Intuitive Programming of Conditional Tasks by Demonstration of Multiple Solutions

Thomas Eiband[1], Matteo Saveriano[1], and Dongheui Lee[1,2]

***Abstract*— Conditional tasks include a decision on how the robot should react to an observation. This requires to select the appropriate action during execution. For instance, spatial sorting of objects may require different goal positions based on the objects properties, such as weight or geometry. We propose a framework that allows a user to demonstrate conditional tasks including recovery behaviors for expected situations. In our framework, human demonstrations define the required actions for task completion, which we term solutions. Each specific solution accounts for different conditions which may arise during execution. We exploit a clustering scheme to assign multiple demonstrations to a specific solution, which is then encoded in a probabilistic model. At runtime, our approach monitors the execution of the current solution using measured robot pose, external wrench, and grasp status. Deviations from the expected state are then classified as anomalies. This triggers the execution of an alternative solution, appropriately selected from the pool of demonstrated actions. Experiments on a real robot show the capability of the proposed approach to detect anomalies online and switch to an appropriate solution that fulfills the task.**

I. Introduction

The execution of a conditional task is affected by changes in the executive context. Therefore, a conditional task has several possible outcomes depending on the conditions of the specific execution. For instance, one can consider sorting of objects by their weight, where light and heavy objects should be placed on different destinations (Fig. 1). Hence, this sorting task has two possible actions and requires two specialized behaviors to be accomplished. We call such behaviors *solutions* and state that multiple solutions (solution pool) have to be assigned to a conditional task. Furthermore, the sorting example requires physical interaction with the objects to estimate the weight, which is a property that cannot be observed visually before the task execution. It is clear that the correct execution of a conditional task requires a continuous monitoring of the executive state and the capability of detecting anomalies, i.e. deviations between the expected state of the current (nominal) solution and the measured state. Such anomalies can trigger the execution of an alternative solution that fulfills the task. Finding an appropriate solution requires a decision on which is the best alternative solution in the solution pool (Fig. 1 right).

Our goal is to develop a framework that allows a user to intuitively program a conditional task. Learning from Demonstrations (LfD) has been applied in a variety of scenarios, including the motion and force domain. Many works consider multiple demonstrations of a task in order to achieve good generalization results and a robust execution, for example task parametrization approaches [1], [2]. These approaches require the selection of a task parameter and do not generalize well if, as for conditional tasks, the required action is too different from what has been demonstrated. Other approaches consider that multiple demonstrations can include information about specific strategies, which need to be treated independently [3], [4].

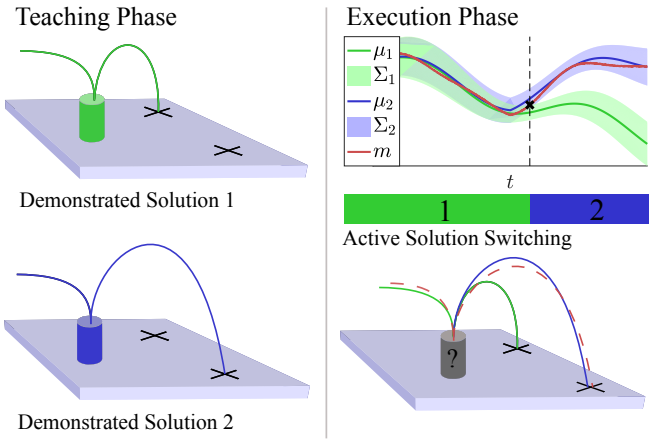

Fig. 1. Teaching multiple solutions for different conditions in a task, e.g. sorting by object weight. The robot executes a nominal solution (1) and monitors measured state m and commanded state μ_1. The scheduler selects an alternative solution (2) with commanded state μ_2, when the confidence bound Σ_1 of the current solution is violated.

It may be hard for a novice user to select either task parameters or to program a higher level description of a task involving decisions or recovery behaviors. Therefore, we propose an approach to allow a user to program a conditional task only by demonstration. In our framework, the user demonstrates several solutions corresponding to different executive conditions that influence the robots behavior. A clustering algorithm is employed to assign multiple demonstrations to a specific solution in an unsupervised fashion, i.e. without explicitly labeling them. The task execution is continuously monitored to detect deviations from the expected state and eventually switch to an alternative solution.

To summarize, the intuitive programming of conditional tasks and recovery behaviors is obtained by

- learning a variety of task solutions without labeled data or a symbolic task representation by clustering of

[1] German Aerospace Center (DLR), Institute of Robotics and Mechatronics, Wessling, Germany
[2] Technical University of Munich, Chair of Human-centered Assistive Robotics, Munich, Germany
{thomas.eiband, matteo.saveriano, dongheui.lee}@dlr.de

motion and force data,
- online switching of task solutions by anomaly detection including forces and grasp status, and
- recovering from an error by choosing the most likely state within the set of alternative task solutions.

We are exploiting the robot's proprioceptive sensing without incorporating an additional vision system, which requires a partially structured environment with fixed object positions. We emphasize that the task is bootstrapped from the demonstrations only, without requesting further knowledge of how it shall be executed and how to react to changes in the environment.

The document is outlined as follows. We first discuss related work (Sec. II), introduce our scheme to learn multiple solutions (Sec. III) and explain the task execution and monitoring system (Sec. IV). We evaluate our approach in the experiments section (Sec. V) and give a short conclusion and outlook about future work (Sec. VI).

II. Related Work

A. Learning Conditional Tasks from Demonstration

Conditional tasks require that the robot adapts to the state of the environment and plans accordingly. Complex strategies can be designed by hand in the form of a finite state machine (FSM) [5]. Alternatively, plans can be directly learned from demonstrations [6], where a knowledge base of manipulation tasks is built from demonstrations by observation of a virtual workspace and by segmentation into predefined skills with pre- and post-conditions. In [4], these plans are interactively built in form of a FSM, given visually tracked object poses. Recovery behaviors are considered for grasp and object pose failures, but unknown anomalies are not detected by the system itself and the user has to intervene if a new behavior shall be added. Our monitoring system observes continuously the process parameters, including expected forces. This allows also fault detection, whether a recovery behavior is available or not.

Although there exist solvers that can handle continuous state spaces and do not require predefined transition probabilities, such as POMCPs [7], they require a black-box model to sample from and the reward (or goal states) need to be defined manually. In [8], a switching scheme between deterministic planning and decision-theoretic planning in the form of POMDP is proposed, but it requires a problem description in the Planning Domain Definition Language.

Common planners, e.g. based on a probabilistic roadmap (PRM) or on a rapidly exploring random tree (RRT) [9], always require a model, which has to be built beforehand. Instead, this is not needed with the presented approach. In [10], multiple demonstrations are encoded in sequences of predefined symbols to find the longest common subsequence. Symbolic actions for planning are extracted from demonstrations in [11] and [12]. In both works, preconditions and effects of symbolic actions are learned from demonstration but the corresponding symbols need to be manually defined. In [13], complex tasks are learned by kinesthetic teaching involving multiple visually tracked objects, where the task structure itself has to be predefined beforehand.

B. Error Detection and Recovery

Our approach can be applied to error detection, where a recovery behavior needs to be provided in advance for expected faults. Faults can be divided into internal robotic system faults and task execution faults [14]. We address task execution faults, where anomalies in robot pose, external wrench or grasp status can be detected. Geometric assembly errors were theoretically described in [15], and possible recovery strategies provided. The authors state that an event is termed an error if no solution exists to handle it. Further, they term an action a recovery strategy, if it is a possible solution to the given problem. In general, an error can be seen as a simple event in a guaranteed plan, if the system can detect it and plan accordingly.

Several approaches in the literature exploit time series of previously observed executions for error detection. In [16], Hidden Markov Models (HMM) were trained as a process monitor in robotic assembly. In [17], a HMM is trained on the wrench and its derivative to monitor force-based interactions with the environment and to classify if the execution is successful or not. As a drawback, the specific modality, which allows for detecting the deviation, needs to be selected beforehand and an expert user need to observe enough executions to manually label them. Finally, the execution strategy is pre-programmed as a finite state machine (FSM). In [18], Mahalanobis distance is used on a subset of sensor time series of unmanned vehicles, but recovery from errors was not scope of this work. Task stratification [19] orders possible faults into classes such as execution/planning/modeling and sensing errors and describes a forward and backward correction process to successfully accomplish an error-prone task. While providing labels for error classes, a strategy how to identify an error given a certain state is not provided. The task outcome prediction in [20] compares sensor signals with successful trials coming from a reinforcement learning system. A z-test predicts, if the observed signals stem from a population of successful trials, which requires cumbersome labeling by hand.

Learning from failed trials has been presented [21] to avoid the human's mistakes and to converge faster to an optimal policy. The basic assumption thereof was that every shown trial has failed and the robot tries to find a better strategy in between these failures. On the other hand, our approach focuses only on successful trials to bootstrap the desired behavior.

C. Force Events in Robotic Manipulation

In assembly tasks, force signatures (i.e. force or torque time-series) have been exploited in [22] to train a support vector machine (SVM) with a bunch of successful and unsuccessful assemblies from hand-labeled trials. The approach does not handle fault states by a recovery strategy. Instead of thresholds, force and torque transients were used in an assembly task to accelerate the event detection in order to

move to the next state [23]. Events were inferred by a SVM, trained on 60 samples of assembly. The state machine and control schemes were predefined. Hand-designed detection schemes for recognition of fault states can also detect abstract error signatures, such as haptically detecting if a hand driller is running or not by analyzing the frequency spectrum of the "feeling" force sensor [5]. However, such abstract detection schemes require great effort in manually designing them and relating them to the right state during execution is not straight-forward. For each task, specific failure detection and recovery behaviors were defined by hand and added to a state machine. Therefore, unexpected situations which were not accounted for in the design cannot be handled. In contrast, we do not need to relate a specific recovery behavior to a fault state during execution but let the anomaly detection identify the fault state and its appropriate solution, if existing.

D. Clustering of Time Series

We rely on time series clustering, which not only aligns temporally distorted series but also uses a time-series prototype where other series are warped to. Our problem involves multidimensional series with unequal length due to demonstration variations. The observed input data consists of continuous variables from multiple modalities such as position, orientation, wrench and grasp status. A survey of time series clustering [24] groups the methods into raw-data-, feature- and model-based approaches. Methods which act in the raw-data domain with multidimensional input and variable length often employ Euclidean distance as the general distance metric. An approach which directly uses dynamic time warping (DTW) distance for clustering is presented in [25]. The authors not only warp multiple time series but also provide a technique to find the average time series representing the cluster center, which they term as the prototype.

As proposed in [3], the number of clusters depends on a distance parameter, which has to be specified by the system designer but leads to an unpredictable number of clusters depending on the scenario. Instead, we just let the user define the number of clusters to make sure that the correct number of solutions is learned. Additionally, we consider temporal variations by DTW and additional modalities, such as force.

III. Learning Multiple Solutions

We assume that our task is not a fixed temporal sequence of skills or actions, but a variable execution strategy considering changes in the environment. We call a successful task execution a solution, whereby several of these are included in a solution pool (SP). Therefore, it does not matter if an execution which deviates from the nominal one, is seen as a recovery behavior or as an alternative solution to a problem. In the same way, the event which leads to the adaptation of execution can be seen as an error or just as the expected environmental state deviating from the nominal execution, which we can handle by an alternative solution.

We think that alternative solutions can be executed by switching from the initial nominal solution to them at the state where the environmental condition is met. This can be compared with a state machine where the transition condition is observed continuously. We switch to another state, in our case the alternative solution, whenever the condition is fulfilled.

In theory, all observable variables during execution can be taken into consideration for error detection. However, we apply our method to a vision-free approach using the following modalities: a) robot pose, which can be simply obtained by the robot measurements, b) wrench, which is obtained by a force-torque (FT) sensor, and c) gripper finger distance in accordance with a discrete grasp status (-1: no object in gripper, 0: gripper moving, 1: object in gripper) provided by the gripper interface.

A. Demonstration System

The user provides multiple demonstrations for varying conditions, to which the robot shall account for. The teaching system shown in [26] is used to transfer knowledge to the robot, where a FT sensor is mounted between robot and gripper. We use three foot pedals to trigger the start and stop of teaching, gripper movements and to switch to the next demonstration. A sample at time t of demonstration i is given by $\boldsymbol{x}_t^i = [\boldsymbol{p}, \boldsymbol{o}, \boldsymbol{w}, g, h]^T \in \mathbb{R}^{15}$, representing Cartesian end effector position $\boldsymbol{p} = [x, y, z] \in \mathbb{R}^3$, orientation in unit quaternions $\boldsymbol{o} = [q_w, q_x, q_y, q_z] \in \mathbb{R}^4$, wrench $\boldsymbol{w} = [f_\mathrm{x}, f_\mathrm{y}, f_\mathrm{z}, t_\mathrm{x}, t_\mathrm{y}, t_\mathrm{z}] \in \mathbb{R}^6$, gripper finger distance $g \in \mathbb{R}$ and grasp status $h \in \{-1, 0, 1\}$. The wrench values are filtered by a 1st order Butterworth low-pass filter with cutoff frequency of 1 Hz. The trajectories of demonstration $i \in \{1, \ldots, I\}$ with according sample length N_i for demonstration i are stored in a matrix $\boldsymbol{X}_i = [\boldsymbol{x}_1^i, ..., \boldsymbol{x}_{N_i}^i]$. Steady states where there is nearly no change in position and orientation are removed.

B. Clustering

Clustering is applied on all demonstrations to find a set of solutions. In the pre-processing, all demonstrations are dimension-wise standardized by subtracting the mean and dividing by the standard deviation (z-transform). This allows to use Euclidean distance metric over all dimensions for DTW. In order not to favor signal modalities with rather high amplitudes (e.g. force signals), standardization makes sure that all dimensions contribute equally to the warping error. For a common standardization, all demonstrations are stacked in $\bar{\boldsymbol{X}} = [\boldsymbol{X}_1 \ldots \boldsymbol{X}_I]$. The mean and standard deviation are computed row-wise over $\bar{\boldsymbol{X}}$ to obtain standardized demonstrations from $\hat{\boldsymbol{X}}_1$ to $\hat{\boldsymbol{X}}_I$. A pairwise distance matrix between the demonstrations is obtained by the Dynamic Time Warping (DTW) distance. We denote this distance as $\mathrm{DTW}(\boldsymbol{A}, \boldsymbol{B})$, for some multi-dimensional time series $\boldsymbol{A}$ and $\boldsymbol{B}$. The distance matrix over all demonstrations is given by

$$\boldsymbol{D}_\mathrm{DTW} = \begin{bmatrix} \mathrm{DTW}(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_1) \ldots \mathrm{DTW}(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_I) \\ \ddots \\ \mathrm{DTW}(\hat{\boldsymbol{X}}_I, \hat{\boldsymbol{X}}_1) \ldots \mathrm{DTW}(\hat{\boldsymbol{X}}_I, \hat{\boldsymbol{X}}_I) \end{bmatrix}.$$

We use single linkage hierarchical clustering [27], which uses $\boldsymbol{D}_{\text{DTW}}$ as distance matrix. At the current state, the user needs to specify the number of clusters, which is simply the number of demonstrated solutions S. In comparison, density based clustering algorithms require a density parameter instead of a desired cluster number, which is again task specific as multiple demonstrations might have variable similarity depending on user performance and task goals. In our preliminary studies, such clustering approaches did not lead to reliable results.

A number of S clusters is obtained by flattening the hierarchical cluster structure. Hereby, a minimum threshold is computed on the cophenetic distance between two observations in the same cluster such that no more than S flat clusters are created.

The cluster medoid is found by the minimum sum of squared distances to all other demos within the same cluster. DTW is applied again between the medoid and all other demos in the same cluster. The resulting warping path realigns all demos with the medoid. Subsequently, the warped demos are resampled to share the same length (N_s) within one cluster. The warped demonstrations $\tilde{\boldsymbol{X}}_s^i$ related to a cluster corresponding to solution s, are stacked into a common matrix $\hat{\boldsymbol{C}}_s$.

C. Trajectory Learning

The clustered data is converted into a generalized trajectory for each cluster s. Remember that the data has been standardized for clustering and warping, whereas we use the original data in the trajectory learning by row-wise multiplication with the standard deviation and addition of the mean, resulting in a non-standardized cluster $\boldsymbol{C}_s$. For each solution s, a time-based Gaussian Mixture Model (GMM) is used to generalize multiple demos into a common model $\mathcal{M}$. The input matrix

$$\boldsymbol{G}_s = \begin{bmatrix} \boldsymbol{C}_s \\ \boldsymbol{u}, \ldots, \boldsymbol{u} \end{bmatrix} \in \mathbb{R}^{16 \times N_s I_s} \tag{1}$$

is used to learn a joint probabilistic model of all input dimensions, denoted as

$$\mathcal{M} = \text{GMM}(\boldsymbol{G}_s), \tag{2}$$

with a time vector $\boldsymbol{u} = [1, \ldots, N_s]$. Gaussian Mixture Regression (GMR) provides a trajectory $\boldsymbol{Y}_s$ and a time-series of covariance matrices $\boldsymbol{Z}_s$ by conditioning the model on the time vector $\boldsymbol{u}$ with GMR($\mathcal{M}|\boldsymbol{u}$). The trajectory of each solution s is stored in $\boldsymbol{Y}_s = [\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{N_s}]$ and the according covariance time-series in $\boldsymbol{Z}_s = [\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_{N_s}]$. The symmetric covariance matrix at time-step t is represented by

$$\boldsymbol{\Sigma}_t = \begin{bmatrix} \boldsymbol{\Sigma}_{p,p} & \boldsymbol{\Sigma}_{p,o} & \boldsymbol{\Sigma}_{p,w} & \boldsymbol{\Sigma}_{p,g} & \boldsymbol{\Sigma}_{p,h} \\ \cdots & \boldsymbol{\Sigma}_{o,o} & \boldsymbol{\Sigma}_{o,w} & \boldsymbol{\Sigma}_{o,g} & \boldsymbol{\Sigma}_{o,h} \\ \cdots & \cdots & \boldsymbol{\Sigma}_{w,w} & \boldsymbol{\Sigma}_{w,g} & \boldsymbol{\Sigma}_{w,h} \\ \cdots & \cdots & \cdots & \boldsymbol{\Sigma}_{g,g} & \boldsymbol{\Sigma}_{g,h} \\ \cdots & \cdots & \cdots & \cdots & \boldsymbol{\Sigma}_{h,h} \end{bmatrix}. \tag{3}$$

The quaternions of the orientation trajectory are normalized at this point. This is due to the regression and DTW, which both act in Euclidean vector space and do not preserve the quaternion properties. The deviation we faced is small enough to promote the usage of such compact orientation encoding alongside other modalities. Having the trajectories for pose, wrench, gripper distance and grasp status, as well as the covariance matrix at each time-step, the required data has been computed for executing the task on the robot. This data is generated for each solution and added to a solution pool (SP).

IV. Execution and Monitoring of Conditional Tasks

A. Scheduler

The *Scheduler* (Fig. 2) is a decision module, which manages a solution pool (SP) and takes care of events that occur during task execution (Algorithm 1). Possible events are anomalies or the finishing of a task. The initial nominal solution for a task can be randomly preselected from the pool or simply by user choice. We decided to select the nominal solution with shortest length in the number of samples in order to favor short execution times and less complex behaviors. Whenever a solution is selected, its trajectory is passed to the *Execution*.

Algorithm 1 Scheduler

Input: solution pool: SP; anomaly threshold: ϵ

Initialization :
$t \leftarrow 1$ ▷ set trajectory starting index
$s \leftarrow$ get_nominal_solution(SP) ▷ (section IV-A)
while not_empty(SP) **do**
  goto_start_point(s, t)
  execute_solution(s)
  remove solution s from pool SP
  event $\leftarrow$ wait_for_event(ϵ) ▷ anomaly or finished task
  if event **is** anomaly **then**
    if not_empty(SP) **then**
      given: measured erroneous state: $\boldsymbol{m}_e$
      $s, t \leftarrow$ find_alternative_solution$(\boldsymbol{m}_e, s)$ ▷ (section IV-D)
    else
      return stop_on_error ▷ no more solutions available
    end if
  else
    return finished ▷ successful completion
  end if
end while

B. Execution

The *Execution* module employs a Cartesian Impedance controller with additional wrench term, leading to the joint torque

$$\boldsymbol{\tau}_{\text{cmd}} = \boldsymbol{J}^T(\boldsymbol{K}(\boldsymbol{x}_{\text{d}} - \boldsymbol{x}) + \boldsymbol{w}_{\text{d}} - \boldsymbol{D}\dot{\boldsymbol{x}}) + \boldsymbol{g}(\boldsymbol{q}), \tag{4}$$

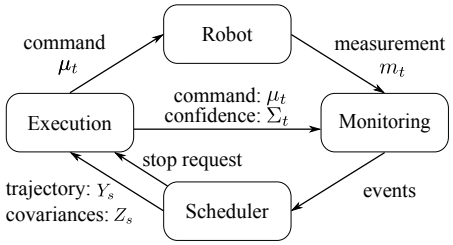

Fig. 2. Modules and data flow of the execution and observation system.

where $\boldsymbol{J}$ is the Jacobian, $\boldsymbol{K}$ the Cartesian stiffness matrix, $\boldsymbol{x}_\mathrm{d}$ and $\boldsymbol{x}$ the desired and measured position, $\boldsymbol{w}_\mathrm{d}$ the desired wrench, $\boldsymbol{D}$ a positive definite Damping matrix and $\boldsymbol{g}$ represents the robot's gravity term, depending on the joint position $\boldsymbol{q}$. We added the desired wrench term in order to reproduce the demonstrated dynamics, which makes sure that an external wrench which has been observed in the demonstrations does not have major influence on the pose of the robot. The gripper fingers are commanded by a simple feed-forward controller.

C. Monitoring

The *Monitoring* module observes both nominal (commanded) and measured state during execution. Hereby, anomalies are detected by a one-class classification [28], where we model a probability distribution and employ an error metric to identify outliers. Whenever an anomaly occurs during execution, an event and relevant process data is sent to the *Scheduler*.

The current measurement during execution is $\boldsymbol{m}_t = [\boldsymbol{p}, \boldsymbol{o}, \boldsymbol{w}, g, h]^T \in \mathbb{R}^{15}$ and the commanded state of the nominal solution is $\boldsymbol{\mu}_t = [\boldsymbol{\mu}_\mathrm{p}, \boldsymbol{\mu}_\mathrm{o}, \boldsymbol{\mu}_w, \mu_g, \mu_h]^T \in \mathbb{R}^{15}$. At each time-step t, the deviation between nominal execution and measured state is computed by the Mahalanobis distance

$$D_\mathrm{M} = \sqrt{(\boldsymbol{m}_t - \boldsymbol{\mu}_t)^T \boldsymbol{\Sigma}_t^{-1} (\boldsymbol{m}_t - \boldsymbol{\mu}_t)} \tag{5}$$

which is compared with a threshold ϵ. This means that a large enough deviation signalizes the failure of the nominal solution. If the threshold is exceed for a few consecutive timesteps, an anomaly event is triggered. Hereby, all modalities contribute equally to the anomaly detection as their errors are scaled by the covariance matrix. Errors can be introduced by deviation in position or orientation, by an abnormal wrench due to unexpected interaction forces or object weights, by gripper finger distance or grasp status due to object geometry or misplaced objects in the environment. If an anomaly is detected at time-step t_e, the measured state $\boldsymbol{m}_\mathrm{e}$ is forwarded to the *Scheduler*.

D. Finding an Alternative Solution

The task of the *Scheduler* is to select a nominal solution in the beginning and an appropriate alternative solution in the case an anomaly is detected. If only two solutions exist and one is currently executed, only one solution remains left as a recovery strategy. Given more than two solutions, we want to identify the best strategy to cope with the situation (see Algorithm 1). Based on the current (erroneous) state $\boldsymbol{m}_\mathrm{e}$ at time t_e and current nominal solution η, the most appropriate alternative solution s^* can be found in the pool. Let $\boldsymbol{Y}_s(t)$ be the sample in the trajectory of solution s at time t and $\boldsymbol{Z}_s(t)$ the respective covariance matrix. Then, the minimum squared Mahalanobis distance over each sample in solution s is given by

$$C_s = \min_{t \in [1, N_s]} \{(\boldsymbol{Y}_s(t) - \boldsymbol{m}_\mathrm{e})^T \boldsymbol{Z}_s(t)^{-1} (\boldsymbol{Y}_s(t) - \boldsymbol{m}_\mathrm{e})\}. \tag{6}$$

The solution with the closest state to erroneous state is found by

$$s^* = \underset{s \in \mathrm{SP} \setminus \eta}{\operatorname{argmin}} \{C_s\}, \tag{7}$$

excluding the currently executed nominal solution η. The identified alternative solution trajectory is stated as $\boldsymbol{Y}_{s^*}$.

Inspired by human behavior, the recovery strategy shall be executed ad-hoc, right after the error is identified. Additionally, the task shall be continued to resolve only the error but not by restarting the whole execution sequence. Therefore, the time-step, in which the recovery strategy shall be started, is identified similarly to (6) with

$$t^* = \underset{t \in [1, N_s^*]}{\operatorname{argmin}} \{(\boldsymbol{Y}_{s^*}(t) - \boldsymbol{m}_\mathrm{e})^T \boldsymbol{Z}_s(t)^{-1} (\boldsymbol{Y}_{s^*}(t) - \boldsymbol{m}_\mathrm{e})\}. \tag{8}$$

The trimmed solution trajectory, where the *Scheduler* switches to at runtime, starts at index t^* and is denoted by $\boldsymbol{Y}_{s^*}(t^*, ..., N_{s^*})$.

V. Experiments

A. Evaluation of Monitoring System

In a baseline experiment, we evaluate that a system without monitoring capabilities cannot detect anomalies from an expected behavior, and show that the proposed monitoring system is able to do so. The anomaly threshold is set to $\epsilon = 6$ throughout the next experiments. In a simple peg-in-hole setup (Fig. 3), the nominal solution is to insert the peg into a hole such that no high external forces or blocking occurs during insertion as demonstrated by the user. Four demonstrations of picking the peg and inserting it in the hole were given. One solution has been learned from a single cluster of demonstrations. We use a DLR Light Weight Robot (LWR IV) [29] mounted on a linear axis, equipped with a 2-finger Robotiq 85 gripper. For analysis, we show how errors from different modalities contribute to the overall error D_M. Therefore, we define $D_\mathrm{p} = \sqrt{(\boldsymbol{p}_t - \boldsymbol{\mu}_{\mathrm{p},t})^T \boldsymbol{\Sigma}_{p,p}^{-1} (\boldsymbol{p}_t - \boldsymbol{\mu}_{\mathrm{p},t})}$ for positional errors; force errors are defined by $D_\mathrm{f} = \sqrt{(\boldsymbol{w}_t - \boldsymbol{\mu}_{\mathrm{f},t})^T \boldsymbol{\Sigma}_{f,f}^{-1} (\boldsymbol{w}_t - \boldsymbol{\mu}_{\mathrm{f},t})}$, where $\boldsymbol{w}_t$, $\boldsymbol{\mu}_{\mathrm{f},t}$, and $\boldsymbol{\Sigma}_{f,f}$ are the corresponding sub-vectors and sub-matrix of the wrench; and gripper finger distance as well as grasp status errors are defined by D_g and D_h respectively. In the *Monitoring* module, we use the full state space to compute D_M, as denoted in (5). We conducted three execution runs, where in the first run, the robot executes the nominal solution without obstacle in the hole (Fig. 3 (c)). Hereby,

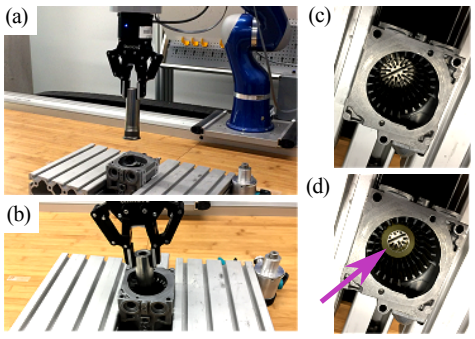

Fig. 3. Robot is approaching the hole (a) and inserts the peg with expected forces during the nominal solution (b). On the right, the empty hole (c) and the hole with obstacle marked with purple arrow (d) are shown.

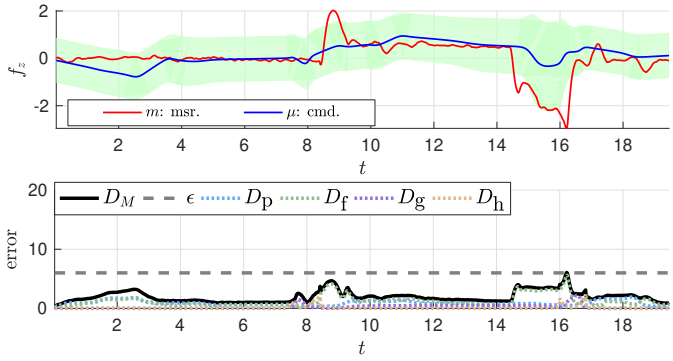

Fig. 4. Peg-in-hole experiment: First run with nominal environment, where running with or without monitoring does not affect the execution. Top: Measured (m) and commanded (μ) force in z-axis and ± 2 standard deviations around commanded state (green). Bottom: Monitored error in different domains with anomaly threshold set to $\epsilon = 6$.

execution regardless of monitoring leads to success (Fig. 4). Second, we insert an obstacle in the hole (Fig. 3 (d)) to simulate unexpected forces during insertion and run the task without monitoring (Fig. 5). In this case, the robot executes the whole commanded motion, without detecting the failed insertion, which could lead to possible robot or object damage. Third, we run the task again with inserted obstacle while the monitoring is activated. Consequently, the monitoring detected the anomaly and stopped the robot to prevent further damage and signalizes that something went wrong during execution (Fig. 6). Since there are no recovery behaviors available, the error cannot be resolved in this state.

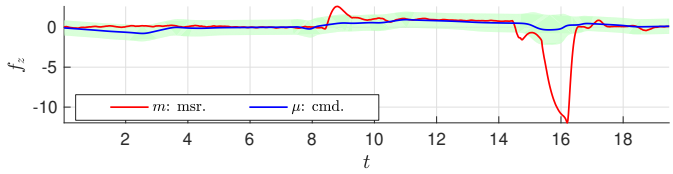

Fig. 5. Second run with obstacle but without monitoring, leading to an undetected error at around 16 s. Plot shows measured (m) and commanded (μ) force in z-axis and ± 2 standard deviations around commanded state (green).

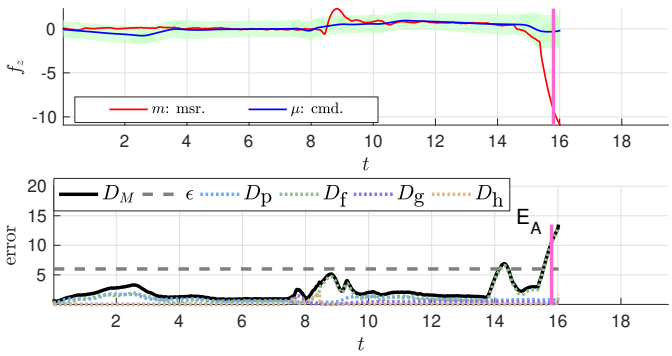

Fig. 6. Third run with obstacle and active monitoring, stopping the robot at the abnormal state (event $\mathrm{E_A}$ marked by horizontal purple bars). Top: Measured (m) and commanded (μ) force in z-axis and ± 2 standard deviations around commanded state (green). Bottom: Monitored error in different domains.

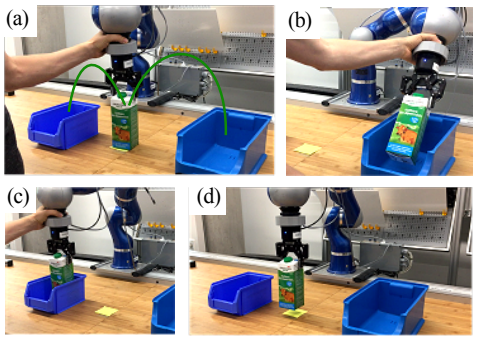

Fig. 7. Human demonstration at pick location and motion paths in green (a) with empty milk carton disposed at the right box (b) and full milk carton placed in the left box (c). Robot execution while picking up the carton (d).

B. Conditional Task: Weight-based Sorting

We focus now on a task, which can either be solved by a nominal solution or where switching to an alternative solution resolves the abnormal state. Figure 7 shows the experimental setup, where full milk cartons shall be packaged in the left box and empty milk cartons shall be disposed in the right box. According to that, three demonstrations were given for the full and three for the empty carton setup. Given the number of two solutions, these demonstrations are assigned autonomously to two clusters. Two solutions were learned and added to the solution pool, namely *full_carton* and *empty_carton*.

In the execution phase, the nominal solution is *full_carton*, which is successfully executed if also a full carton is present. Figure 8 shows the force in z-axis and the monitored errors. In the next run, an empty carton is present and the nominal solution *full_carton* causes an anomaly, as can be seen from the force measurements and monitored error in Fig. 9. Hereby a switch occurs from *full_carton* to *empty_carton*. The same visual appearance of the full and empty carton does not allow to detect their state by vision.

C. Switching with Multiple Alternatives

We evaluate that multiple alternatives can be used within a task, may it be intended task goals such as sorting or

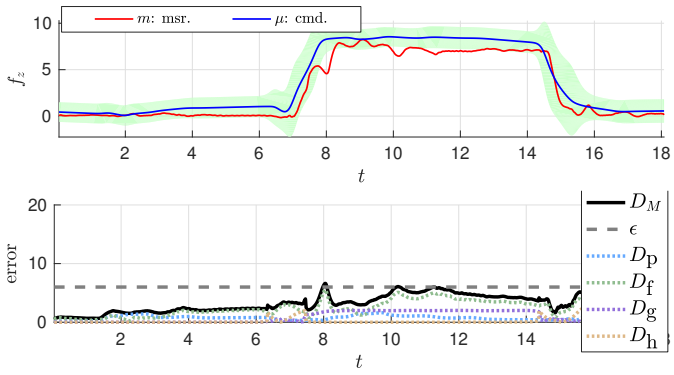

Fig. 8. Top: Measured (m) and commanded (μ) force in z-axis for execution with solution *full_carton* and ± 2 standard deviations around commanded state (green). Bottom: Monitored error in different domains. The monitoring does not affect the execution.

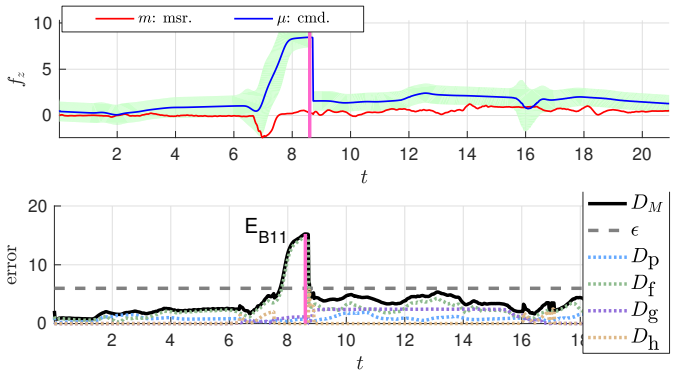

Fig. 9. Top: Measured (m) and commanded (μ) force in z-axis for execution with solution *full_carton* but environment with empty carton, while monitoring is active. In green, ± 2 standard deviations around commanded state. Bottom: Anomaly during execution at event E_{B11} leading to a switch from *full_carton* to *empty_carton*. Vertical purple bars mark switching event.

recovery behaviors. Hereby, we consider a partially structured environment where objects with same properties (e.g. geometry and weight) are located at the same position during execution. Figure 10 shows the experimental setup. A conveyor belt (in blue to the right side of the robot) is used to deliver new objects in a random sequence. It stops whenever an object enters the light barrier and restarts after the object has been removed. The teaching phase consists of 9 demonstrations, showing three different behaviors to the robot. Demonstrations are given in a random order, specified by the sequence of objects arriving at the conveyor belt. In three demonstrations according to solution *empty_box*, the user shows the desired behavior to sort empty boxes (mass $m = 0.15\,\text{kg}$, marked with yellow square) to a desired goal pose. Similarly, three more demonstrations are provided for solution *full_box*, where full boxes (mass $m = 0.85\,\text{kg}$, marked with pink square) are placed at a different goal pose, according to Fig. 10. Three more demonstrations are dedicated to handle the profile object (mass $m = 0.70\,\text{kg}$) with solution *profile*. The clustering method assigns the demonstrations to one of the three solutions ($S = 3$).

In the following, we evaluate that the robot is able to

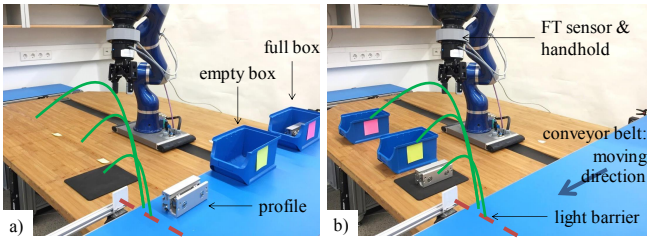

Fig. 10. Possible initial setup a) and target configuration b). The blue conveyor belt moves in direction of the thick blue arrow until an object interferes with the light barrier.

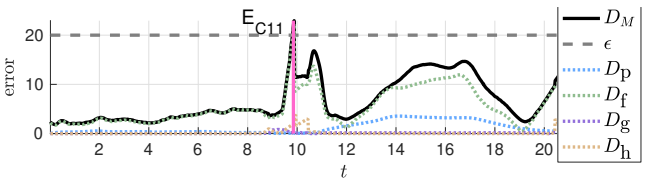

Fig. 11. Error during execution with solution *empty_box* with a switch at event E_{C11} to solution *full_box*. Vertical purple bar marks the switch.

detect anomalies during execution and act without requiring a symbolic task representation. We set the anomaly threshold based on preliminary experiments to $\epsilon = 20$. The task is to manipulate the arriving objects (full box, empty box, profile) onto the target locations as shown in Fig. 10.

When the robot executes solution *empty_box* and having an empty box present at the pick location, the task is solved without anomalies. We analyze now the case where the nominal solution is *empty_box* but the robot faces a full box at the pick location. Figure 11 shows the monitored error and the detected anomaly during the lifting of the box, mainly caused by the deviations in the force domain. This triggered a switching event E_{C11}, where the *Scheduler* switches from *empty_box* to *full_box*.

In the next run, we consider again the nominal solution *empty_box*, but the robot faces the profile at the pick location. Figure 12 on the top plot shows the grasp status, which is a good indication if objects has been picked as intended.

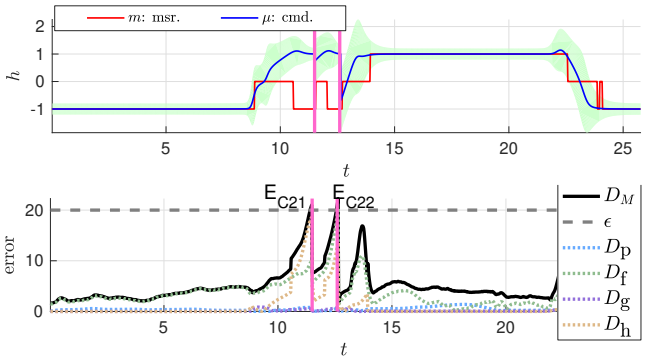

Fig. 12. Top: Measured (m) and commanded (μ) grasp status h, when robot tries to pick an unavailable object. In green, ± 2 standard deviations around commanded state. Bottom: Error during execution with solution *empty_box* with a switch at event E_{C21} to solution *full_box* and a 2nd switch at event E_{C22} to solution *profile*. Vertical purple bars marks the switch in both plots.

The bottom plot shows the Mahalanobis distance and how different modalities contribute to the overall error, especially from grasp status (D_h) and force (D_f). The first anomaly when picking failed triggered a switching event $\mathrm{E}_{\mathrm{C}21}$ from *empty_box* to *full_box*, which has been identified as the next closest solution mainly because of the proximity of the pick location. The next picking attempt triggered the switching event $\mathrm{E}_{\mathrm{C}22}$ and the start of solution *profile*. Finally, the profile object is manipulated to the target position successfully.

VI. Conclusion and Future Work

The proposed approach uses an anomaly detection to trigger an alternative solution to the current observed erroneous state. The alternative is parametrized such that it starts at a proximal region, where the error occurred. Multiple alternatives can be provided, where the most appropriate is selected by minimizing the error to the failed state. This framework allows a user to specify conditions and recovery behaviors within a task by demonstration only.

We have shown that the solution switching works in our evaluated scenarios but guaranteeing a successful transition to an alternative may depend on the initially selected solution and the demonstrator's performance, which is a topic for further investigation. Finding or learning a general anomaly threshold on the state space, which is invariant of the task goals and does not require any parametrization is an interesting challenge for future work, where also learning from executed trials could be considered to increase the monitoring performance. Furthermore, handling of failed demonstrations would increase the robustness of the system. The proposed system capabilities might be fused with other sensors, such as vision, to allow a more adaptive framework in unstructured environments.

References

[1] S. Calinon, "Robot learning with task-parameterized generative models," in *Robotics Research*. Springer, 2018, pp. 111–126.

[2] A. Pervez and D. Lee, "Learning task-parameterized dynamic movement primitives using mixture of gmms," *Intelligent Service Robotics*, pp. 1–18, 2017.

[3] J. Aleotti and S. Caselli, "Trajectory clustering and stochastic approximation for robot programming by demonstration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1029–1034.

[4] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto, "Learning grounded finite-state representations from unstructured demonstrations," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 131–157, 2015.

[5] L. Righetti, M. Kalakrishnan, P. Pastor, J. Binney, J. Kelly, R. C. Voorhies, G. S. Sukhatme, and S. Schaal, "An autonomous manipulation system based on force control and optimization," *Autonomous Robots*, vol. 36, no. 1-2, pp. 11–30, 2014.

[6] R. Zöllner, M. Pardowitz, S. Knoop, and R. Dillmann, "Towards cognitive robots: Building hierarchical task representations of manipulations from human demonstration," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2005, pp. 1535–1540.

[7] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," in *Advances in neural information processing systems*, 2010, pp. 2164–2172.

[8] M. Hanheide, M. Göbelbecker, G. S. Horn, A. Pronobis, K. Sjöö, A. Aydemir, P. Jensfelt, C. Gretton, R. Dearden, M. Janicek, *et al.*, "Robot task planning and explanation in open and uncertain worlds," *Artificial Intelligence*, vol. 247, pp. 119–150, 2017.

[9] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[10] T. Abbas and B. A. MacDonald, "Generalizing topological task graphs from multiple symbolic demonstrations in programming by demonstration (pbd) processes," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3816–3821.

[11] N. Abdo, H. Kretzschmar, and C. Stachniss, "From low-level trajectory demonstrations to symbolic actions for planning," in *ICAPS Workshop on Combining Task and Motion Planning for Real-World App*. Citeseer, 2012.

[12] S. R. Ahmadzadeh, A. Paikan, F. Mastrogiovanni, L. Natale, P. Kormushev, and D. G. Caldwell, "Learning symbolic representations of actions from human demonstrations," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3801–3808.

[13] R. Caccavale, S. Matteo, A. Finzi, and D. Lee, "Kinesthetic teaching and attentional supervision of structured tasks in human-robot interaction," *Autonomous Robots (AURO)*, 2017.

[14] M. L. Visinsky, J. R. Cavallaro, and I. D. Walker, "Robot fault detection and fault tolerance: a survey," *Reliability Engineering and System Safety*, vol. 46, no. 2, pp. 139–158, 1994.

[15] B. R. Donald, "A geometric approach to error detection and recovery for robot motion planning with uncertainty," *Artificial Intelligence*, vol. 37, no. 1-3, pp. 223–271, 1988.

[16] G. E. Hovland and B. J. McCarragher, "Hidden markov models as a process monitor in robotic assembly," *The International Journal of Robotics Research*, vol. 17, no. 2, pp. 153–168, 1998.

[17] E. Di Lello, M. Klotzbucher, T. De Laet, and H. Bruyninckx, "Bayesian time-series models for continuous fault detection and recognition in industrial robotic tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 5827–5833.

[18] R. Lin, E. Khalastchi, and G. A. Kaminka, "Detecting anomalies in unmanned vehicles using the mahalanobis distance," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 3038–3044.

[19] A. Nakamura, K. Nagata, K. Harada, N. Yamanobe, T. Tsuji, T. Foissotte, and Y. Kawai, "Error recovery using task stratification and error classification for manipulation robots in various fields," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 3535–3542.

[20] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3828–3834.

[21] D. H. Grollman and A. Billard, "Donut as i do: Learning from failed demonstrations," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3804–3809.

[22] A. Rodriguez, D. Bourne, M. Mason, G. F. Rossano, and J. Wang, "Failure detection in assembly: Force signature analysis," in *Automation Science and Engineering (CASE), 2010 IEEE Conference on*. IEEE, 2010, pp. 210–215.

[23] A. Stolt, M. Linderoth, A. Robertsson, and R. Johansson, "Detection of contact force transients in robotic assembly," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 962–968.

[24] T. W. Liao, "Clustering of time series data - a survey," *Pattern recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[25] H. Izakian, W. Pedrycz, and I. Jamal, "Fuzzy clustering of time series data using dynamic time warping distance," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 235–244, 2015.

[26] T. Eiband, M. Saveriano, and D. Lee, "Learning haptic exploration schemes for adaptive task execution," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[27] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[28] S. S. Khan and M. G. Madden, "A survey of recent trends in one class classification," in *Irish conference on artificial intelligence and cognitive science*. Springer, 2009, pp. 188–197.

[29] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger, "The dlr lightweight robot: design and control concepts for robots in human environments," *Industrial Robot: an international journal*, vol. 34, no. 5, pp. 376–385, 2007.