Enhancing Constraint-Based Robot Task Specification with Dual Controller and Estimator Synthesis

Ruan Viljoen[1,2], Johan Ubbink[1], Goele Pipeleers[1], Wilm Decré[1], Erwin Aertbeliën[1], Joris De Schutter[1]

***Abstract*— Task specification for robot manipulators continues to be a time consuming and expensive process, limiting their rapid deployment for industrial applications. This is especially true for the automation of sensor-based tasks, where the robot should adapt to uncertainties and disturbances in the environment. Previous work showed how robot task specification can be eased by using the task function approach to synthesize a robot controller. The main novelty of this paper is to further ease task specification by also using the task function approach to synthesize an estimator, which can be used to estimate geometric position uncertainties and the disturbances acting on them. Furthermore, as a result of the duality between control and estimation, an existing constraint-based robot task specification framework can be reused to perform both control and estimation without modification. The approach is validated on the well-known force-controlled contour following task. It is shown that, as expected, when disturbances are estimated and compensated for, the force tracking errors can be substantially decreased, while the contour following speed can be substantially increased. The presented approach is quite generic and can be applied in many practical applications to significantly improve the performance of constraint-based controllers for sensor-based robotic tasks.**

I. Motivation and background

While robots have traditionally been used for automating repetitive tasks in highly structured environments, there is an increasing demand for robots to perform more complex tasks in less-structured environments. However, this increased complexity results in longer application development and deployment times, and hence a higher automation cost. To address this issue, robot task specification frameworks have been designed with the goal to enable easy specification of sensor-based controllers that can react to the disturbances in these less-structured environments.

The pioneering research in this field focused on specifying the desired behavior of the robot (e.g. position, velocity, or contact force) directly in the *task space* [1, 2]. It was found that by using the Task Function Approach (TFA) proposed in [3], complex sensor-based behaviors could be implemented with relative ease. This approach is referred to as *constraint-based robot task specification*, and several software frameworks have been developed over time based on these ideas (e.g. COMRADE [4], iTaSC [5], eTaSL [6], SOT [7]). Some of these frameworks perform only kinematic control and rely on the internal motion controllers of the robots [8] for its execution, such as [4–6], while other frameworks also allow for full dynamic control and directly output control torques to the robots, such as [7]. The focus of this paper is on software frameworks that use the former approach. Using the TFA in such frameworks allows for easy tuning of the sensor-based constraint controllers, as it imposes a desired closed-loop response, with a bandwidth that is required to stay sufficiently below the bandwidth of the robot's internal motion controllers, which is easily determined.

However, purely reacting to sensor measurements can lead to tracking errors, caused by unmodeled disturbances. These disturbances can be accounted for by incorporating a model of the environment and explicitly estimating the disturbances based on sensor measurements, as demonstrated in [9] for a 1-degree-of-freedom (1–dof) force tracking task. The Kalman filter family stands out as the most widely used estimator, as it involves a probabilistic approach that is provably optimal under certain assumptions [10]. However, these assumptions do not hold in most applications, making the tuning of the process and measurement noise covariance matrices a tedious and time-consuming effort. An alternative approach is to use the Luenberger observer with pole placement [11], a deterministic estimator that imposes a desired closed-loop response for the estimated state variables.

Given the TFA's control approach, the duality which exists between control and estimation, and the inspiration provided by the Luenberger observer, the following question arises: is it possible to also implement an estimator using the TFA and will this also result in easy tuning of the estimator? This paper addresses this question and shows how the TFA can in fact be used to synthesize an estimator, the tuning of which only relies on knowing the bandwidth of the sensor noise. If this bandwidth is not specified by the manufacturer of the sensor, it can be easily determined experimentally. The design of the estimator is presented in a unified framework that uses the TFA to synthesize both the controller and estimator, exploiting the duality that exists between them.

This concept was tested using an existing task specification framework, eTaSL [6], and it was found that, by also formulating the estimator using the TFA, the framework can be used without modification to implement and easily tune both the controller and estimator. From a pragmatic point of view, this offers a significant advantage, as no additional software has to be developed and maintained.

A force-controlled contour following application was selected to validate the performance of the proposed approach. Contour following has become a benchmark in literature, as it represents a challenging task requiring accurate force control, and is also integral to many industrial applications

This research was supported by project G0D1119N of the Research Foundation - Flanders (FWO - Flanders).

[1] All authors are with the Department of Mechanical Engineering, KU Leuven, and Flanders Make@KU Leuven, Leuven, Belgium.

[2] Corresponding author: ruan.viljoen@kuleuven.be

(e.g. grinding [12], deburring [13], shape recovery [14]). Even though this task can be accomplished using only sensor feedback and without the use of an estimator, it has been shown that the use of an estimator can improve the tracking accuracy [5]. When estimators are used for contour following, it is typically done using an Extended Kalman filter (e.g. [5]) or a use-case-specific approach (e.g. [15]). However, these approaches introduce significant complexity, which we aim to reduce.

It has also been shown that the tracking performance of the contour following can be improved by incorporating a CAD model [16], or using additional sensors, such as vision [17]. This additional information can be used by a feedforward controller [17], or by using preview control [18]. However, for the sake of simplicity, we demonstrate our approach on an application that only relies on force measurements. This is because contour following in itself is not the focus of the paper, but is only used to demonstrate the advantages of the proposed approach. Nonetheless, the approach allows for incorporating CAD models and additional sensors.

This paper has the following outline. Section II shows how the TFA can be extended to specify both the controller and estimator while exploiting the duality between them. Section III describes the contour following application used to validate this approach and formulates its specification. Experimental results are provided in Section IV, which is followed by a discussion and conclusion in Section V.

II. Constraint-based robot task specification

Part A of this section models the overall system, defining the *system states*, *system dynamics*, and *system outputs*. Part B describes the *system constraints*, which are used to specify the desired behavior of the system. Additionally, it introduces the TFA as a way to enforce the system constraints. Part C shows how the control inputs and estimated disturbances can then be found by applying the TFA.

A. System model

Most robot applications involve motion. Motion can always be seen as relative motion between two objects. In many applications, one object is manipulated by the robot, hence its motion can be controlled by the robot, while the other object is external to the robot (i.e. belongs to the robot environment), and hence its motion cannot be controlled by the robot. However, an estimate of its position or orientation is usually available from a CAD model, from on-line teaching of the robot task, or from sensor measurements. Still, the true position or orientation of the external object is uncertain, especially if it is subject to disturbances. These disturbances affect the relative motion between the two objects. This is illustrated in Fig. 1 (a) and (b) for position and orientation, respectively. This simple model can be extended to motion in contact where contact forces and moments occur, as forces and moments are always generated in a contact between two objects. If one object is manipulated by the robot, and the other belongs to the environment, disturbances in the position or orientation of the latter affect the contact force or moment.

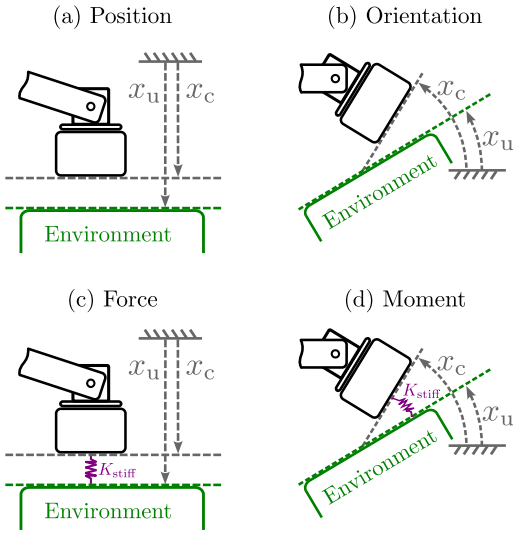

Fig. 1. Illustrations of 1–dof relative motion between the robot and the environment: in free space with a desired position (a) or rotation (b); in contact with resulting force (c) or moment (d). x_c represents a state that can be controlled by the robot, while x_u represents an uncontrollable state.

This is illustrated in Fig. 1 (c) and (d), where a spring with stiffness K_{stiff} is introduced as a simple model for the contact dynamics.

In Fig. 1, x_c represents the 1–dof position or orientation of the manipulated object. Generalizing to multiple degrees of freedom, we introduce controllable states $\mathbf{x}_\text{c}$ comprising of both *robot joint states* $\mathbf{x}_\text{q} \in \mathbb{R}^{n_q}$ and *virtual states* $\mathbf{x}_\text{v} \in \mathbb{R}^{n_v}$, the latter of which are used to model additional controllable states which arise in the specification of tasks, such as an imposed degree of advancement along a path to be followed by the robot. In Fig. 1, x_u represents the 1–dof position or orientation of the environment. Similarly, we introduce uncontrollable states $\mathbf{x}_\text{u}$ that can be used to model geometric position or orientation uncertainties in multiple degrees of freedom in the environment. This concept is inspired by the uncertainty coordinates introduced in [5], where it is argued that a variety of different sensor-based applications can be modeled in this way. More formally, the states of the system are:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_\text{c} \\ \mathbf{x}_\text{u} \end{bmatrix} \qquad \mathbf{x}_\text{c} = \begin{bmatrix} \mathbf{x}_\text{q} \\ \mathbf{x}_\text{v} \end{bmatrix}. \tag{1}$$

Next, the dynamics of the system are modeled as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{x}}_\text{c} \\ \dot{\mathbf{x}}_\text{u} \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ \mathbf{d} \end{bmatrix}. \tag{2}$$

This model of the system assumes that the dynamics of the controllable states can be reliably controlled by using the *control input* $\mathbf{u} \in \mathbb{R}^{n_c}$, while the dynamics of the uncontrollable states are modeled as being affected by *exogenous inputs*, or *disturbances* $\mathbf{d} \in \mathbb{R}^{n_u}$. While the states $\mathbf{x}$ are defined at position level, it follows from (2) that $\mathbf{u}$ and $\mathbf{d}$ are inputs at velocity level. Similarly as for the controllable states, the control inputs are also divided into *desired robot joint velocities* $\mathbf{u}_\text{q} \in \mathbb{R}^{n_q}$ and *virtual control inputs* $\mathbf{u}_\text{v} \in \mathbb{R}^{n_v}$.

It is assumed that the robot can reliably execute the desired robot joint velocities $\mathbf{u}_\text{q}$. This assumption is valid when

using industrial manipulators, as they are equipped with high-performance internal motion controllers, provided the closed-loop bandwidth of the task controllers to be designed stays sufficiently below the bandwidth of the internal motion controllers. It is also assumed that the robot joint states $\mathbf{x}_\text{q}$ are accurately known at all times. Again this assumption is valid when using industrial manipulators, as they are equipped with joint encoders which provide accurate measurements of the robot joint positions at the control sampling rate.

The values of the virtual states $\mathbf{x}_\text{v}$ can be kept track of internally by taking into account the sampling time of the controller ΔT and using forward Euler integration:

$$\mathbf{x}_\text{v}^+ = \mathbf{x}_\text{v} + \Delta\text{T}\ \mathbf{u}_\text{v}, \tag{3}$$

where $\mathbf{x}_\text{v}^+$ is the state $\mathbf{x}_\text{v}$ at the next time step.

Next, we distinguish between two different types of system outputs, namely the *task outputs* $\mathbf{y}$ and the *measured outputs* $\mathbf{z}$. An example of a task output could be the Cartesian position of the robot end-effector, or the distance between a manipulated object and the environment as illustrated in Fig. 1. On the other hand, measured outputs relate to sensors, such as force, laser, or camera sensors. An output can be both a task output and a measured output, such as the distance between the robot and an obstacle, which has to be constrained, but is also measured using a camera.

The task outputs can be modeled using the *task function* $\mathbf{g}$, while the measured outputs can be modeled using the *measurement function* $\mathbf{h}$. Both the task function and the measurement function depend on the states $\mathbf{x}$ of the system, and can also have an explicit dependency on time t:

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, t) \qquad\qquad \mathbf{z} = \mathbf{h}(\mathbf{x}, t). \tag{4}$$

The measurement function can be used to produce *predicted measurements* $\hat{\mathbf{z}}$, provided that a *state estimate* $\hat{\mathbf{x}}$ is available. Since the controllable states $\mathbf{x}_\text{c}$ are either accurately measured, as is the case for the robot joint positions $\mathbf{x}_\text{q}$, or kept tracking of internally, as is the case for the virtual states $\mathbf{x}_\text{v}$, the estimate of the state is formed by the correctly known values for the controllable states $\mathbf{x}_\text{c}$, and estimated values for the uncontrollable states $\mathbf{x}_\text{u}$:

$$\hat{\mathbf{z}} = \mathbf{h}(\hat{\mathbf{x}}, t) \qquad\qquad \hat{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_\text{c} \\ \hat{\mathbf{x}}_\text{u} \end{bmatrix}. \tag{5}$$

B. Dual task function approach for control and estimation

The desired behavior of the system can be described using a constraint-based approach, where the constraints are either *task constraints* or *measurement constraints*. For task constraints, the desired behavior is that the task output $\mathbf{y}$ should track a desired task value $\mathbf{y}_\text{d}$. For measurement constraints, the desired behavior is that the predicted measurements $\hat{\mathbf{z}}$ should correspond to the actual measurements $\mathbf{z}_\text{m}$. The difference between the task output and desired task output is referred to as the *task error* $\mathbf{e}_\mathbf{y}$, while the difference between the predicted measurement and actual measurement is referred to as the *prediction error* $\mathbf{e}_\mathbf{z}$.

$$\mathbf{e}_\mathbf{y} = \mathbf{g}(\mathbf{x}, t) - \mathbf{y}_\text{d} \qquad\qquad \mathbf{e}_\mathbf{z} = \mathbf{h}(\hat{\mathbf{x}}, t) - \mathbf{z}_\text{m} \tag{6}$$

In previous work [6], the TFA was used to satisfy task constraints by imposing a first-order decay on task errors. We propose to similarly use the TFA to satisfy measurement constraints, by imposing a first-order decay on the prediction errors:

$$\frac{\text{d}}{\text{d}t}\mathbf{e}_\mathbf{y} = -\mathbf{K}\mathbf{e}_\mathbf{y} \qquad\qquad \frac{\text{d}}{\text{d}t}\mathbf{e}_\mathbf{z} = -\mathbf{L}\mathbf{e}_\mathbf{z}. \tag{7}$$

The feedback gain matrices $\mathbf{K}$ and $\mathbf{L}$ are chosen to be diagonal with entries K_i and L_i corresponding to the desired closed-loop time constants K_i^{-1} and L_i^{-1} of the individual task and measurement constraints. While the values of both K_i and L_i are chosen as large as possible, K_i should be chosen to be below the bandwidth of the robot's internal motion controllers, while L_i should be chosen below the bandwidth of the sensor noise. Both the bandwidth of the motion controllers and that of sensor noise can be easily determined experimentally if they are not known beforehand. This makes the overall tuning process for the controller and estimator fast and intuitive. It is useful to introduce the following short-hand notation when specifying individual tasks or measurements constraints:

$$y_i \xrightarrow{K_i} y_{\text{d},i} \qquad\qquad \hat{z}_i \xrightarrow{L_i} z_{\text{m},i} \tag{8}$$

While the TFA is typically used to enforce *position-level* constraints, it can also be used to specify *velocity-level* constraints. This can be done by setting the feedback gain to zero. A shorthand similar to (8) is introduced in (11):

$$\frac{\text{d}}{\text{d}t}(\mathbf{y} - \dot{\mathbf{y}}_\text{d} t) = 0 \tag{9}$$

$$\dot{\mathbf{y}} = \dot{\mathbf{y}}_\text{d} \tag{10}$$

$$\dot{y}_i \xrightarrow{\text{vel}} \dot{y}_{\text{d},i}. \tag{11}$$

C. Solving for control inputs and estimated disturbances

Once the task and measurement constraints have been specified, it is possible to solve for the control inputs and estimated disturbances, as shown in this section. However, it should be noted that in practice the following derivations are performed in an automated way, by making use of an automatic differentiation tool (e.g. the *expressiongraph* library of eTaSL [6]).

It is possible to find the control inputs which enable the decay in task errors by expressing the time-derivative of the task errors in terms of the control inputs $\mathbf{u}$:

$$\frac{\text{d}}{\text{d}t}\mathbf{e}_\mathbf{y} = \left.\frac{\partial \mathbf{e}_\mathbf{y}(\mathbf{x}, t)}{\partial \mathbf{x}}\right|_{\mathbf{x}_k, t_k} \dot{\mathbf{x}} + \left.\frac{\partial \mathbf{e}_\mathbf{y}(\mathbf{x}, t)}{\partial t}\right|_{\mathbf{x}_k, t_k} \tag{12}$$

$$= \mathbf{J}_{\mathbf{y},\mathbf{x}}\dot{\mathbf{x}} + \tfrac{\partial \mathbf{e}_\mathbf{y}}{\partial t} \tag{13}$$

$$= \begin{bmatrix} \mathbf{J}_{\mathbf{y},\mathbf{x}_\text{c}} & \mathbf{J}_{\mathbf{y},\mathbf{x}_\text{u}} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{d} \end{bmatrix} + \tfrac{\partial \mathbf{e}_\mathbf{y}}{\partial t}. \tag{14}$$

Equation (13) is the same as (12), but using a more compact notation. Equation (14) is found by splitting up the Jacobian term, and substituting the system dynamics from (2) into (13). Finally, the equation for finding the control inputs $\mathbf{u}$ which enforce the decay in task error can be expressed by

combining (7) and (14), showing the *feedback*, *feedforward* and *disturbance rejection* terms:

$$\mathbf{J}_{\mathbf{y},\mathbf{x}_\mathrm{c}}\mathbf{u} = -\overbrace{\mathbf{K}\mathbf{e}_\mathbf{y}}^{\text{feedback}} - \overbrace{\frac{\partial \mathbf{e}_\mathbf{y}}{\partial t}}^{\text{feedforward}} - \overbrace{\mathbf{J}_{\mathbf{y},\mathbf{x}_\mathrm{u}}\hat{\mathbf{d}}}^{\text{disturbance rejection}}. \tag{15}$$

The feedforward term arises from the fact that both the task function $\mathbf{g}$ and the desired task value $\mathbf{y}_\mathrm{d}$ are allowed to be time-varying. Solving (15) for $\mathbf{u}$ provides the desired control input. To evaluate (15), it is necessary to have both an estimate of the system states $\hat{\mathbf{x}}$, as seen in (12), and an estimate of the disturbance $\hat{\mathbf{d}}$. Because of this added complexity, the disturbance rejection term is often neglected, which results in tracking errors. It will now be shown how this estimation can in fact be performed in a simple way by using a similar approach as used by the controller.

As stated in Section II-A, the values of the controllable states are known. Hence, it is only necessary to estimate the values of the uncontrollable states $\hat{\mathbf{x}}_\mathrm{u}$, as well as the disturbances $\hat{\mathbf{d}}$. The estimates of the uncontrollable states can be updated as in (3), using forward Euler integration:

$$\hat{\mathbf{x}}_\mathrm{u}^{+} = \hat{\mathbf{x}}_\mathrm{u} + \Delta\mathrm{T}\,\dot{\hat{\mathbf{x}}}_\mathrm{u}. \tag{16}$$

This of course requires that $\dot{\hat{\mathbf{x}}}_\mathrm{u}$ is known. It is possible to find $\dot{\hat{\mathbf{x}}}_\mathrm{u}$ by going through a similar procedure as is done in (12) through (15), as a result of the similarity which exists between control and estimation. In the case of the controller, the goal is to accurately track desired task outputs, which is done by applying control inputs $\mathbf{u}$. In the case of the estimator, the task is to accurately estimate the uncontrollable states $\mathbf{x}_\mathrm{u}$, which can be done by applying the estimator correction $\dot{\hat{\mathbf{x}}}_\mathrm{u}$.

Similarly as for (12) through (14), the value of $\dot{\hat{\mathbf{x}}}_\mathrm{u}$ can be found by expressing the time-derivative of the prediction errors in terms of the correction term $\dot{\hat{\mathbf{x}}}_\mathrm{u}$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{e}_\mathbf{z} = \left.\frac{\partial \mathbf{e}_\mathbf{z}(\hat{\mathbf{x}},t)}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k,t_k} \dot{\hat{\mathbf{x}}} + \left.\frac{\partial \mathbf{e}_\mathbf{z}(\hat{\mathbf{x}},t)}{\partial t}\right|_{\hat{\mathbf{x}}_k,t_k} \tag{17}$$

$$= \mathbf{J}_{\mathbf{z},\mathbf{x}}\dot{\hat{\mathbf{x}}} + \frac{\partial \mathbf{e}_\mathbf{z}}{\partial t} \tag{18}$$

$$= \begin{bmatrix}\mathbf{J}_{\mathbf{z},\mathbf{x}_\mathrm{c}} & \mathbf{J}_{\mathbf{z},\mathbf{x}_\mathrm{u}}\end{bmatrix}\begin{bmatrix}\dot{\hat{\mathbf{x}}}_\mathrm{c}\\ \dot{\hat{\mathbf{x}}}_\mathrm{u}\end{bmatrix} + \frac{\partial \mathbf{e}_\mathbf{z}}{\partial t}. \tag{19}$$

The terms $\dot{\hat{\mathbf{x}}}_\mathrm{c}$ and $\dot{\hat{\mathbf{x}}}_\mathrm{u}$ represent the correction we want to make to the estimates of the states. Using the assumption that the controllable states $\mathbf{x}_\mathrm{c}$ are good estimates already, they do not contribute to the innovation and do not have to be corrected. Hence, it is only necessary to correct for the states $\mathbf{x}_\mathrm{u}$. We therefore set the values of $\dot{\hat{\mathbf{x}}}_\mathrm{c}$ to zero, and by combining (7) and (19) we arrive at:

$$\mathbf{J}_{\mathbf{z},\mathbf{x}_\mathrm{u}}\dot{\hat{\mathbf{x}}}_\mathrm{u} = -\mathbf{L}\mathbf{e}_\mathbf{z} - \frac{\partial \mathbf{e}_\mathbf{z}}{\partial t}. \tag{20}$$

The feedforward term $\frac{\partial \mathbf{e}_\mathbf{z}}{\partial t}$ arises because the measurement function $\mathbf{h}$ is allowed to be time-varying.

Correcting the estimate $\hat{\mathbf{x}}_\mathrm{u}$ can then be done using $\dot{\hat{\mathbf{x}}}_\mathrm{u}$ according to (16). Also, $\dot{\hat{\mathbf{x}}}_\mathrm{u}$ can be used as an estimate of the disturbance $\hat{\mathbf{d}}$ in (15). This approximation is valid if a constant-velocity model is a good approximation for the geometric uncertainty $\mathbf{x}_\mathrm{u}$ and when the estimator, and hence the innovation, reaches a steady state.

The sets of linear equations in (15) and (20) can be (partially) over- and/or under-determined. One way to find a solution for the general case is to use the approach of eTaSL [6], where a Quadratic Programming (QP) problem is formulated to find a solution. As explained in [6], this approach can also be used to enforce inequality constraints, which are useful for specifying joint position and velocity limits. The similarity between (15) and (20) is what enables an existing constraint-based task specification framework like eTaSL to synthesize both a controller and estimator.

III. Contour following application

A force-controlled contour following application is selected to validate the performance of the proposed control and estimation strategy. Fig. 2 shows the setup which consists of a 2D contour and a robot arm (Universal Robots UR10), equipped with a tool and JR3 force-torque sensor. Attached to the tool is a roller, which follows the contour while providing near-frictionless contact. For simplicity, it is assumed that the contour is in the horizontal plane of the world.

The code used to implement the application can be found here[1], and makes use of the eTaSL framework. The *expressiongraph* library of eTaSL allows for generating symbolic expressions of the task and measurement functions, based on the URDF file of the robot. These symbolic expressions are then used to automatically generate the necessary Jacobians by making use of automatic differentiation.

A. Geometric modeling

The following reference frames are defined: the world frame {w}, which coincides with the robot base; end-effector frame {ee}, which is known as a function of the joint states $\mathbf{x}_\mathrm{q}$; force sensor frame {sensor} and tool center point frame {tcp}, which are both attached to the end-effector. The {tcp} frame is placed at the center of the roller with its z-axis pointing up towards the robot end-effector.

Furthermore, following the *task frame formalism* [1, 2], a task frame {tf} is defined. Due to the circular symmetry of the roller, the contact between wheel and contour can

[1] https://github.com/ruanViljoen/Contour-Following

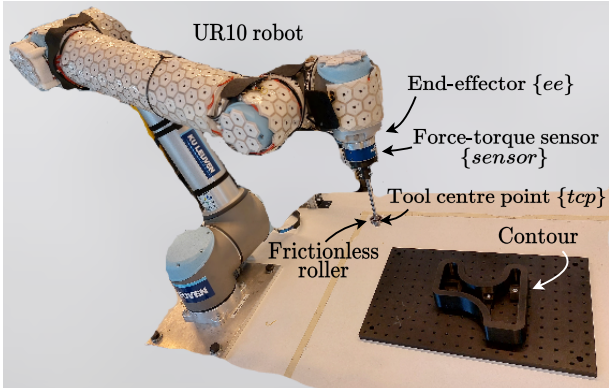

Fig. 2. Contour following setup where the robot is tasked to follow the contour of the object using force feedback.

be modeled as a contact between a point and a contour that is grown by the radius r of the roller. Therefore, the origin of $\{\text{tf}\}$ is placed at the origin of $\{\text{tcp}\}$. In the task specification, the orientation of $\{\text{tf}\}$ will be constrained to align with the contour normal. This can be done using two different scenarios. In scenario 1, $\{\text{tcp}\}$ is fixed in orientation to $\{\text{tf}\}$. This has the limitation that after having completed one revolution around the contour, $\{\text{tcp}\}$ has also rotated by one revolution, and the robot will have reached its joint limits. This problem is avoided in scenario 2, where $\{\text{tf}\}$ is allowed to rotate freely around the z-axis of $\{\text{tcp}\}$ by introducing an additional degree of freedom. Hence, the transformation between $\{\text{tcp}\}$ and $\{\text{tf}\}$ is expressed using a homogeneous transformation matrix

$$ {}^{\text{tf}}_{\text{tcp}}\mathbf{T}(x_{\text{v1}}) = \begin{bmatrix} \mathbf{R}(Z, x_{\text{v1}}) & 0_{3\times 1} \\ 0_{1\times 3} & 1 \end{bmatrix}, \tag{21} $$

where x_{v1} is a virtual state as introduced in Section II-A.

Since the application involves a point contact, no moments are generated, and only forces are considered. The contact forces ${}_{\text{sensor}}\mathbf{f}_{\text{m}}$ are measured within the sensor frame, but can be transformed to the task frame using

$$ {}_{\text{tf}}\mathbf{f}_{\text{m}} = {}^{\text{sensor}}_{\text{tf}}\mathbf{R}\ {}_{\text{sensor}}\mathbf{f}_{\text{m}}. \tag{22} $$

B. Constraint-based task specification

Contour following can be achieved using the following specification. Firstly, the robot should maintain a desired orientation between the task frame and the contour. In particular, the task frame should remain aligned with the contour surface normal. This is an application of the relative motion illustrated in Fig. 1 (b). Secondly, the robot should maintain a desired force between the roller and the contour. This is an application of the relative motion illustration in Fig. 1 (c). Both of these constraints will be affected by disturbances as the contour following progresses, caused by the curvature of the contour as well as changes in contact stiffness. By making use of the approach presented in Section II, these disturbances can be estimated and then compensated for by the controller. Finally, it is also necessary to specify constraints which keep the roller in a horizontal plane, as well as a constraint which enforces a desired tangential velocity along the contour. The specification of all these constraints are discussed in the remainder of this section.

1) Orientation constraint to align the task frame: The orientation of the $\{\text{tf}\}$ is constrained such that its x-axis is aligned with the contour normal, which is not known beforehand, and can change as the tool progresses along the contour. Therefore, the orientation of the contour normal is modeled as a rotational geometric uncertainty x_{u1} defined relative to the x-axis of the world. From Fig. 3 (a) it follows that the alignment error δ_{o} between the contour normal $\mathbf{n}$ and the x-axis of $\{\text{tf}\}$ is modeled as:

$$ \delta_{\text{o}} = \text{relativeAngle}(\, x_{\text{u1}}\, ,\, {}_{\text{tf}}\theta_{\text{yaw}}\,), \tag{23} $$

where ${}_{\text{tf}}\theta_{\text{yaw}}$ is the yaw angle of the task frame, and can be controlled by the robot. The relativeAngle function is used

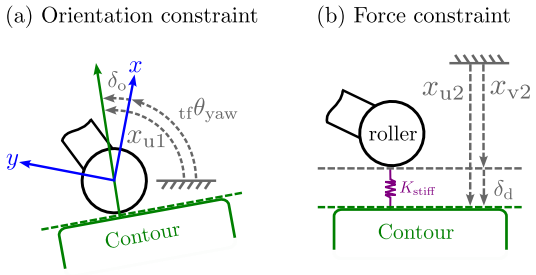

Fig. 3. An illustration of the orientation constraint (a) and force constraint (b). For the force constraint it is assumed that the task frame is aligned the contour. These constraints are examples of the relative motions shown in Fig. 1.

to deal with wrap-around. The alignment error δ_{o} can be measured indirectly, by noting that in the absence of friction, the contour normal coincides with the direction of the force:

$$ \delta_{\text{o,m}} = \text{arctan2}(\, {}_{\text{tf}}f_{\text{m,y}},\ {}_{\text{tf}}f_{\text{m,x}}\,). \tag{24} $$

The alignment error δ_{o} is set both as a task output to be controlled to zero, and as a measured output used to estimated the disturbance acting on x_{u1}. Firstly, δ_{o} is controlled to zero with a task constraint:

$$ g_1(\mathbf{x}) : \delta_{\text{o}} \xrightarrow{K_1} 0. \tag{25} $$

Secondly, δ_{o} is used to estimate the disturbance d_1 acting on uncontrollable state x_{u1} by imposing a measurement constraint:

$$ h_1(\mathbf{x}) : \hat{\delta}_{\text{o}} \xrightarrow{L_1} \delta_{\text{o,m}}. \tag{26} $$

2) Force constraint in the normal direction: The objective is to maintain a desired force f_d. This is achieved with an admittance control type of strategy, where the force is modeled by a deformation multiplied with a stiffness.

Assuming that the orientation constraint ensures that $\{\text{tf}\}$ stays aligned with the contour, the force constraint reduces to regulating the force along the x-axis of $\{\text{tf}\}$ as illustrated in Fig. 3 (b). In this figure, the position of the contour is modeled as a geometric position uncertainty x_{u2}, while the position of the roller is represented by the virtual state x_{v2}, which represents the cumulative movement of the roller along the x-axis of $\{\text{tf}\}$. Hence, x_{v2} is kept track of by integrating the translational velocity of the roller along the x-axis of the task frame, ${}_{\text{tf}}v_x$, during the task execution. Accordingly, the deformation δ_{n} between the roller and the contour is modeled as:

$$ \delta_{\text{n}} = x_{\text{u2}} - x_{\text{v2}}. \tag{27} $$

To obtain the normal contact force, δ_{n} is multiplied by the modeled contact stiffness K_{stiff}. This force is regulated to a desired value using the task constraint:

$$ g_2(\mathbf{x}) : K_{\text{stiff}}\, \delta_{\text{n}} \xrightarrow{K_2} f_{\text{d}}. \tag{28} $$

To estimate the disturbance d_2 acting on the uncontrollable state x_{u2} the following measurement constraint is specified:

$$ h_2(\mathbf{x}) : K_{\text{stiff}}\, \hat{\delta}_{\text{n}} \xrightarrow{L_2} f_{\text{m,x}}. \tag{29} $$

3) Planar constraints: For planar contour following, it is necessary to constrain the movement of the roller to a fixed horizontal plane. This can be done in two steps. First, the z-position p_z^{tf} of $\{\text{tf}\}$ is constrained to the known z-position of the contour p_z^{c} using the shorthand introduced in (8):

$$g_3(\mathbf{x}) : p_z^{\text{tf}} \xrightarrow{K_3} p_z^{\text{c}}. \tag{30}$$

Second, to keep the roller level with the contour, the roll and pitch of $\{\text{tf}\}$, $\theta_{\text{roll}}^{\text{tf}}$ and $\theta_{\text{pitch}}^{\text{tf}}$, are constrained with:

$$g_4(\mathbf{x}) : \theta_{\text{roll}}^{\text{tf}} \xrightarrow{K_4} 0, \qquad g_5(\mathbf{x}) : \theta_{\text{pitch}}^{\text{tf}} \xrightarrow{K_5} 0. \tag{31}$$

4) Velocity constraint in the tangential direction: The tool should move with a desired velocity v_{d} along the contour. Assuming that $\{\text{tf}\}$ is aligned with the contour, this can be achieved by using a task constraint:

$$g_6(\mathbf{x}) : {}_{\text{tf}}v_y \xrightarrow{\text{vel}} v_d, \tag{32}$$

where ${}_{\text{tf}}v_y$ is the y component of the roller velocity in $\{\text{tf}\}$.

IV. Experimental Results

The method was tested using two different contours, a circular contour with an effective radius R_{eff} of 6.5 cm (i.e. circle radius R + roller radius r), and a complex contour consisting of circular arc and straight-line segments as shown in Fig. 4. In all experiments, the desired contact force was 10 N, and a modeled stiffness $K_{\text{stiffness}}$ of 2500 N.m^{-1} was assumed.

The controller gains K_1,K_3,K_4 and K_5 where chosen to be 3 s^{-1}, while the force control gain K_2 was chosen to be 5 s^{-1}. These gains were tuned experimentally to be as large as possible but sufficiently below the bandwidth of the robot's internal motion controllers. As for the estimator gains, L_1 = L_2 = 3 s^{-1} were tuned experimentally, so as to also be as large as possible but sufficiently below the bandwidth of the force sensor's measurement noise.

The controller was implemented as an Orocos component on a desktop PC with a 12-core Intel Xeon E5-1650v3 CPU running Ubuntu 20.04. The control and estimation frequency was set to 200 Hz.

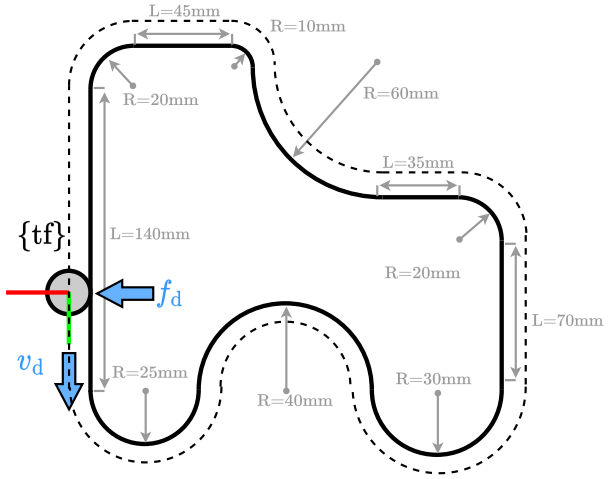

Fig. 4. Complex contour showing starting position and direction. The effective radius R_{eff} is shown with the dashed line.

Three experiments were conducted involving scenarios 1 and 2 (see Section III-A), as summarized in Table I.

TABLE I
Overview of Experiments

exp.	contour	v_{d} [cm.s^{-1}]	scenario	disturbances rejected	fig.
1	circle	4	1 **vs.** 2	none **vs.** d_1	5
2	circle	4	2	none **vs.** d_1 **vs.** d_1 & d_2	6
3	complex	2	2	none **vs.** d_1 & d_2	7, 8

scenario 1: {tcp} aligned with {tf}; scenario 2: {tcp} not aligned with {tf}
d_1: disturbance in orientation; d_2: disturbance in normal direction.

Experiment 1: The aim of experiment 1 was to compare the force tracking error with the circular contour for scenarios 1 and 2, while also evaluating the effect of adding disturbance rejection for d_1 in the orientation direction. For the circular contour, the steady-state force tracking error when using disturbance rejection for d_1 is expected to be zero, while as shown in [8], without disturbance rejection it is given by:

$$\Delta f = \frac{K_{\text{stiffness}}}{K_1 K_2} \frac{v_{\text{d}}^2}{R_{\text{eff}}}. \tag{33}$$

For a desired tangential velocity v_{d} of 4 cm.s^{-1}, (33) results in a steady-state error of 4.1 N.

Fig. 5 shows the results of experiment 1. For scenario 1, the force tracking behaves as expected, with a steady-state error of approximately 4.1 N initially, and approximately 0 N when the disturbance rejection is activated. However, after one revolution around the contour the robot reaches a joint limit and has to stop. This joint limit is avoided in scenario 2, but the force tracking error does not correspond with the predicted behaviors, and there is a sinusoidal tracking error even when rejecting the disturbance d_1. This force tracking error is caused by an unmodeled effect, such as a directional variation of the overall contact stiffness, or an eccentricity in the tool.

Experiment 2: The aim of experiment 2 was to show that the unmodeled disturbance observed in experiment 1, for scenario 2, could be compensated for by also performing disturbance rejection of d_2 in the normal direction. Fig. 6 shows the results of experiment 2. As expected, the force tracking error was reduced by adding the disturbance rejection of d_1, but the sinusoidal tracking error observed in experiment

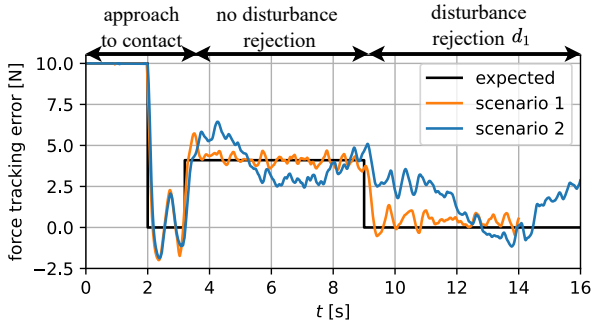

Fig. 5. Experiment 1: force tracking error for circular contour for the two scenarios, without and with disturbance rejection d_1.

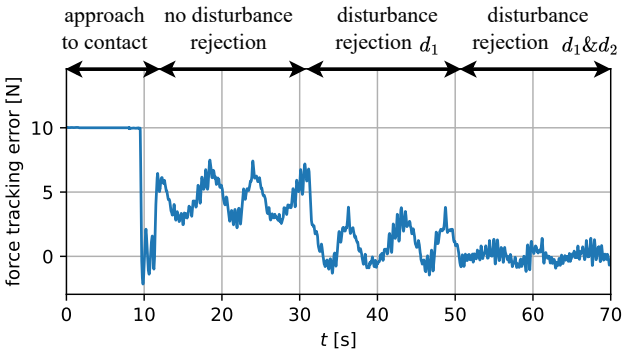

Fig. 6. Experiment 2: force tracking error for circular contour (multiple revolutions) and scenario 2, with different disturbance rejection modes.

1 was still present. When the disturbance rejection of both d_1 and d_2 was used, this tracking error was further reduced.

Experiment 3: The aim of experiment 3 was to show how the force tracking error could also be reduced when using the complex contour of Fig. 4. Here v_d was specified as $2\,\mathrm{cm.s^{-1}}$, the fastest velocity that could be maintained without losing contact when no disturbance rejection was used. Fig. 7 compares the force tracking error with and without the disturbance rejection of both d_1 and d_2. By using the disturbance rejection, the RMS of the force tracking error during the contour following phase could be decreased by 79.2%, from 2.4 N to 0.5 N. Fig. 8 shows both the ground truth and estimated values of the disturbance d_1 in the orientation tracking direction, showing accurate tracking. The ground truth of d_1 was obtained by dividing v_d by the effective radii of the circular arc segments. A benefit of using the disturbance rejection was that faster tangential velocities could be maintained without losing contact during the contour following exercise. Without disturbance rejection, tangential speeds of up to $2\,\mathrm{cm.s^{-1}}$ could be specified without losing

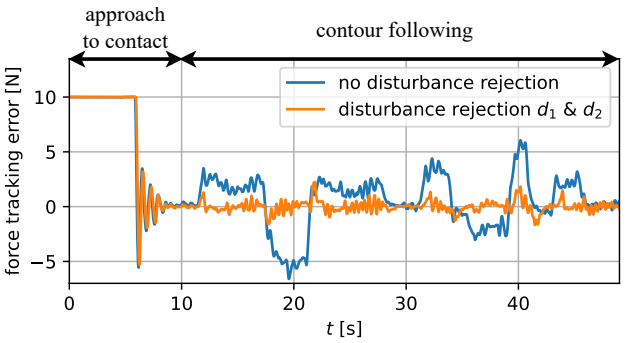

Fig. 7. Experiment 3: force tracking error for complex contour (1 revolution) and scenario 2, without and with disturbance rejection of d_1 and d_2.

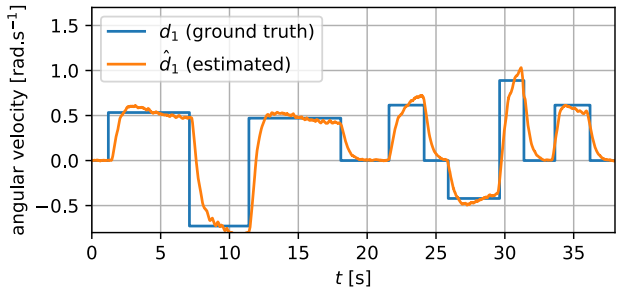

Fig. 8. Experiment 3: round truth versus estimated values of disturbance d_1.

contact with the contour. With disturbance rejection, this value increased to $3.5\,\mathrm{cm.s^{-1}}$, representing a 75% improvement.

V. Discussion and Conclusion

The main novelty of this paper is that it presented a unified framework for the specification of sensor-based robot tasks, enabling the dual synthesis of a controller and an estimator using the TFA. The advantage of this approach is two-fold. Firstly, it allows for easy tuning of the controller and estimator, which enables more rapid application deployment. Secondly, as this approach makes use of the duality between control and estimation, the same software component can be used for both the controller and estimator. In fact, it was found that an existing constraint-based task specification tool, which is normally used only for control, could be used without modification to also perform the estimation. As a further advantage (not shown in the paper), this allows us to easily incorporate hard or soft constraints between the geometric uncertainties that have to be estimated.

While the approach for the estimator was inspired by the Luenberger observer with pole placement, it is conceptually different, as it imposes a decay on the innovations whereas the Luenberger observer imposes a decay on the state estimation errors. Still, if the Jacobian matrix in (20) is diagonal, both approaches yield the same result. We refer to this as the case of *decoupled uncertainties*.

As an illustration, we showed how this approach can be applied to a complex sensor-based task, namely the well-known force-controlled and model-free planar contour following task. As expected based on previous studies, experimental validation confirmed that the performance of the task could be significantly improved by estimating and compensating for the geometric disturbances. More specifically, when tracking the contour with a tangential speed of $2\,\mathrm{cm.s^{-1}}$, the RMS force error could be decreased by 79.2% when the disturbance rejection was used. Additionally, the maximum speed that could be achieved without losing contact was increased by 75% when using the disturbance rejection. The results achieved by using this simpler approach are comparable to those which use more complex estimation strategies, such as in [15] and [18], which validates the usefulness of the proposed approach. Moreover, this experimental validation could be carried out with the existing software framework eTaSL [6], without any modification.

As for the tuning, the control and estimator gains have an intuitive interpretation, being the inverse of the time constants of the first-order decay of the control and estimation errors. All achieved control gains were in the range 3 $\mathrm{s^{-1}}$ to 5 $\mathrm{s^{-1}}$ that we expected based on our previous experience with constraint-based control of the UR10 and comparable robots such as the KUKA LBR iiwa. On the other hand, the two estimator gains that we achieved were only 3 $\mathrm{s^{-1}}$. This is lower than we expected, because usually estimator gains can be tuned higher than control gains, as correcting an estimate does not involve any actual robot motion, hence there is no limitation related to the bandwidth of the robot's motion controllers. In the contour tracking application, these small

estimator gains result from the fact that force sensors have significant sensor noise. In fact, a low-pass prefilter was used in the experiments to attenuate the noise as much as possible. In contrast, in preliminary experiments with laser distance sensors, which are known to have much lower noise levels, we found that much larger estimator gains (up to 15 s^{-1}) could be used, yielding much more accurate performance in a non-contact surface tracking experiment. As a conclusion, estimator gains are limited by the bandwidth of the low-pass filter used to attenuate the sensor noise, if any.

One limitation of this approach is that it only models disturbances acting on the uncontrollable states of the system, and assumes a constant velocity model for these disturbances. Even though a constant velocity model might seem limiting, it is known to provide surprisingly good performance [19]. Nonetheless, allowing for more complex disturbance models will improve the generalizability of the approach.

Also, our validation experiment only involved decoupled uncertainties, resulting in decoupled estimation constraints (26) and (29). While this case is very common in practical applications, it is not a restriction of the proposed estimator.

The approach was only verified using one application. However, as argued in [5], many sensor-based tasks can be described in terms of relative motion between the robot and the environment. For example, as explained in Section III-B, the contour following could be achieved by combining two of the four types of relative motion shown in Fig. 1. Whenever such type of relative motion is part of the application, the proposed approach can be used to estimate and compensate for disturbances caused by motion of the environment. Hence, we believe that the proposed method has the potential to simplify the specification of a wide range of sensor-based tracking tasks, both in free space (e.g. visual servoing) and in contact, allowing for faster robot application deployment.

REFERENCES

[1] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, 1981.

[2] H. Bruyninckx and J. De Schutter, "Specification of force-controlled actions in the 'task frame formalism' : A synthesis," *IEEE Transactions on Robotics and Automation*, 1996.

[3] C. Samson, M. Le Borgne, and B. Espiau, *Robot Control, the Task Function Approach* (Combinatorial Scientific Computing). Clarendon Press, 1991.

[4] W. Witvrouw, P. Van de Poel, and J. De Schutter, "Comrade: Compliant motion research and development environment," *IFAC Proceedings Volumes*, 1995.

[5] J. De Schutter *et al.*, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *The International Journal of Robotics Research*, 2007.

[6] E. Aertbeliën and J. De Schutter, "Etasl/etc: A constraint-based task specification language and robot controller using expression graphs," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[7] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," in *International Conference on Advanced Robotics (ICAR)*, 2009.

[8] J. De Schutter and H. Van Brussel, "Compliant robot motion ii. a control approach based on external control loops," *The International Journal of Robotics Research*, 1988.

[9] J. De Schutter, "Improved force control laws for advanced tracking applications," *1988 IEEE International Conference on Robotics and Automation*, 1988.

[10] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, 1960.

[11] D. Luenberger, "An introduction to observers," *IEEE Transactions on Automatic Control*, 1971.

[12] T. Thomessen and T. K. Lien, "Robot control system for safe and rapid programming of grinding applications," *Industrial Robot*, 2000.

[13] G. Ziliani, G. Legnani, and A. Visioli, "A mechatronic design for robotic deburring," in *Proceedings of the IEEE International Symposium on Industrial Electronics, 2005. ISIE 2005.*, 2005.

[14] S. Ahmad and C. Lee, "Shape recovery from robot contour-tracking with force feedback," in *Proceedings., IEEE International Conference on Robotics and Automation*, 1990.

[15] N. Wang, J. Zhou, K. Zhong, X. Zhang, and W. Chen, "Force tracking impedance control based on contour following algorithm," in *Intelligent Robotics and Applications*, Springer International Publishing, 2022.

[16] S. Demey, H. Bruyninckx, and J. De Schutter, "Model-based planar contour following in the presence of pose and model errors," *The International Journal of Robotics Research*, 1997.

[17] J. Baeten, H. Bruyninckx, and J. De Schutter, "Integrated vision/force robotic servoing in the task frame formalism," *The International Journal of Robotics Research*, 2003.

[18] B. Yong, "Contour-following of a force-controlled industrial robot using preview control," in *KSME International Journal*, 1997.

[19] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll, "What the constant velocity model can teach us about pedestrian motion prediction," *IEEE Robotics and Automation Letters*, 2020.