TrimBot2020 Deliverable 5.4

Clipping site recognition software

Principal Author: UEDIN
Contributors: UEDIN
Dissemination: PU

Abstract: The report describes the software for detecting clipping sites on a rose bush, which can be used to prune it. We discuss several options to implement rules of rose pruning, including branching and buds.
The final software that was integrated on the prototype consists of a pipeline that starts by scanning the rose bush to collect depth images of the bush and color information from different viewpoints. To locate the branches, a CNN is trained to segment them from the rest of the scene using the RGB data. Once the branches are segmented, their 3D locations are obtained using the point cloud from the depth map and the points that are located at the desired cutting height are found. Finally, those points are clustered to find the cutting points.

Deliverable due: Month 42

Contents

Figure 1: General overview of the robot.

1 Introduction

This deliverable describes the software developed for the rose pruning pipeline focusing on the clipping site recognition. Section 2 gives a brief introduction to the robot hardware for which the clipping site software was developed. The next sections describe each component of the software from scanning the plant to the navigation towards each site using continuous visual feedback.

The clipping site detection was based on the following general gardening rules:

1. Cut the branches at a certain height. The height depends on the type of branch.
2. Locate the buds that are pointing outwards and cut the branches approximately 3 to 4 millimeters above them in a 45° angle wrt. the stem.
3. Remove dead and weak stems from the base of the bush. These stems are usually thinner than the rest.

The software is available in the project repository: `https://gitlab.inf.ed.ac.uk/TrimBot2020/Trimming/tree/devel/RoseClipping`.

2 Robot description

The setup of the robot consist of a Kinova Jaco2 arm mounted on a mobile robot (Figure 1). The end-effector of the arm is a clipping tool design specifically for cutting rose stems. On top of the end-effector, a housing containing a pair of stereo cameras, each one with different baseline, is mounted. The left camera of the stereo pair captures color images whereas the right camera captures gray scale.

More details of the robot setup can be found in the deliverables D1.3 and D2.6. The software solution presented in the following sections is based on this hardware design.

Figure 2: Left, right and disparity map obtained by the stereo camera.

3 Scanning of the bush shape

After the robot gets in front of the rose bush, it proceeds to scan the plant 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 of $h = 30$ cm. The scanning captures the point cloud of the bush and the pose of the camera with respect to the base of the robot.

The depth information of the point clouds was obtained using the camera parameters and the disparity map computed by Block Matching Stereo algorithm (BM) with a maximum disparity of 192 pixels. We used the algorithm implemented in the ROS package $stereo\_image\_proc$. Although BM is not as accurate as state-of-the-art methods like Semi Global Block Matching (SGBM) or DispNet [3], it is a faster approach. BM obtains the disparity maps in real-time, whereas SGBM[1] runs at ~ 4 FPS and DispNet[2] at ~ 2 FPS. A sample of the images captured in each pose can be seen in Figure 3. An example of the images captured by the stereo camera and the disparity map is observed in Figure 2.

The scanning is performed because a single viewpoint only provides 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. After some experiments, we found that the best starting distance between the end-effector and the bush is ~ 0.6 m. In this way, the arm is far enough to avoid any collision with the bush and is close enough so the stereo matching algorithm can estimate the disparity of the stems.

4 Stem detection

4.1 Stem segmentation

After collecting the data from the different poses, we proceed to segment the branches from the rest of the scene. To do this task, an encoder-decoder CNN with residual connections similar to [1] was developed. The branch segmentation CNN receives as input an RGB image which is captured by the left camera. Its output is a probability map at pixel level where it indicates the presence of a branch. A *selectional* threshold δ of 0.3 is applied to the output to determine (select) whether a pixel belongs to a branch or not; the complete architecture of the CNN is shown in Table 1.

[1] based on OpenCV implementation
[2] based on version described in D5.2

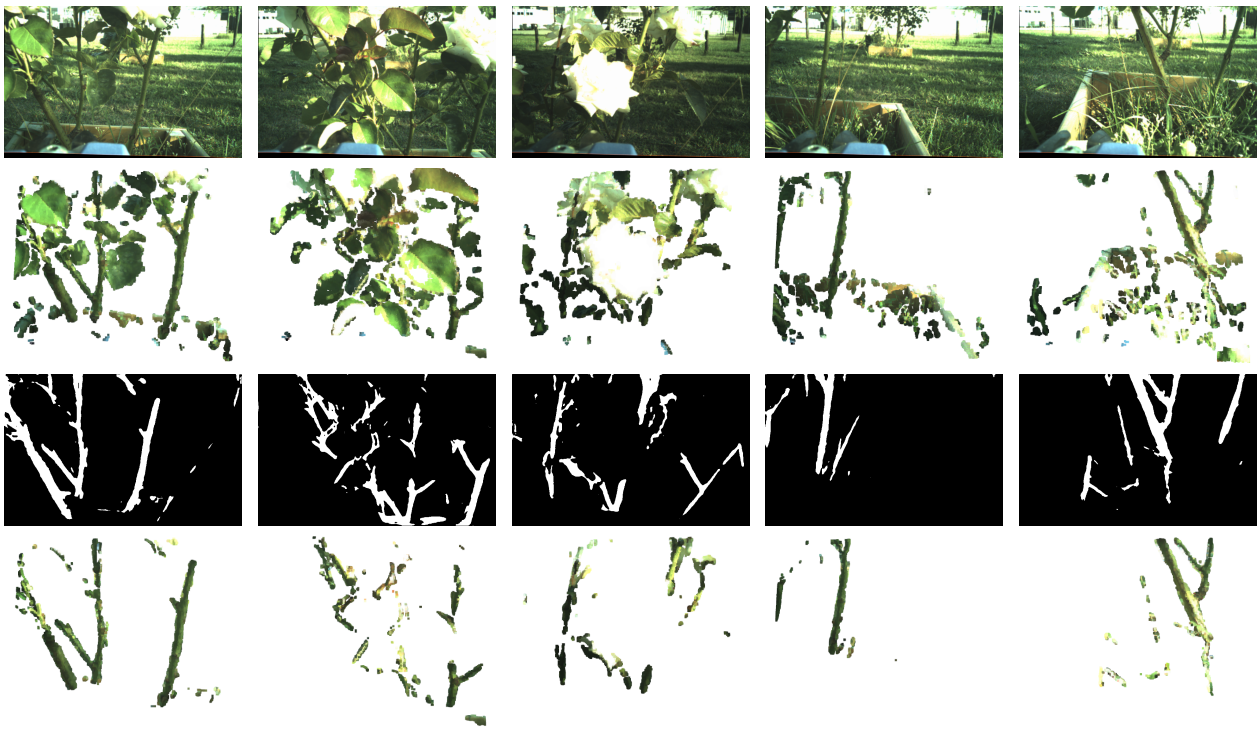

Figure 3: Scanning trajectory at 5 stopping poses. From top to bottom, color images captured by the left camera, point clouds of the rose bush, branch segmentation network output, and the final masked point clouds.

Figure 4: A sample of the dataset collected to train the network.

To train the network we created a new dataset that consists of 1360 manually labeled images of rose stems, each one with a size of 752×480 px. A sample of the dataset can be seen in Fig. 4. 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. The evaluation is done at a pixel-wise level. Table 2 and Figure 3 show the results of the branch segmentation.

4.2 Plant morphology

The segmentation network provides a binary output that indicates whether or not a stem is located in the image. This information, combined with the disparity map, is used to find the morphology of the plant, more specifically, the branching structure of the rose bush in 3D. Firstly, morphological thinning is used to extract the skeleton of the binary image obtained by the network. We use 2D thinning for two reasons, the method is fast enough to be used in

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 1: Branch segmentation CNN architecture.

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

real-time and because we exploit the basic principle of thinning, *a thinning algorithm reduces the components of a binary image to single-pixel thickness lines*, to find the branching structure. This means that, if a pixel belongs to a branch, it should have at most two neighbors. If it has more than two neighbors, it should be considered as a branching node. After obtaining the skeleton and the branching structure, the disparity map and camera parameters are used to get the 3D morphology of the plant. The results can be appreciated in Figure 5.

With the 2D skeleton and segmentation of the bush, we can also find the diameter of the branches. First, we transform the skeleton into a set of lines. We do this by using the probabilistic implementation of the Hough transform [5], allowing a small gap (3 pixels in our implementation) between pixels to create the lines. We also set the threshold parameter to 10 (minimum number of intersecting points to detect a line) and the minimum segment length to 3 pixels in order to better adjust the lines according to the curvature of the branch. Once the line is obtained, we calculate a perpendicular line to each Hough line. These perpendicular lines start and finish at the boundaries of the segmentation image (see Figure 6).

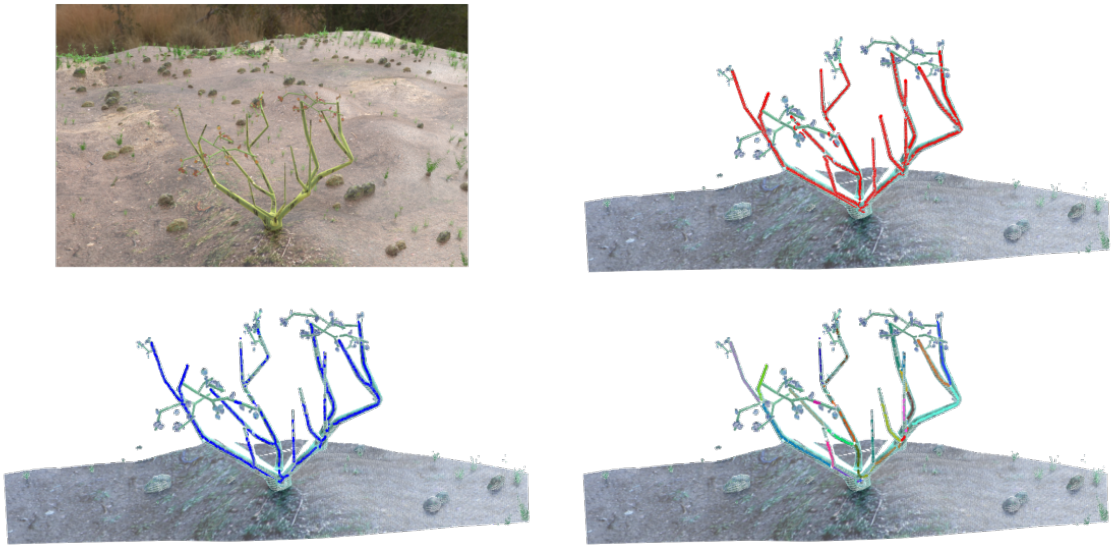

Figure 5: Output example of the 3D skeleton obtained by our method (blue) and the ground truth skeleton (red), both superimposed over the point cloud of the plant. The branches found are also shown at the bottom right image (each branch is marked using different colors). The image at the top left corner is the color input.

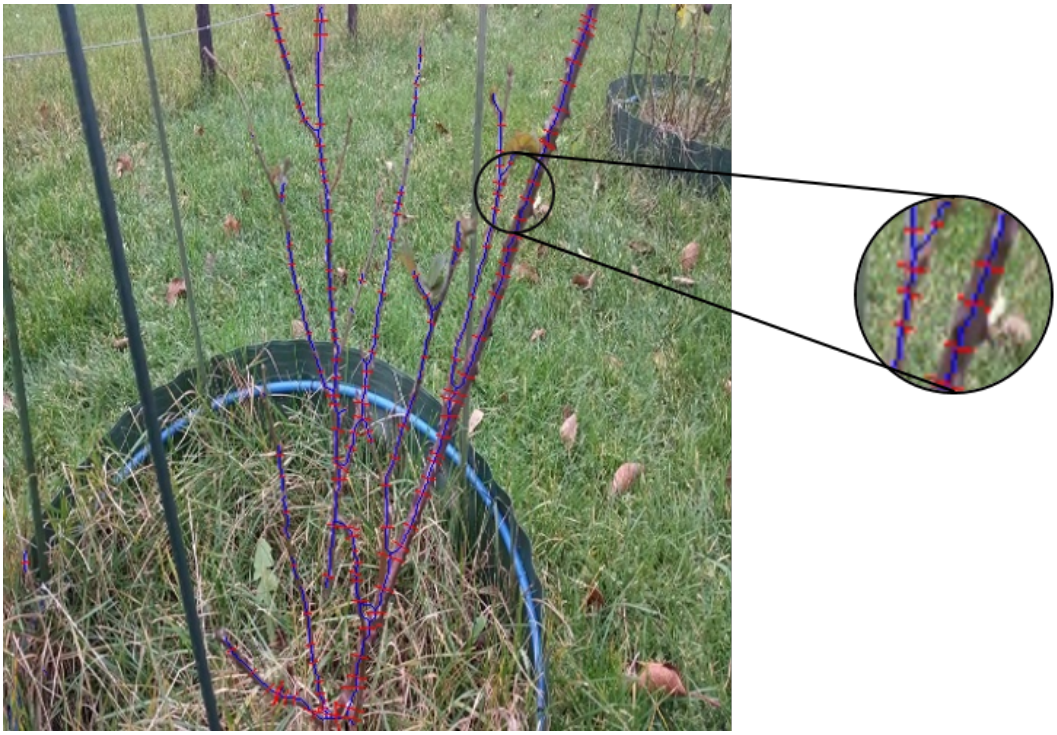

Figure 6: Example of a branch with its skeleton (blue) and diameter (red).

5 Clipping site recognition

The morphology of the plant obtained in the previous section can be used to find specific branches, depending on their width and growing direction, and cut those that are thin and pointing inwards (according to the gardening rules). However, this is constrained by the quality of the disparity map. In practice, this involves a refinement of the disparity and post-processing which are expensive to compute and cannot be used in a visual servoing system, which requires a real-time implementation to work. On the other hand, computing the width of the branches also slowed down the process. Therefore, the whole morphology process was not used in the final pipeline of the project. We also relaxed the gardening rules by only pruning the branches that exceed a certain height.

5.1 Clipping site at a given height

To find the cutting points at a certain height, we use the scanning from Section 3 to collect the point clouds of the rose bush from different poses. Then, the color information is extracted from the point cloud and fed to the branch segmentation CNN to segment the stems from the rest of the scene. The output of the CNN is used to mask the depth values of the point cloud and keep only the points of the stems, as seen in Figure 3. After segmenting the stems, each point cloud is down-sampled using a voxel grid filter with resolution of 0.1 mm. As the next step, the poses of the point clouds 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 point cloud is spatially sub-sampled 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. With this process, we can obtain a 3D model of the bush containing only the segmented branches as seen in Figure 7.

After obtaining the 3D model of the bush with segmented branches, we consider the stems with height above h cm as candidates to be pruned. This height varies depending on the type of 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 point cloud (pc) that are close to the plane in

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

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) [2]. DBSCAN is a non-parametric 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). An example of the localized cutting points can be seen in Figure 7b.

Finally, we use SVD to extract the eigenvector with the highest eigenvalue of each cluster. This eigenvector was used to obtain the tilting angle of the target branch by measuring the

(a) Merged point cloud without branch segmentation.

(b) Merged point cloud after segmenting the stems with cutting points localized.

(c) Cutting points on the original point cloud of the bush.

Figure 7: Scanning trajectory and point cloud of the rose bush with cutting points (red dots).

Figure 8: *Left:* bud detector inference results (blue) on dataset compared with the ground truth (red) on a rectified image from the arm camera. *Right:* No buds detected because the camera is too far.

angular distance between the vector and the plane we fit. Knowing this angle also allows us to rotate the arm $45°$ from the axis of the branch to cut it.

5.2 Bud detection

After locating the branches that exceed the desired height, we should find the nearby buds that are pointing outwards and cut the branches above them. To find the buds, a pre-trained CNN detector called YOLO v3 [4] was used. We fine-tuned the network by replacing the last layer with a new output, which consists of the coordinates of a bounding box and the probability of containing a bud-eye. The network was trained using 350 images and tested on 150. The images in the dataset contain rose bushes captured from different perspectives, light variation and in a natural environment. The performance was measured using the AP metric with two different IoU thresholds. An IoU of 10% gives an AP of 74.45% whereas an IoU of 50% returns an AP of 55.62%. Figure 8 show an example of the output.

The evaluation shows that the performance of the bud detector on the dataset is acceptable. In practice, however, the quality of the stereo camera used in the robot setup and the small size of the buds require that the camera be at least 20 cm close to the bush to find most of the buds. These constraints make it difficult to be implemented in the designed robot configuration

because the tool-tip of the end-effector is already ca. 15 cm from the camera. Therefore, the scanning at 20 cm is not feasible because the end-effector would collide with some parts of the bush, particularly at higher levels, where some branches reach towards the camera. A good example of this problem is found in Figure 8, where the buds closer to the camera were localized successfully, but those farther were ignored. In the image, the branches that contain the localized buds (red+blue) are about 20 cm from the camera, whereas the others exceed that distance (red only).

5.3 Discussion and Conclusions

After finding the cutting sites, we use velocity control to navigate the end-effector towards the targets and clip them one by one. This process is described in *D2.6 - Final Manipulator, tools and algorithms*.

Evaluation of the presented pipeline is given in *D2.7 - Final evaluation and dissemination*, which shows that 90% clipping sites were be successfully detected in a rose clipping experiment with 60 sites at different heights. This is a satisfactory result. The missed 10% have a good chance to be detected from another side of the bush.

With some changes to the setup, eg. increased resolution of the cameras, it should be possible to include also rules for clipping above buds, as our preliminary dataset results suggest.

References

[1] Jorge Calvo-Zaragoza and Antonio-Javier Gallego. A selectional auto-encoder approach for document image binarization. *Pattern Recognition*, 86:37 – 47, 2019. URL: http://www.sciencedirect.com/science/article/pii/S0031320318303091, doi:https://doi.org/10.1016/j.patcog.2018.08.011.

[2] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise.

[3] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. URL: http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16.

[4] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[5] L.G. Shapiro and G.C. Stockman. *Computer Vision*. Prentice Hall, 2001. URL: https://books.google.co.uk/books?id=FftDAQAAIAAJ.

