# A Constraint-Based Programming Approach for Robotic Assembly Skills Implementation

Matteo Parigi Polverini[a,b,*], Andrea Maria Zanchettin[a], Paolo Rocco[a]

[a]*Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza Leonardo Da Vinci 32, 20133, Milano, Italy*
[b]*Advanced Robotics Department (ADVR), Istituto Italiano di Tecnologia, Via Morego 30, 16163, Genova, Italy*

**Abstract**

The features of modern collaborative robots, mainly their kinematic redundancy combined with the light-weight structure, can be fully exploited in parts assembly. Traditional robot-level paradigm to robot programming, that requires to explicitly specify the motion of the robot and allows to use contact forces for motion supervision only, cannot be easily applied to complex interaction tasks, such as robotic assembly. Instead, by shifting paradigm to skill-based programming, it is possible to specify force control actions at task level and inherently provide compliant capabilities, without the need to specify the motions of the robot. To this end, this paper presents a constraint-based programming framework for the implementation of assembly skills for light-weight redundant robots, enabling a reactive generation of motion trajectories based on force control requirements. The effectiveness of the proposed approach is experimentally validated on a bimanual assembly use case performed with the ABB YuMi dual-arm robot, requiring a peg-in-hole insertion and a cap-rotation task. Estimation of the interaction force/torque additionally enables the execution of the assembly operation without the need for exteroceptive sensors.

*Keywords:* Robotic assembly, Collaborative robots, Task-level programming, Constraint-based programming

## 1. Introduction

Kinematic redundancy, dexterity and inherent compliance are some of the features of the new generation robotic arms, commonly referred to as collaborative robots or *cobots*, e.g. KUKA LBR iiwa, Universal Robot platforms, Franka Emika Panda, that make this type of platforms particularly suitable for assembly applications [1, 2]. Furthermore, when employing a dual-arm cobot, e.g. Willow Garage PR2, Rethink Robotics Baxter, Kawada Nextage, ABB YuMi, the design of task specific fixtures is no longer required, enabling a variety of autonomous operations, including: insertion [3, 4], folding [5, 6], screwing [7], taping [8], up to manipulation of highly deformable objects [9].

With the aim to intuitively program a complex assembly operation using a cobot, this paper roots in the *task-level* [10] or *skill-based* programming technique in order to allow the non-robotics expert to specify easy and quick task-related actions, rather than the motions of the robot as expected by the traditional *explicit* or *robot-level* paradigm. Tasks are assembled by concatenating these actions, generally referred to as *skills* of the robot [11, 12]. In this respect, the scope of this work is to propose a *constraint-based* formalism for the *task-level* specification of robotic skills that require force control policies, consequently endowing cobots with assembly *skills* that can be easily adopted for industrial applications.

The main contribution of this work are:

(i) A constraint-based programming framework that allows the skill developer to integrate force control requirements within the specification of an assembly skill, independently of the specific robotic platform and assembly application.

(ii) The real-time generation of reactive force controlled motions based on force-related objectives and constraints.

(iii) The possibility for a factory worker to intuitively program a complex assembly process by simple concatenation of basic assembly skills.

(iv) Demonstrations of a complete assembly process performed with the ABB YuMi dual-arm robot in a sensorless configuration, i.e. without the use of force/torque sensors.

The basic ideas behind the present paper can be preliminarily found in the authors' works [13, 14], although dealing with the control problems inherent in two different dual-arm robotic assembly tasks. This paper contributes with a comprehensive and general formulation of the constraint-based programming framework, which is independent of the specific robotic platform and of the specific assembly application. Therefore, from the perspective of the present work, [13, 14] represent two possible assembly scenarios enhancing the generality of the proposed method. An extensive discussion of the system capabilities further complements the description of the approach.

The paper is organized as follows. We present related works in Section 2 and provide details on the constraint-based programming framework by Zanchettin et al. [15] in Section 3.

[*]Corresponding author.
*Email addresses:* `matteo.parigi@iit.it` (Matteo Parigi Polverini), `andreamaria.zanchettin@polimi.it` (Andrea Maria Zanchettin), `paolo.rocco@polimi.it` (Paolo Rocco)

The architecture of the proposed programming approach to assembly skills implementation, which is built upon [15], is introduced in Section 4. The dual-arm robot used for the experimental validation and its control interface are presented in Section 5, together with the employed model-based observer of interaction forces [16, 17] and the assembly use case. Experimental validation on a peg-in-hole insertion task and on the subsequent cap-rotation task, required by the assembly process, are shown in Section 6 and Section 7, respectively. The experimental execution of the complete assembly process is shown in Section 8. Concluding remarks can be found in Section 9. Guidelines to select admittance parameters are additionally provided in Appendix A.

## 2. Related Works

From a control perspective, the lower inertia combined with the compliant structure of collaborative robots provides an intrinsic degree of safety towards manipulated objects, which is beneficial in assembly operations. On the other hand, the consequent lower position accuracy makes the application of force control algorithms [18] even more crucial for a successful task execution compared to traditional industrial robots. The adoption of force control policies when using collaborative robots can be found e.g. in [19, 20, 21]. In addition, the kinematic redundancy allows for a more dexterous manipulation but demands for redundancy resolution in order to plan a robot trajectory [22].

From a programming perspective, the traditional explicit or robot-level paradigm requires a robot programmer to entirely specify the robot motion through a scripting language, e.g. ABB's RAPID, KUKA KRL or COMAU PDL2. With respect to an assembly process, robot-level programming languages require the robot programmer to be expert not only in computer programming but also in the design of sensor-based motion strategies. According to this paradigm in fact, the data from a force sensor can be used exclusively for motion supervision, see e.g. [23], instead of for control purposes. This eventually results in rather intricate flowcharts of the assembly process, that are based on *what-if* handling strategies to define each possible non-nominal behaviour. Furthermore, the resulting robot motion, which is inherently stiff with respect to the environment, could end up in parts' breakage during the assembly task due to low position accuracy.

In contrast to robot-level programming, task-level or skill-based programming aims at providing easy and quick instructions without need for programming expertise. It refers to a higher abstraction layer than traditional robot-level programming by specifying task-related actions (robot skills), rather than the motions of the robot. So called *device primitives*, i.e. functions provided by a single device as a force/torque sensor, are used by the skill developer to design skills guaranteeing functionality on a higher level. This way the shop floor factory workers can program tasks quickly and efficiently by simply sequencing a number of skills. A major advantage of task-level programming over robot-level programming when it comes to an assembly process consists in the possibility for the skill developer to program assembly skills that enable force controlled compliant motions. As a result, a non-expert user can intuitively program a complex assembly task by simple concatenation of assembly skills.

As an alternative programming approach, Programming by Demonstration (PbD) [24] and statistical learning techniques, allow the automatic generation of robot trajectories from a database of related, but different, situations. According to PbD the robot trajectories are generally encoded into a form that allows for on-line modification, e.g. by using dynamic movement primitives (DMP) [25] or stochastic models as Gaussian Mixture Models (GMMs [26]). Recent work on learning by demonstration [27] through kinesthetic guiding of robotic arms [28, 29], has turned out to be particularly suitable for industrial applications. Nevertheless, although much research in this area has focused on pick-and-place tasks, only few works deal with robotic assembly, see [30, 31].

Following the task-level paradigm, *constraint-based* programming represents a viable solution for skills implementation [15], as it allows the real-time generation of robot motions as an output of a constrained optimization problem based on task-related objectives and constraints. The constraint-based approach, originally introduced in [32] based on the task function formalism [33], has been extensively adopted within the iTask (instantaneous Task Specification using Constraints) [34], the Stack of Tasks approach [35], and in several formulations using Quadratic programming (QP), see e.g. [36, 37]. The iTaSC in particular has been applied to assembly production scenarios, e.g. in [38, 39, 40] for the assembly of an emergency stop button. Although effective, these works tackle the assembly problem from a control perspective, rather than from a programming point of view and the integration within a skill-based programming framework remains an open problem only recently started to be addressed, e.g. in [41, 42].

Of particular interest for this work is the constraint-based programming approach by Zanchettin et al. [15], which will be revised in the following Section, that has been conceived as a skill-based architecture for the implementation of robust and ready-to-use robotic skills for industrial robotic applications.

## 3. Preliminaries

In the following we provide background knowledge on the combined trajectory generation and constraint-based programming approach proposed in [15].

### *3.1. Framework*

The architecture shown in Fig. 1, is based on a real-time trajectory generation algorithm [43], referred to as *trajectory generation* module, combined with a constraint-based *reactive controller*, feeding the inner industrial joint position/velocity controller. Subscript $k$ refers to the value of the related vector at discrete time instant $t_k$. The reactive controller, responsible for kinematic inversion in presence of redundancy and state constraints, generates the joint position and velocity reference

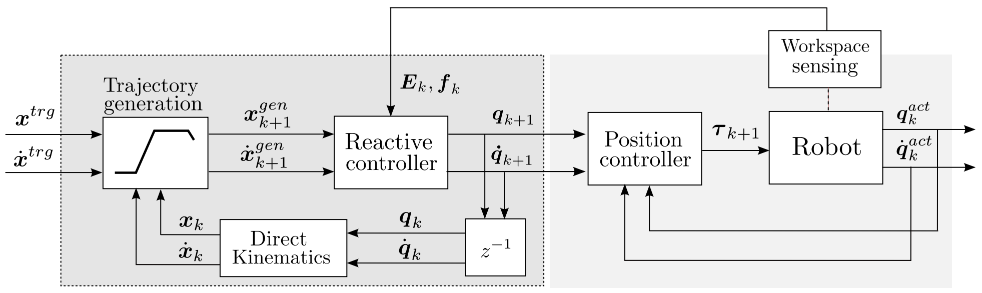


Figure 1: Block diagram of the constraint-based programming framework in [15] for a position controlled robot.

to the robot position/velocity controller ($q_{k+1}$, $\dot{q}_{k+1} \in \mathbb{R}^n$) based on: the state of motion in task coordinates provided by the trajectory generation module ($x^{gen}_{k+1}$, $\dot{x}^{gen}_{k+1} \in \mathbb{R}^m$), the current state of motion in task coordinates ($x_k$, $\dot{x}_k \in \mathbb{R}^m$) and work-space sensing data. If this involves a deviation from the planned trajectory, a new trajectory from the current state of motion to the target state of motion ($x^{trg}$, $\dot{x}^{trg} \in \mathbb{R}^m$) is reactively generated by the trajectory generation algorithm.

The following assumption has been made:

***Assumption***. *The reference joint acceleration vector represents the control input* $\boldsymbol{u} \in \mathbb{R}^n$

$$\ddot{\boldsymbol{q}}_k = \boldsymbol{u}_k \tag{1}$$

Consequently, the joint space process model reduces to a discrete-time double integrator

$$\begin{cases} \boldsymbol{q}_{k+1} = \boldsymbol{q}_k + T_s\dot{\boldsymbol{q}}_k + 0.5\ T_s^2\boldsymbol{u}_k \\ \dot{\boldsymbol{q}}_{k+1} = \dot{\boldsymbol{q}}_k + T_s\boldsymbol{u}_k \end{cases} \tag{2}$$

$T_s$ being the controller sampling time. By further considering the well-known forward kinematic relation between joint coordinates $\boldsymbol{q} \in \mathbb{R}^n$ and task variables $\boldsymbol{x} \in \mathbb{R}^m$

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q}) \quad \dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\,\dot{\boldsymbol{q}} \quad \ddot{\boldsymbol{x}} = \dot{\boldsymbol{J}}(\boldsymbol{q})\,\dot{\boldsymbol{q}} + \boldsymbol{J}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} \tag{3}$$

where $\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{n\times m}$ is the Jacobian matrix, the task space process model is given by

$$\begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + T_s\boldsymbol{J}_k\dot{\boldsymbol{q}}_k + 0.5\ T_s^2\left(\dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}}_k + \boldsymbol{J}_k\boldsymbol{u}_k\right) \\ \dot{\boldsymbol{x}}_{k+1} = \boldsymbol{J}_k\dot{\boldsymbol{q}}_k + T_s\left(\dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}}_k + \boldsymbol{J}_k\boldsymbol{u}_k\right) \end{cases} \tag{4}$$

being $\boldsymbol{J}_k = \boldsymbol{J}(\boldsymbol{q}_k)$ and $\dot{\boldsymbol{J}}_k = \dot{\boldsymbol{J}}(\boldsymbol{q}_k)$.

### 3.2. Constraint Specification

A generic constraint, considered within the reactive controller, has the following linear formulation

$$\boldsymbol{E}_k\boldsymbol{u}_k \leq \boldsymbol{f}_k \tag{5}$$

where $\boldsymbol{E}_k$ and $\boldsymbol{f}_k$ can be constant or time-varying vectors/matrices. Possible constraints can account for typical motion planning limitations (i.e. joint limits, bounds on the maximum velocities and accelerations in joint and task space) but also for sensor-related events occurring at time instant $k$, see [44, 15].

Taking into account the existing dynamics between the state variable subject to constraints and the control variable, i.e. when the relative degree is greater than 0, set invariance theory [45] comes into play to prevent constraint violations. The so-called *invariance control* by [46, 47] has been employed for this purpose, where a set invariance control law is designed off-line to render a subset of the constraint admissible state-space positively invariant. Note that a set is said to be positively invariant with respect to a dynamical system if a trajectory, with initial condition in the set, remains therein for all future times [45]. Constraints' satisfaction is ensured if a suitable set invariance condition holds for all active constraints. This in turn can be expressed consistently with (5) by exploiting Input-Output (I-O) Linearization, leading to

$$\mathcal{A}_{\mathcal{I}}\boldsymbol{u}_k \preceq \boldsymbol{b}_{\mathcal{I}} \tag{6}$$

where $\mathcal{I}$ is the set of active constraints at time instant $k$, $\mathcal{A}_{\mathcal{I}} = [\boldsymbol{a}_i]$ and $\boldsymbol{b}_{\mathcal{I}} = [\gamma_i - b_i]$ with $i \in \mathcal{I}$ and $\gamma_i < 0$. The operator $\preceq$ denotes the inequality $\leq$ for all elements of the vectors. Finally, $\boldsymbol{a}_i(\boldsymbol{x})$ and $b_i$ compose the I-O Linearization equation for the $i$-th constrained state variable $x_i$, i.e.

$$x^{(r_i)}_{i,k} = \boldsymbol{a}_i(\boldsymbol{x})\,\boldsymbol{u}_k + b_i \tag{7}$$

$r_i$ being the relative degree.

### 3.3. Reactive Controller Module

The reactive controller implementation is described in Algorithm 1. The cost function (9a) weighs the difference between the next state of motion $(\boldsymbol{x}_{k+1}, \dot{\boldsymbol{x}}_{k+1})$ and its reference values $\left(\boldsymbol{x}^{gen}_{k+1}, \dot{\boldsymbol{x}}^{gen}_{k+1}\right)$, the latter provided by the trajectory generation algorithm. The following Quadratic Programming (QP) cost function can be used

$$\begin{aligned} \mathcal{L}(\boldsymbol{e},\dot{\boldsymbol{e}}) = {} & 0.5\boldsymbol{e}^T\boldsymbol{Q}_p\boldsymbol{e} + 0.5\dot{\boldsymbol{e}}^T\boldsymbol{Q}_v\dot{\boldsymbol{e}} + \boldsymbol{e}^T\boldsymbol{Q}_{p,v}\dot{\boldsymbol{e}} + \\ & + \boldsymbol{g}_p^T\boldsymbol{e} + \boldsymbol{g}_v^T\dot{\boldsymbol{e}} \end{aligned} \tag{8}$$

where $\boldsymbol{Q}_p, \boldsymbol{Q}_v, \boldsymbol{Q}_{p,v}$ are positive definite matrices, while $\boldsymbol{g}_p$ and $\boldsymbol{g}_v$ are vectors of suitable dimension. Constraints (9b) and (9c) map joint space velocities and accelerations into task coordinates, while (9d) represents the set of all considered constraints accounting for motion planning limitations and sensor-related events. The output of the QP problem consists in the reference joint acceleration $\ddot{\boldsymbol{q}}_k$, i.e. the control input $\boldsymbol{u}_k$, from which

```
Algorithm 1 Reactive Controller
Input: $q_k, \dot{q}_k, x_{k+1}^{gen}, \dot{x}_{k+1}^{gen}, \mathcal{A}_I, b_I$
Output: $q_{k+1}, \dot{q}_{k+1}$
1: solve the following QP problem
       $\min_{u_k} \mathcal{L}\left(\dot{x}_{k+1} - \dot{x}_{k+1}^{gen}, x_{k+1} - x_{k+1}^{gen}\right)$   (9a)
   subject to
       $x_{k+1} = x_k + T_s J_k \dot{q}_k + 0.5 T_s^2 \left(\dot{J}_k \dot{q}_k + J_k u_k\right)$   (9b)
       $\dot{x}_{k+1} = J_k \dot{q}_k + T_s \left(\dot{J}_k \dot{q}_k + J_k u_k\right)$   (9c)
       $\mathcal{A}_I u_k \leq b_I$   (9d)
2: update the state of motion as in (2)
       $\begin{cases} q_{k+1} = q_k + T_s \dot{q}_k + 0.5 T_s^2 u_k \\ \dot{q}_{k+1} = \dot{q}_k + T_s u_k \end{cases}$   (10)
```

the corresponding reference position $q_{k+1}$ and velocity $\dot{q}_{k+1}$ can be computed through (2) and fed to the inner position/velocity controller.

In presence of redundant degrees of freedom or in case of task redundancy, a secondary task can be performed having lower priority with respect to the primary task. To this end, being $u_k^0$ any of the (infinite) optimal solutions to the QP problem in (33), the following equality constraint needs to be additionally considered

$$J_k \left(u_k - u_k^0\right) = \mathbf{0} \tag{11}$$

ensuring that the alternative solution $u_k$ differs from $u_k^0$ in the null space of the task Jacobian [48, 49]. Within an optimization based framework this can be implemented by introducing a second optimization stage, which inherits all the constraints in (33) together with the equality constraint in (11)

$$\begin{gathered} \min_{u_k} \frac{1}{2} u_k^T Q_u u_k + g_u^T u_k \\ \text{subject to} \\ J_k u_k = J_k u_k^0 \\ \mathcal{A}_I u_k \leq b_I \end{gathered} \tag{12}$$

where $Q_u$ is a positive definite matrix and $g_u$ is a vector of suitable dimension.

Note that the constraint-based programming framework in [15] is mainly devoted to the implementation of motion skills that do not require interaction with the environment and has been evaluated in an image-guided grasping task. Building upon this framework, the current paper aims at extending its capabilities by focusing on the design and implementation of force controlled assembly skills.

## 4. Proposed Constraint-Based Programming Framework for Assembly Skills Implementation

Parts assembly involves a variety of autonomous operations (e.g. insertion, alignment, screwing) where the capabilities of a collaborative robot can be naturally exploited. Note that robotic assembly generally requires the adoption of force control policies for a correct task execution. Indirect force control approaches [18], e.g. impedance control [50], are particularly suitable for this purpose compared to direct force control methods, typically applied in robotic polishing, debarring and machining. Note also that, the lower inertia combined with the compliant structure of light-weight cobots cause in fact the deterioration of the controller performance when employing a direct force control approach. Nevertheless, bounding the interaction force, arising from the contact between manipulated parts in assembly operations, is a relevant control feature that becomes crucial for the successful execution of those assembly tasks where contact loss needs to be prevented, e.g. the cap rotation task depicted in Fig. 2. In this respect, as previously addressed in Section 2, a paradigm shift from traditional robot-level programming to task-level or skill-based programming can be beneficial to endow a collaborative robot with the aforementioned force control capabilities.

According to the skill-based paradigm and building upon the approach in [15], this Section presents the mathematical formulation of the proposed constraint-based programming framework for the implementation of assembly skills. Two main force control requirements will be considered in the following: a compliant robot motion, see Section 4.1, and a robust constraint on the interaction force, see Section 4.2.

### 4.1. Compliant Robot Motion

In a variety of assembly processes the motion of the robot should be made compliant to interaction forces in order to compensate for the low position accuracy and prevent parts' breakage. The trajectory followed by the robot is therefore modified by external forces/moments resulting in a compliant motion. Taking into account that in usual industrial scenarios robots are typically equipped with position/velocity controllers, this compliant behavior can be generally achieved through *admittance* control [18]. Assume that the interaction force/torque $\mu \in \mathbb{R}^m$ is applied to the robot TCP (Tool Centre Point), the following equation establishes a mass-damper relation between the interaction force/torque acting on the system and the TCP displacement vector $\Delta x \in \mathbb{R}^m$

$$\mu_k = M \Delta \ddot{x}_k + D \Delta \dot{x}_k \tag{13}$$

where $M \in \mathbb{R}^{m \times m}$ and $D \in \mathbb{R}^{m \times m}$ are positive definite design matrices composing the mass-damper system. By retrieving $\Delta \ddot{x}_k$ from (13) and accounting for the task space process model in (4), i.e.

$$\begin{cases} \Delta \dot{x}_{k+1} = \Delta \dot{x}_k + T_s \Delta \ddot{x}_k \\ \Delta x_{k+1} = T_s \Delta \dot{x}_k + 0.5 T_s^2 \Delta \ddot{x}_k \end{cases} \tag{14}$$

the impedance relation (13) can be adopted to compute the TCP position and velocity displacements $\Delta x$ and $\Delta \dot{x}$, respectively, yielding

$$\begin{cases}\Delta\dot{\boldsymbol{x}}_{k+1} = \Delta\dot{\boldsymbol{x}}_k + T_s\boldsymbol{M}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{D}\Delta\dot{\boldsymbol{x}}_k)\\ \Delta\boldsymbol{x}_{k+1} = T_s\Delta\dot{\boldsymbol{x}}_k + 0.5T_s^2\boldsymbol{M}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{D}\Delta\dot{\boldsymbol{x}}_k)\end{cases} \tag{15}$$

## *4.2. Robust Constraint on the Interaction Force*

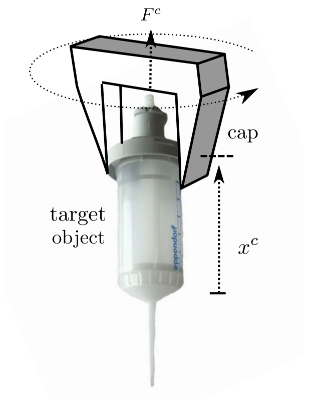


Figure 2: Example of a cap rotation task.

Considering a compliant contact situation (a very common situation, due to the most likely low stiffness of the assembled parts combined with the light-weight robot structure), the relation between the robot end effector position $x_c \in \mathbb{R}$ and the contact force $F_c \in \mathbb{R}$, along one of the directions constrained by the environment, can be modeled as an equivalent spring at the end effector

$$F_c = -K(x_c - x_0) \tag{16}$$

where $x_0 \in \mathbb{R}$ is the nominal undeformed pose of the environment surface in the constrained direction, while $K \in \mathbb{R}$ is the environment stiffness depending on the object material.

***Remark 1****. Possible uncertainties considered in the remainder of this paper are:*

(i) *a rough knowledge of the interaction model, represented by a bounded uncertainty in the environment stiffness*

$$K \in \mathbb{K} := [K_{min}, K_{max}]; \tag{17}$$

(ii) *force measurement (or estimation) noise, i.e. a measurement uncertainty on the state vector*

$$\boldsymbol{\Delta}^F \in \mathbb{D}^F := \left\{\boldsymbol{\Delta}^F : \boldsymbol{\Delta}^F_{min} \leq \boldsymbol{\Delta}^F \leq \boldsymbol{\Delta}^F_{max}\right\}; \tag{18}$$

(iii) *uncertainty in the environment surface pose $x_0$, represented as generic, non structured, modelling error*

$$\boldsymbol{v}^F \in \mathbb{V}^F := \left\{\boldsymbol{v}^F : \boldsymbol{v}^F_{min} \leq \boldsymbol{v}^F \leq \boldsymbol{v}^F_{max}\right\}; \tag{19}$$

In this respect and based on the authors work in [14], the invariance control approach will be employed to robustly ensure the satisfaction of a lower and an upper bound on the interaction force, despite the considered uncertainties. Assuming that (16) has relative degree 2, the related state vector $\boldsymbol{\pi}^F \in \mathbb{R}^2$ is given by

$$\boldsymbol{\pi}^F = \begin{bmatrix}F_c & \dot{F}_c\end{bmatrix}^T. \tag{20}$$

Following from (16)

$$\boldsymbol{\pi}^F = \begin{bmatrix}F_c & -K\,\dot{x}_c\end{bmatrix}^T = \begin{bmatrix}1 & 0\\ 0 & -K\end{bmatrix}\begin{bmatrix}F_c & \dot{x}_c\end{bmatrix}^T \tag{21}$$

where $\dot{x}_c = \boldsymbol{J}_c(\boldsymbol{q})\,\dot{\boldsymbol{q}}$ and $\boldsymbol{J}_c(\boldsymbol{q})$ is the row vector of the Jacobian matrix related to the constrained direction. By introducing the new state vector $\hat{\boldsymbol{\pi}}^F \in \mathbb{R}^2$

$$\hat{\boldsymbol{\pi}}^F = \begin{bmatrix}F_c & \dot{x}_c\end{bmatrix}^T \tag{22}$$

equation (21) can be written as

$$\boldsymbol{\pi}^F = \begin{bmatrix}1 & 0\\ 0 & -K\end{bmatrix}\hat{\boldsymbol{\pi}}^F \tag{23}$$

The corresponding discrete-time state vector $\boldsymbol{\pi}^F_{k+1}$ is obtained through the double-integrator relation

$$\boldsymbol{\pi}^F_{k+1} = \boldsymbol{A}^F\hat{\boldsymbol{\pi}}_k + \boldsymbol{B}^F\ddot{x}_{c,k} \tag{24}$$

where the state matrices $\boldsymbol{A}^F$ and $\boldsymbol{B}^F$ depend on the environment stiffness $K$

$$\boldsymbol{A}^F = \begin{bmatrix}1 & K\,T_s\\ 0 & -K\end{bmatrix} \quad \boldsymbol{B}^F = \begin{bmatrix}0.5\,K\,T_s^2\\ -K\,T_s\end{bmatrix} \tag{25}$$

while the Cartesian acceleration along the constrained direction $\ddot{x}_c$ is given by

$$\ddot{x}_c = \dot{\boldsymbol{J}}_c(\boldsymbol{q})\,\dot{\boldsymbol{q}} + \boldsymbol{J}_c(\boldsymbol{q})\,\boldsymbol{u} \tag{26}$$

Based on Remark 1, the corresponding uncertain system is

$$\boldsymbol{\pi}^F_{k+1} = \boldsymbol{A}^F(\hat{\boldsymbol{\pi}}_k + \boldsymbol{\Delta}^F) + \boldsymbol{B}^F\ddot{x}_{c,k} + \boldsymbol{v}^F \tag{27}$$

while the related reachable set, depicted in Fig. 3, representing the set of all possible values for the state vector $\boldsymbol{\pi}^F_{k+1}$ due to the considered uncertainties, can be computed as

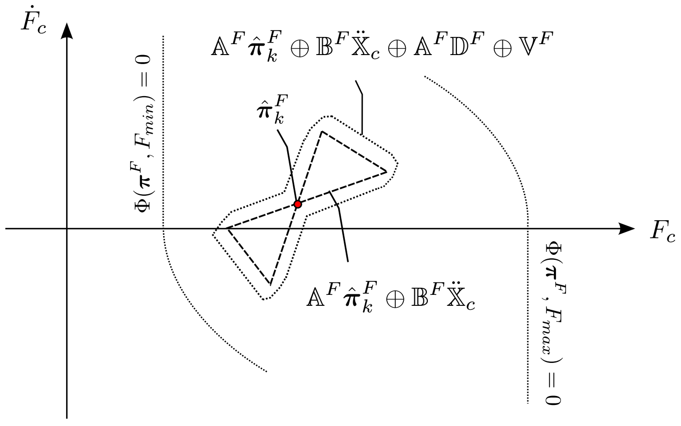
$$\boldsymbol{\pi}^F_{k+1} \in \mathbb{A}^F\hat{\boldsymbol{\pi}}^F_k \oplus \mathbb{B}^F\ddot{\mathbb{X}}_c \oplus \mathbb{A}^F\mathbb{D}^F \oplus \mathbb{V}^F \tag{28}$$

being $\ddot{\mathbb{X}}_c = \{\ddot{x}_c : |\ddot{x}_c| \leq \gamma_x,\ \gamma_x > 0\}$.
As shown in Fig. 3, the reachable set in (28) results in a butterfly shaped non-convex reachable set. This is specifically due to the impact of the considered environment stiffness uncertainty, see (17), in the state matrix $\boldsymbol{A}^F$.

Let's now assume that the controller must ensure a lower and upper bound on the interaction force

$$F_{min} \leq F_{c,k} \leq F_{max},\ \forall k. \tag{29}$$

According to the invariance control approach, a so-called *invariance function* $\Phi$ needs to be computed to ensure constraint satisfaction, which represents a worst case estimation for the future output trajectory of state variable subject to constraint. Since (16) has relative degree 2, the invariance functions $\Phi(\boldsymbol{\pi}^F, F_{max})$ and $\Phi(\boldsymbol{\pi}^F, F_{min})$ related to each force constraints, i.e. $F_{c,k} \leq F_{max}$ and $F_{c,k} \geq F_{min}$, can be analytically computed



$\dot{F}_c$
$F_c$
$\mathbb{A}^F\hat{\pi}_k^F \oplus \mathbb{B}^F\ddot{\mathbb{X}}_c \oplus \mathbb{A}^F\mathbb{D}^F \oplus \mathbb{V}^F$
$\hat{\pi}_k^F$
$\mathbb{A}^F\hat{\pi}_k^F \oplus \mathbb{B}^F\ddot{\mathbb{X}}_c$
$\Phi(\pi^F, F_{min}) = 0$
$\Phi(\pi^F, F_{max}) = 0$

Figure 3: Geometric interpretation of condition (28).

$$\Phi(\boldsymbol{\pi}^F, F_{max}) = \begin{cases} F_c - F_{max} & \dot{F}_c \leq 0 \\ F_c + \frac{\dot{F}_c^2}{2\ddot{F}_c^{max}} - F_{max} & \dot{F}_c > 0 \end{cases} \tag{30}$$

and

$$\Phi(\boldsymbol{\pi}^F, F_{min}) = \begin{cases} F_{min} - F_c & \dot{F}_c \geq 0 \\ -F_c + \frac{\dot{F}_c^2}{2\ddot{F}_c^{max}} + F_{min} & \dot{F}_c < 0 \end{cases} \tag{31}$$

The sub-domain $\mathbb{I}^F$ of the space $\boldsymbol{\pi}^F$ bounded by $\Phi(\boldsymbol{\pi}^F, F_{max}) = 0$ and $\Phi(\boldsymbol{\pi}^F, F_{min}) = 0$ is shown in Fig. 3. By monitoring the value of the two invariance functions with respect to the future force state vector $\boldsymbol{\pi}^F_{k+1}$ in (28), the set $\mathbb{I}^F$ can be made robustly positive invariant. This can be achieved if the controller selects a value of $\ddot{x}_{c,k}$ satisfying the following set invariance conditions

$$\begin{cases} \ddot{x}_{c,k} \leq -\gamma_x, & \text{if } \Phi(\boldsymbol{\pi}^F_{k+1}, F_{max}) = 0 \\ \ddot{x}_{c,k} \geq \gamma_x, & \text{if } \Phi(\boldsymbol{\pi}^F_{k+1}, F_{min}) = 0 \end{cases} \tag{32}$$

where $\gamma_x > 0$ is a controller design parameter, representing the minimum value of the Cartesian acceleration to avoid constraint violation. Despite the non-convexity of the reachable set in (28), by considering the convex hull of the reachable set [51], it is sufficient to evaluate the invariance functions on the vertices of the convex hull.

### *4.3. Reactive Force Controller Module*

From a task-level programming perspective, the force control features presented in Section 4.1 and Section 4.2 represent task-related actions that a skill developer can employ for the specification of an assembly skill. In order to allow this by following a constraint-based programming approach, we propose to improve the framework in [15], previously described in Section 3, by conveniently modifying the reactive controller module, hereafter referred to as *reactive force controller*. The trajectory generation block, shown in Fig. 1, is kept unaltered and completely ignores joint level constraints and work-space sensing data. Instead, the reactive force controller tries to minimize the tracking error w.r.t. the trajectory generation output, i.e. ($\boldsymbol{x}^{gen}_{k+1}$, $\dot{\boldsymbol{x}}^{gen}_{k+1} \in \mathbb{R}^m$), while accommodating for the considered force control requirements based on sensor data, in addition to joint level constraints. The implementation of the reactive force controller is shown in Algorithm 2 and described in details the following remarks.

**Algorithm 2** Reactive Force Controller

```
Input: $\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \boldsymbol{x}^{gen}_{k+1}, \dot{\boldsymbol{x}}^{gen}_{k+1}, \mathcal{A}_{\mathcal{I}}, \boldsymbol{b}_{\mathcal{I}}, \boldsymbol{\mu}_k$
Output: $\boldsymbol{q}_{k+1}, \dot{\boldsymbol{q}}_{k+1}$
1: compute $\Delta\boldsymbol{x}_{k+1}$ and $\Delta\dot{\boldsymbol{x}}_{k+1}$ as in (15)
2: compute $\Phi(\boldsymbol{\pi}^F_{k+1}, F_{max})$ and $\Phi(\boldsymbol{\pi}^F_{k+1}, F_{min})$
3: solve the following QP problem
```

$$\min_{\boldsymbol{u}_k} \mathcal{L}\left(\dot{\boldsymbol{x}}_{k+1} - \dot{\tilde{\boldsymbol{x}}}^{gen}_{k+1}, \boldsymbol{x}_{k+1} - \tilde{\boldsymbol{x}}^{gen}_{k+1}\right) \tag{33a}$$

subject to

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + T_s\boldsymbol{J}_k\dot{\boldsymbol{q}}_k + 0.5T_s^2\left(\dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}}_k + \boldsymbol{J}_k\boldsymbol{u}_k\right) \tag{33b}$$

$$\dot{\boldsymbol{x}}_{k+1} = \boldsymbol{J}_k\dot{\boldsymbol{q}}_k + T_s\left(\dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}}_k + \boldsymbol{J}_k\boldsymbol{u}_k\right) \tag{33c}$$

$$\tilde{\boldsymbol{x}}^{gen}_{k+1} = \boldsymbol{x}^{gen}_{k+1} + T_s\Delta\dot{\boldsymbol{x}}_k + 0.5T_s^2\boldsymbol{M}^{-1}\left(\boldsymbol{\mu}_k - \boldsymbol{D}\Delta\dot{\boldsymbol{x}}_k\right) \tag{33d}$$

$$\dot{\tilde{\boldsymbol{x}}}^{gen}_{k+1} = \dot{\boldsymbol{x}}^{gen}_{k+1} + \Delta\dot{\boldsymbol{x}}_k + T_s\boldsymbol{M}^{-1}\left(\boldsymbol{\mu}_k - \boldsymbol{D}\Delta\dot{\boldsymbol{x}}_k\right) \tag{33e}$$

$$\mathcal{A}_{\mathcal{I}}\boldsymbol{u}_k \leq \boldsymbol{b}_{\mathcal{I}} \tag{33f}$$

$$\boldsymbol{a}_{F_{max}}\boldsymbol{u}_k \leq b_{F_{max}} \text{if } \Phi(\boldsymbol{\pi}^F_{k+1}, F_{max}) = 0 \tag{33g}$$

$$\boldsymbol{a}_{F_{min}}\boldsymbol{u}_k \leq b_{F_{min}} \text{if } \Phi(\boldsymbol{\pi}^F_{k+1}, F_{min}) = 0 \tag{33h}$$

```
4: update the state of motion as in (2)
```

$$\begin{cases} \boldsymbol{q}_{k+1} = \boldsymbol{q}_k + T_s\dot{\boldsymbol{q}}_k + 0.5T_s^2\boldsymbol{u}_k \\ \dot{\boldsymbol{q}}_{k+1} = \dot{\boldsymbol{q}}_k + T_s\boldsymbol{u}_k \end{cases} \tag{34}$$

```
5: update the state of the admittance filter
```

$$\begin{cases} \Delta\boldsymbol{x}_{k+1} = \tilde{\boldsymbol{x}}^{gen}_{k+1} - \boldsymbol{x}_{k+1} \\ \Delta\dot{\boldsymbol{x}}_{k+1} = \dot{\tilde{\boldsymbol{x}}}^{gen}_{k+1} - \dot{\boldsymbol{x}}_{k+1} \end{cases} \tag{35}$$

***Remark 2****. In order to enable a robot compliant motion, the reference state of motion in task coordinates provided by the trajectory generation module* ($\boldsymbol{x}^{gen}_{k+1}$, $\dot{\boldsymbol{x}}^{gen}_{k+1} \in \mathbb{R}^m$) *is added to the admittance-based TCP position and velocity displacements* ($\Delta\boldsymbol{x}_{k+1}$, $\Delta\dot{\boldsymbol{x}}_{k+1} \in \mathbb{R}^m$) *computed through* (15).

The resulting new reference state of motion ($\tilde{\boldsymbol{x}}^{gen}_{k+1}$, $\dot{\tilde{\boldsymbol{x}}}^{gen}_{k+1} \in \mathbb{R}^m$) is given by

$$\begin{cases} \tilde{\boldsymbol{x}}^{gen}_{k+1} = \boldsymbol{x}^{gen}_{k+1} + \Delta\boldsymbol{x}_{k+1} = \boldsymbol{x}^{gen}_{k+1} + T_s\Delta\dot{\boldsymbol{x}}_k + 0.5T_s^2\boldsymbol{M}^{-1}\left(\boldsymbol{\mu}_k - \boldsymbol{D}\Delta\dot{\boldsymbol{x}}_k\right) \\ \dot{\tilde{\boldsymbol{x}}}^{gen}_{k+1} = \dot{\boldsymbol{x}}^{gen}_{k+1} + \Delta\dot{\boldsymbol{x}}_{k+1} = \dot{\boldsymbol{x}}^{gen}_{k+1} + \Delta\dot{\boldsymbol{x}}_k + T_s\boldsymbol{M}^{-1}\left(\boldsymbol{\mu}_k - \boldsymbol{D}\Delta\dot{\boldsymbol{x}}_k\right) \end{cases} \tag{36}$$

Consequently, the cost function (33a) in Algorithm 2 weighs the difference between the next state of motion ($\boldsymbol{x}_{k+1}, \dot{\boldsymbol{x}}_{k+1}$) and the new reference values ($\tilde{\boldsymbol{x}}^{gen}_{k+1}$, $\dot{\tilde{\boldsymbol{x}}}^{gen}_{k+1} \in \mathbb{R}^m$). The state of the admittance filter is then updated as in (35). Useful guidelines to select admittance parameters are provided in Appendix A.

***Remark 3***. *A robust bounding of the interaction force can be achieved through the additional hard constraints* (33g) *and* (33h).

These constraints, linear in the optimization variable ($\boldsymbol{u}_k = \ddot{\boldsymbol{q}}_k$), are obtained from the set invariance conditions in (32) by exploiting the forward kinematics relation in (26), yielding

$$\begin{aligned} \boldsymbol{a}_{F_{max}} &= \boldsymbol{J}_c(\boldsymbol{q}_k), \quad b_{F_{max}} = -\gamma_x - \dot{\boldsymbol{J}}_c(\boldsymbol{q}_k)\,\dot{\boldsymbol{q}}_k \\ \boldsymbol{a}_{F_{min}} &= -\boldsymbol{J}_c(\boldsymbol{q}_k), \quad b_{F_{min}} = -\gamma_x + \dot{\boldsymbol{J}}_c(\boldsymbol{q}_k)\,\dot{\boldsymbol{q}}_k \end{aligned} \tag{37}$$

If required by the assembly operation, a straightforward way for softening the force constraints (33g) and (33h) is to introduce the slack variables $\epsilon_1, \epsilon_2 \in \mathbb{R}$

$$\begin{aligned} \boldsymbol{a}_{F_{max}}\boldsymbol{u}_k &\leq b_{F_{max}} + \epsilon_1 \\ \boldsymbol{a}_{F_{min}}\boldsymbol{u}_k &\leq b_{F_{min}} + \epsilon_2 \end{aligned} \tag{38}$$

The cost function in Algorithm 2 is in turn modified as follows

$$\min_{\boldsymbol{u}_k, \epsilon_1, \epsilon_2} \mathcal{L}\left(\dot{\boldsymbol{x}}_{k+1} - \dot{\tilde{\boldsymbol{x}}}^{gen}_{k+1}, \boldsymbol{x}_{k+1} - \tilde{\boldsymbol{x}}^{gen}_{k+1}\right) + \sum_{i=1}^{2} \rho_i \|\epsilon_i\|_2^2 \tag{39}$$

where $\rho_1, \rho_2 \in \mathbb{R}$ are the related weights.

***Remark 4***. *In order to exploit redundancy to perform a lower-priority task, several formulations of the cost function in* (12) *can be considered.*

By setting

$$\begin{aligned} \boldsymbol{Q}_u &= 2T_s^2 \boldsymbol{I}_n \\ \boldsymbol{g}_u &= 2T_s \dot{\boldsymbol{q}}_k \end{aligned} \tag{40}$$

joint space velocities $\dot{\boldsymbol{q}}_{k+1}$ can be minimized. Alternatively, kinematic redundancy can be exploited for collision-avoidance purposes, including self-collision avoidance. Defining as $\dot{\boldsymbol{q}}_k^0$ the vector of evasive joint displacements, defined as in [52], the candidate cost function in (12) is given by

$$\begin{aligned} \boldsymbol{Q}_u &= 2T_s^2 \boldsymbol{I}_n \\ \boldsymbol{g}_u &= 2T_s(\dot{\boldsymbol{q}}_k - \dot{\boldsymbol{q}}_k^0) \end{aligned} \tag{41}$$

As another option, the robot reflected mass could be minimized, according to [53], in order to decrease the dissipated energy in potential inelastic impacts.

***Remark 5***. *By proper selection of tasks and constraints within the reactive force controller module different assembly skills can be implemented:*

- *Compliant motion skill*: cost function (33a) plus admittance equality constraints (33d)-(33e);
- *Force bounding skill*: cost function (9a) plus (hard/soft) force constraints (33g)-(33h);
- *Force-constrained compliant motion*: cost function (33a), admittance equality constraints (33d)-(33e), (hard/soft) force constraints (33g)-(33h).

Note that these assembly skills can be used in conjunction with the motion skills in [15] described in Algorithm 1. As a matter of fact, different skills, corresponding to different control modes (position/velocity control, admittance control, force bounding) can be specified for each task variable within the reactive force controller module, e.g. a compliant motion along the $z$ Cartesian directions and a position-controlled motion along the $x - y$ directions, thus allowing for hybrid force/position control.

## 4.4. Discussion

The proposed constraint-based programming method to robotic assembly has the following advantages over traditional robot-level programming:

- the possibility for the skill developer to embed force control requirements within the specification of an assembly skill, independently of the specific robotic platform and assembly process;
- the real-time generation of reactive force controlled robot motions;
- the intuitiveness for the non-expert user to program a complex assembly task by simple concatenation of assembly skills.

Using today's motion specification paradigm, reactive and adaptive behaviours are only obtained with complicated what-if logics to be specified at programming time, thus limiting a flexible adoption of robotic manipulators in industrial settings. Moreover, according to the robot-level paradigm, the data from a force sensor can be used only to supervise the robot motion. This eventually results in a rather complex sequence of what-if handling logics, see e.g. the following pseudo-code for a peg-in-hole task:

```
MoveTo apprInsPos;
if (Fz<=10N)
 MoveAlongZ [+10mm/s];
 MoveAlongX [-1mm/s] with Fx>=+2.0 until Fx<=+0.5
 MoveAlongX [+1mm/s] with Fx<=-2.0 until Fx>=-0.5
 MoveAlongY [-1mm/s] with Fy>=+2.0 until Fy<=+0.5
 MoveAlongY [+1mm/s] with Fy<=-2.0 until Fy>=-0.5
else
     StopMove;
```

Furthermore, the motion of the robot, set by the low-level joint position control, remains stiff with respect to the manipulated object, thus increasing the risk of parts' breakage due to low position accuracy. Instead, the proposed approach, that builds upon the skill-based programming paradigm, gives the skill developer the possibility to program assembly skills that inherently enable force controlled motions. This is done by proper specification of force-related actions at task level, rather than the motions of the robot at joint level. As a major advantage, a non-expert shop floor factory worker can intuitively program a complex assembly task by simple concatenation of assembly skills with a reduced number of handling strategies. With reference to a possible novel programming language, an assembly process could be simply specified by means of a single instruction without the need to further define handling strategies, e.g. see the following pseudo-code for a peg-in-hole task:

```
MoveTo apprInsPos;
MoveAlongZ[+10mm/s]-Admittance[XY] until Fz>=10N;
StopMove;
```

The resulting real-time generation of reactive robot motions based on force control requirements endows the robot controller with improved adaptation and robustness capabilities.

It is worth pointing out that the proposed framework represents a programming tool for the skill developer. Developing the high-level programming language to be used by the factory worker for the specification of assembly tasks (based on implemented robot skills) remains a future research direction, which is not addressed in the present paper.

## 5. Experimental Validation: Bimanual Robotic Assembly of a Plastic Pipette

In this Section the effectiveness of the proposed programming framework is experimentally validated on a complete assembly process performed with a dual-arm collaborative robot. These results, that preliminarily appeared in the authors' works [13, 14], are also illustrated in the attached supplementary video. Details on the robotic platform are first provided, together with a description of the employed method for estimation of the interaction force/torque. The assembly scenario is finally presented.

### *5.1. Experimental Platform*

The robot used in this paper for experimental validation is the ABB YuMi, see Fig 4. a position controlled dual-arm cobot with a light-weight skeleton, specifically designed for robotic assembly and human-robot collaboration. Each of the two arms is redundant with 7 degrees of freedom. Aiming at performing assembly tasks, the YuMi used in the experiments has been customized with Schunk pneumatically actuated grippers on both end effectors and 3D printed tools designed for the specific assembly operation. The robot is position controlled by the ABB IRC5 control system. An external controller using a research interface, running at 250 $Hz$, is executed on an external PC with Linux and Xenomai for real-time performance. The external PC communicates with the robot controller through LabComm protocol.
The sequence of QP problems required by the reactive force controller module has been solved with QPOASES [54]. Since the control loop runs at 250 $Hz$, a worst-case computation time of approximately 4 $ms$ is required for real-time control. Reflexxes Motion Libraries [55] have been employed as real-time trajectory generation algorithm within the trajectory generation module.

### *5.2. Force Estimation*

In its current setup the YuMi robot is not equipped with any wrist mounted force/torque sensor or joint torque sensors, therefore estimation of the interaction force/torque is required in order to perform assembly tasks, see [40]. Force estimation is hereafter performed based on the generalized momentum method [16, 17]. The benefit of using this method is that (noisy) joint acceleration measurements are not needed, although knowledge of the robot dynamical model is required. Details on the YuMi dynamical model and on the identification of friction coefficients can be found in [56, 13].

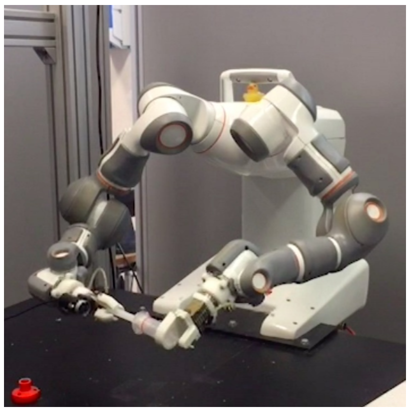


Figure 4: The ABB YuMi robot used for experimental validation equipped with Schunk pneumatically actuated grippers and 3D printed tools, designed for the specific assembly process.

According to [16], estimation of the external torques $\boldsymbol{\tau}_{ext} \in \mathbb{R}^n$ can be performed by computing the residual vector $\boldsymbol{r} \in \mathbb{R}^n$

$$\boldsymbol{r} = \boldsymbol{K}_R \left[ \boldsymbol{p} - \int_0^{t_k} \left( \boldsymbol{\tau} + \boldsymbol{C}^T \dot{\boldsymbol{q}}^{act} - \boldsymbol{\gamma} + \boldsymbol{r} \right) dt \right] \tag{42}$$

where $\boldsymbol{p} \in \mathbb{R}^n$ is the generalized momentum vector, $\boldsymbol{\tau} \in \mathbb{R}^n$ is the commanded joint torque, $\boldsymbol{K}_R \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix. Coriolis/centrifugal forces and torques are represented by $\boldsymbol{C}^T \dot{\boldsymbol{q}}^{act} \in \mathbb{R}^n$, while gravity and friction effects are contained in $\boldsymbol{\gamma} \in \mathbb{R}^n$. From (42), one obtains a first order stable linear relationship between the external torques $\boldsymbol{\tau}_{ext}$ and the residual

$$\dot{\boldsymbol{r}} = \boldsymbol{K}_R \left( \boldsymbol{\tau}_{ext} - \boldsymbol{r} \right) \tag{43}$$

Accounting for robot redundancy and assuming the interaction force/torque $\boldsymbol{\mu} \in \mathbb{R}^m$ (with $m < n$) to be applied to the Tool Center Point (TCP), the following equation can be finally adopted to estimate $\boldsymbol{\mu}$

$$\boldsymbol{\mu} = \left( \boldsymbol{J}(\boldsymbol{q})^T \right)^{\dagger} \boldsymbol{r} \tag{44}$$

where $\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{m \times n}$ represents the Jacobian of the TCP frame, while $^{\dagger}$ stands for the Moore-Penrose pseudo-inverse.

Finally note that, according to Remark 1 in Section 5.3, force estimation noise can be taken into account within the robust constraint on the interaction force as a measurement uncertainty on the state vector in (22).

### *5.3. Assembly Use Case*

The use case considered in this work consists in the bimanual assembly of an Eppendorf combitip® plastic pipette, generally used in laboratory automation. An assembly graph for the assembly task is displayed in Fig. 5.
The assembly operation can be divided in two phases:

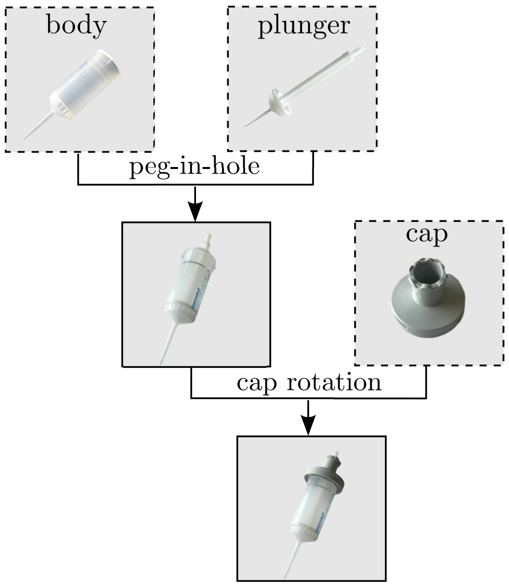


Figure 5: Assembly graph of an Eppendorf combitip® pipette.

(i) The plunger should be initially inserted into the body of the pipette. This is a typical peg-in-hole insertion problem that will be addressed in Section 6.

(ii) Once the plunger has been inserted, a cap rotation task, treated in Section 7, should be performed in order to screw the cap on top of the pipette body. A bayonet mount ensures the fastening between the cap and the body.

Note that, in order to execute these tasks, the traditional robot-level programming paradigm would require a relevant number of what-if handling logics, but also high accuracy of both the robot positioning system and the force sensing system. Due to the low position accuracy of a light-weight robot as YuMi compared to traditional industrial robots, and since no force sensor is available (but only an estimator), an experimental comparison of the proposed method with traditional robot-level programming has not been performed in order to limit the risk of parts' breakage.

## 6. Phase 1: Plunger Insertion

Selection of task variables is first described in Section 6.1, while the finite state machine model [57] governing the the task execution and experimental results are consequently presented in Section 6.2 and Section 6.3, respectively.

### 6.1. Task Specification and Required Assembly Skills

A leader-follower approach [58] has been employed so that a relative motion command is generated for the follower robot arm (the one holding the peg) based on the leader's arm state of motion (the one holding the hole). The following vector of task variables $\boldsymbol{x} \in \mathbb{R}^5$ has been chosen as state vector

$$\boldsymbol{x} = \begin{bmatrix} x & y & z & \phi & \theta \end{bmatrix}^T \tag{45}$$

representing the relative linear and angular displacements (XY Euler angles) of the follower frame with respect to the leader frame, see Fig. 6(a). Due to the cylindrical symmetry of the two work-pieces, the Z Euler angle $\psi$ does not represent a task variable, thus resulting in a redundant degree of freedom. The peg longitudinal axis is hereafter assumed to be aligned with the follower $z$ direction.

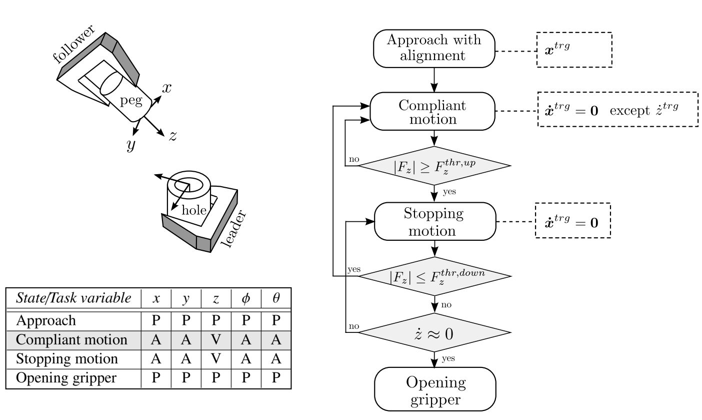
### 6.2. Assembly Sequence

The complete flowchart, shown in Fig. 6(a) together with the control mode for each task variable, is described in the following.

1. *Approach with alignment* - the follower end effector reaches the initial pose $\boldsymbol{x}^{trg}$, corresponding to an approximate alignment between the peg and the hole;
2. *Compliant insertion* - the follower insertion motion along the $z$ direction is velocity-controlled, while a compliant motion skill is set along $(x, y, \phi, \theta)$;
3. Whenever the force in $z$ direction exceeds a given bound ($F_z^{thr,up}$), meaning that the peg has been completely inserted, the robot enters a *Stopping motion* state and starts decelerating towards zero target velocity in $z$ direction;
4. In order to prevent false positives, the robot continues its stopping motion unless the force in $z$ direction drops under a certain threshold ($F_z^{thr,down}$);
5. The task terminates in *Opening follower gripper* state.

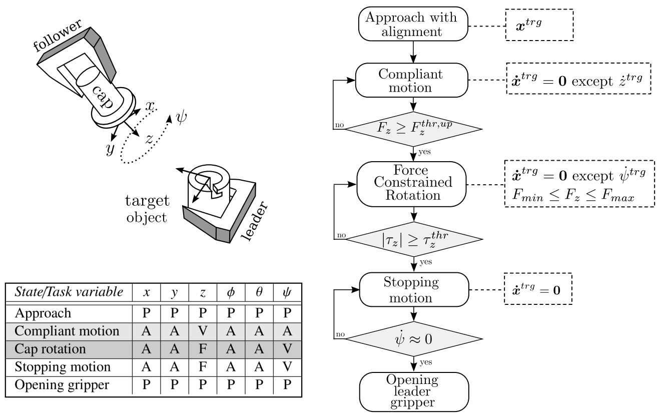
### 6.3. Experimental Results

Robustness to peg/hole misalignment has been evaluated by adding a position offset ($x$ direction) and an orientation offset ($\phi$ Euler angle) calculated with respect to the target end effector pose $\boldsymbol{x}^{trg}$ in the *Approach with alignment* state. An insertion velocity $\dot{z}^{trg}$ of 75 $mm/s$ has been chosen in order to achieve a considerable task execution speed. Selection of the admittance parameters has been performed according to (A.7), based on approximate knowledge of the environment (i.e. plastic) stiffness and damping. The following values have been used: $M = 0.2\ kg$, $M^{rot} = 0.1\ kgm^2/rad$ for the mass and rotational inertia, respectively, $D = 15\ Ns/m$, $D^{rot} = 15\ Nms/rad$ for the linear and rotational damping, respectively. Snapshots of a complete experiment and corresponding time histories of the follower-side estimated interaction force and end effector velocity are shown in Fig. 7 and Fig. 8, respectively. As shown in Fig. 8, the robot successfully performs the compliant peg-in-hole insertion in approximately 1.25 $s$ (red vertical line), achieving a considerable task completion speed. At time instant $t = 1.2\ s$ the absolute value of the force in the $z$ direction exceeds the threshold $F_z^{thr,up} = 1\ N$ (red dashed horizontal line) and the task execution enters the *Stopping motion* state, consistently with the state machine sequence in Fig. 6(a). Nevertheless, as soon as the force drops below the threshold $F_z^{thr,down} = 0.5\ N$ (blue dashed horizontal line), the robot resumes the *Compliant motion* state, until at time instant $t = 1.25\ s$ the peg-in-hole insertion terminates and the robot starts the *Stopping motion*.


follower
peg
$x$
$y$
$z$
hole
leader
Approach with alignment
$\boldsymbol{x}^{trg}$
Compliant motion
$\dot{\boldsymbol{x}}^{trg} = \mathbf{0}$ except $\dot{z}^{trg}$
no
$|F_z| \geq F_z^{thr,up}$
yes
Stopping motion
$\dot{\boldsymbol{x}}^{trg} = \mathbf{0}$
yes
$|F_z| \leq F_z^{thr,down}$
no
no
$\dot{z} \approx 0$
yes
Opening gripper

| *State/Task variable* | $x$ | $y$ | $z$ | $\phi$ | $\theta$ |
|---|---|---|---|---|---|
| Approach | P | P | P | P | P |
| Compliant motion | A | A | V | A | A |
| Stopping motion | A | A | V | A | A |
| Opening gripper | P | P | P | P | P |

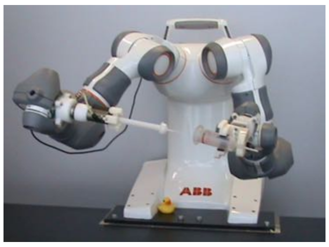
(a) Peg-in-hole insertion task

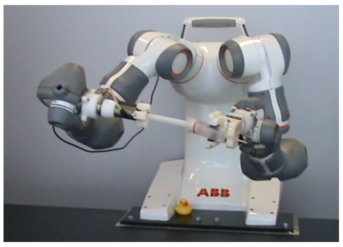

follower
cap
$x$
$y$
$z$
$\psi$
target object
leader
Approach with alignment
$\boldsymbol{x}^{trg}$
Compliant motion
$\dot{\boldsymbol{x}}^{trg} = \mathbf{0}$ except $\dot{z}^{trg}$
no
$F_z \geq F_z^{thr,up}$
yes
Force Constrained Rotation
$\dot{\boldsymbol{x}}^{trg} = \mathbf{0}$ except $\dot{\psi}^{trg}$
$F_{min} \leq F_z \leq F_{max}$
no
$|\tau_z| \geq \tau_z^{thr}$
yes
Stopping motion
$\dot{\boldsymbol{x}}^{trg} = \mathbf{0}$
no
$\dot{\psi} \approx 0$
yes
Opening leader gripper

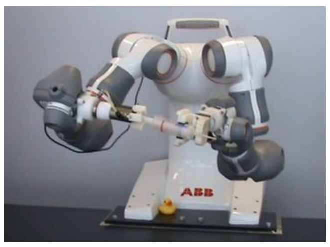
| *State/Task variable* | $x$ | $y$ | $z$ | $\phi$ | $\theta$ | $\psi$ |
|---|---|---|---|---|---|---|
| Approach | P | P | P | P | P | P |
| Compliant motion | A | A | V | A | A | A |
| Cap rotation | A | A | F | A | A | V |
| Stopping motion | A | A | F | A | A | V |
| Opening gripper | P | P | P | P | P | P |

(b) Cap rotation task

Figure 6: Task specification and assembly sequence for the peg-in-hole insertion and the cap rotation tasks. The considered frames are depicted in the upper left side, while the state machine governing the task execution is on the right side. The task variables' control mode during the task execution are reported on the table located on the lower left side (P: position control, V: velocity control, A: admittance control, F: force constraint).


(a) Approach with alignment


(b) Compliant motion skill


(c) Stopping motion

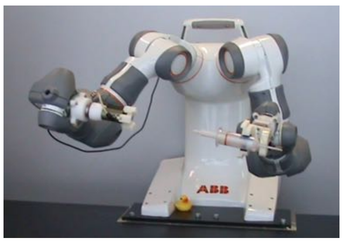
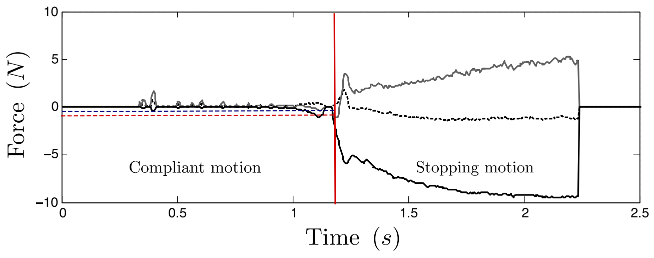

(d) Opening follower gripper

Figure 7: Snapshots from a peg-in-hole experiment.



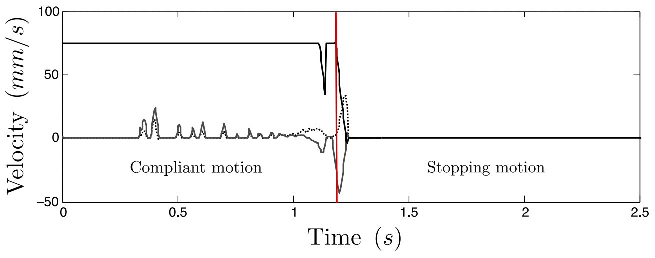


Figure 8: Time histories of the follower estimated interaction force and corresponding end effector linear velocity during the *Compliant motion* and *Stopping motion* phase ($x$: solid gray line, $y$: dotted black line, $z$: solid black line).

## 7. Phase 2: Cap Rotation

The kinematic description of the system state in terms of selected task variables is first introduced in Section 7.1. The finite state machine model governing the the task execution and experimental results are presented in Section 7.2 and Section 7.3, respectively.

### 7.1. Task Specification

The following vector of task variables has been chosen

$$\boldsymbol{x} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T \tag{46}$$

representing the relative linear and angular displacements (in terms of XYZ Euler angles) of the follower frame with respect to the leader frame, see Fig. 6(b). The screwing axis is assumed to be aligned with the follower $z$ direction. Note that now the Z Euler angle $\psi$, i.e. the screwing rotation, represents a task variable considered within the state vector. A bayonet mount ensures the fastening between the assembled parts.

### 7.2. Assembly Sequence

The complete task, shown in Fig. 6(b) together with the corresponding control mode for each task variable, is described in the following.

1. *Approach with alignment* - The follower end effector reaches the initial pose $\boldsymbol{x}^{trg}$.

2. *Compliant motion* - the follower motion along the $z$ direction is velocity-controlled, while a compliant motion skill is set along $(x, y, \phi, \theta, \psi)$;

3. *Force constrained rotation* - Loss of contact between the cap and the pipette body is achieved by conveniently bounding the interaction force arising from the contact along the screwing rotation axis $z$ within a lower and an upper bound $F_{min}$ and $F_{max}$. A velocity controlled rotation is simultaneously performed along $\psi$, while a compliant motion skill is set along $(x, y, \phi, \theta)$.

4. When the norm of the screwing torque $\tau_z$ exceeds the bound $\tau_z^{thr}$ required for the fastening of the bayonet mount, the robot enters the *Stopping motion* state and starts decelerating towards zero angular velocity $\dot{\psi}$;

5. The task terminates in *Opening leader gripper* state.

### *7.3. Experimental Results*

During the *Approach with alignment* phase, an orientation offset of 5 deg has been artificially introduced on the X and Y Euler angles of the follower end effector pose $\boldsymbol{x}^{trg}$, in order to evaluate robustness to possible misalignment. An approaching velocity $\dot{z}^{trg}$ of 1 $mm/s$ has been adopted to limit the impact force when the contact is established. Selection of the admittance parameters has been performed according to (A.7). The following values have been used: $M = 0.5\ kg$, $M^{rot} = 0.2\ kgm^2/rad$ for the mass and rotational inertia, respectively, $D = 40\ Ns/m$, $D^{rot} = 50\ Nms/rad$ for the linear and rotational damping, respectively. A threshold $F_z^{thr,up} = 0.5\ N$ has been used to switch to the subsequent *Force constrained rotation* phase. The lower and upper bounds on the interaction force are $F_{min} = 0.2\ N$ and $F_{max} = 0.5\ N$, respectively, while the considered uncertainty in the environment stiffness is: $K_{min} = 1300\ N/m$ and $K_{max} = 1600\ N/m$. An angular velocity $\dot{\psi}^{trg} = 0.1\ rad/s$ is applied during the screwing phase. Finally, a threshold value $|\tau_z^{thr,up}| = 0.05\ Nm$ has been chosen to complete the task execution by stopping the cap rotation and opening the leader gripper. Snapshots of the complete experiment are shown in Fig. 9, while the time history of the estimated contact force/torque is shown in Fig. 10. After approx. $t = 28\ s$ the interaction force exceeds the threshold $F_z^{thr,up} = 0.5\ N$ and the control execution enters the *Force constrained rotation* phase, during which contact loss is effectively prevented.

## 8. Complete Assembly Execution

An additional experiment has been performed in order to execute the complete pipette assembly sequence, consisting of the sequence of the plunger insertion phase and the cap rotation phase. Snapshots of the experiment are shown in Fig. 11. These results are also illustrated in the attached supplementary video.

The robot starts from an initial configuration, see Fig. 11(a), and inserts the plunger into the pipette body as described in Section 6.2, see Fig. 11(b) and Fig. 11(c). Afterwards, the right arm end effector reaches a predefined grasping pose to pick the cap, see Fig. 11(d) and Fig. 11(e). Due to the pipette bayonet mount fastening system, a suitable initial alignment between the cap and the pipette body, i.e. the initial pose Z Euler angle $\psi^{trg}$ for the cap rotation phase (see Section 7.2), has been achieved in a static "look and move" fashion [59] from RGB camera data, see Fig. 11(f). Finally, the cap rotation phase completes the assembly execution, see Fig. 11(g) and Fig. 11(h).

## 9. Conclusion

The redundancy and inherent compliance of modern collaborative robots naturally motivate their employment in assembly tasks. In this respect, a paradigm shift from traditional robot-level programming to skill-based programming allows to specify force control actions at task level and inherently provide compliant capabilities, without the need to specify the motions of the robot. A constraint-based formalism enabling the task-level specification of robotic skills that require force control policies has been presented for this purpose.

Compared to traditional robot-level programming, the proposed constraint-based programming method gives the skill developer the possibility to embed force control requirements within the specification of an assembly skill, and allows the non-expert user to intuitively program a complex assembly task by simple concatenation of assembly skills. Furthermore, the real-time generation of reactive robot motions, based on force control requirements, endows the robot controller with improved adaptation and robustness capabilities.

The proposed approach has been experimentally validated on the bimanual assembly of an Eppendorf combitip plastic pipette, requiring a plunger insertion phase and a cap rotation phase, using the ABB YuMi dual-arm robot. Estimation of the contact force/torque further enables the execution of the assembly operation without the use of extra force/torque sensors.

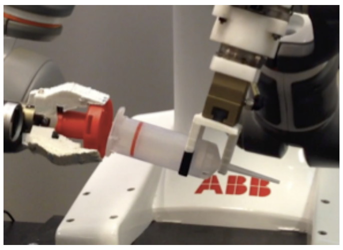
Finally, it is worth pointing out that the proposed framework represents a programming tool for the skill developer. Implementing the high-level software prototype which supports the specification of assembly tasks by sequencing of robotic skills represents a future research direction.

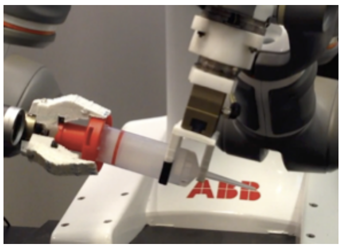
## Acknowledgment

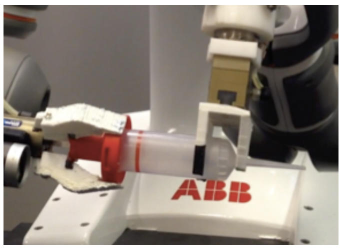
The authors would like to thank Sebastiano Castello, Francesco Incocciati and Alessandro Abbondanza for the valuable work performed on the experimental platform.

## Appendix A. Selection of Admittance Parameters

It is well-known that the selection of admittance parameters poses a major problem of stability. In order to tackle this issue, a simplified 1-DoF peg-in-hole task will be considered in the following. As sketched in Figure A.12, let $\theta$ and $\theta_{hole}$ represent the actual orientation of the peg and the actual orientation of the hole, respectively, while $\theta^{trg}$ represents an available estimate of


(a) Approach phase


(b) Compliant motion skill


(c) Force bounding skill

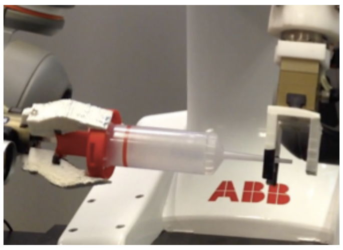

(d) Opening leader gripper

Figure 9: Snapshots taken from an experiment of bimanual cap assembly.

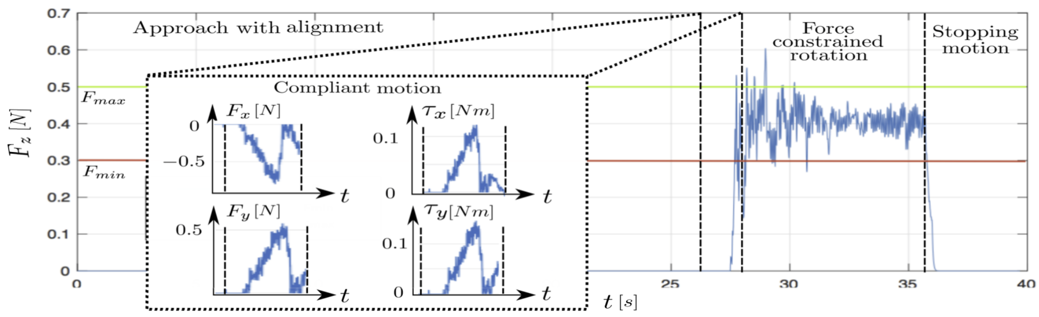
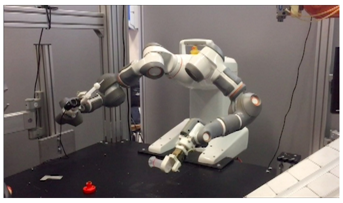
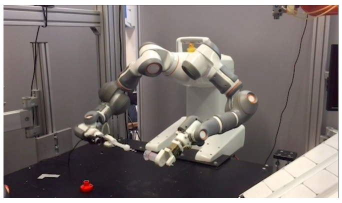
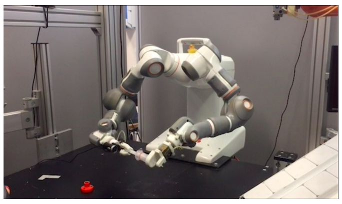
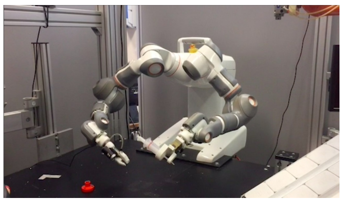


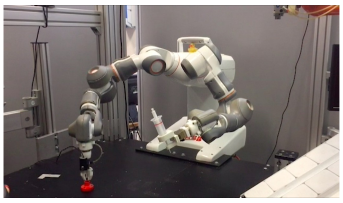
Figure 10: Time history of the estimated interaction force along the screwing axis $F_z$ and considered lower bound $F_{min}$ (solid red line) and upper bound $F_{max}$ (solid green line). Additional time history window of $F_x, F_y$ and $\tau_x, \tau_y$ during the *Compliant motion* phase.

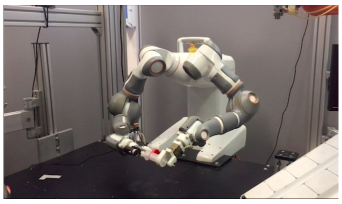

(a) Initial configuration

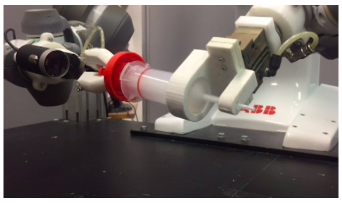

(b) Plunger insertion phase


(c) Plunger insertion phase


(d) Cap grasping phase


(e) Cap grasping phase


(f) Vision-based alignment


(g) Cap rotation phase

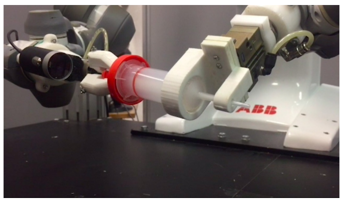

(h) Cap rotation phase

Figure 11: Snapshots taken from an experiment of a complete assembly sequence.

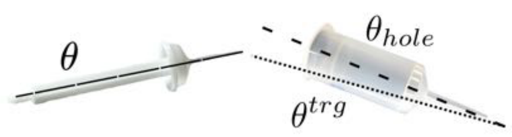


Figure A.12: 1-DOF representation of a peg-in-hole task. $\theta$ and $\theta_{hole}$ represent the actual orientation of the peg and the hole, respectively, while $\theta^{trg}$ represents a inaccurate estimate of the hole orientation.

the hole orientation, in order to consider additional uncertainty in the insertion problem. Due to the misalignment between the peg and the hole, a torque $\mu$ arises, according to the Kelvin-Voigt linear model

$$\mu = -K^{env}\left(\theta - \theta_{hole}\right) - D^{env}\dot{\theta} \tag{A.1}$$

where $K^{env}$ and $D^{env}$ are the environment rotational stiffness and damping, respectively. According to Section 4.1, the impedance relation

$$\mu_k = M\Delta\ddot{\theta}_k + D\Delta\dot{\theta}_k \tag{A.2}$$

$M$ and $D$ being the filter mass and damping coefficients, respectively, can be adopted to compute the orientation and angular velocity increments, $\Delta\theta_{k+1}$ and $\Delta\dot{\theta}_{k+1}$, respectively, responsible for a compliant motion

$$\begin{cases}\Delta\dot{\theta}_{k+1} = \Delta\dot{\theta}_k + T_s M^{-1}\left(\mu_k - D\Delta\dot{\theta}_k\right) \\ \Delta\theta_{k+1} = \Delta\theta_k + T_s\Delta\dot{\theta}_k + \dfrac{T_s^2}{2}M^{-1}\left(\mu_k - D\Delta\dot{\theta}_k\right)\end{cases} \tag{A.3}$$

Finally, as described in Section 4.3, the peg orientation and angular velocity references provided by the trajectory generation module, $\theta_{k+1}^{gen}$ and $\dot{\theta}_{k+1}^{gen}$, respectively, are added to the orientation and angular velocity increments $\Delta\theta_{k+1}$ and $\Delta\dot{\theta}_{k+1}$, respectively.

$$\begin{cases}\theta_{k+1} = \theta_{k+1}^{gen} + \Delta\theta_{k+1} \\ \dot{\theta}_{k+1} = \dot{\theta}_{k+1}^{gen} + \Delta\dot{\theta}_{k+1}\end{cases} \tag{A.4}$$

Selection of the admittance parameters, $M$ and $D$, ensuring closed-loop stability can be performed by considering as state vector

$$\boldsymbol{x}_k = \begin{bmatrix}\theta_k & \dot{\theta}_k & \Delta\theta_k & \Delta\dot{\theta}_k\end{bmatrix} \tag{A.5}$$

According to [13], a simple velocity-based trajectory generation algorithm can be employed as a trajectory generation module

$$\begin{cases}\ddot{\theta}_{k+1}^{gen} = T_s^{-1}(\dot{\theta}^{trg} - \dot{\theta}_k) = -T_s^{-1}\dot{\theta}_k \\ \dot{\theta}_{k+1}^{gen} = \dot{\theta}_k + T_s\ddot{\theta}_{k+1}^{gen} = 0 \\ \theta_{k+1}^{gen} = \theta_k + T_s\dot{\theta}_k + 0.5\ T_s^2\ \ddot{\theta}_{k+1}^{gen} = \theta_k\end{cases} \tag{A.6}$$

where saturation in the angular acceleration $\ddot{\theta}^{gen}$ has been neglected. A state-space representation of the closed-loop system can be obtained through equations (A.1), (A.4), (A.3), (A.6). Jury stability criterion can be subsequently applied in order to obtain necessary and sufficient conditions for the stability of the closed-loop system expressed with respect to the admittance parameters, yielding

$$\begin{cases}D > K^{env}T_s - D^{env} \\ M > \frac{3}{8}\ K^{env}\ T_s^2\end{cases} \tag{A.7}$$

## References

[1] L. Roveda, N. Pedrocchi, M. Beschi, L. M. Tosatti, High-accuracy robotized industrial assembly task control schema with force overshoots avoidance, Control Engineering Practice 71 (2018) 142–153.

[2] M. W. Abdullah, H. Roth, M. Weyrich, J. Wahrburg, An approach for peg-in-hole assembling using intuitive search algorithm based on human behavior and carried by sensors guided industrial robot, IFAC-PapersOnLine 48 (3) (2015) 1476–1481.

[3] D. Surdilovic, Y. Yakut, T. M. Nguyen, X. B. Pham, A. Vick, R. Martin-Martin, Compliance control with dual-arm humanoid robots: Design, planning and programming, in: IEEE-RAS International Conference on Humanoid Robots, 2010, pp. 275–281.

[4] J. Krüger, G. Schreck, D. Surdilovic, Dual arm robot for flexible and cooperative assembly, CIRP Annals - Manufacturing Technology 60 (1) (2011) 5 – 8.

[5] D. Almeida, F. E. Via, Y. Karayiannidis, Bimanual folding assembly: Switched control and contact point estimation, in: IEEE-RAS International Conference on Humanoid Robots, 2016, pp. 210–216.

[6] A. Nakashima, Y. Iwanaga, Y. Hayakawa, A motion planning of dual arm-hand manipulators for origami-folding based on a probabilistic model of constraint transitions within human behavior, in: IEEE International Conference on Robotics and Biomimetics (ROBIO), 2016, pp. 562–569.

[7] N. T. Dantam, H. B. Amor, H. I. Christensen, M. Stilman, Online multi-camera registration for bimanual workspace trajectories, in: IEEE-RAS International Conference on Humanoid Robots, 2014, pp. 588–593.

[8] P. K. Kim, J. H. Bae, H. Park, D. H. Lee, J. H. Park, M. H. Baeg, J. Park, Dual-arm robot box taping with kinesthetic teaching, in: International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), 2016, pp. 555–557.

[9] C. Bersch, B. Pitzer, S. Kammel, Bimanual robotic cloth manipulation for laundry folding, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2011, pp. 1413–1419.

[10] T. Lozano-Perez, Robot programming, Proceedings of the IEEE 71 (7) (1983) 821–841.

[11] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. Schou, S. Bøgh, V. Krüger, O. Madsen, Robot skills for manufacturing: From concept to industrial deployment, Robotics and Computer-Integrated Manufacturing 37 (2016) 282–291.

[12] C. Schou, R. S. Andersen, D. Chrysostomou, S. Bøgh, O. Madsen, Skill-based instruction of collaborative robots in industrial settings, Robotics and Computer-Integrated Manufacturing 53 (2018) 72–80.

[13] M. Parigi Polverini, A. M. Zanchettin, S. Castello, P. Rocco, Sensorless and constraint based peg-in-hole task execution with a dual-arm robot, in: IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 415–420.

[14] M. Parigi Polverini, A. M. Zanchettin, F. Incocciati, P. Rocco, Robust constraint-based robot control for bimanual cap rotation, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 4785–4790.

[15] A. M. Zanchettin, P. Rocco, Motion planning for robotic manipulators using robust constrained control, Control Engineering Practice 59 (2017) 127–136.

[16] A. De Luca, R. Mattone, Sensorless robot collision detection and hybrid force/motion control, in: IEEE International Conference on Robotics and Automation (ICRA), 2005, pp. 999–1004.

[17] A. De Luca, A. Albu-Schaffer, S. Haddadin, G. Hirzinger, Collision detection and safe reaction with the dlr-iii lightweight manipulator arm, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2006, p. 16231630.

[18] L. Villani, J. De Schutter, Force control, in: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, Ch. 7, pp. 161–185.

[19] D. Nicolis, A. M. Zanchettin, P. Rocco, Constraint-based and sensorless force control with an application to a lightweight dual-arm robot, IEEE Robotics and Automation Letters 1 (1) (2016) 340–347.

[20] Y. Li, G. Ganesh, N. Jarrassé, S. Haddadin, A. Albu-Schaeffer, E. Burdet, Force, Impedance, and Trajectory Learning for Contact Tooling and Haptic Identification, IEEE Transactions on Robotics 34 (5) (2018) 1170–1182.

[21] E. Lutscher, E. C. Dean-León, G. Cheng, Hierarchical Force and Positioning Task Specification for Indirect Force Controlled Robots, IEEE Transactions on Robotics 34 (1) (2018) 280–286.

[22] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: modelling, planning and control, Springer Science & Business Media, 2010.

[23] D.-H. Lee, M.-W. Na, J.-B. Song, C.-H. Park, D.-I. Park, Assembly process monitoring algorithm using force data and deformation data, Robotics and Computer-Integrated Manufacturing 56 (2019) 149–156.

[24] A. Billard, S. Calinon, R. Dillmann, S. Schaal, Robot programming by demonstration, in: Springer handbook of robotics, Springer, 2008, pp. 1371–1394.

[25] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: learning attractor models for motor behaviors, Neural computation 25 (2) (2013) 328–373.

[26] S. Calinon, F. Guenter, A. Billard, On learning, representing, and generalizing a task in a humanoid robot, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 37 (2) (2007) 286–298.

[27] B. D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robotics and autonomous systems 57 (5) (2009) 469–483.

[28] M. Hersch, F. Guenter, S. Calinon, A. Billard, Dynamical system modulation for robot learning via kinesthetic demonstrations, IEEE Transactions on Robotics 24 (6) (2008) 1463–1467.

[29] D. Lee, C. Ott, Incremental kinesthetic teaching of motion primitives us-

ing the motion refinement tube, Autonomous Robots 31 (2-3) (2011) 115–131.

[30] A. Muxfeldt, J.-H. Kluth, D. Kubus, Kinesthetic teaching in assembly operations–a user study, in: International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Springer, 2014, pp. 533–544.

[31] A. Kramberger, A. Gams, B. Nemec, D. Chrysostomou, O. Madsen, A. Ude, Generalization of orientation trajectories and force-torque profiles for robotic assembly, Robotics and Autonomous Systems 98 (2017) 333–346.

[32] S. B. Slotine, A general framework for managing multiple tasks in highly redundant robotic systems, in: proceeding of 5th International Conference on Advanced Robotics, Vol. 2, 1991, pp. 1211–1216.

[33] C. Samson, B. Espiau, M. L. Borgne, Robot control: the task function approach, Oxford University Press, 1991.

[34] J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, H. Bruyninckx, Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty, The International Journal of Robotics Research 26 (5) (2007) 433–455.

[35] N. Mansard, O. Khatib, A. Kheddar, A unified approach to integrate unilateral constraints in the stack of tasks, IEEE Transactions on Robotics 25 (3) (2009) 670–685.

[36] Y. Wang, F. Vina, Y. Karayiannidis, C. Smith, P. Ogren, Dual arm manipulation using constraint based programming, IFAC Proceedings Volumes 47 (3) (2014) 311–319.

[37] Y. Wang, H. Liu, W. Ji, L. Wang, Realtime collaborating with an industrial manipulator using a constraint-based programming approach, Procedia CIRP 72 (1) (2018) 105–110.

[38] A. Stolt, M. Linderoth, A. Robertsson, R. Johansson, Force controlled assembly of emergency stop button, in: IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 3751–3756.

[39] A. Stolt, M. Linderoth, A. Robertsson, R. Johansson, Force controlled robotic assembly without a force sensor, in: IEEE International Conference on Robotics and Automation (ICRA), 2012, pp. 1538–1543.

[40] A. Stolt, M. Linderoth, A. Robertsson, R. Johansson, Robotic assembly of emergency stop buttons, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013, pp. 2081–2081.

[41] F. Dai, A. Wahrburg, B. Matthias, H. Ding, Robot assembly skills based on compliant motion, in: ISR 2016: 47st International Symposium on Robotics; Proceedings of, VDE, 2016, pp. 1–6.

[42] L. Halt, F. Nagele, P. Tenbrock, A. Pott, Intuitive constraint-based robot programming for robotic assembly tasks, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 520–526.

[43] L. Biagiotti, C. Melchiorri, Trajectory planning for automatic machines and robots, Springer Science & Business Media, 2008.

[44] A. M. Zanchettin, P. Rocco, Reactive motion planning and control for compliant and constraint-based task execution, in: IEEE Conference on Robotics and Automation (ICRA), 2015, pp. 2748–2753.

[45] F. Blanchini, Set invariance in control, Automatica 35 (11) (1999) 1747–1767.

[46] J. Wolff, M. Buss, Invariance control design for nonlinear control affine systems under hard state constraints, IFAC Symposium on Nonlinear Control Systems (NOLCOS) 37 (1) (2004) 555–560.

[47] J. Wolff, M. Buss, Invariance control design for constrained nonlinear systems, IFAC Proceedings Volumes 38 (1) (2005) 37–42.

[48] O. Kanoun, F. Lamiraux, P. B. Wieber, Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task, IEEE Transactions on Robotics 27 (4) (2011) 785–792.

[49] A. Escande, N. Mansard, P.-B. Wieber, Hierarchical quadratic programming: Fast online humanoid-robot motion generation, International Journal of Robotics Research 3 (7) (2014) 1006 – 1028.

[50] N. Hogan, Impedance control: An approach to manipulation, part I - theory, ASME Journal of Dynamic Systems, Measurement, and Control 107 (1985) 1–7.

[51] B. Barmish, J. Sankaran, The propagation of parametric uncertainty via polytopes, IEEE Transactions on Automatic Control 24 (2) (1979) 346–349.

[52] M. Parigi Polverini, A. M. Zanchettin, P. Rocco, A computationally efficient safety assessment for collaborative robotics applications, Robotics and Computer-Integrated Manufacturing 46 (2017) 25–37.

[53] R. Rossi, M. Parigi Polverini, A. M. Zanchettin, P. Rocco, A pre-collision control strategy for human-robot interaction based on dissipated energy in potential inelastic impacts, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 26–31.

[54] H. Ferreau, C. Kirches, A. Potschka, H. Bock, M. Diehl, qpOASES: A parametric active-set algorithm for quadratic programming, Mathematical Programming Computation 6 (4) (2014) 327–363.

[55] T. Kröger, F. M. Wahl, Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events, IEEE Transactions on Robotics 26 (1) (2010) 94–111.

[56] M. Ragaglia, A. M. Zanchettin, L. Bascetta, P. Rocco, Accurate sensorless lead-through programming for lightweight robots in structured environments, Robotics and Computer-Integrated Manufacturing 39 (2016) 9 – 21.

[57] A. Gill, Introduction to the Theory of Finite-state Machines, McGraw-Hill, New York, NY, USA, 1962.

[58] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, C. J. Taylor, A vision-based formation control framework, IEEE Transactions on Robotics and Automation 18 (5) (2002) 813–825.

[59] L. Weiss, A. Sanderson, C. Neuman, Dynamic sensor-based control of robots with visual feedback, IEEE Journal on Robotics and Automation 3 (5) (1987) 404–417.