TrimBot2020 Deliverable D2.6

Final Manipulator, tools and algorithms

Principal Author: Wageningen University (WU) &
Wageningen Research (WR)
Contributors: A.-Ludwigs-Univ. Freiburg (ALUF)
University of Edinburgh (UEDIN)
Dissemination: CO

Abstract. This report describes the final design of the robotic manipulator, the end-effectors for bush trimming and rose cutting, as well as the vision modules and control components for closed-loop bush trimming and rose clipping. In the final robotic manipulator a spherical wrist was used that allowed easier and faster trajectory planning. To increase the accuracy of the manipulator several adaptations have been made to its control, such as using higher limits for the motor control currents and adapted PID control values. A controller to do trimmming of flat horizontal planes was implemented and the arm behaviour was adjusted to avoid the cables wrapping around the arm. In the final bush trimming tool the upper blade was made stationary to improve cutting performance. No design changes were made in the mechanical design of the rose clipping end-effector. The vision algorithm for rose clipping was adapted to the real outdoor case. The final bush trimming motion planning method makes use of a trajectories knowledge base and exploits the trimming result for closed loop operation. The approach for rose clipping motion planning is visual servoing, similar to Demonstrator 2. Finally, the control of the total system (the state machine) and where relevant, specific properties of the arm control used therein is described.

Deliverable due: Month 36

Contents

1 Introduction

Based on the results of Deliverables 2.4 and 2.5, and testing of the system that was used there and for Demonstrator (Deliverable 7.6), several modifications to the manipulator, tools and algorithms were proposed. This deliverable describes the final design of the manipulator, the end-effectors for bush trimming and rose cutting, as well as the vision modules and control components for closed-loop bush trimming and rose clipping. Details are given on the modifications that were carried out with respect to the V2 system described in Deliverable 2.4. The integration process itself is described in Deliverable 6.4, while Deliverable 7.7 summarizes the demonstration results.

The final evaluation and found performance for this system will be described in Deliverable 2.7.

2 Final manipulator

2.1 Manipulator selection

Based on evaluation of the arm capabilities and trimming performance during the testing on the static rig (as described in Deliverable 2.5 and Demonstrator 2 / Deliverable 7.6), it was observed that the curved wrist of the robotic arm sometimes poses challenges in motion planning and control. As the option of using a spherical wrist on the Kinova Jaco^2 arm also emerged, this was evaluated and found to offer the following benefits:

- Using a spherical wrist allows for analytically solving the inverse kinematics (IK) problem in the trajectory planning. As previously an approximation was used, this is expected to result in better trajectories with faster calculations.
- A spherical wrist offers better options to place the tool in a desired orientation, such that trimming accuracy might be improved.
- The resulting trajectory might be executed in less time, as fewer (time-consuming) retractions are needed to avoid arm singularities.

Figure 1: Kinova Jaco^2 configurations with curved wrist (left) and spherical wrist (right).

Next, this was also tested by using a simulation of both the curved wrist and spherical arm type in path planning, and comparing the results, as shown in Figures 4 and 5. Here, it can be seen that planning and execution time are clearly lower for the spherical wrist (red) compared to the curved wrist (blue). Also the number of retractions was lower, resulting in a higher cutting time ratio (time spent cutting divided by total execution time) and thus a more efficient trimming path.

As result, a spherical wrist module was ordered and implemented for bush trimming. Testing the spherical writst configuration indeed proved to be beneficial for planning and trimming,

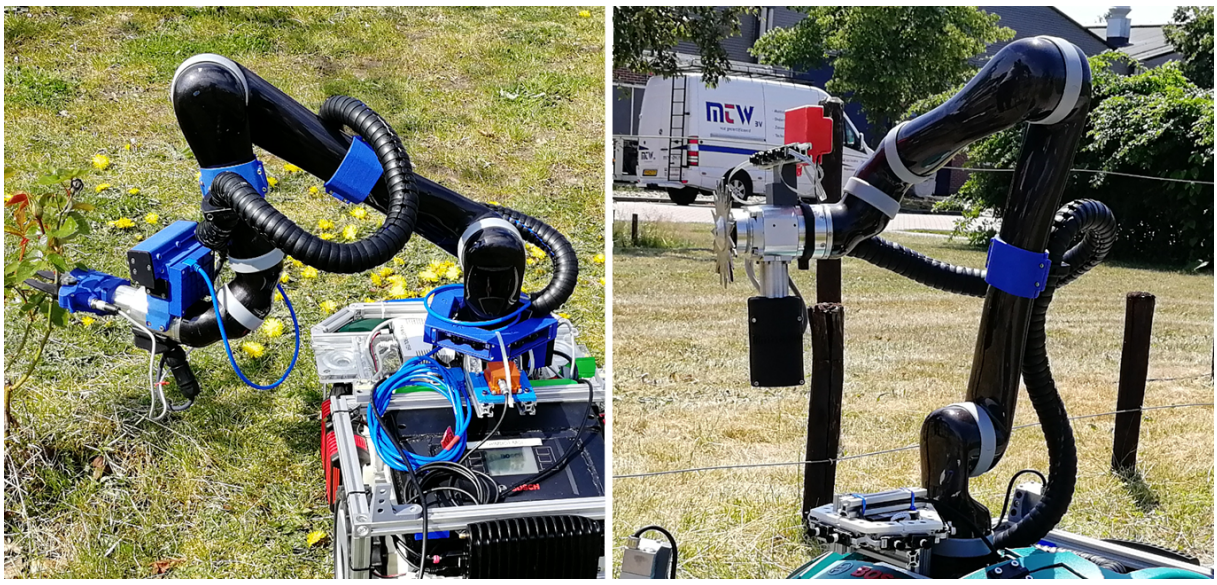

Figure 2: Earlier versions of the robot with curved wrist manipulator. Rose clipper configuration (left) and bush trimmer configuration (right).

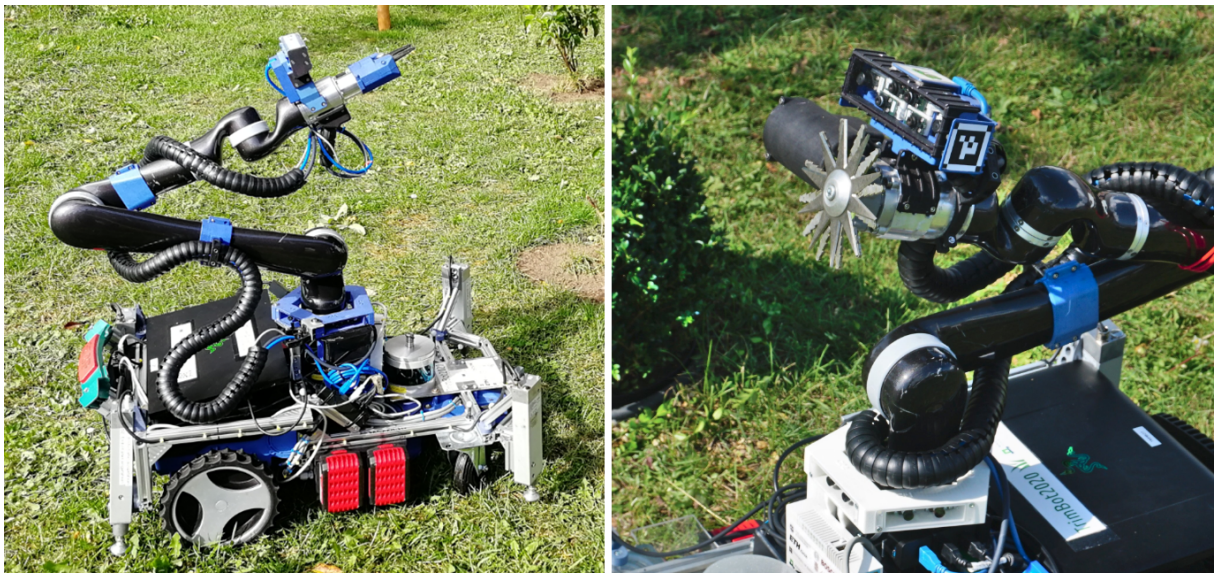

Figure 3: Final versions of the robot with spherical wrist manipulator. Rose clipper configuration (left) and bush trimmer configuration (right).

although the expected time difference could not be observed due to other changes made in the planning setup. Seeing the benefits of a spherical arm for topiary trimming, also its use for rose clipping was considered. As similar benefits could be expected also for this case, and to ensure easier switching between the arms used for trimming and clipping, it was decided to also change this arm. Thus, for the final demonstrator of bush trimming and rose cutting the spherical version of the Kinova Jaco^2 arm was selected. For reference the earlier curved wrist versions of the Trimbot are shown in Figure 2.The final versions of the robot with spherical wrist are shown in Figure 3.

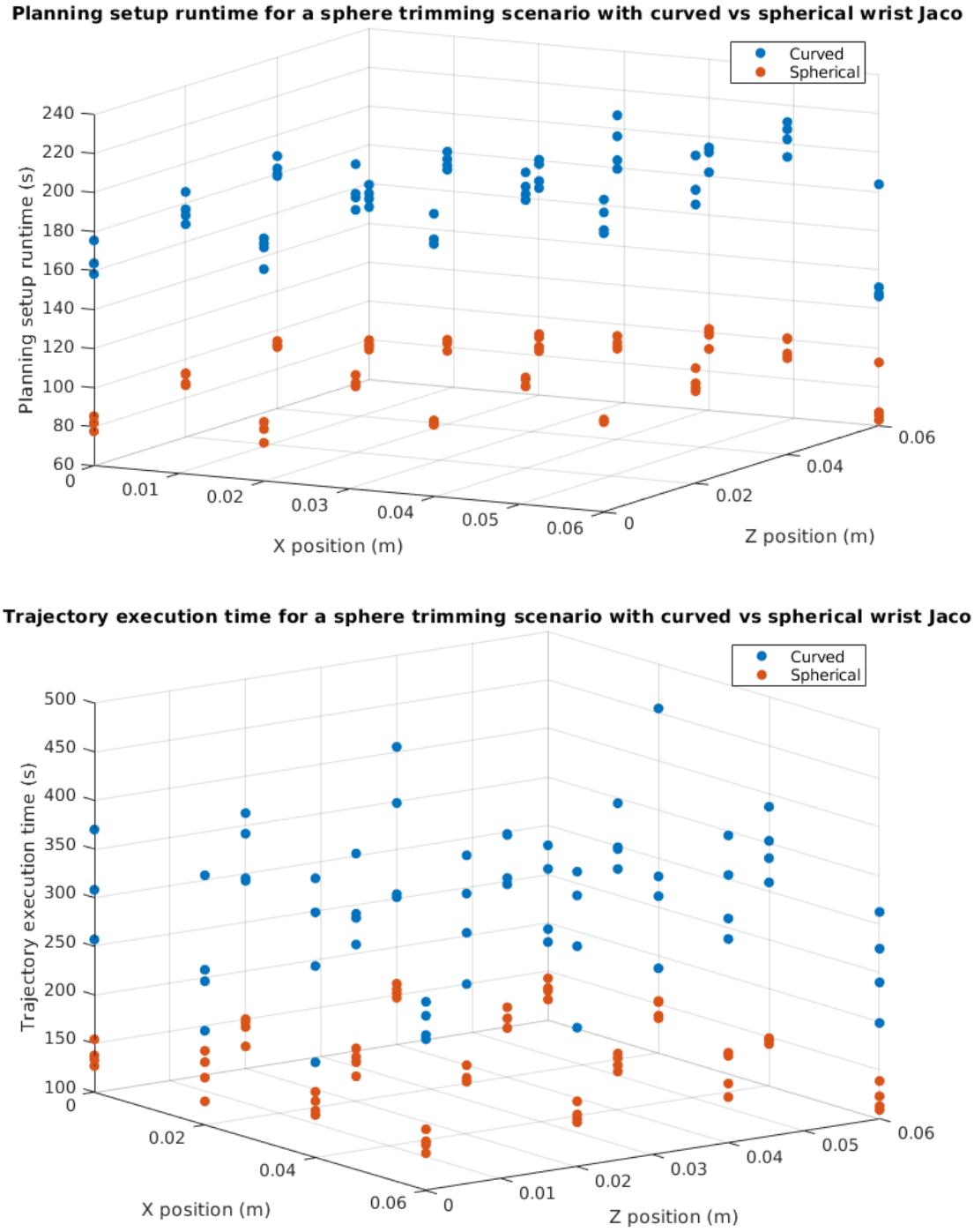

Figure 4: Performance comparison of the spherical wrist Jaco (red) and the curved wrist Jaco (blue) for a simulated sphere trimming task. The target sphere has a 33cm diameter, and its center is alternatively located on several positions on a grid (all referred to the arm base frame). In these tests the y coordinate is always -0.64m, whereas the x and z coordinates vary. The coordinate values (in meters) used in the trials are shown on the lower plane of the plot. Up: Planning setup runtime (including IK querying). Down: Trajectory execution time.

2.2 Control improvement

After selecting the version of the arm to be used, several adaptations had to be made to the arm control software.

First, the relatively large weight of the bush cutting tool (1.5kg) in relation to the specified arm payload (2.2kg peak at full reach) directly affected arm motion performance. This was especially observed in the form of jerky motions and low control accuracy at extended arm reach, caused by the arm firmware limiting the control currents. Such limitation was originally implemented by Kinova for the arm as a whole to avoid damage to the motors due to overloading, but turned out to be limiting for this application. Since torques and currents for the individual motors remained within specification, we implemented an automatic disabling of this current limiter to have the arm execute its motions as desired.

Second, in previous experiments (described in Deliverable 2.4) the arm was found to have some control inaccuracy, especially in the z-direction and likely caused by insufficient gravity compensation. Initially this was resolved by using a lookup table to correct for this error, but the switch to a different arm configuration would at least require an update of this approach. Instead, it was chosen to directly compensate for the gravity effects by updating the parameters of the PID control used in the arm. For this, the I-term in the control for joints 2 and 3, that had the largest contribution to the Tool Center Point (TCP) error, was set to 2 instead of 0. As result, TCP error in z-direction reduced from 15-20mm to 1-3mm max during execution of a cartesian path in the x,y plane. In this, largest errors were observed when arm motions changed direction, causing a small drop in z-direction. When executing a straight path XTE were found to be ¡ 1 mm in x,y direction.

To allow testing of the arm in simulation, it was decided to switch to a different arm controller. Although now implemented by using the official Kinova ROS packages, its technical basis is still the same as the previously used `WPI-controller`, where a time-interpolated trajectory is created for execution by the arm. Also here, PID values for this controller were tuned to achieve suitable TCP positioning accuracy. Values were set to 150, 0 and 0 (with likely internal scaling in the controller). The errorNorm, summed over all joints and indicating if a goal pose was reached or not, was set to 0.001 radians. Arm speed was found to affect control accuracy, with a mean error on the final TCP pose of 3.9mm (SD = 1.9mm) using an arm speed of 0.3 radians/second and 2.6mm (SD = 1.1mm) using an arm speed of 0.15 radians/second. Peak errors ranged between 8 and 12 mm for the fast motion, and between 6 and 10 mm for the slow one. Such accuracy was considered sufficient for further testing of the bush trimming systems. Furthermore, some smaller updates to the control strategy were made. First, a basic controller to do trimming of flat horizontal planes was implemented on top of the original `cartesian\_move` command from the Kinova packages. Second, arm behaviour was adjusted to avoid cable rolling around the arm. For this, after each motion, the arm returned in a reversed motion to its home position, thereby unrolling the cable.

2.3 Trimming motion execution

The planned manipulator trajectory with trimmer speed information is sent as a goal to a ROS action server implemented on top of the Kinova ROS driver. The action server exploits the

Kinova API to send speed commands based on a PID controller, and as input for the error calculation it systematically uses the configuration that should have been reached by the arm at the current time instant. This is computed according to a time-sampled version of the submitted trajectory plan that uses quintic splines and constraints on peak speed and acceleration.

2.4 Proposed future improvements

With these modifications, the final manipulator for integration on the mobile platform was defined. Still, several elements could be addressed for future improvements. These were:

- Reduce the arm load, to improve control performance and lower the risk of arm failure due to overloading.
- Implement more extensive status-checking in the system, to ensure arm loads and usage always remain within safe limits, and adjust control if this is required. This system should also monitor and prevent arm-overheating to prevent arm breakdown.
- Implement a more extensive collision checking in the arm control, that also takes into account vehicle properties and ensures no self-collision in the system will occur to avoid arm damage.

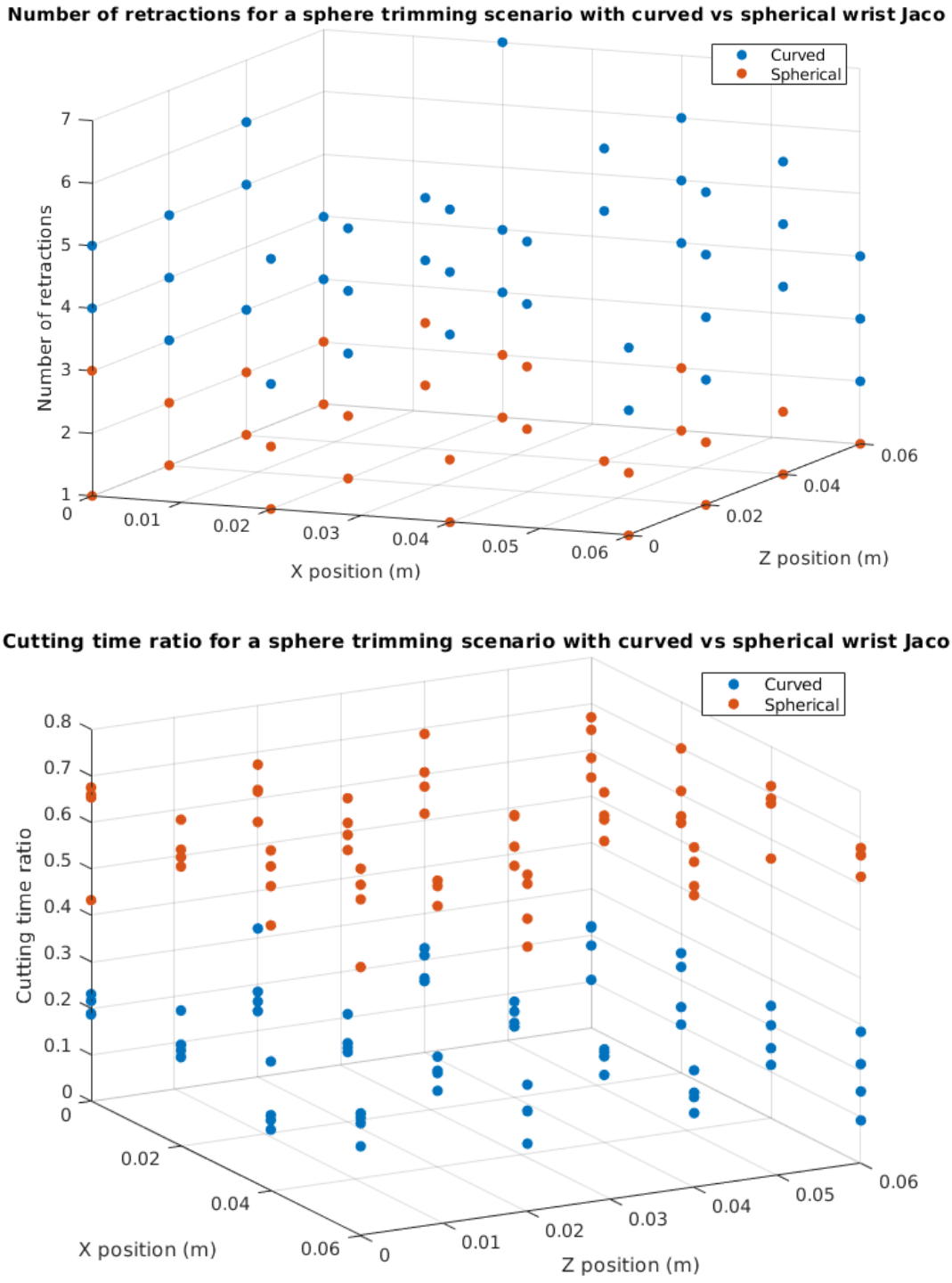

Figure 5: Performance comparison of the spherical wrist Jaco (red) and the curved wrist Jaco (blue) for a simulated sphere trimming task. The target sphere has a 33cm diameter, and its center is alternatively located on several positions on a grid (all referred to the arm base frame). In these tests the y coordinate is always -0.64m, whereas the x and z coordinates vary. The coordinate values (in meters) used in the trials are shown on the lower plane of the plot. Up: Number of retractions. Down: Cutting time ratio (i.e. ratio between the time spent cutting and the whole execution time).

3 Trimming tools

3.1 Bush trimming end-effector

3.1.1 Design changes

Field experiments with the bush trimming end-effector as described in Deliverable 2.5 revealed a shortcoming of the design. The end-effector had two counter rotating blades, with a sharp front blade and a blunt rear blade (see Figure 6). The rear blade has an arrowhead, which prevents a branch from being pushed out from between the blades whilst being cut. However, at the moment of entrance (before the branch passes the hook point), the branch encounters two angled edges which drives the branch out of the cutting unit. This results in a situation where some branches are pushed away all the time and being led around the perimeter of the blades. Furthermore, both the blade and the entry guide had the same number of fingers (see Figure 7 (left)). As result, all fingers will cut at the same time, thus creating a peak load on the drive system.

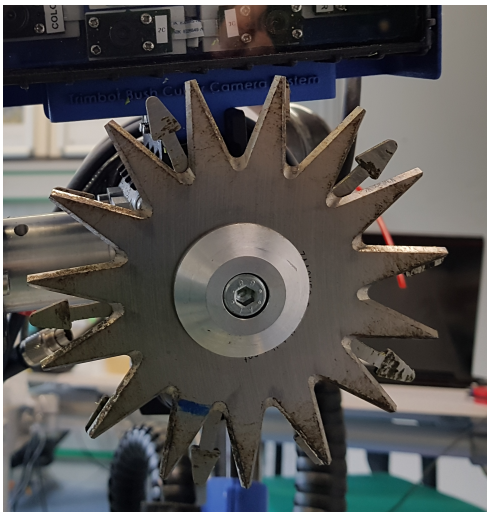

Figure 6: Bush cutter design as described in Deliverable 2.5.

In order to improve the cutting performance, the rear blade was made stationary. In this way, the rear blade can be seen as an entry guide instead of a blade. The entry guide also got two other changes. The arrowhead of the entry guide was made smaller and additional hook points were added to ensure all branches are caught. Also, the number of fingers was increased from 10 to 12. This results in an unequal number of fingers between the cutting blade and the entry guide, resulting in a more evenly distributed load for the end-effector (Figure 7 (right)).

In Deliverable 2.4, it was also indicated that to reduce the weight of the end effector, the drive motor would be placed on the 4th axis of the arm. To drive the cutter head, a flexible drive shaft would be used. As described in Deliverable 2.5 this design was tested but failed. It was decided to revert the design to the previous stage, in which the cutter head was driven directly by the motor. The resulting final design is shown in Figure 8. A photo of the final version is shown in Figure 9.

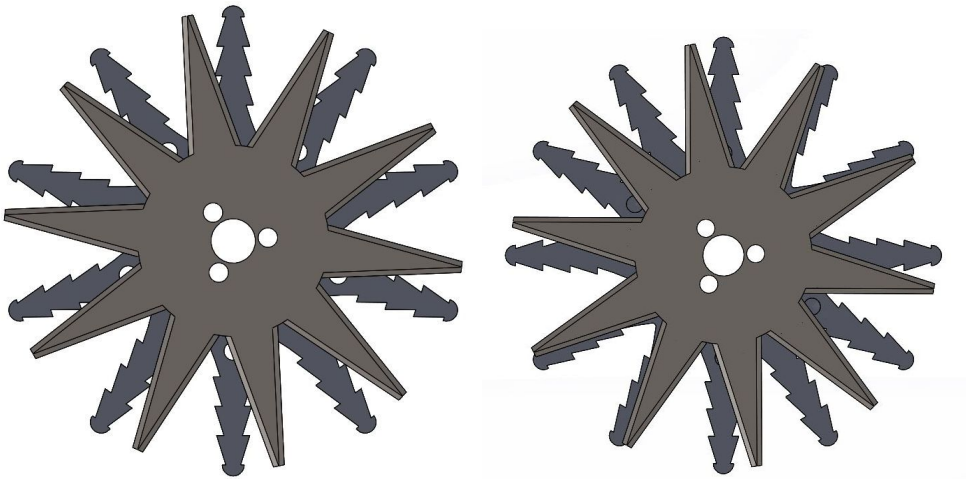

Figure 7: Left: Equal number of fingers (10) for blade and entry guide. Every blade tip reaches the entry guide at the same moment, resulting in higher peak loads ; Right: Blade with 10 fingers and entry guide with 12 fingers. This results in lower peak loads because not all blade tips reach the entry guide at the same time.

Evaluating trimming performance of the new design showed that it provided a better cutting result with less missed branches. Therefore it was decided to keep this configuration for the final demonstrator and bush trimming experiments.

3.1.2 Ideas for future improvements

The changes of the design lead to new possibilities for more improvements of the end-effector design. Due to the limited time between the presentation of Demonstrator 2 (February 2019) and Demonstrator 3 (September 2019), these have not yet been integrated in the design. Since the new design only needs one rotating blade (instead of two), the whole construction can be simplified. A new motor and gearbox arrangement could result in a more compact and lightweight end-effector, which would also benefit arm control. Another point of improvement is cable guidance. The cables are placed in a cable carrier along the robotic arm. This limits the freedom of the arm and can block the view of the navigation cameras. For a future design, the freedom of rotation of the robotic arm should be limited (maximum +/- 180 degrees for each joint) or a robotic arm with integrated power (and data) supply must be chosen.

3.2 Rose clipping end-effector

The design of the rose clipping end-effector was not subject to changes since Deliverable 2.4. In the control software, only minor changes were made to have a more reliable sensor readout and avoid undesired tool motions.

A photo of the final version of the rose clipping end-effector is shown in Figure 10.

Figure 8: CAD drawing of the final design of the bush trimming end-effector. The major change in this design is that only the lower blade is rotating below a stationary entry guide.

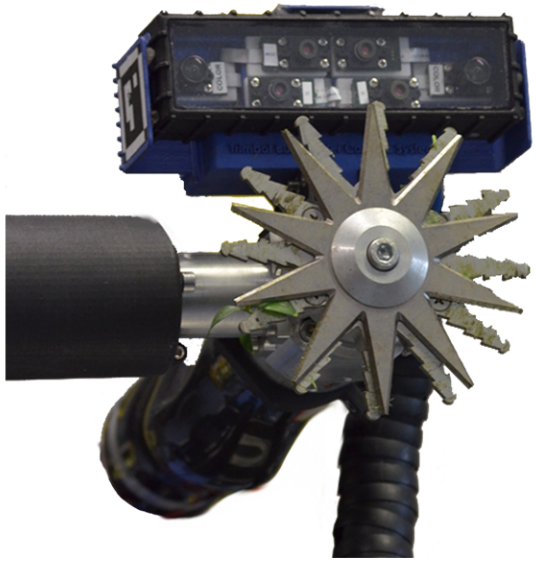

Figure 9: Photo of the final version of the bush trimming end-effector.

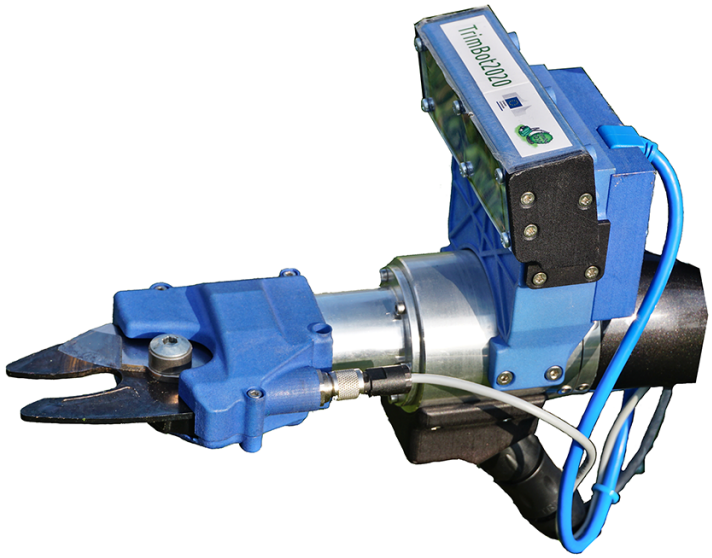

Figure 10: Photo of the final version of the rose clipping end-effector.

4 Vision modules

This section describes the vision modules that take raw visual perception data and use this to generate a target object for the bush trimming planning module. Specifically, these modules (separate for bush and rose) provide information about *where* the object-to-be-cut is located; the planning module then takes care of *how* this cutting is done.

4.1 Bush trimming vision algorithm

The bush trimming vision module operates after the robot platform has navigated to a trimming target and stabilized its own position. This module's tasks are to (a) perceive the 3D environment around the target bush, (b) identify the target and (c) produce a target shape which is sent to the arm motion planning module for further processing. The target shape is parameterized as a fine triangle mesh.

The bush trimming vision algorithm was already extensively described in Deliverable 2.4. For more details, the reader is referred to there.

The improvement brought by multi-view data collection is discussed in Deliverable 2.5 and [2]. Details on the depth sensing and shape-fitting method can also be found in [2].

4.2 Rose clipping vision algorithm

The goal of the rose clipping vision module is to detect clipping site locations relative to the robot. This is obtained by scanning the bush from a set of predefined positions with a stereo camera mounted on the arm. Subsequently, the image is segmented and the regions corresponding to branches are extracted in the form of a point cloud. Finally, clipping sites on branches are detected based on height from the ground and local branching situation (Figure 11). The algorithm is described in detail in *Deliverable 5.4 - Clipping site recognition software*.

Simultaneously, an algorithm was developed to detect buds on stems, which could allow for a more precise site localisation, based on the gardener's rule of cutting above buds. While the results on previous datasets look promising, in practice we found that it is difficult to make it work reliably in the integrated system. The reasons for this are two-fold. First, the resolution of the cameras was not sufficient to capture buds during the bush scan (from ca. 40 cm). The buds could be detected only when they are 20 cm or closer from the camera. Second, the occlusion from leaves prevented seeing the buds at all in many cases, even at close distance. This was particularly problematic during the season before final project demonstration (September), as most buds grow into leaves or small branches during summer, and only few remain dormant and detectable.

Compared to Demonstrator 2, which was based on detection of red tags on branches, the changes in the vision algorithm involved adaptation to the real outdoor case. An important step was to include the branch segmentation, which allows to remove leaves from the plant model. This allowed analysing the branch topology to get good clipping sites even without the bud detector.

Figure 11: Rose clipping site detection. One image from the bush scan set (left) and corresponding detected clipping sites (red) on the segmented pointcloud of the bush (right).

4.3 Hand-eye calibration

The hand-eye calibration for the cameras mounted on top of the trimming tools was already described in detail in Deliverable 2.5. One drawback of that procedure was that it required many manual steps, such as manually recording of the robot poses. For the final system a ROS node was written that performs all steps needed automatically. A master launch file starts the camera driver, robot arm driver and the Halcon hand eye calibation script (implemented using the HDevEngine of Halcon). After initialization the arm moves along a fixed list of waypoints to record images of a calibration pattern from different poses. The recorded images are transferred to Halcon (using the `asr\_halcon\_bridge`[1]). With each acquired image the reached robot pose is communicated to Halcon as well. After collecting a configurable number of images (typically 15 to 20) the movement of the arm is stopped and the hand-eye calibration calculation is carried out. The result and the error measures are displayed on screen and are stored in a result file. For convenience also a yaml file is generated that contains the ROS parameters `arm2cam\_rotation\_euler\_static\_xyz` and the `arm2cam\_translation\_xyz`. This data can directly be used by the Trimbot shape fitting nodes.

[1] http://wiki.ros.org/asr_halcon_bridge

5 Motion planning

Motion planning is described separately for bush and rose trimming, as both tasks used different planning approaches. The description here is based upon previous deliverables, and indicates the modifications that were made to build the final planning system.

5.1 Bush trimming motion planning

5.1.1 Use of trajectories knowledge base

A detailed description of the procedure outlined in this paragraph is provided in [3]. The algorithm is based on discretizing the reachable target area into a set of desired tool poses, and the path used to traverse them is computed by approximately solving a Generalized Travelling Salesmen Problem (GTSP) over a graph whose nodes are multiple IK solutions for each target tool pose, and whose edges are weighted with a custom objective function encoding motion easiness. The major shortcoming of the algorithm is the fact that it is computationally expensive, as it needs to compare a very high number of solutions in order to find a convenient coverage schedule.

In the following the word *scenario* will be used to refer to a specific combination of the shape, position in the robot frame and dimensions of a given target mesh. Storing a working GTSP solution found by the algorithm allows to re-use it in the same scenario multiple times without any computation. If a scenario is lightly perturbed with respect to the one used to generate a solution, the optimal traversal order is expected to be essentially the same, only the actual arm configurations will be slightly different to match the new target poses. This observation leads to the idea of precomputing a database of coverage trajectories by submitting to the GTSP-based planner a grid of size/position instances for each shape of interest.

When a target mesh is acquired, the coverage trajectory corresponding to the closest scenario in the database is retrieved, which is defined as the one whose size and position in x-, y- and z-dimension are the closest to target mesh.

Mathematically, this can be defined as: Given the center coordinates of the target object with respect to the robot base x_{cur}, y_{cur}, z_{cur}, the size of the target object s_{cur} (alternatively defined as the diameter or edge), and the $x_{db,i}$, $y_{db,j}$, $z_{db,k}$, $s_{db,l}$ for all $i = 1,..,N_X$, $j = 1,..,N_Y$, $z = 1,..,N_Z$, $l = 1,..,N_S$ present in the database, the closest scenario is defined as the one having as target the same type of shape that is required in the given trimming task, and whose indices i_{best}, j_{best}, k_{best}, l_{best} are defined as follows:

$$\begin{aligned} i_{best} &= \underset{i=1,...,N_X}{\arg\min} |x_{cur} - x_{db,i}| \\ j_{best} &= \underset{j=1,...,N_Y}{\arg\min} |y_{cur} - y_{db,j}| \\ k_{best} &= \underset{k=1,...,N_Z}{\arg\min} |z_{cur} - z_{db,k}| \\ l_{best} &= \underset{l=1,...,N_S}{\arg\min} |s_{cur} - s_{db,l}| \end{aligned} \tag{1}$$

Next, the intermediate configurations involved in the trimming task are adapted according to the following rule:

$$q_{new} = \begin{cases} q + J(q)^{\dagger}(p_{new} - p) & \text{if } \kappa(J(q)) < \kappa_{MAX} \\ q & \text{elsewhere} \end{cases} \tag{2}$$

where $J(q)$ is the geometric Jacobian of the robot at configuration q, $\kappa(J(q))$ is the condition number of the matrix and κ_{MAX} is a threshold value after which the matrix is regarded as ill-conditioned. Since inverting an ill-conditioned Jacobian in the given equation would lead to a very large configuration displacement, in these cases it is preferred to exploit the unchanged value of q in the stored trajectory. Since this event does not happen frequently, and given that the database has been generated with a resolution of 2cm along all dimensions, the error introduced by such a policy is expected to be negligible. For each arm configuration in the planned trajectory, also the desired trimmer speed for this trajectory slice is stored.

5.1.2 Trimming outcome extraction

Before and after the execution of each trimming motion, the bush pointcloud acquired by the stereo camera from the home configuration viewpoint is stored and used to initialize an octree structure. The two structures are compared in order to find the leaf nodes in the second octree that were not present in the first one. This, in turn, allows reconstructing the point indices i in the second pointcloud that changed with respect to the initial pointcloud. These are interpreted as the points on the trimmed bush where cutting took place, and their signed distance ρ_i from the target mesh is computed.

In order to make the procedure rigorous, the same scanning trajectory used during the target fitting process should be used each time the trimmed bush pointcloud is acquired. This would guarantee to consistently gather information about all points affected by trimming. However, such procedure turned out to be too time-intensive, and using a single view for this comparison turned out to be reliable in most circumstances.

The acquired information about the difference points is exploited in the trimming state machine to trigger trimming motion repetitions (see Section 6.1).

5.1.3 Further updates

The following additional modifications were applied to the coverage trajectory planning routine:

1. A strict constraint was added for direction of motion, so the arm is no longer allowed to go downwards during cutting. Only horizontal and upward directions are allowed, which is expected to improve trimming effectiveness;

2. Angular motions are restricted within the $0-2\pi$ range, so the joint does not try to rotate above 2π to reach angles close to zero. As result, all motions stay within a safe window, and no cable rolling takes place;

3. Mesh triangles affected by cutting are published. They are meant be used by the fitting module, so that it is aware about already impacted areas when fitting a partially trimmed plant.

5.2 Rose clipping motion planning

The approach for motion planning is visual servoing to continouosly updated the targets detected by the arm camera, similar to Demonstrator 2. In this section we detail how the goals are set and updated.

The arm navigates to detected and stored clipping sites 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 1.5 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 Deliverable 5.4 (stem detection, pointcloud post-processing, cutting points localization). This process can be better appreciated in Figure 12.

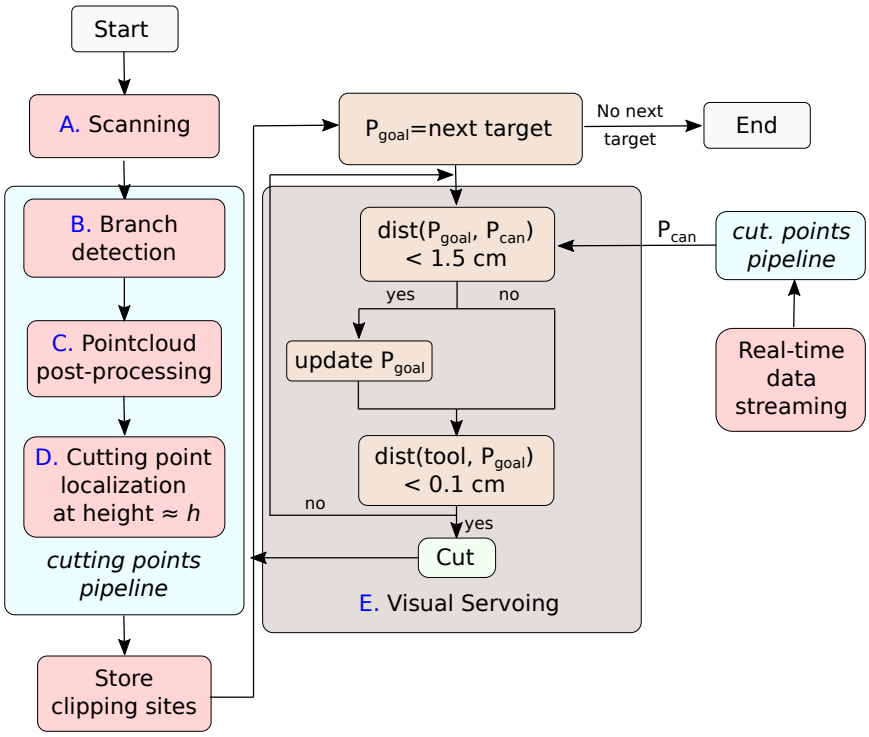

Figure 12: A scheme of the pipeline for rose clipping.

6 System control and arm control for bush trimming

This section describes the control of the total system (the state machine) and where relevant, specific properties of the arm control used therein. The flowchart of the trimming pipeline is shown in Figure 13, starting from the lower-left.

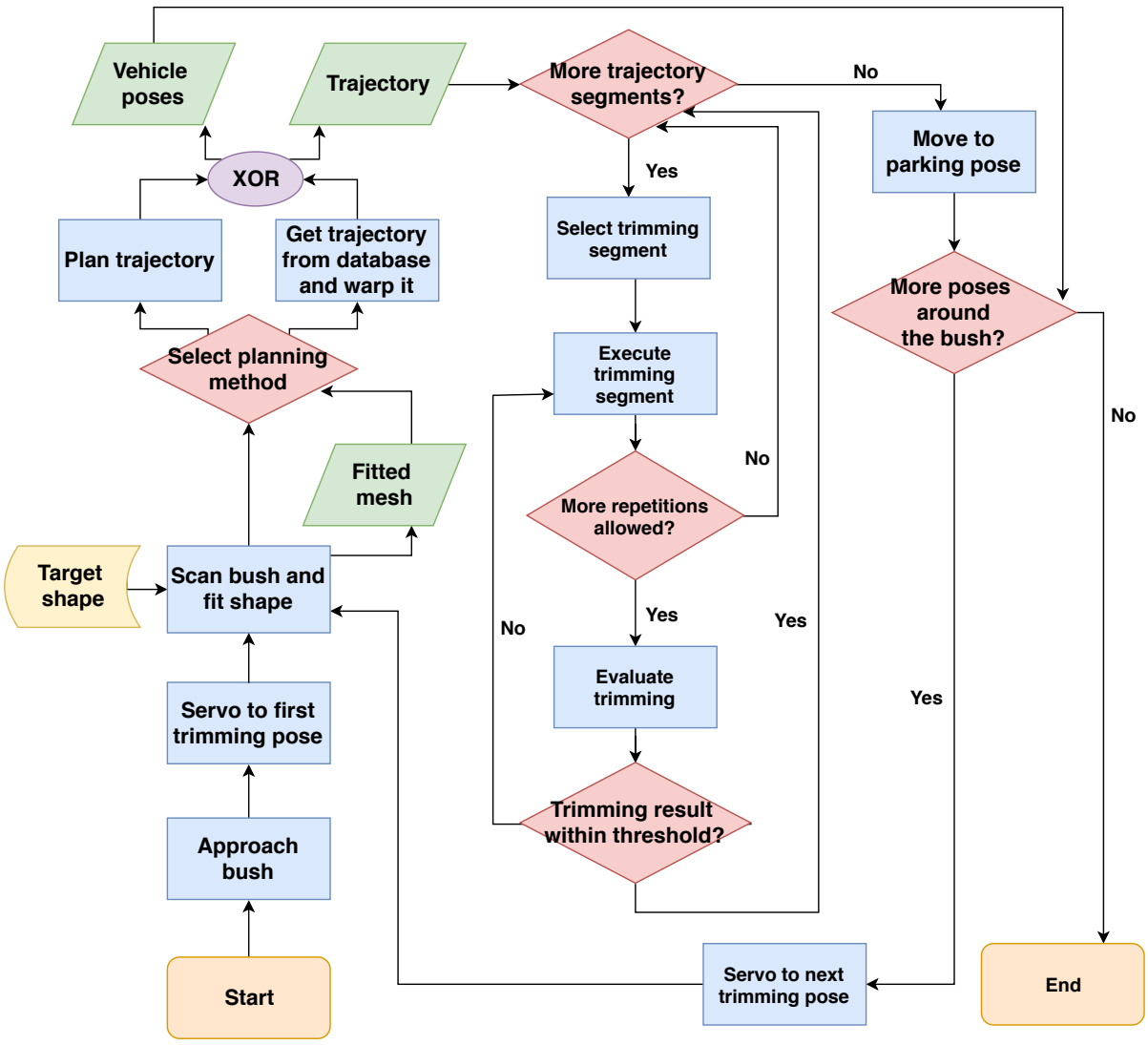

Figure 13: Flowchart of the multi-side trimming pipeline. The process starts lower-left, goes via approaching, scanning and trimming towards the next pose or finish if no poses are left.

6.1 Trimming State Machine

The state machine for the trimming task execution is implemented in ROS FlexBE, and includes various steps from approaching and navigating around the bush up to the actual trimming.

The various steps used in this are explained below, starting from "approach bush", through "scanning", via "trimming" up to "evaluation" and "navigate to next pose". Furthermore, the state machine contained a special state for handling system errors, such that in case of failure the user could indicate to the system what its next action should be: stopping, repeating an operation, or stepping over to another part of the state machine.

6.1.1 Bush approaching

The whole trimming control, including approaching of the bush, starts on a pose about 1 to 2 meters away from the bush, with the robot arriving there by using its garden navigation capabilities. At this moment, the robot switches over to local control for servoing to the bush, as detailed in Deliverable 1.3. Visual servoing is used in a two-step approach: the robot moves forward until it is about 1.5m from the plant, then it turns to have the bush by the right side within the trimming distance of 0.8 m from the bush center (about 0.65m from the bush surface).

6.1.2 Single-pose trimming state machine

After the platform has reached the location at which trimming should be performed, it switches from the general state machine controlling the platform to the state machine that controls the trimming pipeline. At the beginning of the trimming pipeline, the arm moves to a default home configuration, after which it enables the Dispnet and the shape fitting node. At this point the scanning trajectory is executed, and the most recent fitted mesh is continuously published on a ROS topic. At the end of the scanning state, a request is sent to the planning node. The planning node reads the latest published message on the fitted mesh topic and uses it as input to calculate a trimming trajectory. Depending on the received mesh, it will either use a pre-calculated trajectory from the database and warp it to match the bush (as decribed in Section 5.1.1) or plan a new trajectory using the planner. After calculating, it returns the trimming trajectory, together with the indices of the trimming segments in the trajectory and the list of the expected future platform locations around the bush, expressed as angle increments around the bush object starting from its current position. If the current location is the first pose at this bush, the state machine is initialized with an dummy list object, and this newly calculated list is used. If the input list of angles was an actual list object, then this list (which was received at the start) is used as the output list of the state machine after removing the current location.

At this point, for each trimming segment in the planned trajectory the following procedure is followed:

1. The initial bush pointcloud is stored;
2. The arm executes the trimming trajectory segment;
3. The post-trimming bush pointcloud is retrieved;
4. The difference points are computed, as well as their signed distance ρ_i with respect to the target mesh, as defined in Section 5.1.2;

5. The trajectory repetition counter `cnt` (initialized to 1) is checked:
   - if `cnt` is lower than a predefined attempts limit (2 in the implementation), the following quantity is computed

$$\delta = \frac{\sum_{i=1}^{N_P} \rho_i [\rho_i > 0]}{\sum_{i=1}^{N_P} 1 [\rho_i > 0]} \tag{4}$$

     δ is the average distance of the outer difference points (i.e. the ones whose signed distance ρ_i from the target mesh is above zero).
     - if δ is higher than a threshold (1cm in the implementation), then a repetition of the trimming trajectory segment is triggered, and `cnt` is increased by 1;
     - otherwise, the next trimming segment is processed (and `cnt` is set to 1).
   - otherwise, the next trimming segment is processed (and `cnt` is set to 1).

Once the trajectory execution is completed, the shapefitting node and the Dispnet node are disabled, and the arm is moved to a predefined parking configuration that is convenient when the platform is driving. At this point the control is passed back to the outer state machine, to handle moving of the platform.

6.1.3 Multi-pose trimming state machine

Using the list of target trimming poses as provided by the planning node, the following procedure is then executed to move to the next trimming location, as long as the target angle list is not empty:

1. The first angle in the list is read and removed from the list;
2. Visual servoing is used to perform a circular motion attempting to position the robot at the required angle around the plant and at the correct distance;
3. Control is given to the bush trimming state machine with the reduced list of angles as input.

Visual servoing is described in more detail in Deliverable 1.3, while the description of the bush trimming state machine was given in section 6.1.2.

6.1.4 Top trimming

When handling the last pose at the bush (pose 5 in the implementation), the platform will move closer to the bush, to ensure that also the top of the bush can be reached and is properly trimmed. For this, the trajectory planner settings are adjusted such that only the patches whose normal vector is close to vertical are considered in planning. Aside from this, the behaviour of system for this pose is the same as described before.

6.2 Visual servo for rose cutting

The major improvement compared to Demonstrator 2 is in the controller, which was changed from position to velocity control. This allowed getting a smooth motion of the end effector when approaching a moving clipping site on a branch. The scanning and homing motions however still use position control, because the target positions are static.

The navigation of the arm, from the start position to a cutting point, is performed using proportional velocity control (Equation (5)). ROS MoveIt! software [1] 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:

$$\dot{q} = K\Delta q \tag{5}$$

$$\Delta q = J^{\dagger}\Delta X \tag{6}$$

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 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. The velocity of a joint q_i is calculated as follows:

$$\dot{q}_i = \begin{cases} 10[\frac{deg}{s}] & \text{if } K\Delta q_i > 10[\frac{deg}{s}] \\ 3[\frac{deg}{s}] & \text{if } K\Delta q_i < 3[\frac{deg}{s}] \\ K\Delta q_i & \text{otherwise} \end{cases} \tag{7}$$

References

[1] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit! [ros topics]. *IEEE Robotics & Automation Magazine - IEEE ROBOT AUTOMAT*, 19:18–19, 03 2012.

[2] 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.

[3] D. Kaljaca, B. Vroegindeweij, and E. van Henten. Coverage Trajectory Planning for a Bush Trimming Robot Arm. *Journal of Field Robotics*, pages 1–26, 2019.

