Combining Imitation Learning With Constraint-Based Task Specification and Control

Cristian Alejandro Vergara Perico, Joris De Schutter, and Erwin Aertbeliën

***Abstract*—This letter combines an imitation learning approach with a model-based and constraint-based task specification and control methodology. Imitation learning provides an intuitive way for the end user to specify context of a new robot application without the need of traditional programming skills. On the other hand, constraint-based robot programming allows us to define complex tasks involving different kinds of sensor input. Combination of both enables adaptation of complex tasks to new environments and new objects with a small number of demonstrations. The proposed method uses a statistical uni-modal model to describe the demonstrations in terms of a number of weighted basis functions. This is then combined with model-based descriptions of other aspects of the task at hand. This approach was tested in a use case inspired by an industrial application, in which the required transfer motions were learned from a small number of demonstrations, and gradually improved by adding new demonstrations. Information on a collision-free path was introduced through a small number of demonstrations. The method showed a high level of composability with force and vision controlled tasks. The use case showed that the deployment of a complex constraint-based task with sensor interactions can be expedited using imitation learning.**

***Index Terms*—Learning from demonstration, compliance and impedance control, sensor-based control.**

I. Introduction

A GENERAL tendency in manufacturing is the move towards more flexible automation methodologies that can deal with smaller batch sizes of more customized products. To this end, robotic applications must be rapidly reconfigurable to deal with a great variety of tasks while speeding up their deployment on the factory floor. When deploying robotic applications, traditional methods such as manual programming and teleoperation can be time-consuming and they require a high level of expertise from the operator. These drawbacks have been addressed by two distinct communities in the robotic field.

First, the learning community argues that learning methodologies can be used to facilitate task programming while at the same time speeding up deployment [1]. Particularly, in the imitation learning approach a motion model is generalized based on movement primitives learned from observation of humans performing a task [2]. When generalizing robotic tasks with learning methodologies, two aspects should be considered. On one hand, complex controllers are fully developed to deal with applications that require complex robot behaviors that depend on sensor input. On the other hand, a great number of parameters are needed to learn such robotic task, therefore, in most cases, a large number of demonstrations need to be performed. To demonstrate as many robot trajectories as needed, a possible approach consists of introducing demonstrations incrementally, in other words, a robot can already start to perform a task after a small number of demonstrations and the operator will intervene only when he considers it necessary to refine the motion model.

Second, the constraint-based community addresses these problems in a totally different way by proposing a framework which separates task specification from control while offering a high level of composability [3]. In such framework a great variety of reactive robotic behaviors can be combined based on several sensor inputs such as force and vision. As a result, a user with a low level of expertise in control can develop a robotic application more quickly by specifying, reusing and combining application constraints, such as task related constraints (e.g. command a robot to follow a path with a certain impedance), robot related constraints (e.g. robot joint limits), environment related constraints (e.g. application workspace), and constraints related to the interaction with a human operator.

The main contribution of this article is the integration of an imitation learning algorithm with real-time constraint-based reactive control. This combination allows an application design engineer to expedite the development of a robotic application that requires reactive control by offering a flexible way to introduce information about the task: either by demonstrating or by specifying aspects of the task. The specification of these aspects can be done by directly introducing constraints, or complementarily, by explicitly selecting the degrees of freedom that will be learned. This way of specification decreases the amount of information that needs to be generalized by a learning methodology, hence, it reduces the number of demonstrations needed. For instance, the translational part of a trajectory can be learned from a small number of demonstrations, while the orientation along the path, and other aspects, such as impedance behavior, joint limits, and workspace limits can be specified conveniently using the constraint-based methodology. In this letter, as an example of this approach, we integrate the learning algorithm:

Manuscript received September 10, 2018; accepted January 13, 2019. Date of publication February 7, 2019; date of current version February 28, 2019. This letter was recommended for publication by Associate Editor M. Howard and Editor D. Lee upon evaluation of the reviewers' comments. This work was supported by Flanders Make Projects YVES and FINROP. *(Corresponding author: Cristian Alejandro Vergara Perico.)*

The authors are with the Department of Mechanical Engineering, Katholieke Universiteit Leuven, Core Lab ROB, Flanders Make, Leuven 3001, Belgium (e-mail: cristian.vergara@mech.kuleuven.be; joris.deschutter@mech.kuleuven.be; erwin.aertbelien@mech.kuleuven.be).

This letter has supplemental downloadable multimedia material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2019.2898035

probabilistic principal component analysis (PPCA) [4] with the constraint-based reactive framework: *expression graph based task specification language* (eTaSL) [5], [6]. This choice is motivated by the fact that the PPCA algorithm encodes the trajectory shape information from a small number of demonstrations: it generalizes new trajectories in function of latent variables which can be specified directly as controlled variables using eTaSL feature variables (variables that allow the specification of extra degrees of freedom in the system). This approach offers three main advantages: (i) the capability to generalize trajectories towards non-demonstrated targets; (ii) the ability to assist a human by generating an impedance depending on the variability of the demonstrations; and (iii) the possibility of combining the mentioned behaviors with sensor inputs (force and vision signals) as well as other aspects of the robotic task, such as joint limits, workspace limits, etc. As a result, the proposed synergy facilitates the specification while expediting deployment.

II. Related Work

Besides eTaSL, several other constraint-based methodologies have been reported in literature: iTaSC [3] offers a systematic way to formulate constraints; stack-of-tasks [7] avoids conflicting constraints by putting the constraints in a priority stack. Borghesan and De Schutter [8] extended the iTaSC framework towards tasks that involve physical interaction with the environment. They designed a task-level force and impedance control using conflicting constraints based on signals coming from the back-drivability of the robot. Advantages of eTaSL are the complete separation between specification and control, and the possibility of defining auxiliary control variables as feature variables, which offers a tight integration with our learning algorithm.

The use of point distribution models to encode trajectory information while reducing dimensionality has been used to tackle problems of trajectory classification and/or generation. For instance, Roduit et al. [9] used a PCA algorithm for trajectory classification by assessing the variability between trajectories generated by different mobile platforms, and between model predictions and real evaluations. In this work, similarly as in [10], a generative model based on the PPCA method is used to learn basis functions directly from the variations in demonstrations. However, instead of estimating the latent variables by using a Kalman filter, these values are calculated by taking advantage of the feature variables and the behavior composability offered by eTaSL.

Other probabilistic learning methodologies that encode information from demonstrations have been reported in literature: *Gaussian mixture models* (GMM) proposed by Calinon et al. [11] to encode trajectory signals in a mixture of Gaussian functions; *dynamic movement primitives* (DMPs) used in [12], [13] to encode dynamic information in the weights of Gaussian basis functions; and *probabilistic motion primitves* (ProMPs) used in [14], [15], which avoid to encode dynamic information that can lead to dynamic instabilities. Further research has increased the flexibility of the mentioned algorithms to generalize trajectories towards non-demonstrated contexts. This is the case for *task parameterized GMM* (TP-GMM) [16] and *task parameterized DMP* (TP-DMP) [17], which parameterize relevant aspects of the task such as poses at the initial and end location of a trajectory, different reference frames, obstacles in between etc. Dedicated controllers have been developed to tailor the dynamic behaviors of each of the mentioned algorithms. One example is the use of a linear quadratic regulator (LQR) in [16], which allows the projection of extra task parameters in a form of soft constraints prioritized by probabilities encoded with the TP-GMM. Although these algorithms allow the specification of a great variety of tasks, to the authors' knowledge no implementations have been reported that handle tasks expressed using inequality constraints. Hence, these algorithms cannot deal with tasks that require the specification of joint, velocity, or workspace limits.

Another advantage of the proposed integration over the state of the art is the possibility of defining force-based constraints. This allows the system to deal with applications that require narrow tolerances that cannot be achieved with the accuracy of standard robots and vision systems. Moreover, the capability of defining auxiliary control variables allows the system to define more compelling force behaviors such as commanding a robot end effector to advance along a generated trajectory with a specific velocity profile, while still allowing the possibility of holding, or even moving the end effector backwards along the trajectory by a human applying appropriate interaction forces.

The synergy between the PPCA algorithm and eTaSL allows us to guide the user when performing additional demonstrations, by imposing a variable impedance in function of the variability of previous demonstrations. Several authors have reported other strategies for incremental learning. For instance, Calinon and Billard [18], [19] proposed a method where an initial observation is done by mapping human motions to a humanoid robot. The embodiment issues (e.g. difference in length of human and robot links) are solved by refining the motion model, i.e. perturbing the performed task using kinesthetic teaching. Similarly, Lee and Ott [20] proposed a method in which the motion model is incrementally taught by perturbation via kinesthetic teaching with a variable impedance in function of the deviation from a generated trajectory. Ewerton et al. [15] also improved a motion model by giving information about new contexts between demonstrations. This was done by either fully demonstrating new trajectories or by perturbing executed trajectories. Abi-Farraj et al. [13] proposed an incremental learning strategy using human/robot shared control with impedance behavior in function of the variability encoded in the learned model. In contrast, in this work the variable impedance changes along the generated trajectory, i.e. a close correspondence between demonstrations in a particular section of the trajectory generates a high impedance, while a large variability generates low impedance.

III. Motion Model Learning Algorithm

In this letter, we define a *motion model* as a way to describe the preferred and allowable motions of the robot end effector. This model is extracted from a small number d of demonstrations which are made invariant with respect to their velocity profile.

This allows us to add a desired velocity, later, when generating new trajectories. However, tasks that are velocity dependent cannot be generalized with this model.

The proposed learning model defines the motions in an o-dimensional space. Besides specifying constraints on the motion (see Sec. V), we can explicitly select the degrees of freedom (DOF) to learn. For instance, three DOF representing orientation of an end effector, or seven DOF representing joint space. In the use case of this letter, we explicitly choose to learn the shape information corresponding to a pure 3D position of a trajectory ($o = 3$). However, our framework is not limited to this choice.

A. Data Pre-Processing

To enable similarity between demonstrations based on their shape, we propose the following procedure:

1) All demonstrations are re-sampled to have the same number n of samples. Then, as described in [21], demonstration vectors $\hat{\boldsymbol{h}}_{\boldsymbol{i}} \in \mathbb{R}^{n \times o}$ are made invariant with respect to time and linear scale by making signals dependent on the normalized *path coordinate* s (related to the arc length) instead of time t: $\hat{\boldsymbol{h}}_{\boldsymbol{i}}(t_j) \rightarrow \hat{\boldsymbol{f}}_{\boldsymbol{i}}(s_j)$, where $s_j = \frac{\sum_{k=1}^{j} \|\hat{\boldsymbol{h}}_{\boldsymbol{i}}(t_k) - \hat{\boldsymbol{h}}_{\boldsymbol{i}}(t_{k-1})\|}{\sum_{k=1}^{n} \|\hat{\boldsymbol{h}}_{\boldsymbol{i}}(t_k) - \hat{\boldsymbol{h}}_{\boldsymbol{i}}(t_{k-1})\|}$, and $\hat{\boldsymbol{f}}_{\boldsymbol{i}}(s_j) \in \mathbb{R}^{n \times o}$. In this way trajectories with different scale and velocity can be compared, while allowing generation of new trajectories with different motion profiles.
2) To find the sample correspondences between demonstrations, each signal is compared against the pairwise trajectory average using a combination of Fast Dynamic Time Warping (FaDTW) [22] and Nonlinear Alignment and Averaging Filter (NLAAF) [23]. With this methodology the similarity between demonstrations is maximized, obtaining a remapped vector $\boldsymbol{f}_{\boldsymbol{i}}^{*}(s_j) \in \mathbb{R}^{n \times o}$ for each demonstration.
3) As in [10], a matrix $\bar{\boldsymbol{F}} \in \mathbb{R}^{d \times o\, n}$ is assembled, with each row $\bar{\boldsymbol{F}}_i$ containing the concatenation of all *DOFs* corresponding to a demonstration, as follows:

$$\bar{\boldsymbol{F}} = \begin{bmatrix} \left[\boldsymbol{f}_{1,1}^{*}(s_1) \ldots \boldsymbol{f}_{1,1}^{*}(s_n)\right] \ldots \left[\boldsymbol{f}_{1,o}^{*}(s_1) \ldots \boldsymbol{f}_{1,o}^{*}(s_n)\right] \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ \left[\boldsymbol{f}_{d,1}^{*}(s_1) \ldots \boldsymbol{f}_{d,1}^{*}(s_n)\right] \ldots \left[\boldsymbol{f}_{d,o}^{*}(s_1) \ldots \boldsymbol{f}_{d,o}^{*}(s_n)\right] \end{bmatrix}. \quad (1)$$

B. Learning Model

Assuming a unimodal Gaussian distribution of each row in 1, a trajectory $\bar{\boldsymbol{f}} \in \mathbb{R}^{o\, n \times 1}$ can be represented using the following *Latent Variable* model, referred to as *Probabilistic Principle Component Analysis* (*PPCA*) in [4]:

$$\bar{\boldsymbol{f}} = \bar{\boldsymbol{W}} \boldsymbol{\chi}_{f,2} + \bar{\boldsymbol{b}} + \bar{\boldsymbol{\epsilon}}, \quad (2)$$

where $\bar{\boldsymbol{W}} \in \mathbb{R}^{o\, n \times m}$ is a matrix with m basis functions, $\boldsymbol{\chi}_{f,2} \in \mathbb{R}^{m \times 1}$ a vector with m latent variables, $\bar{\boldsymbol{b}} \in \mathbb{R}^{o\, n \times 1}$ an offset vector, and $\bar{\boldsymbol{\epsilon}} \in \mathbb{R}^{o\, n \times 1}$ a noise vector. The latent variables and the noise can be approximated by a Gaussian distribution $\boldsymbol{\chi}_{f,2} \sim \mathcal{N}(0, \boldsymbol{I})$ and $\bar{\boldsymbol{\epsilon}} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$, respectively. Therefore, $\bar{\boldsymbol{f}} \sim \mathcal{N}(\bar{\boldsymbol{b}}, \bar{\boldsymbol{W}} \bar{\boldsymbol{W}}^T + \sigma^2 \boldsymbol{I})$, $\bar{\boldsymbol{W}}$ and $\bar{\boldsymbol{b}}$ are estimated using *maximum likelihood*. As explained in [4] the parameters can be found using the *PPCA* algorithm:

$$\bar{\boldsymbol{b}} = \frac{1}{d} \sum_{i=1}^{d} \bar{\boldsymbol{F}}_i^T, \quad (3)$$

$$\boldsymbol{D} = \begin{bmatrix} \bar{\boldsymbol{F}}_1^T - \bar{\boldsymbol{b}} & \ldots & \bar{\boldsymbol{F}}_d^T - \bar{\boldsymbol{b}} \end{bmatrix}, \quad (4)$$

$$\sigma_{ML}^2 = \frac{1}{o\, n - m} \sum_{j=m+1}^{d} \lambda_j, \quad (5)$$

$$\bar{\boldsymbol{W}} = \boldsymbol{U}_m (\boldsymbol{\Lambda}_m - \sigma_{ML}^2 \boldsymbol{I})^{1/2} \boldsymbol{R}, \quad (6)$$

where $\boldsymbol{U}_m$ and $\{\lambda_1, \ldots, \lambda_m\}$ are the eigenvectors and eigenvalues of the covariance matrix $\boldsymbol{D}\boldsymbol{D}^T$, $\boldsymbol{\Lambda}_m$ is a diagonal matrix containing the eigenvalues $\boldsymbol{\Lambda}_m = diag(\lambda_1, \ldots, \lambda_m)$, and $\boldsymbol{R}$ corresponds to an arbitrary rotation matrix that, for convenience, is chosen as the identity $\boldsymbol{I}$. Since the PPCA starts from a covariance matrix $\boldsymbol{D}\boldsymbol{D}^T$ it scales linearly with the number of demonstrations d. For most practical purposes the computation of the eigenvalues dominates the computation of the covariance matrix. Although involving the eigenvalue decomposition of a large matrix, the computations are still relatively fast (less than 1 second).

A point in a trajectory is expressed in function of a path coordinate and the latent variables by:

$$\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2}) = \boldsymbol{W}(s) \boldsymbol{\chi}_{f,2} + \boldsymbol{b}(s) \quad (7)$$

where $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2})$, $\boldsymbol{b}(s) \in \mathbb{R}^{o \times 1}$, $\boldsymbol{W}(s) \in \mathbb{R}^{o \times m}$, and $\boldsymbol{\chi}_{f,2} \in \mathbb{R}^{m \times 1}$. As in [10], $\boldsymbol{W}(s)$ and $\boldsymbol{b}(s)$ are calculated by interpolating corresponding neighbors for each *DOF* in $\bar{\boldsymbol{W}}$ and $\bar{\boldsymbol{b}}$, respectively, followed by a vertical concatenation of the resulting interpolated functions.

In this model, the latent variables $\boldsymbol{\chi}_{f,2}$ explain correlation between demonstrations, while $\bar{\boldsymbol{\epsilon}}$ explains variability within a single demonstration. $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2})$ can be easily represented as an expression in the eTaSL language as described in Sec. IV.

IV. eTaSL Framework

This section highlights the aspects from the eTaSL framework presented in [5] that are particularly relevant for this work.

eTaSL builds up an *expression graph* that completely describes robot-related variables used during the task specification, such as vectors, transformation matrices, rotation matrices, along with others. During controller execution, the value of these expressions can be computed. Also time derivatives and partial derivatives can be automatically computed using *automatic differentiation* [24]. The transformation of the task specification into an optimization problem is fully automated, such that the robot programmer only needs to specify the expressions and the variables that are used. In eTaSL, soft conflicting constraints are defined using *weighted* constraints.

Besides *time* variable t and *robot joints* variables $\boldsymbol{q}$, eTaSL introduces *feature* variables $\boldsymbol{\chi}_f$. These variables are used to specify extra *degrees of freedom* in the system. Therefore, eTaSL

allows the definition of variables that command the behaviour of geometric and task expressions (e.g. the shape and velocity along a path).

The main types of task constraints and their transformation into constraints on velocities are described below:

Nominal Constraint: In 8 a velocity-resolved controller enforces this constraint such that a task expression $e_i(\boldsymbol{q}, \boldsymbol{\chi}_f, t)$ evolves towards *zero* value while behaving as a first-order system with time constant k_i^{-1}.

Time Derivative Constraint: In 9 a velocity-resolved controller commands the time derivative of a task expression $g_i(\boldsymbol{q}, \boldsymbol{\chi}_f, t)$ to follow a desired velocity v. This is introduced as nominal constraint 8 with $k_i = 0$.

Task Function Constraint: The eTaSL task specification framework [5], [6] further splits up the task function such that a *nominal constraint* in 8 can be expressed as 10: $\text{mdl}_i(\boldsymbol{q}, \boldsymbol{\chi}_f, t)$ describes the model of a task variable on the task space; meas_i is a corresponding measurement of the task variable; and tgt_i is the desired target towards which the task variable needs to evolve. This allows us not only to specify geometric constraints, but also constraints based on sensor values using an admittance control strategy.

$$\frac{d}{dt} e_i(\boldsymbol{q}, \boldsymbol{\chi}_f, t) = -k_i e_i(\boldsymbol{q}, \boldsymbol{\chi}_f, t), \tag{8}$$

$$\frac{d}{dt} (g_i(\boldsymbol{q}, \boldsymbol{\chi}_f, t) - v\, t) = 0, \tag{9}$$

$$\frac{d}{dt} \text{mdl}_i(\boldsymbol{q}, \boldsymbol{\chi}_f, t) = -k_i(\text{meas}_i - \text{tgt}_i). \tag{10}$$

For each of these constraints a weight w_i is specified.

At each time instant the eTaSL controller computes a new value for its controller output by solving an optimization problem in terms of joint velocities, feature variable velocities and slack variables. The optimization problem is formulated as a *Quadratic Programming* (QP) problem and solved using the optimization toolbox *qpOASIS* [25]:

$$\underset{\boldsymbol{x}}{\text{minimize}} \qquad \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} \tag{11a}$$

$$\text{subject to} \qquad \boldsymbol{L_A} \leq \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{U_A} \tag{11b}$$

The argument $\boldsymbol{x}$ is a vector $\begin{bmatrix} \dot{\boldsymbol{q}}^T & \dot{\boldsymbol{\chi}}_f^T & \boldsymbol{\varepsilon}^T \end{bmatrix}^T$ that contains: the time derivative of the robot joints $\dot{\boldsymbol{q}}$, the time derivative of the feature variables $\dot{\boldsymbol{\chi}}_f$, and the slack variables $\boldsymbol{\varepsilon}$, which are added automatically to implement *weighted* constraints. Weights w_i are included in the Hessian matrix $\boldsymbol{H}$ to weigh the square of each slack variable.

To map the defined constraints to 11b in the optimization problem, we first expand the total time derivative of a task variable e_i in its partial derivatives with respect to time, with respect to the joint variables, and with respect to the feature variables [5]:

$$\frac{de_i}{dt} = \frac{\partial e_i}{\partial t} + \sum_{j=1}^{n_r} \frac{\partial e_i}{\partial q_j} \dot{q}_j + \sum_{k=1}^{n_f} \frac{\partial e_i}{\partial \chi_{f,k}} \dot{\chi}_{f,k}, \tag{12}$$

where n_r is the number of robot joints and n_f the number of feature variables.

Applying the above to 8 to 10 and rewriting the partial derivatives using a Jacobian matrix, gives constraints in the form of 11b:

$$\boldsymbol{J}_i \begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{\chi}}_f \end{bmatrix} = -k_i e_i - \frac{\partial e_i}{\partial t} + \varepsilon_i, \tag{13}$$

$$\boldsymbol{J}_i \begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{\chi}}_f \end{bmatrix} = v - \frac{\partial g_i}{\partial t} + \varepsilon_i, \tag{14}$$

$$\boldsymbol{J}_{\text{mdl}_i} \begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{\chi}}_f \end{bmatrix} = -k_i(\text{meas}_i - \text{tgt}_i) - \frac{\partial \text{mdl}_i}{\partial t} + \varepsilon_i. \tag{15}$$

The task function Jacobians $\boldsymbol{J}_i$ and $\boldsymbol{J}_{\text{mdl}_i}$ map the time derivatives of states $\boldsymbol{q}$ and $\boldsymbol{\chi}_f$ into the time derivatives of the task variables.

The eTaSL framework allows us not only to define equality constraints, but also inequality constraints with lower bounds $\boldsymbol{L_A}$ and/or upper bounds $\boldsymbol{U_A}$ in 11b.

V. Model-Based and Constraint-Based Task Specification

In this section we formulate general-purpose constraints using the eTaSL specification language. These general-purpose constraints can be combined to create a variety of behaviors, as needed by a particular application.

A. Constraint Definition

Constraint 1 *Force Control*: To interact with the environment, as in [26], an admittance control based on sensed forces is proposed to command the end effector translations. This behaviour can be specified in eTaSL using 10:

$$\text{mdl}_1 := \boldsymbol{p}(\boldsymbol{q}) \quad \text{meas}_1 := -\boldsymbol{C_a}\, \boldsymbol{u} \quad \text{tgt}_1 := -\boldsymbol{C_a}\, \boldsymbol{u_d}, \tag{16}$$

where $\boldsymbol{p}(\boldsymbol{q}) \in \mathbb{R}^{3\times 1}$ is the position vector of a tool point (dependent on the robot joints $\boldsymbol{q}$); $\boldsymbol{C_a} \in \mathbb{R}^{3\times 3}$ is a matrix with compliance coefficients; $\boldsymbol{u} \in \mathbb{R}^{3\times 1}$ a vector with the measured force signals; and $\boldsymbol{u_d} \in \mathbb{R}^{3\times 1}$ corresponds to the desired force. $\boldsymbol{u_d}$ is chosen zero when the robot kinesthetically needs to follow human motion (see Fig. 1).

Constraint 2 *Latent variables*: The m latent variables $\boldsymbol{\chi}_{f,2}$ described in III-B are introduced in the optimization problem 11a as feature variables. Nominal constraints on these latent variables are specified such that their preferred value is close to zero:

$$\boldsymbol{e_2} := \boldsymbol{\chi}_{f,2} - \boldsymbol{0}_{m\times 1}. \tag{17}$$

This constraint pushes $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2})$ towards the mean trajectory $\boldsymbol{b}(s)$ of the learned model (see Fig. 1).

Constraint 3 *Trajectory generation*: A pose in a generated trajectory is composed by an orientation $\boldsymbol{\psi}_f(s) \in \mathbb{R}^{3\times 1}$ represented using Euler angles, and a position $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2}) \in \mathbb{R}^{3\times 1}$. The orientation $\boldsymbol{\psi}_f(s)$ is calculated by a linear interpolation in the axis-angle representation in function of s between a known initial orientation at s_0 and an orientation obtained from the camera, which corresponds to the end of the path coordinate

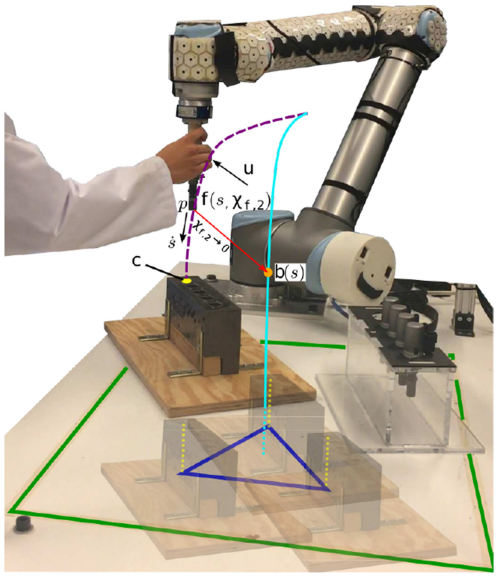

Fig. 1. (taken from the use case in Sec. VI) Guided kinesthetic teaching behavior when adding an extra demonstration towards a target position c. While the green rectangle represents the total application work space of target positions, the blue triangle represents the space covered by the target positions of previous demonstrations. Target position c is obtained by a camera. The deviation of the generated trajectory $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2})$ from the average of previous demonstrations $\boldsymbol{b}(s)$ is given by the local variance $\boldsymbol{W}(s)$ times the latent variables $\boldsymbol{\chi}_{f,2}$. As a result the operator has to apply a variable force $\boldsymbol{u}$ along the new demonstration.

s_n. The model for the desired trajectory position $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2})$ is obtained from 7. The feature variables $\boldsymbol{\chi}_{f,2}$ in this expression are adapted by eTaSL in order to satisfy a constraint between the trajectory end position $\boldsymbol{f}(s_n, \boldsymbol{\chi}_{f,2})$ and position $\boldsymbol{c} \in \mathbb{R}^{3\times 1}$, also obtained from the camera:

$$\boldsymbol{e_3} := \boldsymbol{f}(s_n, \boldsymbol{\chi}_{f,2}) - \boldsymbol{c}. \tag{18}$$

Constraint 4 *Velocity along the path*: A similar approach to [27] is introduced to control the velocity of $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2})$ along the path (Fig. 1). First, the path coordinate s is defined as a feature variable. Then, a trapezoidal motion profile v_f is defined such that: (i) during the starting phase v_f rises at a_{max} rate until reaching v_{max}, (ii) during the middle phase v_f keeps value v_{max}, (iii) during the final phase v_f decreases at $-a_{max}$ rate until zero. For short paths, the middle phase could be skipped. The path coordinate s (see Fig. 1) is constrained to follow this motion profile v_f by defining the task variable using 9:

$$g_4 := s \qquad \text{with } v := v_f. \tag{19}$$

Constraint 5 *Remain on path*: Constraints 2, 3 and 4 describe the motion of a point along a path. This constraint relates the motion of the robot end effector to this point by adding a nominal constraint 8 to minimize distance between the tool position $\boldsymbol{p}(\boldsymbol{q})$ and the trajectory point $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2})$. The resulting task function is specified as:

$$\boldsymbol{e_5} := \boldsymbol{p}(\boldsymbol{q}) - \boldsymbol{f}(s, \boldsymbol{\chi}_{f,2}). \tag{20}$$

Constraint 6 *Tool orientation*: In a similar way as in *Constraint 5* the orientation of the end effector represented by Euler angles $\boldsymbol{\psi}_p(\boldsymbol{q}) \in \mathbb{R}^{3\times 1}$ is constrained to the trajectory orientation $\boldsymbol{\psi}_f(s)$. The resulting task function is specified as:

$$\boldsymbol{e_6} := \boldsymbol{\psi}_p(\boldsymbol{q}) - \boldsymbol{\psi}_f(s). \tag{21}$$

B. Combining Constraints

When generalizing robot motions from human demonstrations there is no unique metric to define the best solution. Tuning parameters w_i and k_i offer a way to explore the range of possible solutions. To guide the intuition of the user, as mentioned in Sec. IV, these parameters have a physical meaning: weights w_i penalize deviation from a constraint while control gains k_i correspond to the inverse of the time constant of first order control systems which suppress constraint violations. When combining constraints, the robot behavior is influenced by the relation between the constraint weights. Below this is explained in detail for each of the defined constraints.

The relevant constraints for the variable impedance behavior are constraints 1 and 2. The resulting stiffness is proportional to $\boldsymbol{W}(s)$, a matrix that contains local trajectory variability (see 7), multiplied by w_2/w_1. Therefore, increasing the ratio w_2/w_1 will increase the overall impedance along the trajectory.

For constraint 3, used to constrain the target position of a generated trajectory $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2})$, the values of w_3 and k_3 are high with respect to the parameters of other constraints in such a way that the generation of trajectory $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2})$ is fast without being affected by other constraints.

The ratio w_4/w_1 is very small, in this way constraints 1 and 4 cause the robot motion to stop or even reverse as soon as a force is sensed in the opposite direction of the end effector motion.

Constraints 5 and 6 keep the point $\boldsymbol{p}(\boldsymbol{q})$ in the generated trajectory $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2})$. In this work they are not affected by other constraints. However, [27] describes interactions with these type of constraints where the robot reactively deviates from a trajectory to avoid collision with obstacles sensed using proximity sensors.

VI. Use Case Implementation

The proposed framework is tested in a use case inspired by an industrial application. A rack with five solenoids and a hub, in which the solenoids have to be assembled, are placed in a workstation. The locations and orientations of the solenoids and the corresponding hub holes vary during the day-to-day operation of the application. To automate this process, these locations and orientations are sensed by a camera system Cognex In-Sight 7400.

A. Robot Behaviors

The following behaviors are designed by combining the general-purpose constraints defined in Sec. V to fit the needs of the use case.

1) Kinesthetic Teaching: The use of constraint 1 with $\boldsymbol{u_d} = \boldsymbol{0_{3\times 1}}$ N allows the end effector to follow human motions. This behavior is used to demonstrate four paths (see Fig. 2 a) that approach vertically towards the hole in the hub (target at the end of the path in Fig. 1). Due to the configuration of the proposed use case, generated trajectories that preserve this shape will

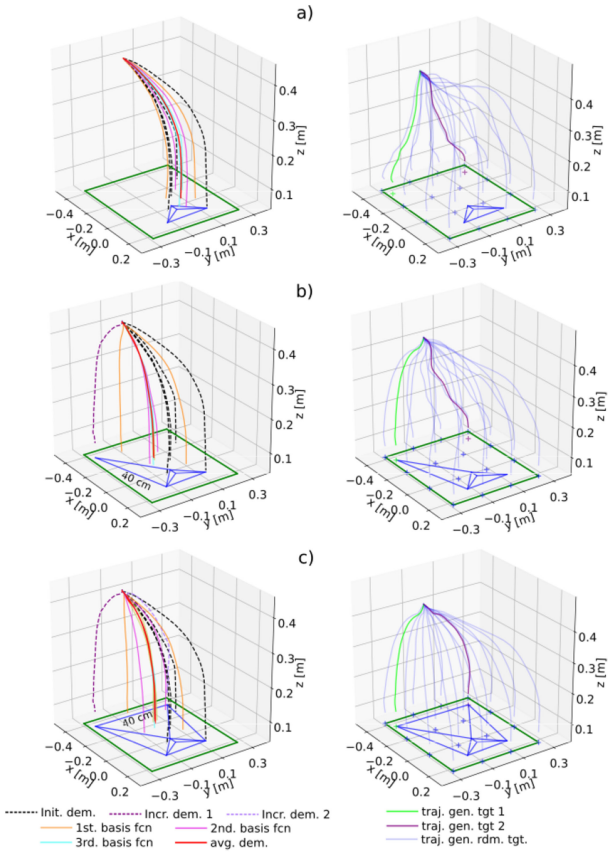

Fig. 2. Motion model refinement by the addition of incremental demonstrations. The green rectangle, the solid blue lines, and the pluses (+) represent projections in horizontal plane of the application workspace, space covered by the demonstrations end positions, and desired targets, respectively. a) corresponds to the initial motion model mod_4 trained with four initial demonstrations, b) corresponds to the first refinement mod_5 by the addition of an incremental demonstration towards target 1, and c) corresponds to the final refinement mod_6 by the addition of an incremental demonstration towards target 2. The left column contains the demonstrations performed, the average of these demonstrations $\boldsymbol{b}(s)$, and the graphical representation of the 1-sigma limits of the demonstrations encoded in the first three basis functions of $\boldsymbol{W}(s)$. The right column contains path generalization towards targets along the workspace.

avoid collisions with other solenoids in the hub (see Fig. 5). No camera information is used in this behavior.

2) Move Along Generated Trajectories: At each time instance, with a rate of $200Hz$, a trajectory is generated towards a pose obtained from the vision system (cf. constraint 3). Starting from $\boldsymbol{b}(s)$ ($\boldsymbol{\chi}_{f,2} = \mathbf{0}$, see 7), the geometric shape of the trajectory $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2})$ evolves such that the end point of the trajectory $\boldsymbol{f}(s_n, \boldsymbol{\chi}_{f,2})$ converges to the target position $\boldsymbol{c}$ according to a first-order system with time constant $k_3^{-1} = 0.125s$. This time constant implies that the distance between the end point of the trajectory and the target position has decreased with 99% after $0.5s$. Because of this short time, the end effector can start immediately moving along the instantaneous shape of the generated trajectory (both position and orientation, cf. constraints 4, 5, and 6). If a force projected in the tangential component of the path is measured, an interaction between constraints 1 and 4 allows the controller to modify $\dot{s}$. As a result, the end effector can hold or even move backwards along the trajectory (see Fig. 4 a).

3) Guided Kinesthetic Teaching (Incremental Demonstrations): Similar to kinesthetic teaching, constraint 1 with

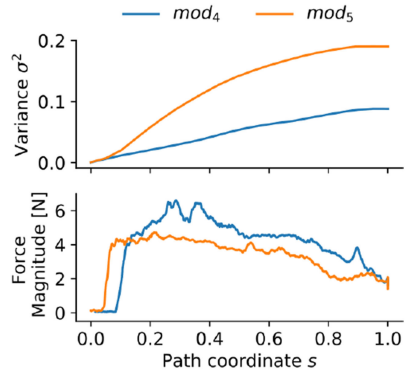

Fig. 3. The variability from the prior demonstrations and the force magnitudes measured during the addition of incremental demonstrations outside the previously demonstrated space. Higher force magnitudes are registered at the beginning of the trajectory (low variability, high impedance), while lower force magnitudes are registered at the end (high variability, low impedance).

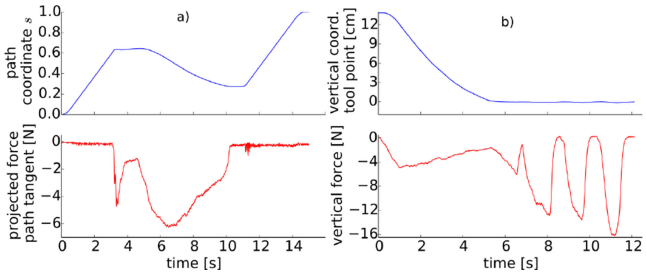

Fig. 4. Examples of robot behaviors due to the composition of constraints. a) The robot end effector advances along a trajectory with a velocity profile (path coordinate s increases); as soon as a force is sensed the end effector holds its position (s keeps its value); then the end effector motion is reversed (s decreases) by a higher force sensed; finally, the end effector resumes its nominal motion when the force is not sensed. b) The robot is driven downwards following the user forces applied by the user, while an inequality constraint avoids that the z coordinate of the tool point goes below zero.

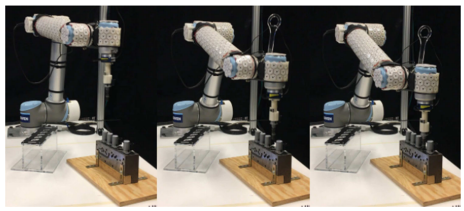

Fig. 5. Snapshots of a robot moving along a trajectory generated towards a non-demonstrated target. Information about the desired vertically approach is encoded, hence preserved, from the demonstrations.

$\boldsymbol{u_d} = \mathbf{0_{3\times 1}}$ N allows the end effector to follow human motions. This behavior is combined with: *(i)* constraint 2 that pushes the generated path $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2})$ towards the average of the demonstrations $\boldsymbol{b}(s)$ by pushing the latent variables $\chi_{f,2}$ toward zero, *(ii)* constraint 5 that attaches the generated path to the end effector, and *(iii)* constraint 4 that provides forward motion along the path. From 7 it can be seen that a deviation of the generated path $\boldsymbol{f}(s, \boldsymbol{\chi}_{f,2})$ from the average of the demonstrations $\boldsymbol{b}(s)$ is proportional to the section variability $\boldsymbol{W}(s)$ weighted by the latent variables $\boldsymbol{\chi}_{f,2}$. I.e., deviations in sections where the variability is small require large values for the latent variables causing a bigger violation of constraint 2. This results in

TABLE I
CONSTRAINT PARAMETERS

Constraint	Parameter	Value	Unit
1: Force control	$diag(\boldsymbol{C_a})$	$\mathbf{3.5_{3\times1}}$	mm/N
	$\boldsymbol{k}_1$	$\mathbf{3_{3\times1}}$	s^{-1}
	$\boldsymbol{w}_1$	$\mathbf{[0,1]_{3\times1}}$	
2: Latent variables	$\boldsymbol{k}_2$	$\mathbf{1_{5\times1}}$	s^{-1}
	$\boldsymbol{w}_2$	$\mathbf{0.5_{5\times1}}$	
3: Trajectory Generation	$\boldsymbol{k}_3$	$\mathbf{8_{3\times1}}$	s^{-1}
	$\boldsymbol{w}_3$	$\mathbf{50_{3\times1}}$	
4: Velocity along path	v_{max} for v_f	0.22	m/s
	a_{max} for v_f	0.5	m/s^2
	k_4	0	s^{-1}
	w_4	0.01	
5: Remain on path	$\boldsymbol{k}_5$	$\mathbf{0.6_{3\times1}}$	s^{-1}
	$\boldsymbol{w}_5$	$\mathbf{1.5_{3\times1}}$	
6: Tool orientation	$\boldsymbol{k}_6$	$\mathbf{4_{3\times1}}$	s^{-1}
	$\boldsymbol{w}_6$	$\mathbf{1_{3\times1}}$	

a variable impedance behavior. No camera information is used in this behavior.

4) Insertion: The use of constraint 1 with $\boldsymbol{u_d}$ equal to $[0\,0\quad -100]^T$ N allows the robot to insert the solenoids by exerting 100 N downwards, while being compliant in the other directions. As a result, the system can deal with uncertainties in the pose estimated by the vision system. No camera information is used in this behavior.

B. Selection of Constraint Parameters

The constraint parameters in table I are specified during the modeling phase according to the guidelines explained in V-B.

To avoid drift and obtain smooth activation/deactivation of constraint 1, similarly to [27], two conditions that modulate weights $\boldsymbol{w_1}$ are implemented. First, if the force magnitude exceeds 0.8 N, the corresponding weight rises at $4\,s^{-1}$ rate until it reaches 1. Second, if the force magnitude falls below 0.8 N, the corresponding weight decreases at $-4\,s^{-1}$ rate until it reaches 0, i.e. deactivating the constraint.

C. Discussion

In this study, a tight integration between an imitation learning algorithm with a constraint-based reactive control framework was developed and evaluated in a collaborative use case inspired by an industrial application. To the authors' knowledge such integration is novel. The strongest benefits of this synergy as mentioned in Sec. I are discussed bellow:

Fig. 2 illustrates the trajectory generalization capability towards non-demonstrated targets done by an initial learned model mod_4 and its two consecutive refinements mod_5 and mod_6. To quantify the quality of the trajectories approaching vertically towards non-demonstrated targets, a volume with a cylinder shape ($h = 0.06m$ and $r = 0.02m$) is defined on top of each target hole to simulate the available free space between two inserted solenoids. A generated trajectory is considered successful only if it intersects this cylinder at its top surface. The evaluation of the 18 generated trajectories in Fig. 2 is presented in table II. These results reflect the improvement of the initial motion model after each refinement. As for the insertions, which were not learned but specified using constraints, it has been corroborated in separate experiments that every time the end effector reached a target hole provided by the camera the subsequent insertion was successful.

TABLE II
SUCCESS IN THE TRAJECTORY GENERATION

Motion model	mod_4	mod_5	mod_6
Successful trajectories	$10/18$	$15/18$	$18/18$

The guided kinesthetic teaching behavior assists an operator when performing additional demonstrations by generating a variable impedance in function of the variability of prior demonstrations along the trajectory. It can be seen in Fig. 3 that the user felt a larger force, due to a larger impedance, at the starting section, in which corresponding samples of the demonstrations are closer to each other, i.e. with low variability. In contrast, the user felt a lower force, due to a lower impedance, at the end of the demonstrations, in which corresponding samples are more widely spread. We believe that this behavior can help the operator to expedite the refinement of the learned model by giving stronger feedback in sections in which previous demonstrations are close to each other.

The proposed synergy between imitation learning and constraint-based task specification allows the combination of robot behaviors needed during both the learning and execution phases with more general task constraints, such as joint limits, velocity limits, and workspace limits. Fig. 4 presents two examples of constraint combination. In Fig. 4 a, while moving along the trajectory, the operator perturbs the motion of the robot by holding and then pushing the end effector backwards along the trajectory. In Fig. 4 b, while executing the kinesthetic teaching behavior, an inequality constraint prevents the operator to drive the end effector below a given threshold in the vertical position z.

The different types of behaviors discussed above show that the proposed approach facilitates the specification of a variety of tasks by combining the general-purpose constraints in different ways. For example in [27], which is not focusing on imitation learning, constraints 4 and 5 are combined with an additional constraint involving proximity signals sensed by an artificial skin mounted on the robot. The goal of [27] is to implement a human-robot collaborative strategy in which the robot end effector moves along a trajectory while avoiding collisions with humans.

These advantages allows us to deploy the presented application in less than 10 minutes (see attached video). Hence, we believe that the proposed framework may expedite the deployment of several robotic tasks while interacting with different sensor inputs. Future work will study the composability of these behaviors to address the deployment of different types of robotic applications, for example, bin picking.

VII. CONCLUSION

This letter focuses on expediting the deployment of robot applications with complex sensor-based interactions using

imitation learning. A new methodology is proposed to combine constraint-based reactive task specification and control with imitation learning.

The chosen learning algorithm can be tightly integrated in the constraint-based controller of eTaSL. This combination offers several advantages: the framework is able to generalize trajectories that preserve shape information outside, but in the neighborhood of, the space covered by a small number of demonstrations. The generated motion can be reactively adapted to sensor inputs while still being as close as possible to the demonstrated motions; in this work these motions are adapted to a target end position given by a vision sensor. The motion models extracted from the demonstrations can be further refined by incrementally adding new demonstrations. Incremental demonstrations are facilitated by a variable impedance behavior in function of a path coordinate along the trajectory. Regions with low variability in the demonstrations require more force to deviate from the mean trajectory than regions with high variability. This makes incremental demonstrations more intuitive.

The proposed method has some limitations. Our motion model is only sufficient for a uni-modal distribution, assuming convexity and homogeneity in the trajectory distribution. This limits application of the learning algorithm when used on its own to handle cases where there is a non-homogeneous spatial deformation in local sections of the trajectory (e.g. when an obstacle is circumvented by different sides). However, the combination of the learning algorithm with the constraint-based methodology provides a potential solution. For example, in [27] a constraint-based reactive strategy is proposed in which the robot can circumvent a dynamic obstacle by deviating from the generated trajectory. Another limitation is that the addition of new demonstrations is done through a supervised learning procedure subject to the user judgment. In this procedure, sensitivity to the inclusion of new demonstrations and to the computational cost is not assessed. Further research is intended to address both composition of obstacle avoidance behavior and incremental learning aspects.

As described in Sec. V and VI, there are many parameters in the constraint-based specification. Fortunately, these parameters are intuitive. The control constants k_i are easily determined and the maximally allowable k_i typically depend on the robot capabilities. For the weights, all the constraints have to be considered together. The parameters are completely determined during the modeling phase and remain fixed during the deployment of the robotic application.

Our approach contributes towards generically applicable 'robot apps' in two ways: the model-based, constraint-based programming approach is very modular and makes the application more robust to changes in the environment, and secondly, the use of imitation learning expedites the application deployment.

REFERENCES

[1] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends Cogn. Sci.*, vol. 3, no. 6, pp. 233–242, 1999.

[2] A. G. Billard, S. Calinon, and R. Dillman, "Learning from humans," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., Cham, Switzerland: Springer, 2016, pp. 1995–2014.

[3] J. De Schutter *et al.*, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *Int. J. Robot. Res.*, vol. 26, no. 5, pp. 433–455, 2007.

[4] M. E. Tipping, and C. M. Bishop, "Probabilistic principal component analysis," *J. R. Statist. Soc. B*, vol. 61, no. 3, pp. 611–622, 1999.

[5] E. Aertbeliën and J. De Schutter, "eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs," in *Proc. IEEE RSJ Int. Conf. Int. Robots Syst.*, Chicago, IL, USA, 2014, pp. 1540–1546.

[6] E. Aertbeliën, "The eTaSL task specification language," 2016. Accessed: 2017. [Online]. Available: https://people.mech.kuleuven.be/eaertbel/etasl

[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 *Proc. Int. Conf. Adv. Robot.*, Kobe, Japan, 2009, pp. 1–6.

[8] G. Borghesan and J. De Schutter, "Constraint-based specification of hybrid position-impedance-force tasks," in *Proc. IEEE Int. Conf. Robot. Autom.*, Hong Kong, 2014, pp. 2290–2296.

[9] P. Roduit, A. Martinoli, and J. Jacot, "A quantitative method for comparing trajectories of mobile robots using point distribution models," in *Proc. IEEE/RSJ Int. Conf. Int. Robots Syst.*, 2007, pp. 2441–2448.

[10] E. Aertbeliën and J. De Schutter, "Learning a predictive model of human gait for the control of a lower-limb exoskeleton," in *Proc. IEEE RAS/EMBS Int. Conf. Biomed. Robot. Biomechatronics*, 2014, pp. 520–525.

[11] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Trans. Syst., Man, Cybern. B Cybern.*, vol. 37, no. 2, pp. 286–298, Apr. 2007.

[12] D. Forte, A. Ude, and A. Gams, "Real-time generalization and integration of different movement primitives," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2011, pp. 590–595.

[13] F. Abi-Farraj, T. Osa, N. P. J. Peters, G. Neumann, and P. R. Giordano, "A learning-based shared control architecture for interactive task execution," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 329–335.

[14] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems*, vol. 26, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2013, pp. 2616–2624.

[15] M. Ewerton, G. Maeda, G. Kollegger, J. Wiemeyer, and J. Peters, "Incremental imitation learning of context-dependent motor skills," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2016, pp. 351–358.

[16] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intell. Service Robot.*, vol. 9, no. 1, pp. 1–29, 2016.

[17] A. Pervez and D. Lee, "Learning task-parameterized dynamic movement primitives using mixture of GMMs," *Intell. Service Robot.*, vol. 11, no. 1, pp. 61–78, 2018.

[18] S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot," in *Proc. ACM/IEEE Int. Conf. Human–Robot Interact.*, 2007, pp. 255–262.

[19] S. Calinon and A. Billard, "Active teaching in robot programming by demonstration," in *Proc. IEEE Int. Workshop Robot Human Interact. Commun.*, 2007, pp. 702–707.

[20] D. Lee and C. Ott, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Auton. Robots*, vol. 31, pp. 115–131, 2011.

[21] J. De Schutter, "Invariant description of rigid body motion trajectories," *Trans. ASME J. Mech. Robot.*, vol. 2, no. 1, 2010, Art. no. 011004.

[22] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, 2007.

[23] L. Gupta, D. L. Molfese, R. Tammana, and P. G. Simos, "Nonlinear alignment and averaging for estimating the evoked potential," *IEEE Trans. Biomed. Eng.*, vol. 43, no. 4, pp. 348–356, Apr. 1996.

[24] L. B. Rall, *Automatic Differentiation*. Berlin, Germany: Springer, 1981.

[25] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Math. Program. Comput.*, vol. 6, no. 4, pp. 327–363, 2014.

[26] H. Bruyninckx and J. De Schutter, "Specification of force-controlled actions in the "Task frame formalism": A synthesis," *IEEE Trans. Robot. Autom.*, vol. 12, no. 5, pp. 581–589, Aug. 1996.

[27] C. Vergara, G. Borghesan, E. Aertbeliën, and J. De Schutter, "Incorporating artificial skin signals in the constraint-based reactive control of human-robot collaborative manipulation tasks," in *Proc. IEEE Int. Conf. Adv. Robot. Mechatronics*, 2018, pp. 280–287.