Constraint Based Control for Manipulation in Agro-Robotics

Verhees, E.D.T.

Award date:
2021

TU/e Department of Mechanical Engineering
Control Systems Technology Research Group

Constraint Based Control for Manipulation in Agro-Robotics

Master Thesis

E.D.T. (Elise) Verhees
Student Nr. 0950109

Master Systems and Control
Report Nr. CST2021.056

Supervisors

prof. dr. ir. H.P.J. (Herman) Bruyninckx
dr. ir. M.J.G. (René) van de Molengraft
ir. J.P.F. (Jordy) Senden, PDEng

Eindhoven, Monday 13th September, 2021

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

08-09-2021
……………………………………………………..……………..

Name

Elise Verhees
……………………………………………………..……………..

ID-number

0950109
……………………………………………………..……………..

Signature

Elise
……………………………………………………..……………..

Submit the signed declaration to the student administration of your department.

[i] See: https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/
The Netherlands Code of Conduct for Scientific Integrity, endorsed by 6 umbrella organizations, including the VSNU, can be found here also. More information about scientific integrity is published on the websites of TU/e and VSNU

Constraint Based Control for Manipulation in Agro-Robotics

Elise Verhees, Jordy Senden, Herman Bruyninckx, and René van de Molengraft
Department of Mechanical Engineering, Eindhoven University of Technology

***Abstract*—Robotic solutions in environments with a high amount of variation and uncertainty require a different control than mainstream trajectory-based approaches. This paper integrates guard monitors, the Closed-Loop Inverse Kinematic (CLIK) method, and Adaptive Gain Adaptive Bias (ABAG) controllers to achieve a reactive control that is robust against disturbances. Furthermore, a systematic analysis to identify the limits of using cobot technology in external force detection and estimation is provided, including the introduction of Situation Aware Active Force Sensing (SAAFS) to actively take these limitations into account in this control. The use of disturbance monitors and SAAFS is experimentally validated through application on a redundant robotic manipulator, for the agro-robotic use case of harvesting tomatoes.**

I. Introduction

Within the last decades, enormous progress within the field of robotics has been made. This has enabled robots to offer feasible solutions in numerous situations. With appropriate sensor and actuator technology, robots can execute tasks in which they have to interact physically with their environment. These environments can range from factories, where optimization of the physical interactions is a major design driver, to environments with a high amount of variations such as greenhouses (Fig. 1) or hospitals, where safety and robustness against uncertain and incomplete information are major challenges.

Fig. 1: A greenhouse is subject to many uncertainties, like variations in the position, orientation, and geometry of different plants.

In agriculture, robots can be used to solve the ever increasing shortage in available human labour and to reduce high labour costs. This is also apparent from the numerous companies working on agro-robotic solutions, see Appendix A for a list of examples. This paper focuses on a particularly generic use case, namely to make contact with a plant's stem and use it to find the connected leaves and crops (Fig. 2). Two sub-tasks can be distinguished: The former task (#1 to #2 in Fig. 2) will be referred to as ”ApproachStem”, and the latter (#2 to #3) as ”FollowStem”. While the list in Appendix A represents a range of different solutions, the same fundamental underlying approach is taken: there's one predefined task to be executed by one dedicated system, which is often solved through a trajectory-based method (see Section III). This paper aims to let go of these restrictions, by recognizing that reactivity and composability is imperative to creating a robust system.

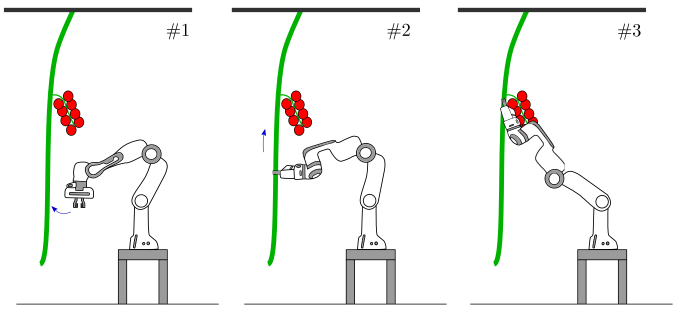


Fig. 2: The tomato harvesting use case. The robot approaches the plant, and then follows the stem of the plant until it reaches the tomato truss.

The layout of the remainder of the paper is as follows. First, the problem is defined in Section II, followed by some related work in Section III. Two main research themes can be distinguished in this paper: monitor-based control and situation aware active perception. The proposed control is elaborated on in Section IV, and the implementation of this, for the tomato harvesting use case, is explained in Section VI. The required background information on manipulator kinematics is provided in Section V. Complementary to the control, perception is required; an analysis to identify the limits of using cobot technology in external force estimation is provided in Section VII. This is followed by suggestions on how to take these into account, introducing Situation Aware Active Force Sensing (SAAFS), in Section VIII. This includes an experimental validation. Lastly, the conclusions and recommendations for future work are given in Section IX.

II. Problem Formulation

This section formulates the addressed problem. First, the term ”task” is elaborated on. Then, some examples of disturbances to which the control should be robust against are given. Lastly, the research questions that are considered in this paper are introduced.

A. *Constrained task control problem*

A task can be defined as an objective subject to certain constraints. The objective usually includes reducing a certain system error and/or optimising some performance criterion like time or energy. For the sub-tasks shown in Fig. 2, the objective includes reducing the distance between the robot end-effector (EE) and the plant or tomato truss. Furthermore, the set of constraints that apply is dynamic, i.e. it can change over the course of the task execution. In general, a robotic manipulation task compasses different aspects, for example:

- T1: Achieving the goal of the task by controlling the motion and/or forces required accordingly,
- T2: Avoiding collisions with known, and possibly also unknown, obstacles,
- T3: Avoiding singular robot configurations as well as algorithmic singularities,
- T4: Ensuring the observability of task-related objects (through e.g. torque sensors or cameras).

In the remainder of this document, the term sub-task will be used for T1–T4. These sub-tasks may require different objectives subject to different constraints, both varying based on the current state. The extend to which different sub-tasks can be executed simultaneously is an important topic of this paper.

B. *Uncertainties in the greenhouse*

The control to solve the constrained task control problem(s) as described in the previous subsection, must be robust against disturbances. Below two examples of disturbances in the tomato harvesting use case (Fig. 2) are described. Both require a more reactive method than the mainstream trajectory-based approaches can offer.

Example 1. Recall respectively the ApproachStem and FollowStem tasks (Fig. 2). The position of respectively the plant and tomato truss (goal region) relative to the robot EE is not exactly known beforehand, because this will vary from plant to plant and measurement uncertainties occur. ■

Example 2. When another plant or object blocks the current motion of the EE, sub-task T2 is required on top of sub-task T1. However, this is not necessarily known before the start of the task execution. Moreover, trajectory planning fixes all Degrees of Freedom (DoF), which compromises composability of multiple partial motion specifications [4]; partial task specifications could possibly allow superimposing T2 on T1. ■

C. *Research questions*

The challenges presented in the previous subsections require (i) clear definition of (sub-)tasks, (ii) discrete control for the scheduling of these tasks, (iii) continuous control for computing control inputs for the robot, and lastly (iv) acquisition of the right sensor data. This gives rise to the following research questions that need to be answered to realize the desired control:

- P1: How to formulate an incomplete task specification (e.g. as required in Example 2)?
- P2: How to resolve incomplete task specifications, i.e. solve redundancy in case of remaining free DoF?
- P3: How to combine several task specifications (e.g. as required in Example 2)?
- P4: How to resolve contradicting constraints? In other words, the opposite of P2, so how to decide what to do when not enough DoF are available to satisfy all the constraints?
- P5: How to coordinate different tasks (discrete control): which variables need to be monitored, when do events need to be triggered, and which state transitions need to be taken when these events are generated?
- P6: How can the limits of (proprioceptive) sensors be actively taken into account into the control (T4)?

III. Related Work

For robotic manipulation tasks in controlled world settings, a lot of research already exists. Most of these approaches are trajectory-based; [8] gives an overview of path and trajectory planning algorithms that can be used in the field of robotics. Generally speaking, first a sensing action is performed, then a trajectory is calculated, and finally this trajectory is accurately tracked without any higher level feedback. This is an effective solution for industrial machines, i.e. high precision motion control in a controlled world. These trajectory-based approaches are most often applied blindly in other contexts too – [2], [3], and [5] use trajectory-based approaches in agro-robotics usecases – although the needs in these contexts are very different, due to variation and uncertainty in the environment: (i) the area of arrival at the end of a motion is what counts, not the exact trajectory towards that area, and (ii) during the motion other tasks should be realised, such as pointing sensors actively in particular directions. A shift is thus required in the trade-off between accurate tracking and robustness against disturbances.

Furthermore, trajectory-based approaches often lack reactivity. [12] introduces a Model Predictive Control (MPC) based joint velocity controller that modulates the joint velocity constraints in real-time. MPC has become a very popular approach to join reactivity to trajectory-based motion, by re-computing new trajectories every time ”significantly new” sensor data has been received. In its Hybrid Constrained Optimization (HCO) form, MPC moreover allows multiple tasks to be computed at the same time, because the core primitives of MPC, namely objective functions and constraints, are highly composable. Although this is a significant improvement with respect to mainstream trajectory-based approaches, it often easily becomes computationally complex. To reduce this complexity, [9], [14], [19], and [10] relax the desired trajectory to a desired region, and thus minimize a cost function in which the minimally required output to keep the state within a bounded zone is calculated. However, in robotics, control is often more a satisfaction problem than an optimization problem, i.e. until some constraint is violated, the robot may move without any interventions of the controller. With this in mind, [1] introduces a ”lazy” control, to control an anthropomorphic robotic arm to open a drawer. This control also expands a trajectory to an acceptable region of motion, and moreover monitors interaction with the environment, to detect contact with the drawer, to trigger events that govern the controller's behavior. The explicit monitoring of state variables can be used in a broader sense, to monitor all active constraints, as is done in [13]. Note however, that the approach given by [13] is still limited by thinking in terms of moving from A to B (instead of moving relative to task-related objects). This paper aims to develop a monitor-based control without this restriction.

Force feedback is required when the interaction between the robot and the environment needs to be monitored. This can be acquired through mounting an external force sensor, but these are expensive, and in many cases overkill. [11] reviews model-based algorithms for real-time collision detection, isolation, and identification that only use proprioceptive sensors. These methods are used in commercial cobots, for example to calculate the external wrench on the EE, but in many configurations such an estimate is not possible (due to mathematical singularities). This paper tries, where possible, to make smart use of partial sensor data to circumvent this. In conclusion, the contributions are formulated as follows:

C1: A methodology to design controllers robust against disturbances, such as measurement uncertainties, or non-nominal contacts or motions. The method is based on two complementary insights: to add a monitor for each disturbance one wants the task to be robust against, and to adapt, pro-actively, to the force sensing capabilities of the robot to reduce the impact of the expected disturbances.

C2: Experimental validation of C1, on a redundant ”cobot” manipulator realising the tomato harvesting use case of Fig. 2.

IV. Proposed Discrete and Continuous Control

This section introduces the primary characteristics of the control: reactivity, limited computational expenses, partial (motion) specifications, and composability. This constitutes the first part of C1 and aims to lay the foundation for solving P1–P5. First, an explanation is given on the concept of guarded motions [4], which is suggested for the discrete control. Furthermore, the proposed complementary continuous control is introduced, and lastly the use of a Closed-Loop Inverse Kinematics (CLIK) scheme as part of this continuous control, to solve for the control inputs in a computationally efficient manner, is suggested.

A. *Discrete control: guarded motions & disturbance monitors*

Reactivity is increased, with respect to trajectory-based approaches, by adding a tolerance constraint for each of the disturbances one wants the task execution to be robust against. So, the generated motions (see later Subsection IV-B) have a guard that triggers when one of these tolerances is violated, so that the discrete control can take the appropriate decision to react to this disturbance. The input of the discrete control thus includes (i) guards and (ii) dependency relations, such as the order in which different (sub-)tasks must be executed, while the output is the trajectory. All the constrained variables are monitored in an online fashion. This approach can also be applied to resolve the problem illustrated by Example 1: By simply setting the robot EE in motion in the direction of its goal region, and actively monitoring when it is reached [1] (while also monitoring disturbances), makes pre-planning the trajectory unnecessary. The implementation of this will be discussed in more detail in Section VI-A.

Furthermore, the described use of *guarded motions* leads to explainable decision-making, because it can be traced back based on why specific guards are triggered. Also, a guarded motion specification allows to only constrain those DoF that are relevant for creating a good enough behavior (through a partial motion specification). This implies that not all DoF have to be specified at the start of execution [4]. Only when necessary, constraints are enforced, resulting in a tube-like control. Not constraining the full task space in combination with run-time re-configurability, generally leaves more options for the composition of multiple partial specifications. This is valuable in many situations, including the one in Example 2.

B. *Continuous control: Hybrid Constrained Optimization*

The continuous control is formulated as a HCO problem, which is suitable due to the composable nature of the primitives used. The formulation of a HCO problem includes [4]:

- State variables, i.e. the continuous state of the system, including task space and joint space variables.
- Desired state variables. Not only those that specify the nominal task of the robot, but also those of the ”background” tasks, such as optimizing the internal posture of the robot to maximize its sensitivity to expected contacts.
- Objective function. This represents the desired dynamics of how the actual state variables must evolve towards the desired ones. This representation comes in the form of a 'cost function' to optimize, taking into account the constraints below.

- Inequality constraints and their tolerances. Not only the ones connected to the nominal state evolution, but also one for each of the disturbance monitors.

In the context of the proposed continuous control, hybrid means that the solver also reacts to monitor events, and then (possibly) switches its algorithm. The computation of the solution thus requires:

- Monitors; algorithms that compute the values of the tolerances, and turn these into events to initiate switching when necessary.
- Solver; computes the instantaneous control input, based on the problem formulation as described above.

The next Subsection IV-C elaborates on the solver used.

C. *Solver: kinematics vs. dynamics*

A CLIK scheme [6] is used to compute the control inputs (joint velocities or torques) for the robot. CLIK offers a computationally faster algorithm than a dynamics solver (for example [15], [17]), because a less complex model is used, e.g. using unity masses. The resulting motion is expected to still be good enough, because the required accuracy for the tasks considered in this paper is relatively low (compared to e.g. industrial machines). Especially for robots with build-in gravity compensation; gravity acting on the robot is usually the biggest contributor to inertial effects. Furthermore, implementing a Jacobian transpose control law [6] in the CLIK scheme offers a good middle ground between a purely kinematic solver (no forces or accelerations) and a full dynamics solver, since control inputs, realising certain virtual forces on the robot, can still be applied without a very extensive model. Moreover, often (some of) the dynamics parameters of the system are either inaccurate or not known at all. In this case, it simply does not give any performance advantages anymore to implement a more complex (dynamics) model. Thus, CLIK and implementing a Jacobian transpose control law offer a good enough approach for the use case considered in this paper.

V. Background: Kinematics

This section aims to provide the reader with some background information for understanding the continuous control that is used in this paper. Kinematics is the study of the motion of bodies, solely based on geometric constraints, i.e. no forces or accelerations are taken into account. First, the basics of the kinematic equations of a robot manipulator are summarized. Then, a quick introduction to quasi-dynamic relations as an alternative to the purely kinematic equations is given. Lastly, the concept of CLIK schemes is explained using a simple example.

A. *Manipulator kinematics*

A task specification is generally given in Cartesian space. Therefore, a mapping from Cartesian to joint space must be made. Given the $(n \times 1)$ joint vector q and the $(m \times 1)$ task space vector p, a (static) geometric mapping g between the two is given by:

$$p = g(q), \tag{1}$$

which is also known as the forward kinematics. Eq. (1) can be differentiated with respect to time to obtain the forward velocity kinematics:

$$\dot{p} = J(q)\dot{q}. \tag{2}$$

This is the mapping between joint velocity vector $\dot{q}$ and task velocity vector $\dot{p}$, through the $(m \times n)$ configuration dependent Jacobian matrix $J(q)$. When $m = n$, it is possible to invert Eq. (2) to obtain the backward or inverse velocity kinematics, which can be used to compute the desired joint velocities.

However, if $m < n$, a manipulator is said to be kinematically redundant with respect to its task, and in this case it is not possible to simply invert Eq. (2). A common way of solving redundancy is using the Moore-Penrose psuedo-inverse, i.e.:

$$\dot{q} = J(q)^{\dagger}\dot{p}, \tag{3}$$

which minimizes the joint velocities [18]. Here $J(q)^{\dagger}$ indicates the pseudo-inverse of $J(q)$, a $(n \times m)$ matrix defined as $J^{\dagger} = J^T(JJ^T)^{-1}$, where dependency of J on q is omitted for clarity. However, other solutions than minimizing the joint velocities may be useful. Moreover kinematic singularities are not avoided using Eq. (3). [6]

Lastly, when $m > n$, the system is over-constrained. In this case it is physically impossible to resolve the motion of the robot without violating one or more constraints. Thus contradicting constraints must be weighted or prioritized.

B. *Quasi-dynamics*

Velocity-resolved schemes using the Moore-Penrose pseudo-inverse, as described in Subsection V-A, have two disadvantages: they introduce numerical problems around singularities, and they are computationally expensive with their $O(n^3)$ numerical complexity (with n being the number of joints in the robot). The reason for this is that they require to calculate a matrix inverse. This can be avoided by using the transpose of the the Jacobian (with $O(n^2)$):

$$\tau = J(q)^T F. \tag{4}$$

Here, a vector of forces and moments F applied at the EE is transformed to a vector of joint torques τ. Eq. (4) represents a *static* relation, meaning it is instantaneous, hence the term quasi-dynamics. It can be easily derived through the principle of virtual work. The reader is referred to section 4.10 of [16] for this derivation.

C. *Closed-loop Inverse Kinematics*

The motion models mentioned in sections V-A and V-B are open loop solutions mapping task space variables to robot space variables. By defining a tracking error which is consistent with the task, it is possible to close the loop which results in a CLIK scheme. An example of such a scheme for an unconstrained (redundant) manipulator is given by Fig. 3 [6]. In this figure, $x_{ee,d}$ and x_{ee} represent respectively the desired and actual EE position, e_{ee} the tracking error, K_{ee} a positive definite gain matrix, $J_{ee}^T(q)$ the transpose of the EE Jacobian, and $g_{ee}(q)$ the forward kinematics of the robot.

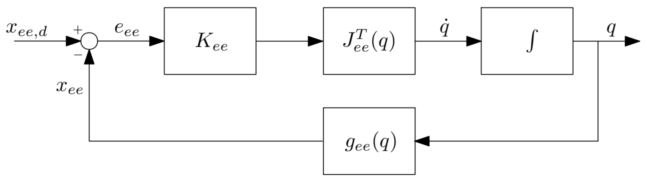


Fig. 3: CLIK scheme for an unconstrained EE task [6]

The CLIK scheme in Fig. 3, uses the position of the EE to calculate the tracking error for the feedback control system, according to the following "control law" [6]:

$$\dot{q} = J_{ee}^T(q) K_{ee} e_{ee}. \tag{5}$$

However, other control laws can be chosen depending on the task specification, robot control interface, or programmer's preference. Note that the term $K_{ee}e_{ee}$ in Eq. (5) is equivalent to a (virtual) force pulling the EE to follow the desired trajectory.

VI. Control Implementation

This section describes the suggested implementation of the control introduced in section IV, for the tomato harvesting use case seen in Fig. 2. It therefore also constitutes the first part of C1, and presents solutions to P1–P5. First, the implementation of the guarded motions is discussed. Next, task descriptions for the two tasks of the tomato harvesting use case are elaborated on, followed by the CLIK scheme for the continuous control.

A. Finite State Machine

Guarded motions, as described in Section IV-A, can be implemented through a Finite State Machine (FSM), with a guarded motion specification in each state. An FSM for the tomato harvesting use case is provided in Fig. 4. On a lower level, one state in this FSM can be implemented as another FSM itself. The need for this becomes apparent when different sub-tasks (Section II-A) of the task need to be coordinated, for instance as required in Example 2. When multiple sub-tasks, that require the same resources, need to be executed (partially) simultaneously, the standard FSM can be extended to a Petri Net [4]. A Petri net contains tokens, that enable transitions to multiple states, or places. When constraints are contradicting, inconsistency is solved by prioritization based on the dependency relations (Subsection IV-A). E.g., in Example 2, safety (T2) has priority over progressing towards the plant/truss (T1).

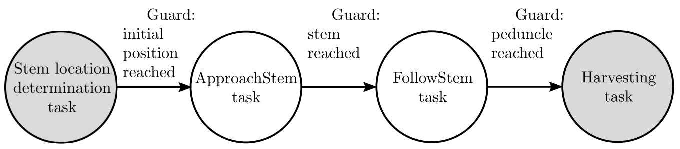


Fig. 4: High level FSM for the tomato harvesting use case. The implementation of the grey states is outside the scope of this paper. For the first task, a force constraint to detect the stem is proposed. Similarly, for the second task, another force constraint to detect the peduncle is proposed. Once a guard is satisfied, the task execution continues to the next task or state.

B. Task descriptions

An example of a (partial) task description for respectively the ApproachStem and FollowStem task is provided in this subsection. In Fig. 4 these task descriptions live in their respective state.

Task description 1: The first task is depicted in Fig. 5. The approximate location of the point on the stem closest to the robot is assumed to be known from a perception skill that falls outside the scope of this paper. Furthermore, the assumption is made that this point is located below the tomato truss. The initial pose of the robot EE is assumed to be similar to the pose shown in Fig. 5. Translation in the direction of the plant is simply generated based on the delta in position between the EE and the output of the perception skill, through Fig. 7 (next subsection), instead of calculating some optimal solution.

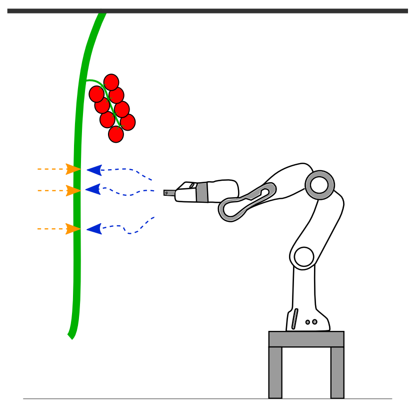


Fig. 5: ApproachStem task: approaching the tomato plant with the EE is not restricted to a certain path (in 6 DoF task space), so the desired path is relaxed to a desired region or tube. Possible (position) paths are displayed by the dashed blue arrows. The guard indicating that the task is completed, a force felt due to contact with the plant, is shown by the orange dashed arrows.

The robot motion is subject to a set of constraints. Firstly, the higher level force guard indicating contact with the plant (see Fig. 5). Furthermore, bounds on the orientation of the EE with respect to the plant are enforced, such that at the end of the task, the gripper is faced towards the stem of the plant correctly. However, the constraints on the orientation are position dependent (no effort has to be wasted on this when the EE is still relatively far away from the plant). Also, bounds on the position of the EE are chosen based on the lay-out of the greenhouse: the plant spacing, row spacing, and height of the stem and trusses. These environment dependent variables are given in Table I, for a standard Dutch (vine tomato) greenhouse.

TABLE I: Greenhouse geometry

Property	**Value**
Plant spacing	45 [cm]
Row spacing	1.6 [m]
Height (max. ripe truss)	0–1.8 [m]

Task description 2: The second task is depicted in Fig. 6. At the beginning of the task, the EE is located where it was at the end of the ApproachStem task, which is assumed to be below the tomato truss. Translation in the direction of the truss can thus be generated by moving the EE upwards. Furthermore, following the stem, without damaging it, requires controlling the interaction force between the EE and the plant. So contrary to the ApproachStem task, for this task, a continuous feedback loop is closed between external force on the EE and the position of the EE, which influences motion in the plane perpendicular to the direction of the velocity of the EE, such that:

$$F_{min} < F_{external,\perp v} < F_{max} \tag{6}$$

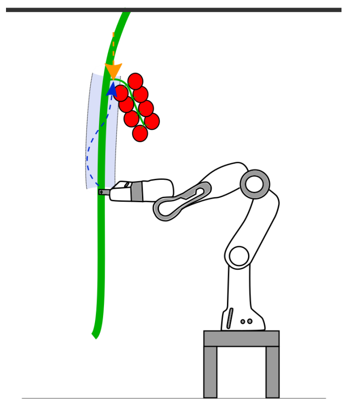


Fig. 6: FollowStem task: for approaching the tomato truss with the EE, through following the stem of the plant, the path may be within a certain region (which is moreover subject to force constraints, to prevent damaging the stem), for example as indicated by the shaded area. A possible path is displayed by the blue dashed arrow. The guard indicating that the task is completed, a force felt due to contact with the peduncle (attachment of the truss), is shown by the vertical orange dashed arrow.

Other constraints include the higher level force guard indicating contact with the peduncle (see Fig. 6). Furthermore, there are angular force constraints to limit twisting of the stem of the plant. Lastly, constraints on the position still apply, due to the environment and geometry of the greenhouse.

C. Closed-Loop Inverse Kinematics scheme

The proposed control scheme mapping Cartesian space specifications, which follow from the task descriptions in Subsection VI-B, to joint space control inputs, is depicted in Fig. 7. A CLIK scheme, as was introduced in Subsection V-C, is used. The following "control law", note the similarity with Eq. (4), is implemented:

$$\tau_d = J(q)^T F_{virtual}(e). \tag{7}$$

The loop is closed through a virtual force $F_{virtual}$ that is dependent on the system error e. For example, in case of proportional control, $F_{virtual}(e)$ becomes Ke (similar to Eq. (5)). The virtual force is mapped to desired joint torques τ_d, which are sent to the robot. Here, e and thus also $F_{virtual}$ ($\leq$6 DoF in Cartesian space) on the EE do not have to be position-based, as is the case in Eq. (5). They can even be based on a combination of motion and/or force constraints in Cartesian space, as is required for the FollowStem task.

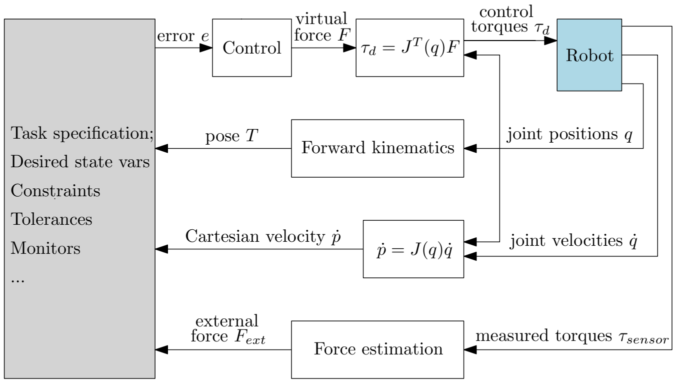


Fig. 7: CLIK scheme to compute control inputs τ_d. The "Task specification" block contains all the information (inputs to the hybrid constrained optimization problem, section IV-B) required for computing the tracking error and for creating events to inform the higher level discrete control when to switch (part of) the task specification.

A 1D controller for each of the specified elements in the $(\leq 6) \times 1$ error e is implemented, in order to convert e to a virtual force or torque on the EE in the corresponding direction. The example in Fig. 3 (section V-C), uses linear proportional feedback regulators, as defined on the diagonal of K_{ee}, to try and drive each element in e_{ee} to zero. This paper however, includes tolerances. One way of implementing this is pre-processing e, as shown in Appendix B. Fig. 8 shows measurement results of the integration of this in linear proportional control in a CLIK scheme. Here, the robot EE is sent to a hard-coded position (or region due to the tolerances). The controller does not react well to a step in the desired position, so feedforward control is added to gradually increase the EE velocity, and thus avoid a step in the control signal.

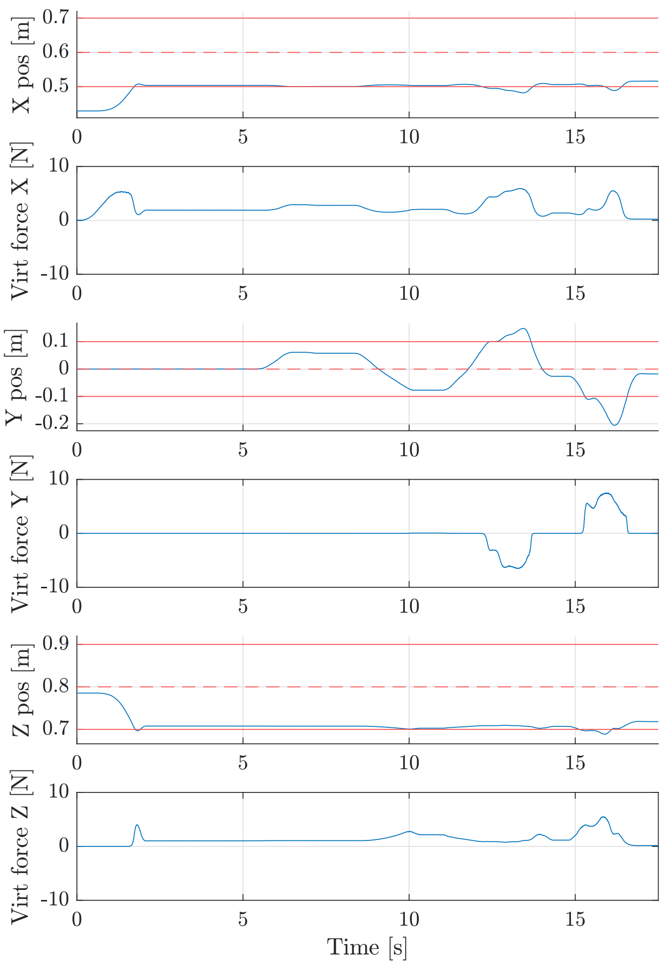


Fig. 8: Measurement data of sending the robot EE to a certain goal region, using a CLIK scheme with linear proportional controllers ($K = 10$) with tolerances. The start position is $[0.43\ 0.00\ 0.79]$, and the goal region is $[0.6\ 0.0\ 0.8]$ (red dashed lines) with tolerances of $[0.1\ 0.1\ 0.1]$ (red lines), all in the base frame as given in Appendix C. Within the first few seconds, the system approaches the goal region. From 5 seconds, the system is excited in the Y direction, and as can be seen, only when the Y position exceeds the bounds, a virtual force, which is directly related to the control torques, is realized (in the Y direction).

Alternatively, an Adaptive Bias Adaptive Gain (ABAG) [7] controller (Appendix D) can be used, which iteratively updates its control parameters, and therefore has no difficulties with a step in the desired position, thus eliminating the need for a pre-planned trajectory. Some other features of ABAG control include:

- Scalable solution. Due to the fact that the output is a factor of the maximum control input instead of an absolute number, it will not "blow up" for increasing errors (in contrast to linear proportional controllers). In this case the maximum control input is the maximum virtual force allowed in each DoF.
- Adaptions to the effects of unknown and/or non-modelled dynamics parameters of the controlled system are performed [7]. While the CLIK scheme used does not include any models of non-instantaneous dynamic effects, when certain tolerances are exceeded, the ABAG controllers will compensate for these effects, by adapting the terms in the ABAG algorithm accordingly.
- Tolerances can be implemented by the explicit dead-zones that are part of the ABAG algorithm (see Appendix D).
- Very low complexity implementation.

The use of ABAG control is thus advised. Implementation of this remains future research.

The feedback loop is completed by using data from the robot state to construct e (grey block in Fig. 7). Using the forward kinematics $g(q)$ and the manipulator Jacobian $J(q)$, the EE pose T and Cartesian velocity $\dot{p}$ can be found based on measured joint positions and velocities (q, $\dot{q}$). The external force on the EE is moreover required for force feedback. This is further investigated in sections VII and VIII.

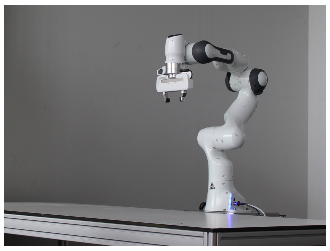
VII. Force Feedback using Proprioceptive Sensors

This section provides a systematic analysis to identify the limits of using cobot technology in external force detection and estimation. The system used in this paper is the *Panda* redundant cobot manipulator by Franka Emika (FE). The implementation in the previous section requires some form of force sensing for two complementary purposes: (i) the monitoring of force guards, e.g. to trigger the state transitions as shown in Fig. 4 and (ii) continuous force feedback control, e.g. to follow the stem of the plant. Here (i) is less demanding, in terms of (dynamic) force feedback, than (ii). First, the accuracy of the wrench estimates (combination of a 3 DoF force and a 3 DoF torque) given by the Franka Control Interface (FCI) is discussed, followed by an investigation of the subset of robot configurations in which a wrench estimate can be provided. Lastly, some conclusions are drawn.

A. External force estimates

The accuracy of the force estimates by the FCI is investigated by attaching a known load to the EE of the robot in some weight measuring experiments. The FCI provides wrench estimates, for the external wrench acting on the robot's EE, based on its proprioceptive sensors. For a detailed explanation of how these calculations are done, the reader is referred to [11]. Three different loads, of respectively 0.5, 1.0 and 2.0 kg are attached to the EE of the robot. Measurements are conducted in two different robot configurations as shown in Fig. 9, such that possible configuration dependency can be observed. Configuration 1 is well within the robot's workspace, while configuration 2 is more close to its workspace boundary. The results of the measurements are plotted in Fig. 10.

To help interpret the measurement results, note that the torque $\vec{\tau}_i$ felt in joint i, caused by (external) force $\vec{F} =$


(a) Configuration 1

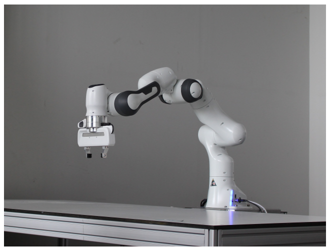

(b) Configuration 2

Fig. 9: Configurations in the weight measuring experiments

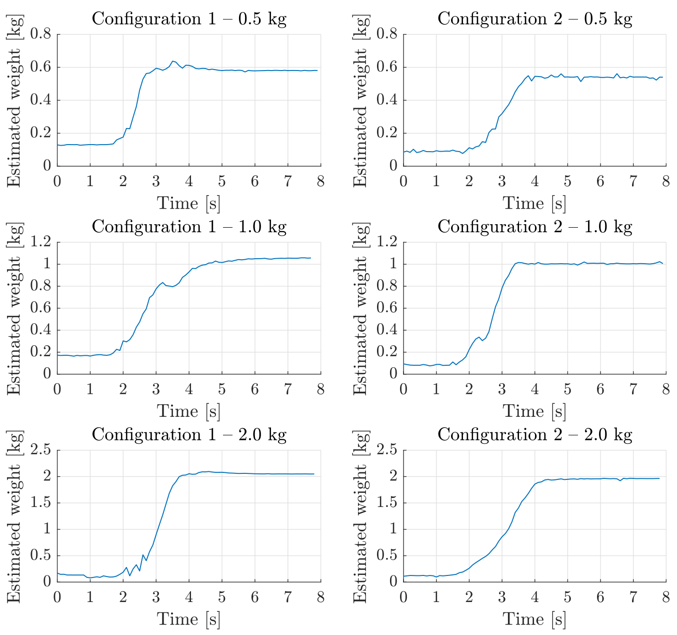


Fig. 10: Results of the weight measuring experiments. Around $t = 2$ seconds, the load is felt by the robot. The estimated weight on the Y-axes of the plots is calculated by dividing the estimated external wrench in the vertical direction (i.e. z_{ee} in Fig. 17, Appendix C) by a gravitational acceleration of 9.81 m/s^2.

$[F_x\ F_y\ F_z]^T$ on the EE, with position vector $\vec{r}_i$, is defined by[1]:

$$\vec{\tau}_i = \vec{r}_i \times \vec{F}. \tag{8}$$

Here $\vec{r}_i$ is a vector from joint i, about which the torque is measured, to the the EE, where the force is applied. The projection of Eq. (8) on the motor axis in joint i is given by:

$$\tau_i = (\vec{r}_i \times \vec{F}) \cdot \hat{n}_i, \tag{9}$$

where $\hat{n}_i$ is a unit vector representing the direction of the motor axis. Provided that $\vec{r}_i$, F, and $\hat{n}_i$ are defined in the base coordinate system (Fig. 17, Appendix C), F only has one non-zero component, namely F_z. This reduces Eq. (9) to:

$$\tau_i = r_{i,y}F_z n_{i,x} - r_{i,x}F_z n_{i,y}, \tag{10}$$

which after substitution of $F_z = -mg$ results in:

$$\tau_i = -r_{i,y}mgn_{i,x} + r_{i,x}mgn_{i,y}, \tag{11}$$

where m is the mass of the load and g is the gravitational acceleration.

Assumption 1. Note that for the configurations in Fig. 9 $r_{i,y} << r_{i,x}$ applies. It is therefore assumed that the second term in Eq. (11) has the most significant influence on τ_i. ■

When the estimated external torque in joint i increases in magnitude, i.e. when the torque τ_i as defined in Eqs. (9)–(11) increases in magnitude, this has two expected results:

- An offset in the estimated external torque, e.g. due to unmodelled or incorrectly modelled effects like friction or gravity compensation, becomes relatively smaller.
- The sensor resolution stays the same, so the "percentual change" in joint torque that can be measured improves.

Therefore, the external wrench estimates, which are calculated through the external torque estimates, are expected to be more accurate for higher τ_i, and after taking Assumption 1 into account, thus for:

H1: Configurations which have, on average, larger $r_{i,x}n_{i,y}$ (for a constant load).
H2: A higher load mg (for a constant configuration).

In order to investigate whether the above holds, the plots in Fig. 9 are summarized in Table II. For both configurations, the Steady State (SS) values (at $t = 8$ seconds) and relative changes Δ (value at $t = 8$ seconds minus value at $t = 0$ seconds) are provided for each of the loads. Furthermore, percentual errors are given, which are calculated as:

$$\text{percentage} = \frac{\text{measurement} - \text{weight}}{\text{weight}} \cdot 100. \tag{12}$$

TABLE II: Steady State (SS) weight values and weight increase (Δ) values, of the weight measuring experiments. Percentual errors are provided between brackets.

Weight	Configuration 1		Configuration 2	
	SS (%)	Δ (%)	SS (%)	Δ (%)
0.5	0.58 (16)	0.45 (-10)	0.54 (8)	0.45 (-10)
1.0	1.06 (6)	0.88 (-12)	1.00 (0)	0.91 (-9)
2.0	2.05 (2.5)	1.88 (-6)	1.96 (-2)	1.85 (-7.5)

From Table II, it can be concluded that the SS estimates in the experiments with configuration 2 are more accurate (both absolute as well as percentual), which is in accordance with hypothesis H1. Furthermore, the SS estimates for larger

[1]Note that Eq. (8) only considers a force on the EE, while in principle, also a torque around the EE may be applied. However, in the context of the current experiment, it will suffice to only include a force, because the weights will (mainly) merely exert a force on the EE.

loads are more accurate as well (with the exception of the experiment with a load of 1.0 kg in configuration 2), as predicted by H2.

One could argue to look at the weight increase Δ instead of the SS values, because at $t = 0$ the estimated weight is never actually 0. Also, when comparing the plots for configuration 1 and 2 in Fig. 9, the whole plot for the respective load is shifted down, which could suggest an offset is present which must be subtracted. However, the expected configuration dependency (H1) and load dependency (H2) are not visible anymore in the Δ data, except for a decreased percentual error for larger loads, see Table Table II. A possible explanation for this is that the suspected offset, which is likely caused by inaccuracies in the models used, is expected to be both configuration and load dependent. On top of that, because both values are subject to model inaccuracies, the error when subtracting these values could be accumulated. Thus, it is not possible to simply subtract the value at $t = 0$ from the value at $t = 8$ seconds.

Note that only the accuracy, i.e. the closeness of measurement data to the real value, in this case the known load, as required for qualitative analysis, is considered in this subsection. Not the precision, i.e. the closeness of measurement data to itself or variation, which is required to be of such small magnitude that measurements are not dominated by it. While the precision is expected to be lower for higher average $r_{i,x}n_{i,y}$ and higher loads (noise is magnified), the measurement data does not show huge differences. However, because the sensor data is low-pass filtered, no further comments can be made regarding this.

B. *Configuration dependency*

There exist configurations where the FCI does not provide any external wrench estimates (note that external joint torque[2] estimates *are* always available though). According to FE, this is an implicit error signal indicating that the robot is near or in a singular configuration. This is confirmed by looking at the singular value decomposition of the manipulator Jacobian: (some of) the singular values are indeed small when the FCI fails to output external wrench estimates, i.e. the Jacobian becomes rank deficient. Near singular configurations, bad numerical conditioning is thus expected to be the cause of the established limits, which this subsection will further investigate.

Considering that there are no wrench estimates available near the singular configurations, the "inverted" problem is investigated: in some configurations, giving control inputs τ_i that correspond to a(n increasing) virtual force on the EE does not move the robot. This behavior is expected to have the same cause as the inability to estimate external forces in some configurations and can be easier investigated, because a known virtual force can be realized on the EE.

[2]The torques as measured by the torque sensors, minus torques resulting from e.g. user input or gravity compensation. In other words, the torques resulting from external forces acting on the robot.

An interactive experiment is set up, such that the robot's response can immediately be felt by the user. After the robot is initialized, the user will pull the robot EE away from its initial position, to which the robot reacts by calculating the required virtual force on the EE and corresponding control inputs to counter this, see Appendix E for the controller used. When the robot is initialized in configuration 1, see Fig. 9, the virtual force will move the robot EE back as expected. When the robot is initialized in configuration 3 however, see Fig. 11, upon perturbing the EE in the X direction (of the base frame, as depicted in Fig. 17, Appendix C, or Fig. 11), no resistance is felt, and the robot will not move the EE back. The remainder of this subsection provides an explanation for this.

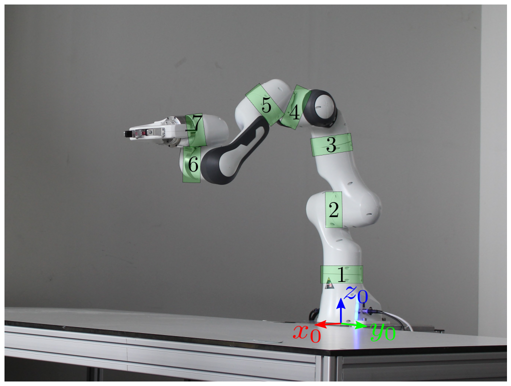


Fig. 11: Configuration 3. The joints (green) are numbered for easy reference, and the base coordinate frame is drawn.

Eq. (9) still applies, but in this context τ_i is the torque about the motor axis in joint i corresponding to the virtual force $F = F_{virt}$. Again, provided that $\vec{r}_i$, F_{virt}, and $\hat{n}_i$ are defined in the base coordinate system, F has one dominant component, namely $F_{x,virt}$, which means that Eq. (9) can be approximated by:

$$\tau_i = r_{i,z}F_{x,virt}n_{i,y} - r_{i,y}F_{x,virt}n_{i,z}. \tag{13}$$

To simplify the calculations in Eq. (13), joint 6 (see Fig. 11) is investigated. The reason for this is that for this joint $\hat{n}_i = [0\ 1\ 0]^T$, resulting in:

$$\tau_6 = r_{6,z}F_{x,virt}. \tag{14}$$

Joint 6 is moreover close to the EE, limiting error propagation in the measurement results.

The results of the experiments are plotted in Fig. 12. The first row shows the desired virtual force as a result of a perturbation in the X direction. The corresponding desired joint torque in joint 6, as calculated by the controller (Appendix E), is given in the second row. For configuration 3, almost no resulting torque is visible. To understand this, first Eq. (14) is inverted to:

$$F_{x,virt} = \frac{\tau_6}{r_{6,z}}, \tag{15}$$

of which the result is plotted in the third row of Fig. 12. This approximation of the virtual force resembles the actual desired virtual quite well, meaning that the approximation of Eq. (13) is reasonable, and that the torques in row two may be approximated by Eq. (14). In the fourth row of Fig. 12, the force arm $r_{6,z}$ of the virtual force in the X direction is plotted[3]. For configuration 3, $r_{6,z}$ is smaller, resulting in a smaller joint torque through Eq. (14), which thus explains the low resistance felt in the X direction.

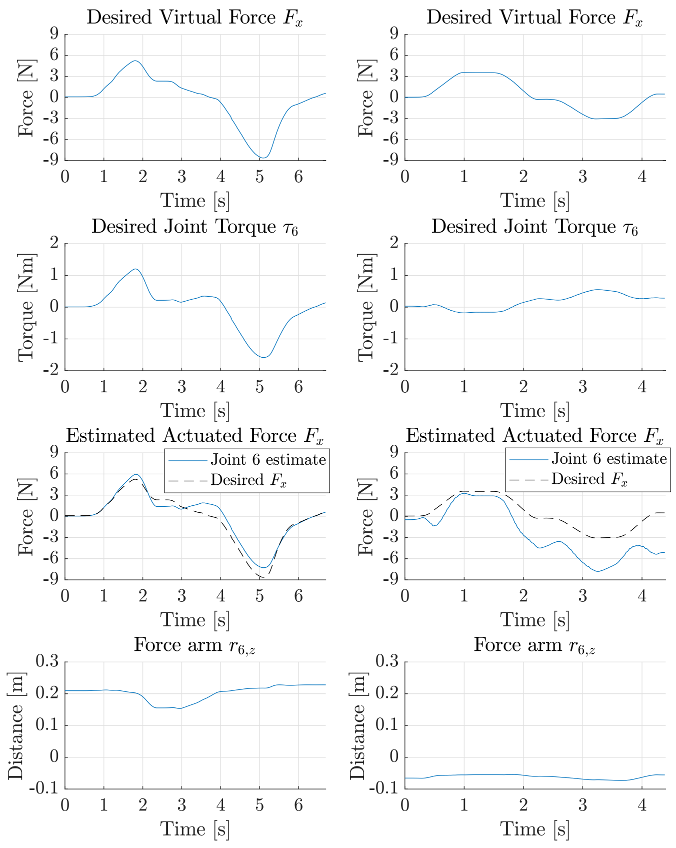


Fig. 12: Results for the configuration dependency experiments. The left column presents data for configuration 1. The right column for configuration 3.

To conclude, the following is expected to be true: When $r_{6,z}$ approaches zero, when applying a virtual force $F_{virt,x}$ on the robot EE, (the part of) τ_6 (contributing to direction x) will approach zero as well, independent of the magnitude of the virtual force. Based on the same underlying principle, when $r_{6,z}$ approaches zero, τ_6 resulting from an external force F_x also decreases, and an estimate of F_x becomes less accurate, until eventually no estimate can be calculated anymore at $r_{6,z} = 0$. This is due to the bad numerical conditioning resulting from dividing by a small number, which is measured using an encoder with finite resolution. This can be generalized to the multidimensional case.

[3]Here $r_{6,z}$ is calculated as the difference between the EE position and the joint position of joint 6 along the Z-axis of the base frame.

C. Conclusions of sensor analysis

The results shown in Subsection VII-A look promising for using the proprioceptive sensors in a cobot to measure external wrenches on the EE. The external wrench estimates are accurate enough to use in the monitoring of forces to detect the stem of the plant and the peduncle. A truss, for example, weighs around 0.4-0.6 kg, which should be detectable based on the results from the weight measuring experiments. Further research should be directed towards application specific tests and dynamic weight measuring experiments. Especially to investigate whether continuous (so dynamic) force feedback control, to realize stem following, is possible. Moreover, both monitoring and feedback control are also complicated by the results found in Subsection VII-B, because for some configurations no external wrench estimates can be calculated. The next section (Section VIII) will suggest a solution to this.

VIII. Situation Aware Active Force Sensing

This section provides suggestions on how to take the configuration dependent force sensing limitations, as identified in Subsection VII-B, into account in the developed control, and hereby offers a solution to P6. It thus constitutes the second part of C1. Also, experimental results showing an implementation of this for the tomato use case are included in this section, constituting C2.

A. Partial force sensing

This paper suggests to control the pose of the robot such that partial force feedback is possible to circumvent the sensing limitations. Instead of calculating a full 6 DoF wrench estimate, (some of) the joints are configured such that at least the relevant partial force feedback is possible. This "method" is from now on referred to as Situation Aware Active Force Sensing, or SAAFS for short. The next Subsection VIII-B exemplifies this through the tomato use case.

The motivation behind SAAFS is that in many configurations, some relevant partial force estimates can still be calculated. In these cases, SAAFS eliminates the need to design a singularity avoidance algorithm. This is advantageous because singularity avoidance, which in current research is often solved by designing some kind of artificial manipulability metric to optimize, might in many cases be unnecessary or even disadvantageous, due to the increased (computational) effort and the compromised robot workspace. Note however that this is a "spectrum", i.e. SAAFS in more DoF approaches the same computational effort and limited workspace as singularity avoidance. This is essentially a trade-off between force estimation in more DoF and workspace.

B. Experimental validation: set-up

The following experiment illustrates SAAFS in combination with the use of disturbance monitors (as introduced in Subsection IV-A). The tomato use case (Fig. 2) is imitated using the experimental set-up with a dummy plant as shown in Fig. 13. For simplicity, in this experiment, the ApproachStem task is reduced to a hard-coded[4] point-to-point movement, and the focus lies on the FollowStem task. Continuous force feedback to follow the stem, in 6 DoF Cartesian space, is not possible in many configurations (see Subsection VII-B). Therefore, SAAFS is applied as follows. An extra state, to actively search for the stem, is added to the FollowStem task, see Fig. 14. The robot is rotated around its base (joint 1, Fig. 11), and using the knowledge that a reaction force from the plant is expected, force feedback, in 1 DoF joint space, in that same direction is used to monitor whether the stem is still being followed. Similarly, 1 DoF force feedback (joint 6, Fig. 11) is used to find the tomato. Here, the assumption is made that no other disturbances are present. This shows that this method's limitations include that solely based on the torque feedback, it is not possible to determine the source of the force. To conclude, SAAFS is used because continuous 6 DoF Cartesian wrench feedback failed to provide sufficient information; the results can be found in the following subsection.

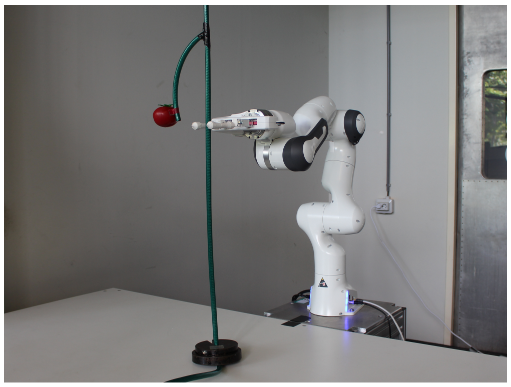


Fig. 13: Experimental set-up with dummy plant and *Panda* cobot. The dummy plant consists of a garden hose with a mock tomato attached, at a height of 88 cm measured from the table. The "stem" is suspended from a height of about 3 times that. Furthermore, because the dummy plant is very light-weight, a weight of 3 kg is used to pre-tension the "stem". This is enough to keep the stem fixed at both ends during the experiment. The EE is mounted with rollers that smoothly roll along the dummy plant stem.

C. Experimental validation: results

This subsection gives the results of the experiment described in Subsection VIII-B. Fig. 15 shows the nominal case: the system starts in state 1, and then switches between the two states until the guard on the torque in joint 6 is triggered, indicating that the robot EE has reached the tomato. Fig. 16

[4]In future work this could be determined using for example a camera.

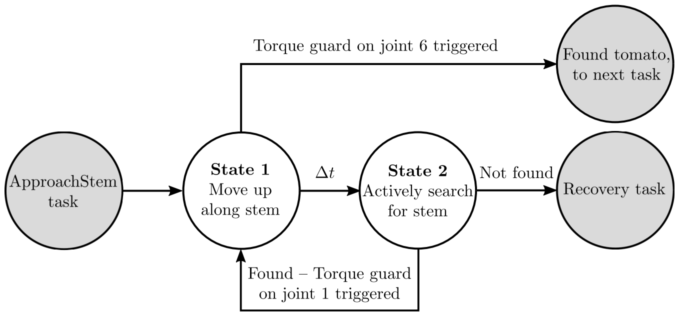


Fig. 14: FSM for the FollowStem task. The grey states are not part of the experiment. For the torque guards, the estimated external torque in a joint with joint axis perpendicular to the direction of the expected force is monitored.

on the other hand, illustrates the influence of a disturbance moving the stem out of the robot's grasp: during the third iteration of state 2 (after 14 seconds), the torque guard on joint 1 is *not* triggered, and instead of continuing to state 1 again, a recovery task (which falls outside the scope of this paper) is to be initiated. Appendix F discusses some more details that can be seen in the plots, for the interested reader. These are not considered very relevant for the main aspects that this experiment illustrates. The results show a successful implementation of disturbance monitors and SAAFS.

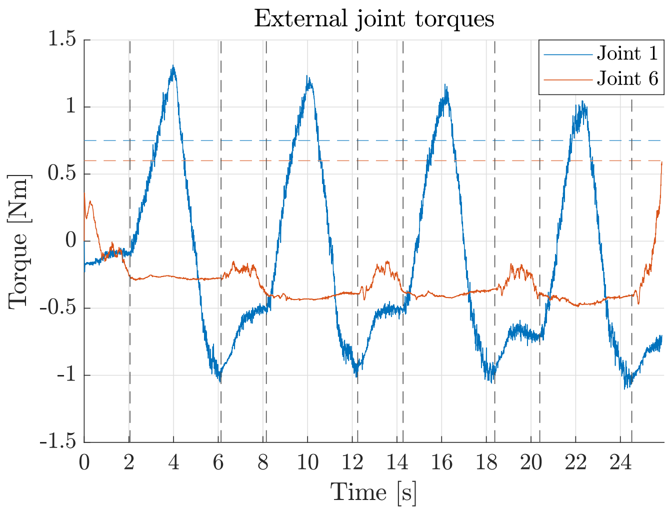


Fig. 15: Results of the FollowStem task, where the task is terminated after the tomato is reached. State transitions are indicated by the vertical black dashed lines. The torque guards on respectively joint 1 and 6 are indicated by the horizontal blue and orange dashed lines.

IX. Conclusions and Recommendations

In this paper, a methodology to design controllers robust against disturbances is introduced, as stated in C1. So-called "lazy" control, i.e. allowing the robot to do only what is sufficient, which leaves more room for robustness, contributes to C1. The tube-like monitor-based control described in

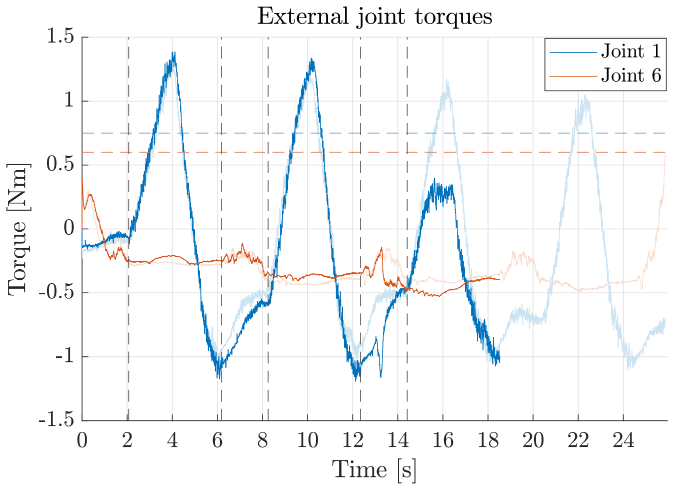


Fig. 16: Results of the FollowStem task, showing a prematurely terminated task. The stem is pulled away, so the robot cannot detect it anymore with its current sensors.

Subsection IV-A, is one way of creating such a lazy control, because of the ”good enough” approach. For the continuous control as described in Subsection IV-B, something similar applies. While the problem is formulated as a constrained optimization problem, all solutions that fall within the tolerances as specified in the task specification are acceptable. In other words, the first feasible control input is chosen. Lastly, Subsection IV-C is also in line with the lazy vision, because the kinematics solution is considered to be good enough. Moreover, reducing computational expenses, which in itself is also a lazy behavior, is achieved by the described lazy behaviors. Note that essentially precision is traded for robustness. This means that lazy control is only desirable in situations where the latter is more important then the former, which is the case in this paper. Where precision plays an important role though, and disturbances are limited, this is not a great approach.

Furthermore, the results of the experiment given in Section VIII-C show that monitoring disturbances through smart interpretation of sensor data can circumvent limitations in higher DoF force feedback as found in Subsection VII-B. As for the disturbance monitors, no pre-planned trajectory is required, because motion can just be generated until a guard is satisfied (reactivity); in this simple case state 1 and 2 are repeated until the tomato is found. Significant limitations of this approach lie in the extend to which task-related objects can be detected. SAAFS increases the set of configurations in which task-related force feedback is possible, under the assumption that no other disturbances act on the used sensors. Moreover, this does not mean that a 6 DoF force measurement is possible, it only exploits that in some configurations it is not necessary to actually have the 6 DoF force measurement. This means that tasks that need 6 DoF force feedback in all or many configurations, still require another method.

Future research includes integration of all the aspects of the proposed control as well as SAAFS on a real (not dummy) use case. Secondly, SAAFS can be improved by further automating the control of the pose of the robot in a way that (some of) the joints are configured such that the relevant partial force feedback is possible. Also, more research is required for isolating different sources of disturbances. Lastly, the term ”lazy” needs to be more clearly defined.

X. Acknowledgements

I would like to thank my day-to-day supervisor Jordy Senden for his support during my MSc thesis, his inputs gave me the opportunity to grow both on a professional as well as personal level. Furthermore, I want to thank René van de Molengraft, who continuously helped to keep my research goals sharp, and reminded me to sometimes take a step back, and look at my work from a more distant perspective, in order to prevent derailing from the goal too much. Also, I would like to thank Herman Bruyninckx, for always critically looking at my work and giving me valuable feedback and guidance, which helped me to keep progressing during the project. Lastly, I want to thank my friends and family, for encouraging me throughout this project. Especially Pim, who had to endure me during the most stressful periods, but always kept patient and supportive.

References

[1] J. Avelar, M. J. G.. van de Molengraft, and H. P. J. Bruyninckx. Lazy control of an anthropomorphic robotic arm, 2019.

[2] C.W. Bac, T. Roorda, R. Reshef, S. Berman, J. Hemming, and E.J. van Henten. Analysis of a motion planning problem for sweet pepper harvesting in a dense obstacle environment. *Biosystems Engineering*, 146:85–97, 2016.

[3] M. Boryga, A. Graboś, P. Kołodziej, K. Gołacki, and Z. Stropek. Trajectory Planning with Obstacles on the Example of Tomato Harvest. *Agriculture and Agricultural Science Procedia*, 7:27–34, 2015.

[4] H. Bruyninckx. *Building blocks for the Design of Complicated Systems featuring Situational Awareness*. 2021. Retrieved April, 2021, from https://robmosys.pages.gitlab.kuleuven.be/.

[5] X. Cao, X. amd Zou, C Jia, M Chen, and Z. Zeng. RRT-based path planning for an intelligent litchi-picking manipulator. *Computers and Electronics in Agriculture*, 156:105–118, 2019.

[6] Pasquale Chiacchio, Stefano Chiaverini, Lorenzo Sciavicco, and Bruno Siciliano. Closed-Loop Inverse Kinematics Schemes for Constrained Redundant Manipulators with Task Space Augmentation and Task Priority Strategy. *The International Journal of Robotics Research*, 10(4):410–425, 1991.

[7] Antonio Franchi and Anthony Mallet. Adaptive Closed-loop Speed Control of BLDC Motors with Applications to Multi-rotor Aerial Vehicles. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5203–5208, Singapore, 2017.

[8] Alessandro Gasparetto, Paolo Boscariol, Albano Lanzutti, and Renato Vidoni. path planning and trajectory planning algorithms: A general overview. pages 3–27, 03 2015.

[9] A. Gonzalez and D. Odloak. A stable mpc with zone control. *Journal of Process Control - J PROCESS CONTROL*, 19:110–122, Jan 2009.

[10] P. J. M. de Groot, C. A. Lopez Martinez, M. J. G. van de Molengraft, and H. P. J. Bruyninckx. Low-cost End-effector and Controller Design for a Compliant Autonomous Mobile Robot, 2018.

[11] Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. Robot collisions: A survey on detection, isolation, and identification. *IEEE Transactions on Robotics*, 33(6):1292–1312, 2017.

[12] Lucas Joseph, Joshua K. Pickard, Vincent Padois, and David Daney. Online velocity constraint adaptation for safe and efficient human-robot workspace sharing. *IEEE International Conference on Intelligent Robots and Systems*, pages 11045–11051, 2020.
[13] Nicolas Mansard and François Chaumette. Task Sequencing for High Level Sensor-Based Control. Technical Report 1, 2007.
[14] A. Nikou, C. K. Verginis, and D. V. Dimarogonas. A tube-based mpc scheme for interaction control of underwater vehicle manipulator systems. In *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, pages 1–6, Nov 2018.
[15] Azamat Shakhimardanov, Herman Bruyninckx, Marieke Copejans, and Ruben Smits. Popov-Vereshchagin algorithm for linear-time hybrid dynamics, control and monitoring with weighted or prioritized partial motion constraints in tree-structured kinematic chains. 2014.
[16] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 1 edition, 2005.
[17] A. F. Vereshchagin. Modelling and control of motion of manipulational robots. *Soviet journal of computer and systems sciences*, 27(5):29–38, 1989.
[18] Daniel E. Whitney. Resolved Motion Rate Control of Manipulators and Human Prostheses. *IEEE Transactions on Man-Machine Systems*, 10(2):47–53, 1969.
[19] M. Wuyts. Model Predictive Control Schemes using a Feasible Tunnel Representation for Motion Planning in an Unstructured Environment. Master's thesis, KU Leuven, 2020.

APPENDIX A
CURRENT AGRO-ROBOTIC SOLUTIONS

Table III presents a (non-exhaustive) list of existing autonomous agricultural systems.

TABLE III: Examples of agro-robotic solutions

Project (year)	**Location**	**Fruit**
Abundant Robotics (2015)	Hayward, CA, USA	Apple
Dogtooth (2014)	Royston, England	Strawberry
FFRobotics (2014)	Bnei Darom, Israel	Apple
Fieldwork Robotics (2017)	Plymouth, England	Raspberry
Octinion (2014)	Leuven, Belgium	Strawberry
Priva (2016)	De Lier, Netherlands	Tomato
Ripe Robotics (2018)	Shepparton, Vic, Australia	Various fruits
Root AI (2018)	Somerville, MA, USA	Tomato
Saia (2017)	Wageningen, Netherlands	Tomato
Tevel (2017)	Tel Aviv, Israel	Apple

APPENDIX B
TOLERANCES AND REFERENCE REGIONS

Including tolerances in the error signals, realizes a reference region instead of a reference trajectory. The error is multiplied by 0 while it remains within the specified tolerance, and by 1 when it is not. Gradual changes in the error, to avoid discontinuities in the control signal, are realized by using sigmoid functions. The function used is the logistic function. The pseudo-code is presented in Algorithm 1. Note that the tolerance σ_e can be made variable, e.g. it can be made dependent on the current position. Furthermore, this algorithm gives a symmetric tolerance around e_k, which can be changed by using two different σ_e's.

Algorithm 1: Implement tolerance, such that when the error value is within the tolerance the error is zero (for a 1 DOF system error)

```
Input      : $e_k$  // System error
Output     : $e_k^*$  // System error after tolerance
Parameters: $L$  // Maximum of the S-curve
            $k$  // Steepness of the S-curve
            $\sigma_e$  // Midpoint of the S-curve, tolerance
Variables  : $f_e$  // Multiply error by this factor

1 // Error scaling factor computation
2 if $e_k > 0$ then
3     $f_e = L/(1 + \exp(-k * (e_k - \sigma_e)))$;
4 else
5     $f_e = L/(1 + \exp(-k * (-e_k - \sigma_e)))$;
6 // Error after tolerance applied
7 $e_k^* = e_k * f_e$;
```

Appendix C
Panda Robotic Manipulator Reference Frames

The reference frames that are used in this document are given in this appendix. A fixed world frame is attached to the base of the robot. Furthermore, a frame is attached to the EE. Both are depicted in Fig. 17.

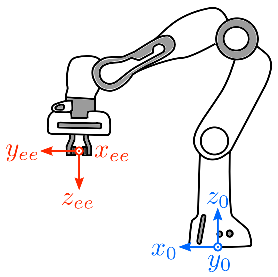


Fig. 17: Base frame (blue) and EE frame (red)

Appendix D
Adaptive Bias Adaptive Gain Control

In Algorithm 2 the ABAG controller, as developed by [7], is presented. The output of the controller is a scaling factor. This is based on the current and past error signs (the higher the value for α, the bigger the influence of the past error signs), i.e. the magnitude of the error is not considered. The scaling factor is eventually multiplied with the maximum value of the control input. For a more in-depth explanation of the algorithm, the reader is referred to [7].

Algorithm 2: ABAG Control Algorithm [7]

```
Input      : $e_k \in \mathbb{R}$  // System error
Output     : $u_k \in [0,1]$  // Control input scaling factor
Parameters : $\alpha \in (0,1)$  // Error sign filtering factor
             $\bar{e}_b \in (0,1)$  // Bias adaption threshold
             $\delta_b \in (0,1)$  // Bias adaption step
             $\bar{e}_g \in (0,1)$  // Gain adaption threshold
             $\delta_g \in (0,1)$  // Gain adaption step
Variables  : $\bar{e}_k \in [-1,1]$  // Low-pass filtered error sign
             $b_k \in [0,1]$  // Adaptive bias term
             $g_k \in [0,1]$  // Adaptive gain term

1  $k = 0, u_0 = \bar{e}_0 = g_0 = b_0 = 0$;
2  while ++$k$ do
3      // Error sign low-pass filtering
4      $\bar{e}_k = \alpha \bar{e}_{k-1} + (1-\alpha)\text{sgn}(e_k)$;
5      // Adaptive Bias term update
6      $b_k = \text{sat}_{[0,1]}(b_{k-1} + \delta_b \text{hside}(|\bar{e}_k| - \bar{e}_b)\text{sgn}(\bar{e}_k - \bar{e}_b))$;
7      // Adaptive Gain term update
8      $g_k = \text{sat}_{[0,1]}(g_{k-1} + \delta_g \text{sgn}(|\bar{e}_k| - \bar{e}_g))$;
9      // Control input scaling factor computation
10     $u_k = \text{sat}_{[0,1]}(b_k + g_k \text{sgn}(e_k))$;
```

Appendix E
Configuration Dependency Experiment: Feedback Control Law

The desired joint torques τ_d in the configuration dependency experiment (subsection VII-B) are calculated through Eq. 7 (subsection VI-C), where the virtual force on the EE is calculated as:

$$F_{virtual} = K(p_0 - p_c), \tag{16}$$

resulting in:

$$\tau_d = J^T(q)K(p_0 - p_c). \tag{17}$$

Here, $F_{virtual}$ is a 3×1 force vector, K is a 3×3 diagonal gain matrix, p_0 is a 3×1 (initial) position vector, and p_c is a 3×1 (current) position vector. τ_d is a 7×1 joint torque vector and $J(q)^T$ is the transpose of the 6×7 manipulator Jacobian. The loop is closed around the position[5] of the EE, taking the initial position as reference.

Appendix F
FollowStem External Torque Analysis

This appendix elaborates on some interesting details that can be seen in the plots of the external torques plotted in Fig. 15 and Fig. 16, subsection VIII-C. First, The height of the peaks in the torque in joint 1 during the state 2 iterations decreases while traveling up the stem. The fake plant is essentially a string fixed at both ends, meaning that for a similar displacement, the least force is felt in the middle of the string. The robot cannot reach above the lower half of the stem, which could explain the decrease in torque seen. To strengthen this belief, two similar experiments are executed. One where the stem is present, and one where it is removed, see Fig. 18. From this it can be concluded that the peaks indeed decrease more when there is interaction with the stem. Except for the last peak, where an increase is seen (for both experiments); this is probably due to the fact that the robot has reached the end of its workspace.

Second, the torque in joint 1 increases during the state 1 iterations, creating a "pre-bump" before the peak in state 2. Because this bump is also visible in the plots in Fig. 18, it can be concluded that it is not caused by the dummy plant dynamics, nor by the roller dynamics. A potential explanation is thus the robot dynamics. In the upper plot of Fig. 19, the estimated external joint torque in joint 1 is plotted for rotating joint 1 in the opposite direction (left instead of right, when facing the front of the robot). The corresponding lower plot (in Fig. 19) shows that during the state 1 iterations, the joint position $q1$ is more or less constant in both cases, as expected. The "bump" in the upper plot is not visible when rotating left though, which could still imply some configuration dependency.

[5]Orientation errors are considered zero in this experiment, because no or negligible perturbation in the rotational directions is applied

Third, there is a slight increase and subsequent decrease in the torque in joint 6 during the state 1 iterations, so where the robot EE goes up, i.e. between approximately 6–8, 12–14, and 18–20 seconds in Fig. 15. This could be caused by inertial effects from accelerating up.

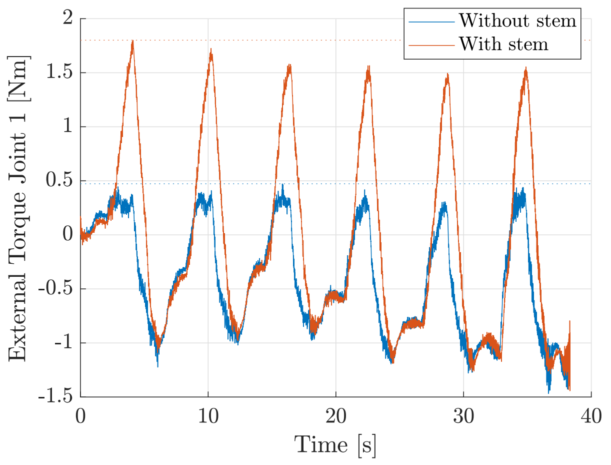


Fig. 18: FollowStem task with vs. without stem. Note that in the experiment with stem, no tomato is present, and motion continues until the end of the robot workspace, similar to the experiment without stem. Furthermore, the guard on joint 1 is temporarily turned off, and the rollers are removed from the EE, i.e. during state 1 there is no contact with the stem.

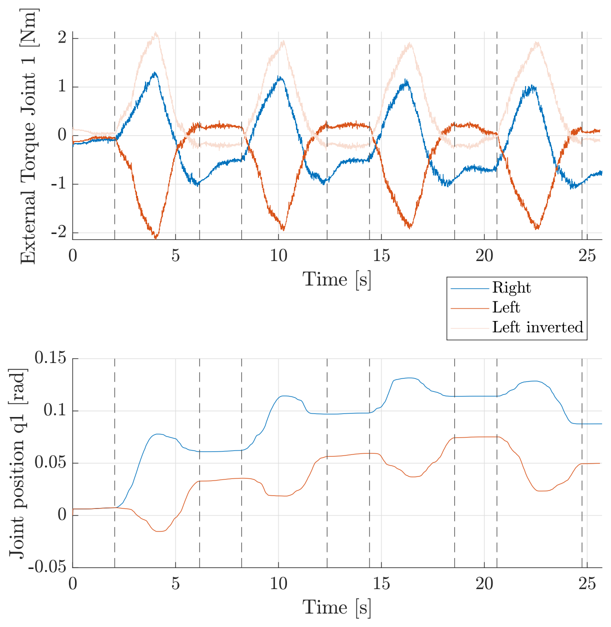


Fig. 19: FollowStem task, rotating joint 1 in opposite directions during state 2. To make comparing the plots easier, a mirrored (in the Y-axis) plot for the external torque in joint 1 when rotating left is also provided.