Real-time Stereo Visual Servoing for Rose Pruning with Robotic Arm

Citation for published version:

Digital Object Identifier (DOI):
10.1109/ICRA40945.2020.9197272

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
2020 IEEE International Conference on Robotics and Automation (ICRA)

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Real-time Stereo Visual Servoing for Rose Pruning with Robotic Arm

Hanz Cuevas-Velasquez1, Antonio-Javier Gallego2, Radim Tylecek1
Jochen Hemming3, Bart van Tuijl3, Angelo Mencarelli3 and Robert B. Fisher1

Abstract—The paper presents a working pipeline which integrates hardware and software in an automated robotic rose cutter. To the best of our knowledge, this is the first robot able to prune rose bushes in a natural environment. Unlike similar approaches like tree stem cutting, the proposed method does not require to scan the full plant, have multiple cameras around the bush, or assume that a stem does not move. It relies on a single stereo camera mounted on the end-effector of the robot and real-time visual servoing to navigate to the desired cutting location on the stem. The evaluation of the whole pipeline shows a good performance in a garden with unconstrained conditions, where finding and approaching a specific location on a stem is challenging due to occlusions caused by other stems and dynamic changes caused by the wind.

I. INTRODUCTION

Gardening is a repetitive and human-intensive task, however, it is difficult to automate given the challenges that it involves. These challenges are related to the unconstrained environment in a real garden. There are simple gardening tasks that have being automated like grass cutting [1], [2], [3], stationary fruit harvesting [4] and plant irrigation [5]. Although these tasks can be considered solved and highly commercialized, there are others that require more dexterity in terms of manipulation and approach.

One of these complex jobs is rose pruning. A rose bush require regular pruning in late winter or early spring to keep it healthy, aesthetic and allow it to grow new and strong roses. This job requires a person to cut stems according to some gardening rules and be able to approach the stems in different positions to cut them. As rose pruning is complex and demanding, its automation is useful and helpful in garden maintenance. The benefits go from small gardens at a house to big ones, e.g. national gardens.

There are very few previous works on rose pruning robots, with [6], [7] one of the few in the area. In these papers, they localize cutting points on a branch by using a relation between the diameter and the length of the branch. After that, they use position based servoing to reach the points. Unlike our approach, they cut only a single rose stem under a controlled environment with constant light and background.

A close approach to rose pruning is tree pruning [8], [9], [10], [11] is the closest approach to our work. Here, the authors model a grape vine using multiple stereo cameras that surround the plant to find cutting points on stems. This process is done in a controlled environment inside a box, which covers completely the whole plant from its surroundings, resulting in a constant illumination and background. They, where they assume the stems are static, our approach uses visual servoing to update constantly the target stem.

Rose pruning can be considered as lying in the research field of precision agriculture, together with bush trimming [12], and fruit harvesting [13]. In [12], they trim bushes based on 3D shape fitting using an eye-in-hand arm robot. They send all the cutting positions as programmed trajectories to the robot without allowing any re-planning while approaching the bush. In [13], they harvest tomatoes using an image based visual servoing approach. They find the position of the fruit based on their round shape and characteristic color. Further work on fruit harvesting can be found in [14], [15].

Rose pruning and fruit harvesting share the same principle: find key points where the robot should cut or pick a target. Fruit harvesting can be considered an easier task given that the targets usually have distinctive characteristics like color and shape, making them easier to recognize. To prune a rose bush, the robot has to find cutting points on a stem in a highly populated1 and small space where all the stems look alike and do not have any distinctive characteristic that shows the location of the cutting point.

Our task can be seen as a special case of grasping based on visual servoing. The difference to a normal grasping model is that these methods usually work with big objects with a defined body to grasp, whereas to prune a rose, the robot has to be really precise to fit a thin stem into the cutting tool (see Fig. 2).

To the best of our knowledge there is no robot able to prune a complete rose bush in a natural environment. To

1Highly populated by stems.
prune a rose bush aesthetically and keep it healthy, a robot has to cut its stems at a certain height from the ground. Therefore, the robot should be able to locate the stems and find where it should cut. Since the robot is supposed to prune roses under bright sunlight, active light sensors are not suitable, and laser scanners are expensive and limited by their coverage. Thus, RGB stereo cameras suit well in this task. Another important feature is robustness under changing conditions. The robot should be robust to track the cutting points when the stems are moved by the wind. Therefore, close-loop control based on vision is used to update the information while it approaches to the target.

Unlike normal grasping or fruit harvesting, having a proper end-effector is important for rose pruning because stems are thin and rose bushes are usually populated by many stems. In addition, the tool must be light enough to be carried by a small size mobile robot as end-effector. Currently there is no pruner that can work and interact with a robot on the market. For this reason, designing a tool capable of cutting thin stems under program control is important for the success of the process.

The paper presents a novel approach which is divided in the following steps.

1) Scan the rose bush based on a pre-planned robot arm trajectory to acquire a complete 3D description of the bush in the form of a 3D pointcloud.
2) Segment the stems using an encoder-decoder CNN to separate the stems from the background and leaves.
3) Evaluate the pointcloud and find cutting locations based on the desired height.
4) Navigate towards the target stem location updating in real-time the location of the current target.
5) If the end-effector reaches the desired target (stem), send a signal to the cutting tool to cut it.

These steps result in not only a good estimation of the cutting points but also a robust servoing under changing conditions without compromising real-time target update and execution. The navigation of the mobile robot is not part of this pipeline because we assume the robot is already positioned in front of the bush at cutting distance, and the surface around it is approximately planar and horizontal as in [16]. The proposed approach has the following contributions:

- **A novel rose pruning pipeline** based on stereo visual servoing and real time target update.
- **A real-time 2.5D servoing algorithm** using pointcloud and 2D images to locate stems and find cutting locations on them.
- **A dataset with more than 1200 labelled images of rose stems** used to train the encoder-decoder CNN to segment rose bushes.
- **A novel cutting tool design for rose clipping** that works with small/medium size arm robots to cut bushes with thin stems.

---

**II. ROBOT SETUP**

This section describes the setup of the robot. Fig. 1 shows an overview of the robot and its components. Although the figure shows a specific design, the proposed pipeline is more general and is not constrained to any type of robot or stereo camera, making it flexible for a re-implementation on different hardware.

**A. Robot arm**

The arm consists of a Kinova Jaco2 robot [17], a lightweight manipulator with lower power consumption mounted on a mobile robot; the details of the mobile robot base can be found in [16]. The 6 rotational joints provide the dexterity needed for the scanning and cutting actions. The arm has a spherical wrist configuration rather than a curved wrist. This configuration allows easier inverse kinematics (IK) computation and smoother movements than the usual curved wrist configuration.

**B. Clipping tool**

Market research started the design process of the clipping tool to find readymade solutions to cut stems in a cluttered environment. The commercially available Ciso pruner from Bosch was chosen as the base because of its simple on-off control system and the built in feedback mechanism, by means of position switches, to sense whether the knife is open or closed. The pruner is able to cut stems up to 14 mm in diameter and was mechanically modified in order to mount it on the Kinova robot arm (Fig. 2). Because of the ROS control architecture used for the main platform, a Maxon motor controller (Epos 2, 24/5, Maxon Motor AG, Switzerland) is used to drive the DC motor that opens and closes the jaws of the knife. An in-house modified version of the Maxon motor ROS wrapper epos_hardware [18], that allows the reading of the digital I/O of Epos 2, is used to control the motor and read out the state of the position switches.

**C. Stereo camera**

On the last joint of the arm, on top of the clipping tool, a custom stereo camera is mounted (Fig. 2). The camera module is mechanically configured to not obstruct the cutting action and to see as much of the scene as possible. The stereo camera has a resolution of $752 \times 480$ px, a diagonal $FoV \approx 68^\circ$ and a baseline $\approx 3$ cm. The stereo calibration was done using Kalibr [19]. The camera uses as its global coordinate the base frame of the robot. Figure 2 shows that the camera housing has two pairs of stereo cameras with different baselines, however, only the camera with the smallest baseline is used because the cutter needs to get close to the target.

**III. METHOD**

The proposed method is divided into 5 steps which can be seen in Fig. 5. The following sections will describe each step of the process individually.

---

2The dataset is at: http://trimbot2020.webhosting.rug.nl/resources/public-datasets/
C. Pointcloud post-processing

After segmenting the stems, each individual pointcloud is downsampled using a Voxel Grid filter of size 0.1 mm. Then, the pose of the pointclouds are transformed to the global frame (robot base) and merged by accumulating all the points. To make the cutting point localization process faster, the merged pointcloud is spatially subsampled and the noise removed. All the points that do not have at least 20 neighbors within a range of 0.5 mm are considered noise.

D. Cutting points localization

Here, the criterion to find the cutting point is based on the height of the plant. Those stems with height above \( h \) cm will become candidates to be pruned. This height varies depending on the type of the rose. The cutting points \((CP)\) are found by creating a virtual plane at \( h \) cm above the ground and finding all the points \( P_{pc} \) in the pointcloud \((pc)\) that are close to the plane (1).

\[
P_{pc} = \text{dist}(pc, \text{plane}) < 1.5 \text{ cm}
\]  

(1)

where:

\[
\text{dist}(pc, \text{plane}) = \left| \frac{\hat{n}_{\text{plane}}}{\|\hat{n}_{\text{plane}}\|} \cdot (pc - \bar{P}_{\text{plane}}) \right|
\]  

(2)

Equation 1 outputs all the points \( P_{pc} \) in the bush that are 1.5 cm or closer to the desired height \( h \).

The points \( P_{pc} \) are clustered to find the parts of the plant where the cutting points are located. These points are clustered based on distance using DBSCAN (Density Based Spatial Clustering of Applications with Noise) [23]. DBSCAN is a density based clustering method, known to work well with groups that are closely packed together and where the number of clusters is not known beforehand. A group of points is considered a cluster if the distance between them is less than a threshold \( d_{\text{cluster}} \) and the quantity of points existing below this threshold is higher than a minimum \( \text{min}_p \). A minimum number of points \( \text{min}_p \) of 40 and a \( d_{\text{cluster}} = 0.8 \) cm was used in our configuration (these values were found empirically).

A simple approach would choose the center of gravity of those clusters as cutting points, however, it might lead to false positives. This could happen for two main reasons. First, the points inside a cluster might belong to a branching part of the plant. Second, two stems that were really close might get clustered together. In both cases, the cutting point would be wrongly located between the middle of two stems. This problem is tackled by dividing the clusters into horizontal slices (7 in our implementation) and clustering each slice with a distance threshold of \( d_{\text{cluster}} = 0.5 \) cm and minimum number of points of 5. In this way, thinner branches in the cluster can be found. This second clustering will be called sub-cluster. The cutting points are found by evaluating these sub-clusters in the following way:

- If at least 4 of the upper slices have two sub-clusters, this means that the cluster is located in a branching section of the plant, therefore, two cutting points are
generated. These two cutting points are the center of gravity of the two sub-clusters that have the farthest distance between each other (yellow slice in Fig. 4a). This criteria aims to have two points in the cluster that are far from each other so the cutting tool does not get stuck between both stems.

- If all the upper slices are divided but there are at least 4 slices at the bottom part of the cluster that have not been divided, the center of the bottom slice is chosen as a cutting point (pink slice in Fig. 4b).
- If all the slices have one sub-cluster, the center of gravity of the middle slice is chosen as cutting point.

Note that because DBSCAN relies on the distance between points to form a cluster, it usually groups the points of a thorn as part of the branch it belongs to; this is because the size of the thorns are small (∼1.5 cm) and do not tend to grow far from the branch.

E. Visual servoing

The robot starts the servoing by rotating the end-effector 45° for sideways cut following gardening rules. The arm navigates to the stored cutting locations, starting with the closest cutting point and ending with the farthest one. While it navigates to the current cutting goal \( \vec{P}_{goal} \), the visual servoing looks for new cutting point candidates \( \vec{P}_{can} \) in a neighborhood of radius 0.15 cm around \( \vec{P}_{goal} \). If there is any cutting point candidate in this radius, the goal gets updated using a convex combination (3) between the current goal position and the new goal. The convex combination is weighted by a “blending” factor \( \alpha \in (0, 1] \) with \( \alpha = 0.1 \) in our implementation. This combination guarantees a smoother update of \( \vec{P}_{goal} \) by avoiding fast changes in position between consecutive times \( (t, t + 1) \), where \( \vec{P}_{goal}(t + 1) \) is the new goal and \( \vec{P}_{goal}(t) \) is the current goal.

\[
\vec{P}_{goal}(t + 1) = \alpha \vec{P}_{can}(t) + (1 - \alpha) \vec{P}_{goal}(t) \tag{3}
\]

The neighbors of the cutting goal are found by running a process in parallel which captures the position of the arm and pointcloud at the current time \( t \), and outputs the positions of the cutting points on the scene using the methods from Section III-C, III-B and III-D (stem detection, Pointcloud post-processing, Cutting points localization). This process can be better appreciated in Fig. 5.

The navigation of the arm, from the start position to a cutting point, is performed using proportional velocity control. ROS MoveIt! software [24] is used to find the inverse of the Jacobian \( J^\dagger \) to obtain the joint difference \( \Delta \vec{q} \) from the distance \( \Delta X \); the \( \Delta X \) is the distance between the end-effector of the robot and the target \( \vec{P}_{goal} \). For the approach, a proportional controller is enough to have a smooth trajectory. The proportional value \( K \) is not a constant but a dynamic value that changes based on \( \Delta X \) because the robot should "decelerate" when it gets close to the cutting location and
Fig. 4: Clustering and slicing approach to find cutting points. The first column shows the slices (colored) found inside a cluster. The second column shows the resulting cutting points after evaluating the slices. The red circles indicate the chosen cutting points and the blue circles the center of gravity of the cluster. (a) Shows the case where 6 out of 7 slices in the cluster can be split in two, generating, in this way, two cutting points. (b) Shows the case where the 4 bottom slices of the cluster are not divided, which creates a cutting point in the bottom slice.

Fig. 5: Pipeline for cutting roses.

should increase the velocity when the end-effector is far from the target. However, in practice, it is not desirable that only the distance between the end-effector and the target controls the value of $K$, specially because if $\Delta X$ is big, $\dot{q}$ will have an undesirable high speed. Thus, a maximum velocity must be set to avoid this. Similarly, a lower bound is set to avoid the velocity becoming 0 when the end-effector gets really close to the target stem. Equation 4 shows how the velocity of a joint $q_i$ is calculated.

$$\dot{q_i} = \begin{cases} 
10 \left[ \frac{deg}{s} \right] & \text{if } K\Delta q_i > 10 \left[ \frac{deg}{s} \right] \\
3 \left[ \frac{deg}{s} \right] & \text{if } K\Delta q_i < 3 \left[ \frac{deg}{s} \right] \\
K\Delta q_i & \text{otherwise}
\end{cases}$$

IV. EXPERIMENTS AND EVALUATION

The pruning pipeline was tested using several rose bushes. This includes rose stems placed inside a pot and a real bush in a garden. The thickness of the stems ranged between 0.6 to 1.0 cm. All the tests were performed outdoors in a garden, meaning that the system was tested in an uncontrolled environment. A sample of the plants used in the evaluation is shown in Fig. 6 and the result of the rose pruning in Fig. 7.

The system was tested using a Razer Blade 14 i7 with 8 cores and a GTX 1060 Nvidia GPU. The connection between the software and hardware was done through the Robot Operating System (ROS) [25].

<table>
<thead>
<tr>
<th>Fold</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.8482</td>
<td>0.8228</td>
<td>0.8353</td>
</tr>
<tr>
<td>1</td>
<td>0.8120</td>
<td>0.8224</td>
<td>0.8171</td>
</tr>
<tr>
<td>2</td>
<td>0.8166</td>
<td>0.8265</td>
<td>0.8215</td>
</tr>
</tbody>
</table>

Macro Avg. 0.8266 ± 0.020 0.8239 ± 0.002 0.8246 ± 0.010

TABLE I: Pixel-wise branch detector results of the different folds and macro averages.

A. Branch segmentation evaluation

The branch segmentation CNN was trained and evaluated using our new dataset of rose bushes which consist of 1360 manually labelled images, each image with a size of 752 × 480 px. A sample of the dataset can be seen in Fig. 8.
The architecture of the branch segmentation CNN is shown in Table II. The performance of the network was evaluated using k-fold cross validation with $k = 3$, $F_1$ score for each fold and macro $F_1$ for the whole network. Table I and Fig. 3a show the results of the branch segmentation.

<table>
<thead>
<tr>
<th>Input image size:</th>
<th>480×320 px</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers:</td>
<td>444</td>
</tr>
<tr>
<td>Filters per layer:</td>
<td>128</td>
</tr>
<tr>
<td>Kernel size:</td>
<td>5×5</td>
</tr>
<tr>
<td>Normalization type:</td>
<td>Standard</td>
</tr>
<tr>
<td>Data augmentation:</td>
<td>10 %</td>
</tr>
</tbody>
</table>

**TABLE II:** Branch segmentation CNN architecture.

### B. Evaluation of cutting point detection from scanning

The accuracy of the scanning was measured by comparing the total number of targets found after scanning a bush and processing it (Section III-C, III-B and III-D) and the real number of targets. This value is also compared against the number of targets found by considering only the data captured by a single pose. The number of cutting points found by a single view was obtained by counting the total cutting locations found for each view and averaging them together. The real targets are the stems that exceed the desired cutting height $h$. The heights used for the evaluation are 10, 15, 20, 25, and 30 cm. The data is taken by moving the robotic arm in a square trajectory of 20 cm and stopping at 18 locations. The data was captured using 19 different bushes, each bush containing 3 to 4 cutting locations. The total number of cutting locations was 60. Table III shows that the fused and segmented pointcloud, after scanning the bush, led to a detection accuracy of 90% (true positive rate). However, if only one single pose (one single view) is used to find the cutting locations, the accuracy drops to 30%. The cutting points that were difficult to find were usually those that belong to the stems at the back of the bush. It was caused mostly by the dense population of stems and leaves covering them.

<table>
<thead>
<tr>
<th>Fused views from scanning</th>
<th>Detected cutting points</th>
<th>Detection accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>54</td>
<td>0.90</td>
</tr>
<tr>
<td>Single view (average)</td>
<td>17.89</td>
<td>0.30</td>
</tr>
</tbody>
</table>

**TABLE III:** Cutting point detection accuracy of targets found (out of 60) after scanning and post-processing the pointcloud, and the average cutting points found by a single view.

### C. Visual servoing navigation quality

The quality of the visual servoing system was evaluated by letting the robot navigate towards the cutting locations after the scanning. This result was compared against a blind navigation. The blind navigation gives the location of the targets to the planner and moves the manipulator towards them without updating the target position. A navigation is considered successful if the robot reaches the target location and the target stem gets into the cutter. The total number of cutting locations found by the robot after scanning the bush were 54 out of 60 real cutting locations. Therefore, the evaluation of the visual servoing process was done using only the 54 locations found by the scanning process. Table IV shows that blind navigation is not sufficient to drive the end-effector to the target location. In practice, the stem randomly moves up to 2 cm sideways due to the combined effect of external forces (like wind) and the interaction of the end-effector with other connected parts of the bush. This makes the end-effector usually end up on one side of the stem. On the other hand, our visual servoing approach is robust enough to make the robot navigate and reach the cutting points 94% of the time under dynamic conditions.

<table>
<thead>
<tr>
<th></th>
<th>Reached points</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual servoing</td>
<td>51</td>
<td>0.94</td>
</tr>
<tr>
<td>Blind navigation</td>
<td>27</td>
<td>0.5</td>
</tr>
</tbody>
</table>

**TABLE IV:** Visual servoing and blind navigation results out of 54 detected cutting targets.

### V. CONCLUSIONS

The presented approach was designed to effectively cut rose bushes in a garden in an unconstrained outdoor environment with dynamic targets. The experimental evaluation shows that the neural network is capable of segmenting the stems of rose bushes from the background, even when the background and the stems have similar color. This result also demonstrates that the large dataset introduced in the paper can indeed be used to successfully train a neural network to segment branches of different type of roses.

The proposed target localization approach, which consists of the combined process of stem detection, clustering and pointcloud merging can successfully find the cutting points 90% of the times. These targets are found even when they are occluded by other stems or leaves. This approach also proves to be robust but fast enough to be used by the visual servoing to update the target location on the fly.

The visual feedback is a key element to navigate in a garden where the wind can change the position of the stems, thus change the location of a target. The proposed visual servoing performs a good navigation with an accuracy of 94%. The combination of these steps results in a pipeline capable of finding cutting points in stems that are occluded by other stems or leaves and navigating towards them successfully in ~ 12 secs with an average initial distance between the center of the cutting tool and a target stem of 0.6 m.

Scanning the rose bush in a square path is a simple yet effective to capture the structure of the plant. Different scanning methods can be done to improve the scanned bush model, like having different poses instead of a square shape or scanning the bush by navigating around it, however this will lead to further problems like localization and drifting.

This work is a significant step towards an automated robotic rose pruning system with visual input as close-loop feedback able to work in a real environment. Future work will consider more gardening rules to prune rose bushes, like cutting the stems above the bud eyes, crossing inward branches and dead branches.
REFERENCES


