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

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

Hanz Cuevas-Velasquez[1], Antonio-Javier Gallego[2], Radim Tylecek[1]
Jochen Hemming[3], Bart van Tuijl[3], Angelo Mencarelli[3] and Robert B. Fisher[1]

***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.

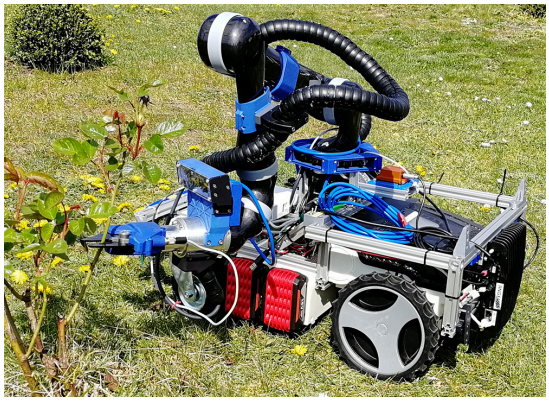


Fig. 1: Overview photo of the rose cutter robot.

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 populated[1] 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

*Authors acknowledge the support of EU project TrimBot2020.

[1]School of Informatics, University of Edinburgh, Edinburgh, EH8 9AB, UK; [2]Department of Software and Computing Systems, University of Alicante, 03690 Alicante, Spain. [3]Wageningen University & Research, Greenhouse Horticulture, P.O. Box 644, 6700 AP, Wageningen, The Netherlands. Contact: hanz.c.v@ed.ac.uk

[1]Highly 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[2].
- **A novel cutting tool design for rose clipping** that works with small/medium size arm robots to cut bushes with thin stems.

[2]The dataset is at: `http://trimbot2020.webhosting.rug.nl/resources/public-datasets/`

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 light-weight 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_harware* [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 752×480 px, a diagonal $FoV \approx 68^\circ$ and a baseline ≈ 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.

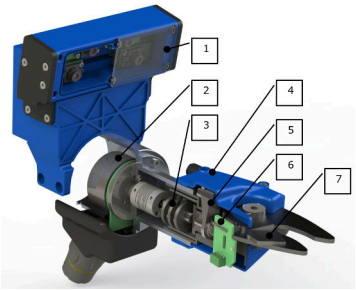


Fig. 2: 3D design of the clipping tool. 1: 2 pairs of cameras in a 3D printed housing, 2: DC motor, 3: gear box, 4: protective cover, 5: drive lever of cutting blade, 6: position feedback switches, 7: cutting blade.

A. Scanning

The pipeline starts by scanning a rose bush in a predefined square shape trajectory, making a short pause at given poses to avoid motion blur when recording an image pair. The square trajectory has a size of 20×20 cm with a center located at the cutting height h. This is the starting position of the arm. The end-effector of the arm is ~ 0.6 m away from the bush. A sample of the images captured in each pose can be seen in Fig. 3a. The scanning is performed because a single viewpoint can only provide limited information due to complex occlusions between stems. Also, a rose bush is usually too big to be captured in a single shot by an eye-in-hand camera. One can argue that the robot can be placed far away from the plant to capture the whole bush at once, however, the farther it gets, the more difficult it is for a stereo matching algorithm to find the disparity of a single stem.

B. Stem detection

For each pose in the trajectory, a color image from the left camera and a disparity map are obtained. The disparity maps are computed by using Block Matching Stereo (BM). Although BM is not as accurate as state-of-the-art methods like Semi Global Block Matching (SGBM) or DispNet [20], it is a faster approach. BM obtains the disparity maps in real-time, whereas SGBM runs at ~ 4 FPS and DispNet at ~ 2 FPS[3].

The image from the left camera is used as input to an encoder-decoder CNN with residual connections to segment the stems from the background, similar to [21]. The network outputs a binary image where it assigns a value of 1 to all the pixels that form part of a stem and 0 to the background. This network outperforms most of the state-of-the-art segmentations for the branch segmentation task [22]. The binary output is used to mask the disparity image to obtain only the disparities of the stems. This masked disparity is then converted into a pointcloud. Figure 3c shows the pointcloud of the bush after performing the segmentation and post-processing.

[3]Under our setup (see Section IV).

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).

$$P_{pc} = dist(pc, plane) < 1.5\,cm \tag{1}$$

where:

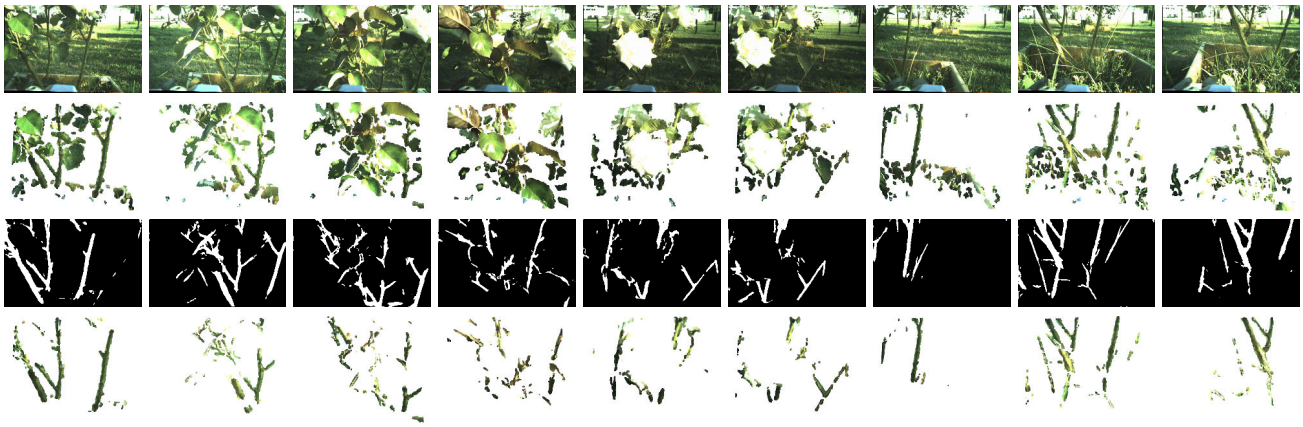
$$dist(pc, plane) = \left| \frac{\vec{n}_{plane}}{\|\vec{n}_{plane}\|} \cdot \left(pc - \vec{P}_{plane} \right) \right| \tag{2}$$

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

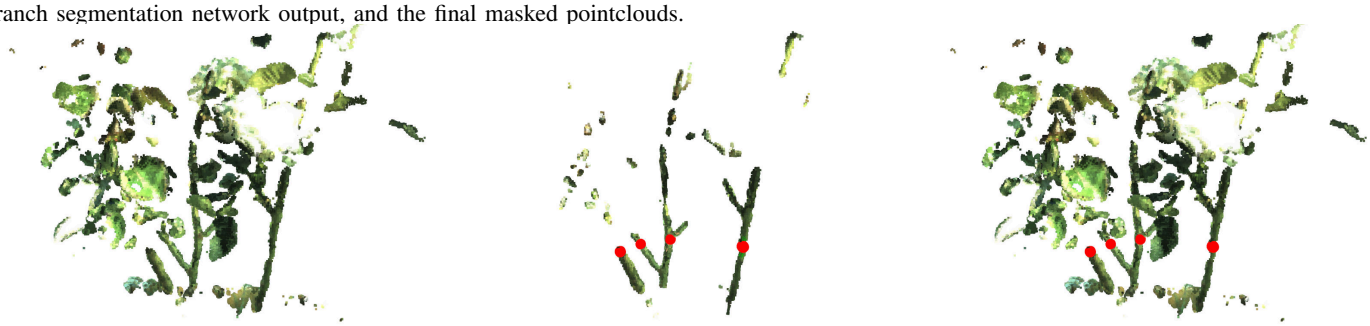
The points P_{pc} are clustered to find the parts of the plant where the cutting points are located. These points are clustered based on distance using DBSCAN (Density Based Spatial Clustering of Applications with Noise) [23]. DBSCAN is a density based clustering method, known to work well with groups 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 $d_{cluster}$ and the quantity of points existing below this threshold is higher than a minimum min_p. A minimum number of points min_p of 40 and a $d_{cluster} = 0.8$ 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 $d_{cluster} = 0.5$ 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



(a) Scanning trajectory at 9 stopping poses. From top to bottom, color images captured by the left camera, pointclouds of the rose bush, branch segmentation network output, and the final masked pointclouds.



(b) Merged pointcloud without branch segmentation.

(c) Merged pointcloud after segmenting the stems with cutting points localized.

(d) Cutting points on the original pointcloud of the bush.

Fig. 3: Scanning trajectory and pointcloud of the rose bush with cutting points (red dots).

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.

$$\vec{P}_{goal}(t+1) = \alpha \vec{P}_{can}(t) + (1-\alpha)\vec{P}_{goal}(t) \qquad (3)$$

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 Δq from the distance ΔX; the ΔX 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 ΔX because the robot should "decelerate" when it gets close to the cutting location and

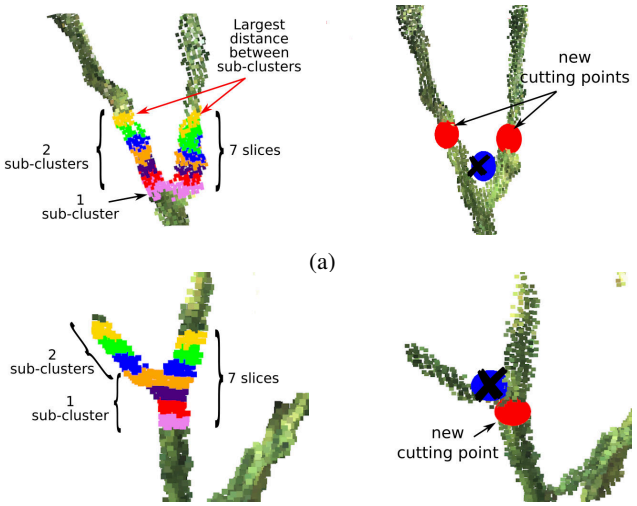


Fig. 4: Clustering and slicing approach to find cutting points. The first column shows the slices (colored) found inside a cluster. The second column shows the resulting cutting points after evaluating the slices. The red circles indicate the chosen cutting points and the blue circles the center of gravity 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.

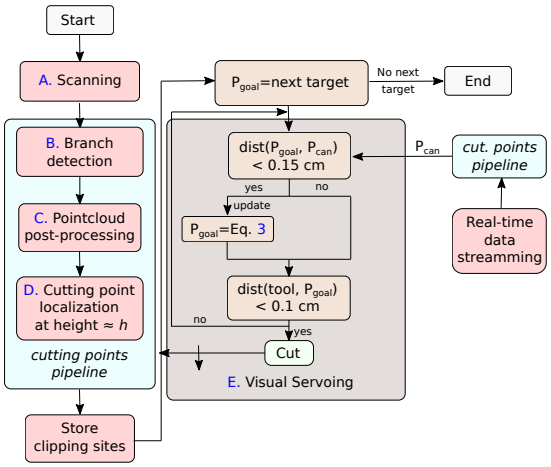


Fig. 5: Pipeline for cutting roses.

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 ΔX 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.

$$\dot{q}_i = \begin{cases} 10[\frac{deg}{s}] & if\ K\Delta q_i > 10[\frac{deg}{s}] \\ 3[\frac{deg}{s}] & if\ K\Delta q_i < 3[\frac{deg}{s}] \\ K\Delta q_i & otherwise \end{cases} \quad (4)$$

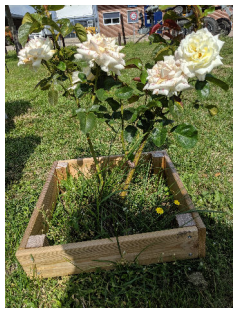


Fig. 6: A sample of the rose bushes used in the evaluation.

Fig. 7: Rose bush after pruning. Figure 6 shows the plant before being cut.

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].

Fold	Precision	Recall	F_1
0	0.8482	0.8228	0.8353
1	0.8120	0.8224	0.8171
2	0.8166	0.8265	0.8215
Macro Avg.	0.8256 ± 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.

Fig. 8: A sample of the dataset collected to train the network.

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.

Input image size:	480×320 px
Number of layers:	4+4
Filters per layer:	128
Kernel size:	5×5
Normalization type:	Standard
Data augmentation:	10 %

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.

	Detected cutting points	Detection accuracy
Fused views from scanning	**54**	**0.90**
Single view (average)	17.89	0.30

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.

	Reached points	Accuracy
Visual servoing	**51**	**0.94**
Blind navigation	27	0.5

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

[1] L. Bangyu and X. Fang, "Outdoor grass target recognition based on gabor filter," in *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 2017, pp. 432–435.

[2] A. Adeodu, I. Daniyan, T. Ebimoghan, and S. Akinola, "Development of an embedded obstacle avoidance and path planning autonomous solar grass cutting robot for semi-structured outdoor environment," in *2018 IEEE 7th International Conference on Adaptive Science & Technology (ICAST)*. IEEE, 2018, pp. 1–11.

[3] Bosch, accessed on 19/07/2019. [online]. Available: https://www.bosch-garden.com/gb/en/garden-tools/indego-home.jsp.

[4] Agrobot, accessed on 19/07/2019. [online]. Available: http://agrobot.com/.

[5] R. H. Rafi, S. Das, N. Ahmed, I. Hossain, and S. T. Reza, "Design & implementation of a line following robot for irrigation based application," in *2016 19th International Conference on Computer and Information Technology (ICCIT)*. IEEE, 2016, pp. 480–483.

[6] C. Gürel, M. H. G. Zadeh, and A. Erden, "Rose stem branch point detection and cutting point location for rose harvesting robot," in *The 17th International Conference on Machine Design and Production, UMTIK 2016*, 2016.

[7] C. GÜrel, M. H. G. Zadeh, and A. Erden, "Development and implementation of rose stem tracing using a stereo vision camera system for rose harvesting robot," in *8th International Conference on Image Processing, Wavelet and Applications (IWW 2016)*, 2016.

[8] S. Paulin, T. Botterill, J. Lin, X. Chen, and R. Green, "A comparison of sampling-based path planners for a grape vine pruning robot arm," in *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*. IEEE, 2015, pp. 98–103.

[9] S. Paulin, T. Botterill, X. Chen, and R. Green, "A specialised collision detector for grape vines."

[10] T. Botterill, R. Green, and S. Mills, "Finding a vine's structure by bottom-up parsing of cane edges," in *2013 28th International Conference on Image and Vision Computing New Zealand (IVCNZ 2013)*. IEEE, 2013, pp. 112–117.

[11] 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.

[12] D. Kaljaca, N. Mayer, B. Vroegindeweij, A. Mencarelli, E. van Henten, and T. Brox, "Automated boxwood topiary trimming with a robotic arm and integrated stereo vision," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[13] N. Correll, N. Arechiga, A. Bolger, M. Bollini, B. Charrow, A. Clayton, F. Dominguez, K. Donahue, S. Dyar, L. Johnson, *et al.*, "Indoor robot gardening: design and implementation," *Intelligent Service Robotics*, vol. 3, no. 4, pp. 219–232, 2010.

[14] P. Li, S.-h. Lee, and H.-Y. Hsu, "Review on fruit harvesting method for potential use of automatic fruit harvesting systems," *Procedia Engineering*, vol. 23, pp. 351–366, 2011.

[15] 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.

[16] N. Strisciuglio, R. Tylecek, N. Petkov, P. Bieber, J. Hemming, E. van Henten, T. Sattler, M. Pollefeys, T. Gevers, T. Brox, and R. B. Fisher, "Trimbot2020: an outdoor robot for automatic gardening," in *50th International Symposium on Robotics*. VDE Verlag GmbH - Berlin - Offenbach, 2018.

[17] K. Robotics, accessed on 19/07/2019. [online]. Available: http://www.kinovarobotics.com.

[18] M. Wills, accessed on 19/07/2019. [online]. Available: http://wiki.ros.org/epos_hardware.

[19] J. Maye, P. Furgale, and R. Siegwart, "Self-supervised calibration for robotic systems," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 473–480.

[20] E. Ilg, T. Saikia, M. Keuper, and T. Brox, "Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 614–630.

[21] J. Calvo-Zaragoza and A.-J. Gallego, "A selectional auto-encoder approach for document image binarization," *Pattern Recognition*, vol. 86, pp. 37 – 47, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320318303091

[22] H. Cuevas-Velasquez, A.-J. Gallego, and R. B. Fisher, "Segmentation and 3d reconstruction of rose plants from stereoscopic images," *Computers and electronics in agriculture*, 2020.

[23] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise."

[24] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics Automation Magazine - IEEE ROBOT AUTOMAT*, vol. 19, pp. 18–19, 03 2012.

[25] M. Quigley, J. Faust, T. Foote, and J. Leibs, "Ros: an open-source robot operating system."