier i. Let $\overrightarrow{\mathbf{c}_i} \in SE(3)$ represent the base camera pose with corresponding principal point $\mathbf{c}_i \in \mathbb{R}^3$; let Im_i be the selected pixel and

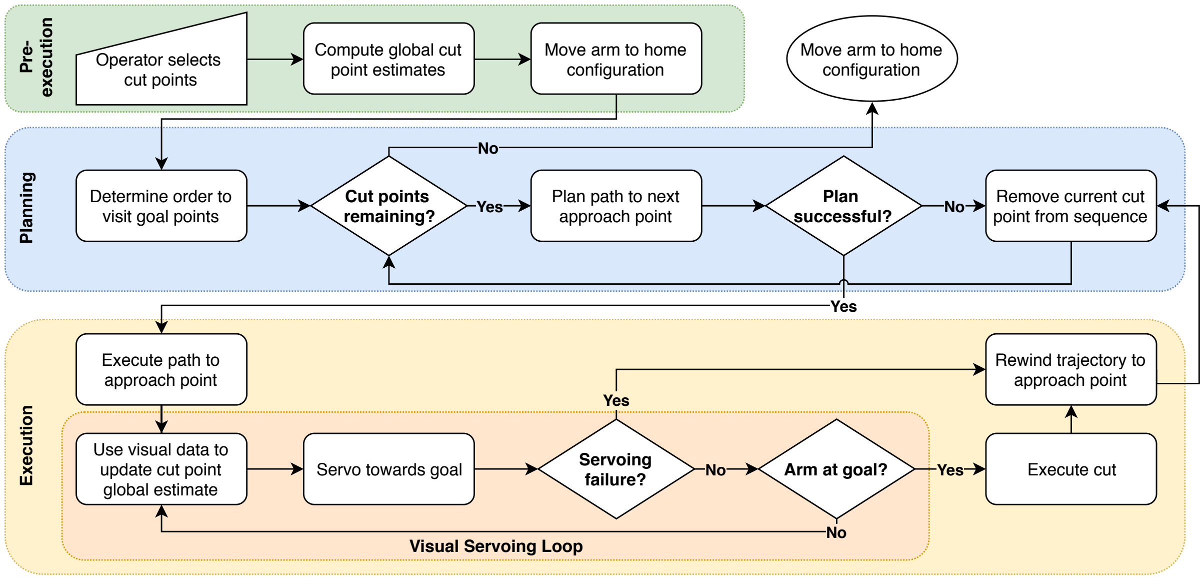
Pre-execution
Operator selects cut points
Compute global cut point estimates
Move arm to home configuration
Move arm to home configuration
Planning
Determine order to visit goal points
Cut points remaining?
Yes
No
Plan path to next approach point
Plan successful?
No
Yes
Remove current cut point from sequence
Execution
Execute path to approach point
Rewind trajectory to approach point
Yes
Use visual data to update cut point global estimate
Servo towards goal
Servoing failure?
No
Arm at goal?
Yes
No
Execute cut
Visual Servoing Loop

Fig. 4. A flowchart of the cutting process from start to finish. Steps fall into three main phases: pre-execution, planning, and execution.

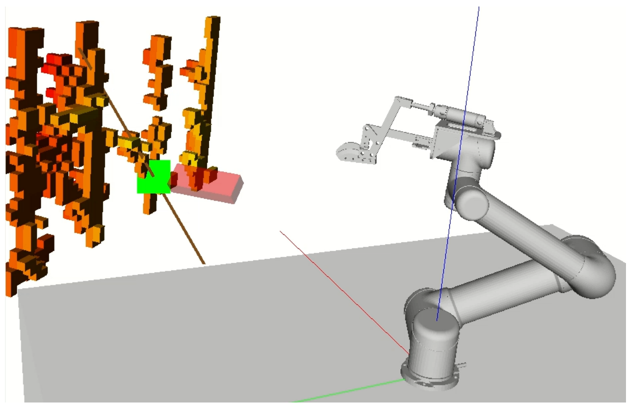

Fig. 5. An image from the simulation showing the OctoMap representation of the tree system. The target branch is shown as a brown line, and the red box represents the potential collision area with the cutter and the environment on the approach.

$\mathbf{Im}_i$ be the 3D location of the pixel in the camera's image plane, computed using the camera's intrinsic parameters. For two distinct 3D points $\mathbf{p}_1$ and $\mathbf{p}_2$, let $\overrightarrow{(\mathbf{p}_1, \mathbf{p}_2)}$ represent the line drawn between $\mathbf{p}_1$ and $\mathbf{p}_2$. The line $\overrightarrow{\mathbf{l}_i} = \overrightarrow{(\mathbf{c}_i, \mathbf{Im}_i)}$ represents all possible locations of Im_i in 3D; we set the initial cut point estimate $\mathbf{g}_i = \mathscr{P} \cap \overrightarrow{\mathbf{l}_i}$.

To estimate branch orientation, we utilize simple color-based image segmentation to classify image pixels as branch or non-branch. For some exogenous parameter δ, we compute the set of all branch pixels within a δ-pixel radius around Im_i. We perform a singular value decomposition on this set and utilize the first right-singular vector to define a second pixel location $\widetilde{Im}_i$ representing the branch linear approximation with respect to Im_i. Finally, we use the corresponding image plane point $\widetilde{\mathbf{Im}}_i$ to obtain $\widetilde{\mathbf{g}}_i = \mathscr{P} \cap \overrightarrow{(\mathbf{c}_i, \widetilde{\mathbf{Im}}_i)}$, yielding a linear branch estimate $\overrightarrow{\mathbf{b}_i} = \overrightarrow{(\mathbf{g}_i, \widetilde{\mathbf{g}}_i)}$.

C. Image Coregistration

The estimated position $\mathbf{g}_i$ is updated to approach the true position $\mathbf{g}_i^*$ as the cutter moves closer to the branch. To do so, we use the image coregistration method shown in Fig. 6. This method takes advantage of the fact that $\mathbf{g}_i$ and $\mathbf{g}_i^*$ should both lie on $\overrightarrow{\mathbf{l}_i}$ (shown in the figure as a solid blue line).

From a new camera view, consider the projection $\overrightarrow{\mathbf{l}_i}$ into the image, shown in Fig. 6 as a dotted blue line. We examine branch pixels which intersect with this projection and consider where the corresponding image plane lines intersect $\overrightarrow{\mathbf{l}_i}$. The goal estimate $\mathbf{g}_i$ is updated to match the line intersection closest to $\mathbf{g}_i$.

For example, in Fig. 6, the two pixel candidates (marked with red and green circles) intersect with $\overrightarrow{\mathbf{l}_i}$ at the green and red stars. The green star is chosen as the new goal estimate due to being more consistent with the original estimate.

D. Depth Refinement

While image coregistration can be effective in updating the goal estimate, it is sensitive to calibration errors. To overcome this limitation we utilize the camera's depth data. Let $\hat{\mathbf{p}}_j$ represent the estimate of the goal in iteration $j \in \mathbb{N}$; for $j = 0$, we use the coregistration process previously described to update $\mathbf{g}_i$ and set $\hat{\mathbf{p}}_0 = \mathbf{g}_i$. Then for $j > 0$, we run the image coregistration process to obtain a candidate pixel Im'_j and transform $\hat{\mathbf{p}}_{j-1}$ into the current camera frame to obtain an estimate of the goal's depth $\hat{p}'_{z,j-1}$. This is used to obtain a new camera-frame estimate of the goal $\hat{\mathbf{p}}_j^{*'}$.

To incorporate the depth data, let $\mathbf{P}'_j$ denote the set of 3D camera-frame points. For some exogenous value ε, let $\mathbf{P}'_{\varepsilon,j} = \{\mathbf{p}' \in \mathbf{P}'_j \mid \left|\mathbf{p}' - \hat{\mathbf{p}}_j^{*'}\right| < \varepsilon\}$. If $\mathbf{P}'_{\varepsilon,j}$ has a sufficient number of points, we use linear regression to predict a depth value z' from the corresponding pixel coordinates (Im'_{x_k}, Im'_{y_k}):

$$\hat{z}' = \alpha + \beta_x Im'_x + \beta_y Im'_y \tag{1}$$

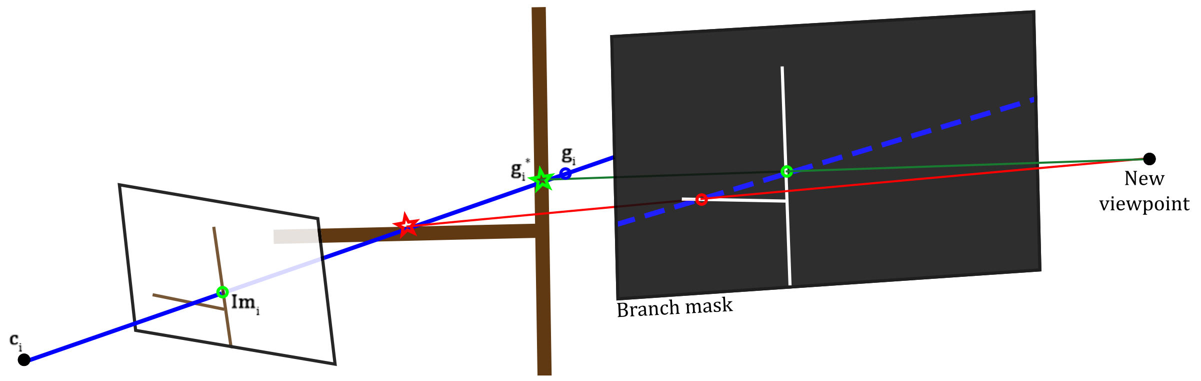

Fig. 6. An illustration of the image coregistration process. Having selected the pixel corresponding to $\mathbf{Im}_i$ as the desired cut point, in the new camera frame we utilize a branch mask along with the ground truth ray (shown in blue) between $\mathbf{c}_i$ and $\mathbf{Im}_i$ to refine the initial estimate $\mathbf{g}_i$ to $\mathbf{g}_i^*$.

By plugging Im'_j into this model, we obtain a corresponding depth estimate $\hat{z}'_j$; with Im'_j and $\hat{z}'_j$, we can compute the camera-frame goal estimate $\hat{\mathbf{p}}'_j$, which is then transformed into the new global cut point estimate $\hat{\mathbf{p}}_j$.

V. Task and Motion Planning

This section describes the planning components of the system. To position the cutter accurately, the system initially moves the cutter to an intermediate *approach pose* and then follows a linear *approach path*. Motion planning is implemented using FREDS-MP, which precomputes a database of optimistic trajectories offline and then queries this database online to initialize a set of trajectory optimization algorithms. We also utilize the trajectory costs stored in the database to optimize the cut sequence.

A. Preliminaries

The robot operates in a 3D Euclidean workspace $W = \mathbb{R}^3$. The system can command the robot's end effector to achieve any arbitrary 6D pose $T \in SE(3)$ given that a valid collision-free inverse kinematic (IK) solution $\theta \in Q(T)$ exists. The obstacle region O is the union of O_{robot}, the model of the robot and its attachments, and W_{env}, the model of environmental obstacles such as the tree and trellis structure.

Let $\theta \in C$ denote a configuration in the configuration space, i.e. the set of all possible configurations of the robot. We denote $A(\theta) \subset W$ as the space occupied by the robot model at configuration θ. The obstacle region is defined as $C_{obs} = \{\theta \in C \mid A(\theta) \cap O \neq \emptyset\}$, from which we obtain the free space region $C_{free} = C \setminus C_{obs}$.

Given two poses T_{start} and T_{end}, we can represent a trajectory with N steps for a K-DOF arm as the following $N \times K$-dimensional vector:

$$\pi(T_{start}, T_{end}) = \{\theta_1, \theta_2, ..., \theta_N\}, \tag{2}$$

Given some distance function d, we define the trajectory cost as follows:

$$f(\pi(T_{start}, T_{end})) = \sum_{i=1}^{N-1} d(\theta_i, \theta_i + 1), \tag{3}$$

B. Database Generation

Although the set of all possible cut poses in $SE(3)$ is infinite, in many applications the robot will operate in a limited sub-volume of the workspace. The offline database generation works by discretizing this sub-volume into a grid of poses $\hat{t}_{offline}$ representing the approximate poses to which we expect to plan. For the pruning application, our discretization strategy is to assume that the cut points lie in some vertical 2D plane and generate a set of uniform 6D poses normal to this plane, as shown in Fig. 7.

Each pose $t \in \hat{t}_{offline}$ may have many IK solutions $Q(t)$. Our goal is to assign a single optimal configuration $\theta^* \in Q(t)$ to each t, considering factors such as the configuration's manipulability and the ease of moving between neighboring poses. By preassigning a single configuration to each pose, we facilitate fast online planning and task sequencing while maintaining high trajectory efficiency and reliability.

To choose θ^*, we run an optimization routine which treats $\hat{t}_{offline}$ as an undirected graph G, with each node $v \in G$ corresponding to a pose $t \in \hat{t}_{offline}$. The routine then iteratively assigns a single IK solution to each node and attempts to minimize the sum of all edge costs in the graph. The edge cost between nodes $u, v \in G$ is defined as:

$$c(v,u) = \|\theta_v - \theta_u\| + \frac{1}{K}\sum_{j=1}^{K} \tanh(2\left|\frac{\theta_u^j - \theta^{lower_j}}{\theta^{upper_j} - \theta^{lower_j}} - \frac{1}{2}\right|). \tag{4}$$

Intuitively, this cost encourages IK solutions for a given graph to be close to each other in C-space. It also penalizes configurations near the joint limits in order to prevent joint limit and manipulability failures during the approach phase.

After determining the optimal graph, the optimizer computes optimistic trajectories between every pair of nodes and their corresponding costs; here, the distance function d is L_2 distance. These costs are optimistic since they assume a $W_{env} = \emptyset$ prior (though one could substitute a bounded volume), but they do take into account the non-linearity of the robot arm. Further details can be found in [9].

C. Approach Pose Selection

Our goal in picking an approach pose is to move the cutter to a pose such that the cutter is pointed towards the

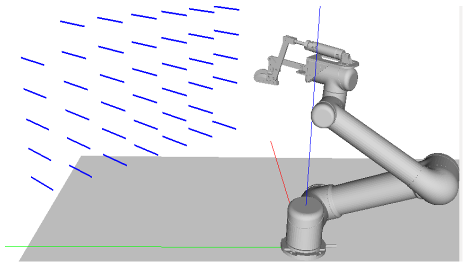

Fig. 7. Discretization strategy used for database generation.

branch and the branch is roughly orthogonal to the plane containing the cutter. Recall that in Sec. IV-B, we generated linear branch approximations $\overrightarrow{\mathbf{b}_i}$ for each goal.

To generate the candidate approach poses, we assume the pitch of the approach pose is 0, i.e. the approach path is parallel to the global xy frame. We then choose the roll and yaw of the approach pose such that the cutter plane is orthogonal to $\overrightarrow{\mathbf{b}_i}$. If such an approach is not possible, either due to collisions or lack of IK solutions, we vary the yaw within some threshold until a viable approach pose is found or the goal is deemed unreachable and discarded.

To check for collisions along the approach, we create a bounding box as shown in Figure 5 which represents the likely path of the cutter during the approach, slightly inflated to account for deviations in the path; we then check if the box collides with W_{env}. The assumption we make is that because the arm is moving on a short linear path, the arm is unlikely to perform any large movements which will collide with the environment or itself, and as such we only need to check for environment-cutter collisions.

D. Cut Point Sequencing

Given a set of desired approach poses $\hat{t}_{online}$, in order to use the offline database, we first associate each pose $t \in \hat{t}_{online}$ with an offline pose as follows. For some $k \in \mathbb{Z}^+$, let $\Theta_{t,k}$ denote the configurations of the k-closest poses in $\hat{t}_{offline}$ to t. We then compute the IK set $Q(t)$ and find the pair $(\theta_t^{min}, \theta_{t,offline}^{min}) \in Q(t) \times \Theta_{t,k}$ which minimizes the pairwise L_2 distance, with v_t^{min} denoting the database node corresponding to $\theta_{t,offline}^{min}$.

Then for every pair of input poses $t_i, t_j \in \hat{t}_{online}$, to compute a trajectory from t_i to t_j we retrieve the corresponding trajectory $\pi(v_{t_i}^{min}, v_{t_j}^{min}) = \{\theta_1, ..., \theta_N\}$ and simply append $\theta_{t_i}^{min}$ and $\theta_{t_j}^{min}$ to the start and end to obtain the trajectory $\pi(v_{t_i}^{min}, v_{t_j}^{min}) = \{\theta_{t_i}^{min}, \theta_1, ..., \theta_N, \theta_{t_j}^{min}\}$. The corresponding path cost $f(\pi(v_{t_i}^{min}, v_{t_j}^{min}))$ is computed by retrieving the offline path cost $f(\pi(v_{t_i}^{min}, v_{t_j}^{min}))$ and adding on $\|\theta_{t_i}^{min} - \theta_1\|$ and $\|\theta_{t_j}^{min} - \theta_N\|$, i.e. the L_2 offsets between the online and offline trajectory endpoints.

This process provides us with edge costs which we can use to generate a weighted adjacency matrix for each pose in $\hat{t}_{online}$. We can then utilize a travelling salesman problem (TSP) solver to determine the optimal pose sequence.

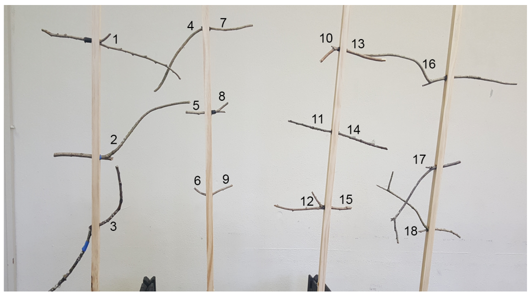

Fig. 8. The test setup used for experimental evaluation. Possible cut points are enumerated.

E. Moving to the Approach Pose

Moving between different approach poses may require complex joint movements to ensure a collision-free path. We take advantage of the optimistic adapted trajectories computed in Sec. V-D by choosing the trajectory corresponding to the desired start and end poses and using a series of trajectory optimizers [19], [20] which adapt the trajectory to W_{env}, yielding a time-continuous safe trajectory. If these optimizers fail, we fall back to the Batch Informed Trees (BIT*) algorithm [21] to generate a plan from scratch.

F. Approaching the Cut Point

Once the cutter has reached the offset pose, the approach routine moves the cutter linearly towards the goal. We compute the desired linear and angular velocities v and ω using a proportional controller and then map them to joint velocities with the Jacobian J and the following equation:

$$\dot{\theta} = J^+ \begin{bmatrix} v \\ \omega \end{bmatrix}, \; J^+ = J^T (JJ^T)^{-1} \tag{5}$$

During this process, we use the eye-in-hand camera configuration to update the cut point location as described in Secs. IV-C and IV-D. We also detect if the configuration approaches a joint limit or falls below a given manipulability threshold, in which case we terminate the approach; if the robot does not encounter a servoing failure, it executes the cut once at the goal. Once complete, we reverse the approach trajectory to return the cutter to the approach pose.

VI. Experiments

A. Experimental Setup

Our experiments use the test setup shown in Fig. 8., designed to resemble a UFO sweet cherry training system (shown earlier in Fig. 2) with horizontal branches emerging from vertical leaders on the tree. For this orchard system, one desirable pruning strategy is simply to cut the branches extending from the leaders. Similarly, our setup consists of four vertical leaders, each of which has holes at three different vertical offsets to hold the branches used for testing.

We evaluate the performance of an end-to-end pruning system in terms of accuracy, speed and reliability. For comparison we use a "naive planner" that sequences cut points based on Euclidean distance in the workspace and

		1	2	3	4	5	6	7	8	9	10	Avg
Success rate	Ours	83%	100%	100%	**83%**	100%	**89%**	**88%**	**100%**	75%	100%	**92%**
	Naive	83%	100%	100%	33%	100%	33%	50%	86%	75%	100%	76%
Sequencing time (s)	Ours	0.33	0.23	0.24	0.30	0.22	0.24	0.27	0.21	0.25	0.23	0.25
	Naive	**0.00**	**0.00**	**0.00**	**0.01**	**0.00**	**0.01**	**0.01**	**0.01**	**0.00**	**0.00**	**0.00**
Planning time (s)	Ours	**0.50**	**0.23**	**0.29**	**0.40**	**0.28**	**0.29**	**0.38**	**0.38**	**0.34**	**0.31**	**0.34**
	Naive	0.57	0.41	0.48	1.81	0.46	3.54	2.84	0.47	0.68	0.47	1.17
Trajectory time (s)	Ours	7.63	4.10	3.89	**5.73**	4.29	**5.52**	**5.46**	5.68	**4.40**	4.48	**5.12**
	Naive	**3.34**	**2.88**	**2.93**	16.12	**2.46**	23.95	15.74	**2.87**	4.51	**3.02**	7.78
Throughput (s)	Ours	8.46	4.57	4.42	**6.43**	4.79	**6.05**	**6.11**	6.26	**5.00**	5.01	**5.71**
	Naive	**3.91**	**3.29**	**3.41**	17.94	**2.92**	27.49	18.59	**3.35**	5.20	**3.49**	8.96

TABLE I

RESULTS FROM TEN EXPERIMENTS. ALL TIMES ARE NORMALISED BY NUMBER OF EXECUTION SUCCESSES WITHIN EACH EXPERIMENT.

plans trajectories as our method does, but with no prior trajectory information. Goal configurations for each approach pose are determined greedily at plan time by choosing the closest IK solution to the current robot configuration.

Each experiment involves 6-10 randomly selected cut points chosen from the 18 possible locations shown earlier in Fig. 8. We randomly configured the branches between experiments to obtain a variety of poses for the planners.

The planning and perception processes ran on a laptop with a six-core Intel i7 2.20 GHz processor. The UR5e robot was limited to 250 mm/s end-effector velocity for safety. The minimum voxel size for the OctoMap was 2 cm, and the approach yaw was allowed to vary ± 45 degrees.

B. Evaluation Criteria

There are three stages during which failures can occur: planning to the approach pose; executing the approach path (e.g., if the robot encounters a servoing error); and cutting the branch (e.g., if the mouth of the cutter does not align properly with the branch). An *execution success* is when no failures occur during the first and second stages, whereas a *cut success* is when no failures occur at any stage. Sequencing, planning, and trajectory times are total times for each experiment normalised by the number of execution successes. Approach path failures only consider executions that were able to plan to the approach pose in the first stage.

C. Results and Discussion

Table I shows results averaged within each of the ten experiments as well as total throughput. Our system executed more cuts than the naive planner (92% versus 76% success). Although sequencing times were higher, planning and trajectory times were significantly lower, and approach path failures were lower (1.7% versus 20.2% failures). Independent of planning, the average time of a successful servoing phase, including cutting and rewinding the approach, was about 13 s, and the overall cut success rate was 75%.

The results indicate that the system is computationally efficient enough to carry out pruning in real time. They also indicate that the system is able to prune with high accuracy while avoiding damaging collisions with the environment. Our system achieved a 1.5x speedup in planning throughput with fewer approach path failures than the naive method.

A particularly notable feature shown in Table I is the variation in trajectory times. Though the naive method was faster in certain experiments, its myopic behavior sometimes caused it to do significantly worse when it fell back to the more costly sampling-based planner. By utilizing stored non-myopic plans, our method performs more consistently.

The sequencing time component of our planner was higher since IK solutions are computed during sequencing as opposed to during planning, but this could easily be parallelized. Furthermore, while our system experienced fewer failures, we believe that the success rate could be further increased through minor improvements such as increasing the size of the trajectory database.

The quality of the depth estimates from the RealSense camera limited the performance of the visual servoing. We found the global estimates of cut points to be unreliable, motivating the need for the planar estimate $\mathscr{P}$ mentioned in Sec. IV. In some cases it was difficult to distinguish branch segments from the background in the camera data, leading to visual servoing failures. Despite this, our 75% cutting success rate demonstrates the robustness of the system.

Our experiments provide useful insights towards further development of the system. Planning time is already in the range of hundreds of milliseconds and any improvement here would not contribute greatly to overall system performance. There is, however, an opportunity to reduce execution time. Because our robot safety settings were very conservative, there is significant room to speed up the arm movements. Accurate sequencing is effective, but is limited by the kinematics of the 6-DOF manipulator. [9] demonstrated empirically that the greater agility of a 7-DOF arm is beneficial in improving travel time within a sequence of goal poses. Further, placing the robot on a gantry would expand the workspace and possibly reduce the need for large changes in joint angles. Improved sensing would help mitigate issues with the visual servoing.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we described an algorithmic framework and system design for robotic pruning which allows for flexible environment representation and collision modelling. We validated the system in end-to-end experiments and report promising results in terms of speed and accuracy.

Guided by lessons learned from these experiments, immediate avenues for future work include exploring the performance of 7-DOF manipulators and of a manipulator mounted on a gantry. We also plan to integrate a high-resolution stereo camera that can scan the entire tree, investigate autonomous branch selection, and evaluate the system in the field.

References

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

[2] 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*, vol. 31, no. 6, pp. 888–911, 2014.

[3] K. Gallardo, M. Taylor, and H. Hinman, "Cost estimates of establishing and producing gala apples in washington," *Washington State University*, 2009.

[4] F. A. Fathallah, "Musculoskeletal disorders in labor-intensive agriculture," *Applied Ergonomics*, vol. 41, no. 6, pp. 738–743, 2010.

[5] D. Ferree and A. Lakso, "Effect of selected dormant pruning techniques in a hedgerow apple orchard." *Journal American Society for Horticultural Science*, vol. 104, pp. 736–739, 1979.

[6] L. He and J. Schupp, "Sensing and automation in pruning of apple trees: A review," *Agronomy*, vol. 8, no. 10, p. 211, 2018.

[7] V. Robotics 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] F. Sukkar, "Fast, reliable and efficient database search motion planner (FREDS-MP) for repetitive manipulator tasks," Master's thesis, University of Technology Sydney, 2018.

[10] A. Silwal, J. R. Davidson, M. Karkee, C. Mo, Q. Zhang, and K. Lewis, "Design, integration, and field evaluation of a robotic apple harvester," *Journal of Field Robotics*, vol. 34, no. 6, pp. 1140–1159, 2017.

[11] C. W. Bac, J. Hemming, B. Van Tuijl, R. Barth, E. Wais, and E. J. van Henten, "Performance evaluation of a harvesting robot for sweet pepper," *Journal of Field Robotics*, vol. 34, no. 6, pp. 1123–1139, 2017.

[12] C. Lehnert, A. English, C. McCool, A. W. Tow, and T. Perez, "Autonomous sweet pepper harvesting for protected cropping systems," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 872–879, 2017.

[13] P. R. Soria, F. Sukkar, W. Martens, B. C. Arrue, and R. Fitch, "Multi-view probabilistic segmentation of pome fruit with a low-cost RGB-D camera," in *Proc. of ROBOT*, 2017, pp. 320–331.

[14] F. Sukkar, G. Best, C. Yoo, and R. Fitch, "Multi-robot region-of-interest reconstruction with Dec-MCTS," in *Proc. of IEEE ICRA*, 2019, pp. 9101–9107.

[15] Y. Zhao, L. Gong, Y. Huang, and C. Liu, "A review of key techniques of vision-based control for harvesting robot," *Computers and Electronics in Agriculture*, vol. 127, pp. 311–323, 2016.

[16] T. T. Andersen, "Optimizing the universal robots ros driver." Technical University of Denmark, Department of Electrical Engineering, Tech. Rep., 2015. [Online]. Available: http://orbit.dtu.dk/en/publications/optimizing-the-universal-robots-ros-driver(20dde139-7e87-4552-8658-dbf2cdaab24b).html

[17] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, 1982.

[18] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[19] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Proc. of RSS*, 2013.

[20] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[21] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proc. of IEEE ICRA*, 2015, pp. 3067–3074.