Learning of Exception Strategies in Assembly Tasks

Bojan Nemec, Mihael Simonič, Aleš Ude

Abstract— **Assembly tasks performed with a robot often fail due to unforeseen situations, regardless of the fact that we carefully learned and optimized the assembly policy. This problem is even more present in humanoid robots acting in an unstructured environment where it is not possible to anticipate all factors that might lead to the failure of the given task. In this work, we propose a concurrent LfD framework, which associates demonstrated exception strategies to the given context. Whenever a failure occurs, the proposed algorithm generalizes past experience regarding the current context and generates an appropriate policy that solves the assembly issue. For this purpose, we applied PCA on force/torque data, which generates low dimensional descriptor of the current context. The proposed framework was validated in a peg-in-hole (PiH) task using Franka-Emika Panda robot.**

I. INTRODUCTION

Robot task executions often stop due to a variety of errors that cannot be forseen in advance. In such cases it is most often necessary for a human cooperating with a robot to manually eliminate the cause of the error and restart the task [1]. In the vast majority of cases, the robot does not learn anything from such experiences. If a similar or even the same situation is again encountered, the intervention of a human will be needed again and again. The frequency of such events depends on the process. The less the process is structured and determined, the more such events occur. In view of this, we can expect that this problem will be even more pronounced by the upcoming generation of humanoid and service robots that will perform a variety of tasks in domestic environments, which are often not well structured.

Despite the impressive development of robotics in recent years, there are only a few research works dealing with the above-mentioned problems [2]. The most common solution is to appended the existing control policy with a fixed search/rescue pattern, such as stochastic search patterns, spiral search, raster search [3], tilt strategy, dithering and hopping, etc. [4]. A more systematic approach to the policy execution failures was considered in [5], where the robot recognizes when it is unable to proceed and requires human intervention to complete the task. In [6] this approach was extended with the ability to correct the state sequencing by a human demonstrator. A framework where the teacher corrects a continuous action selection was proposed in [7].

More work has been done in the field of automatic classification of robot failures. Tovar et al. [8] proposed Bayesian network classifier, which was able to classify between three different failure situations during PiH operation using force-torque data. A similar approach was realized using multilayer neural networks [9]. Neural networks were also applied for fault detection of robot actuators [10]. Karapinar et al. [11] developed experience-based learning of failure contexts from sensor data. Investigation of the cause of the failure using Hidden Markov Models was studied in the work of Altan et al. [12]

Our research aims to develop an integrated solution for automatic handling of failures in assembly processes. The proposed approach combines incremental kinesthetic learning, failure detection and classification, and statistical learning. Exception learning is initially supervised by an operator who first resolves the issue on the occurrence of the error, and then demonstrates appropriate action that enables the continuation of the given assembly task. Robot builds a database of demonstrated actions and associates them with the detected error context. Using statistical learning, it generates appropriate action for unforeseen errors from the demonstrated actions. The robot becomes more and more autonomous and eventually does not require any human intervention to resolve assembly failures.

The paper is organized into 6 sections. In the next section, we present our main idea. The proposed framework is composed of three main technologies: 1) an algorithm which allows to incrementally update of a nominal trajectory along the refinement tube, 2) error classification using principal component analyses (PCA) and 3) non-parametric statistical learning using locally weighted regression (LWR). They are presented in Sections III, IV and V, respectively. Experimental verification on a generic assembly task, peg-in-hole, is described in Section VI. Our final conclusions and discussion about limitations and possible future extensions of the proposed framework are given in Section VII.

II. FRAMEWORK FOR LEARNING EXCEPTION STRATEGIES

In this work, we assume that the basic control policy to execute the desired task was appropriately learnt and optimized. An efficient way to learn the assembly policy is kinesthetic guidance [13], but other methods can also be used, e.g. off-line programming using CAD models etc. Next, we assume that the policy is parameterized with Cartesian space DMPs [14], although our framework enables to use also other popular parameterization techniques, such as Gaussian Mixture Models and Gaussian Mixture Regression (GMM-GMR) [15], Probabilistic Motion Primitives (ProMP)

Humanoid & Cognitive Robotics Lab, Department of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia, `(bojan.nemec, mihael.simonic,ales.ude)@ijs.si`

[16], Radial Basis Functions (RBF), etc. Optimization of the desired control policy can be done using standard techniques, such as iterative learning control (ILC) [17] or reinforcement learning (RL) [18]. Our framework does not aim to change the demonstrated control policy, here denoted by π_d. Rather, it enables the generation of an alternative strategy at the onset of an unexpected situation, which results in failure and consequently the suspension of the task. The reasons for failure can vary, from the incorrect grasping of parts, deviations in the geometry of components, damaged parts, etc.

The follow-up actions are demonstrated once the failure occurs. These demonstrations are captured together with the sensory information, which is used later to classify the cause of the failure. The basic strategy is illustrated in Fig. 1. Whenever a failure occurs, the robot stops. Initially, the robot has no knowledge how to continue, therefore it expects the intervention of the operator. The operator first rolls back the robot action to the point from which it is possible to continue the task. Next, using incremental learning along the refinement tube [19], [20], the human operator demonstrates an alternative policy, which allows the robot to perform the given task from the current context. The context is determined from sensor signals. In assembly operations, we typically rely on a force-torque sensor, but other sensors such as pressure sensors, vision sensors, etc. can also be used.

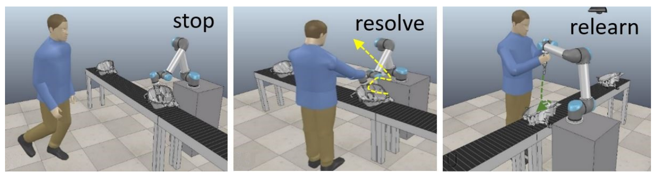

Fig. 1. Left: A failure occurs and the robot stops and waits for the intervention of the operator. Center: Operator rolls back the robot actions to resolve the issue. Right: Operator demonstrates alternative policy

As explained above, the robot analyzes the cause of failure using sensory data. It memorizes the current context and the alternative control policy and saves both of them in a database. When a failure occurs for the second time, the robot checks if it has any experience about the failures in similar contexts. If this is the case, the robot generates an alternative policy using statistical learning [21] and executes it. If the robot either does not have any previous experience or the alternative policy was not successful, it stops and waits for the operator to demonstrate the appropriate policy for the current context and stores both of them to the database. Eventually, the robot does not require any human intervention to resolve failures. The flow chart of the proposed framework is shown in Fig. 2.

The process of partially autonomous database expansion for learning motor primitives was also considered by Petrič et al. [22], who focused on analyzing the required size of the database to ensure accurate task execution and the stability of the resulting control policies. The distinguishing feature of our work is the automatic determination of features that are used to guide the process of database expansion

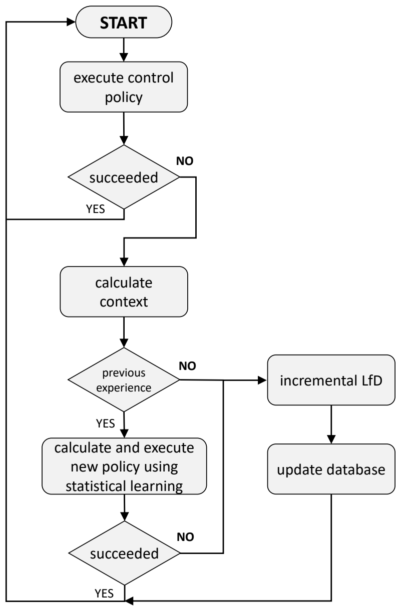

Fig. 2. Flow chart of the proposed approach with learning and execution of the exception strategies

and the generation of query points for statistical learning. While feature selection can sometimes be easily performed manually, there are many tasks where features can be selected only by a computational method.

The framework explained in this section is general and can be applied to most of the robot policies. In the continuation of the paper, we will focus on the assembly policies.

III. Teaching of Follow-up Actions

As the robot stops due to an error, it needs an intervention from the operator, as explained in the previous section. First, the operator needs to rollback the robot actions to the point from which the robot can continue the task. Next, he has to demonstrate a new policy, which fits the current situation. In most cases, only minor modifications of the existing policy are necessary. After that, the operator has to test the demonstrated policy applied in the current context and refine it, if appropriate. Finally, the operator demonstrates a new velocity profile if the original does not suit the newly demonstrated policy. To fulfil all of these requirements, the operator must be able to freely move the robot forward and backward along the existing policy at any speed and change only parts where changes are needed. For this purpose, we applied our previously developed method [20] based on kinesthetic guiding within a refinement tube [19]. In this method, actions are parameterized with Cartesian speed-scaled dynamic movement primitives (CSDMP) (see Appendix). Here, we will review the main idea of this method and present some modifications that enable more efficient trajectory refinement.

In order to allow the operator to move the robot forward and backward along the demonstrated trajectory π_d, we replace the speed scaling factor τ associated with the demonstrated CSDMP [30], [24] with a new speed scaling factor, which is inversely proportional to the force projected to the tangent of the path defined by trajectory π_d. The tangent of the path is calculated as $\mathbf{t}_p(s) = \frac{\dot{\mathbf{p}}_d(s)}{\|\dot{\mathbf{p}}_d(s)\|}$, where $\mathbf{p}_d \in \mathbb{R}^3$ are commanded (demonstrated) positions and s denotes the phase. The corresponding tangential direction for the rotational motion is given by $\mathbf{t}_r(x) = \frac{\boldsymbol{\omega}_d(s)}{\|\boldsymbol{\omega}_d(s)\|}$, where $\boldsymbol{\omega}_d \in \mathbb{R}^3$ is the commanded robot end-effector angular velocity in Cartesian space. We compute the new speed scaling factor τ as follows

$$\tau(s) = \frac{1}{k_1 \mathbf{F} \cdot \mathbf{t}_p(s) + k_2 \mathbf{M} \cdot \mathbf{t}_r(s)}, \quad (1)$$

where $(\cdot)$ denotes dot product and k_1, k_2 are positive scalars, used to scale the velocities of the translational and rotational motion along the $\pi_d(s)$. $\mathbf{F} \in \mathbb{R}^3$ and $\mathbf{M} \in \mathbb{R}^3$ are the measured vectors of forces and torques at the robot tool and expressed in robot base coordinate system. If $\mathbf{F} \cdot \mathbf{t}_p(s) + \mathbf{M} \cdot \mathbf{t}_r(s) \to 0$, then $\tau(s) \to \infty$, which stops the CSDMP integration. We apply this τ to the DMP and phase integration [30] to move the robot along the trajectory $\pi_d(s)$ in the direction of the applied forces and torques, with the speed proportional to them. This makes the guiding process extremely intuitive. An operator just pushes the robot along the tangent of the trajectory. In order to prevent uncontrolled robot movement along the trajectory due to the force/torque sensor noise, a threshold is usually applied to (1).

CSDMP as a dynamical system becomes unstable for negative τ, i.e. when the motion is reversed. In such a case we have to apply a reverse CSDMP, learnt from the time-reversed trajectory $\pi_r(s) = \pi_d(e^{-\alpha_s}/s)$ (see also Eq. (6)).

In order to modify the originally demonstrated trajectory represented by a CSDMP, the robot should be compliant in the directions of normal and binormal directions of the path [23] and stiff in the tangential direction, which is ensured by applying an appropriate control law [24]. This way we are allowed to displace a robot in a plane along these two directions and sample new poses. The modification to the original trajectory at phase s is given by the error vector $\mathbf{e}$

$$\mathbf{e}(s) = \left[\begin{array}{c} \mathbf{e}_p(s) \\ \mathbf{e}_o(s) \end{array} \right] = \left[\begin{array}{c} \mathbf{p}(s) - \mathbf{p}_d(s) \\ \log(\mathbf{q}(s) * \overline{\mathbf{q}_d}(s)) \end{array} \right], \quad (2)$$

where $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{q}$, $\mathbf{q}_d$ denote the positions and quaternions that describe the current orientation and the positions and quaternions computed by the CSDMP, respectively. $*$ denotes the quaternion product and $\overline{\mathbf{q}}$ the conjugate quaternion, whereas the quaternion logarithm is defined as

$$\log(\mathbf{q}) = \log(v, \mathbf{u}) = \begin{cases} \arccos(v) \dfrac{\mathbf{u}}{\|\mathbf{u}\|}, & \mathbf{u} \neq 0 \\ [0,0,0]^{\mathrm{T}}, & \text{otherwise} \end{cases}. \quad (3)$$

It maps the quaternion describing orientation to the angular velocity that rotates the identity orientation to the current orientation within unit time.

In [20] we proposed to sample robot poses during the above described kinesthetic guiding process and calculate the new nonlinear forcing term of the CSDMP using batch regression whenever the sign of $\tau(s)$ changes. For this process to work, the modified robot positions and orientations must be sampled at exactly the same phase s as the original trajectory. This means that the phase of the modified trajectory should be determined very accurately. Even small deviations of s can lead to a wrong sequential order of the captured end-effector poses, which can corrupt the modified trajectory.

In this paper, we propose an alternative solution which is less sensitive to the accuracy of phase calculation. Instead of capturing the complete modified trajectory and updating the nonlinear forcing term at the change of the direction, we concurrently modify weights of the CSDMP's nonlinear forcing terms, $\mathbf{W}(s) = [\mathbf{W}_p(s)^{\mathrm{T}},\ \mathbf{W}_o(s)^{\mathrm{T}}] \in \mathbb{R}^{N \times 6}$, using recursive regression formulas

$$\mathbf{W}(s) = \mathbf{W}(s_{-1}) + \mathbf{P}(s)\mathbf{x}(s)\mathbf{e}(s)^{\mathrm{T}}\mathbf{K}_l \quad (4)$$

$$\mathbf{P}(s) = \frac{1}{\lambda}\left(\mathbf{P}(s_{-1}) - \frac{\mathbf{P}(s_{-1})\mathbf{x}(s)\mathbf{x}(s)^{\mathrm{T}}\mathbf{P}(s_{-1})}{\lambda + \mathbf{x}(s)^{\mathrm{T}}\mathbf{P}(s_{-1})\mathbf{x}(s)} \right), \quad (5)$$

where $\mathbf{P}(s) \in \mathbb{R}^{N \times N}$ is the error covariance matrix and $\mathbf{x}(s) \in \mathbb{R}^N$ is a vector of Gaussian kernel functions (see Appendix). N is the number of kernel functions, s_{-1} denotes the phase in the previous step, λ is a forgetting factor, which is usually kept close to 1, and $\mathbf{K}_l \in \mathbb{R}^{6 \times 6}$ is a diagonal estimation gain matrix that modifies the compliance of the robot during the kinesthetic guidance. We reset the covariance error matrix to the default value, $\mathbf{P}(s) = \gamma\mathbf{I}$, whenever the temporal scaling factor τ changes its sign. $\mathbf{I}$ denotes identity matrix and $\gamma > 0$ is a suitably chosen scalar. This is necessary because the recursive algorithm becomes increasingly less sensitive to the new trajectory updates with the number of iterations. Namely, from (5) it follows that the error covariance matrix $\mathbf{P}(s)$ is independent of measurements and has monotonically decreasing eigenvalues. Consequently, the magnitude of updates computed by (4) is also decreasing and thus influencing the resulting control policy less and less. Note that the magnitude of updates is also affected by the choice of forgetting factor λ.

The procedure described above is simultaneously applied to the reversed CSDMP of the demonstrated trajectory. Based on the sign of $\tau(s)$ defined in Eq. (1), we either integrate the original (in case of positive sign) or reversed CSDMP (in case of negative sign, but in this case $-\tau(s)$ is used for integration). To compute the current phase s_r of the reversed CSDMP from the phase s of the original CSDMP and vice versa, we exploit the following relationship

$$s s_r = e^{-\alpha_s t/\tau_0} e^{-\alpha_s(\tau_0 - t)/\tau_0} = e^{-\alpha_s}. \quad (6)$$

This is true because the temporal constants in the original and reversed CSDMP are constant. Hence $s_r = e^{-\alpha_s}/s$ and $s = e^{-\alpha_s}/s_r$. The update formulas (4) and (5) are then applied to both the original and reversed CSDMP.

Using the procedure described above, we generate a new exception policy directly in the CSDMP form, where the forcing term weights $\mathbf{W}_p$ and $\mathbf{W}_o$ are new but all other CSDMP's parameters are taken from the CSDMP representing the originally demonstrated insertion policy. Thus the speed (or equivalently, the temporal scaling factor $\tau(s)$) of the resulting CSDMP is still determined by the demonstrated policy, which is suboptimal. Therefore, we demonstrate a new speed profile by executing the newly learnt CSDMP while the user is pushing the robot along the trajectory. The new temporal scaling factor $\tau(s)$ is computed from the user-applied forces and torques as specified in Eq. (1). During this demonstration, the robot is stiff in all directions to prevent it from deviating from the learnt path. We sample the resulting $\tau(s)$, which is then associated with the CSDMP representing the exception strategy instead of the temporal scaling factor obtained from the originally demonstrated insertion policy.

For statistical learning it is beneficial to store the learnt exception strategy as a time-dependent trajectory (see Section V). Thus the resulting CSDMP is integrated with the newly sampled $\tau(s(t))$ one more time (without executing the generated motion with the robot) and the points on the resulting trajectory are sampled to generate the training data set (11) for statistical learning. Finally, we re-compute the CSDMP parameters from the sampled data (11), setting (for the i-th exception strategy) $\tau_i(s) = \tau_{0,i} = t_{i,T_i}$, $\mathbf{g}_{p,i} = \mathbf{p}_{i,T_i}$, $\mathbf{g}_{o,i} = \mathbf{q}_{i,T_i}$, and computing the suitable $\mathbf{W}_{p,i}$ and $\mathbf{W}_{o,i}$.

IV. Determination of Failure Context from Force-Torque Sensor Data

During the assembly task execution, it is necessary to monitor the exerted forces on the robot hand in order to prevent damaging of the parts or even robot itself at the occurrence of an unexpected situation. Forces and torques are typically used also to actively guide the assembly process. Execution failures are in most cases characterized by a sudden increase of forces and torques. In our work we therefore used a simple approach where a failure is detected if the sensed forces and torques exceeds a predefined threshold. An autonomous robot should have the ability to detect the reason for the execution failure, which enables it to plan an appropriate recovery action. In this section we propose an algorithm for the calculation of low dimensional features that characterize the detected failures based on the sensed forces and torques. We map the sensed forces and torques to a low dimensional feature space because feature dimensionality is important for statistical learning. For this purpose, we apply Principal Component Analyses (PCA) as a popular dimensionality reduction technique.

Let's assume that we have m measurements of forces and torques, $\mathbf{h}_i = [\mathbf{F}_i^{\mathrm{T}}\ \mathbf{M}_i^{\mathrm{T}}]$, $i = 1, \ldots, m$, captured at the time when the i-th failure has been detected. Each measurement thus corresponds to exactly one failure during the assembly. We form a data matrix

$$\mathbf{Z} = \begin{bmatrix} \mathbf{h}_1 - \bar{\mathbf{h}} \\ \vdots \\ \mathbf{h}_m - \bar{\mathbf{h}} \end{bmatrix} \in \mathbb{R}^{m\times 6}, \tag{7}$$

where $\bar{\mathbf{h}}$ is the row vector of average values of all forces and torques. PCA is an orthogonal linear transformation that maps the data $\mathbf{Z}$ to a new coordinate system such that the biggest variance occurs in the first coordinate, which is called the first principal components. All subsequent coordinates have a lower variance than the previous one. PCA can be calculated by applying singular value decomposition in the form

$$\mathbf{Z} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{8}$$

where matrix $\mathbf{V} \in \mathbb{R}^{6\times 6}$ is orthogonal and maps the data to a new coordinate system

$$\mathbf{C} = \mathbf{Z}\mathbf{V}, \tag{9}$$

such that the principal components are sorted as columns in $\mathbf{C}$. Also, the singular values that form the diagonal matrix $\mathbf{\Sigma} = \mathrm{diag}(\sigma_i) \in \mathbb{R}^{m\times 6}$ are nonnegative and sorted from the biggest to the lowest value $\sigma_j,\ j = 1, \ldots, 6$. The magnitude of singular values determines the significance of each direction determined by eigenvectors in $\mathbf{V}$. The dimensionality reduction is performed in such a way that we keep only the first p columns of $\mathbf{V}$, which correspond to the first p biggest singular values. Whenever a new failure occurs, we calculate the corresponding context data from the measured force-torque vector $\mathbf{h}$ using

$$\mathbf{c} = (\mathbf{h} - \bar{\mathbf{h}})\mathbf{V}_p, \tag{10}$$

where $\mathbf{V}_p \in \mathbb{R}^{6\times p}$ denotes the matrix composed of first p principal eigenvectors of $\mathbf{V}$. The resulting context vector $\mathbf{c}$ is used as query for statistical learning.

V. Statistical Learning of Exception Strategies from Failure

Initially, every failure requires that a user demonstrates a new exception strategy as described in Section III. An exception strategy enables the robot to continue the task after a failure has occurred. It is fully defined by the time evolution of tool poses given in Cartesian coordinates and the associated context. Let's define a set of m exception strategies as

$$\mathcal{G} = \{\mathbf{p}_{i,k}, \mathbf{q}_{i,k}, \dot{\mathbf{p}}_{i,k}, \boldsymbol{\omega}_{i,k}, \ddot{\mathbf{p}}_{i,k}, \dot{\boldsymbol{\omega}}_{i,k}, t_{i,k}; \mathbf{c}_i\}_{i=1}^{m}{}_{k=1}^{T_i}, \tag{11}$$

where $\mathbf{p}_{i,k} \in \mathbb{R}^3$ are the positions, $\mathbf{q}_{i,k} \in \mathrm{S}^3$ are the unit quaternions describing orientation, S^3 is a unit sphere in $\mathbb{R}^4$, i is the demonstration index, k are trajectory samples, and T_i is the number of samples on the i-th exception strategy. Each exception strategy is associated with the context vector $\mathbf{c}_i$ calculated according to Eq. (10) from the measured forces and torques $\mathbf{h}_i$ at the time the failure occured.

Once a sufficient number of exception strategies becomes available, we can exploit previously learnt strategies to generate new ones. This is accomplished using statistical learning techniques. For this purpose we applied locally weighted regression, due to its simplicity and efficiency. LWR belong to a class of non-parametric statistical approximation methods [25] and it has been successfully applied to may robotics applications such as throwing, reaching, drumming, etc. [26].

When the next failure occurs, we first determine the current context $\mathbf{c}$ using the measured forces and torques and Eq. (10). The resulting context vector is used as query point for LWR. Given this query point and a number of existing exception strategies, LWR can compute a new exception strategy. Recall from Section III that in our work, exceptions strategies are defined by CSDMPs. A CSDMP contains a number of parameters (see Appendix), but in the context of exception strategies only some of them change; the weights specifying the nonlinear forcing term, the temporal scaling factor τ_0, the goal position $\mathbf{g}_p$, and the goal orientation $\mathbf{g}_o$. Thus a function that maps query points into a new exception strategy can be written as follows

$$\mathbf{G}(\mathcal{G}) : \mathbf{c} \mapsto \{\mathbf{W}_p, \mathbf{W}_o, \tau_0, \mathbf{g}_p, \mathbf{g}_o\}. \tag{12}$$

Note that the temporal scaling factor $\tau(s)$ is constant for all exception strategies as nonlinear speed scaling is applied only to the initially demonstrated trajectory. Once the initially demonstrated trajectory is adapted by kinesthetic teaching and a new exception strategy is generated, the resulting control policy is resampled to (11) with constant temporal scaling factor as described at the end of Section III.

As explained in [26], $\mathbf{G}(\mathcal{G})$ becomes a smooth function of $\mathbf{c}$ only if example trajectories, in our case exception strategies, are similar and transition between each other smoothly. This is the case in our work because the policy adaptation method described in Section III ensures that the adapted exception strategy is similar to the original control policy. Thus the solution trajectory computed by LWR is similar to other exception strategies but adapted to the current context $\mathbf{c}$.

To compute the generalized CSDMP forcing terms weights $\mathbf{W}_p, \mathbf{W}_o \in \mathbb{R}^{3\times N}$, we apply the following optimization problem

$$\min_{\mathbf{W}_p,\mathbf{W}_o} \sum_{i=1}^{m} ||\mathbf{X}_i[\mathbf{W}_p^{\mathrm{T}}, \mathbf{W}_o^{\mathrm{T}}] - [\tilde{\mathbf{P}}_i, \tilde{\mathbf{Q}}_i]||^2 \mathrm{K}(\mathbf{c}, \mathbf{c}_i), \tag{13}$$

with $\tilde{\mathbf{P}}_i \in \mathbb{R}^{T_i\times 3}$ being a matrix with rows $\tilde{\mathbf{p}}_{i,k} = (\tau_{0,i}^2 \ddot{\mathbf{p}}_{i,k} + \alpha_z \tau_{0,i} \dot{\mathbf{p}}_{i,k} - \alpha_z\beta_z(\mathbf{g}_p - \mathbf{p}_{i,k}))^{\mathrm{T}}$, and $\tilde{\mathbf{Q}}_i \in \mathbb{R}^{T_i\times 3}$ a matrix with rows calculated as $\tilde{\mathbf{q}}_{i,k} = (\tau_{0,i}^2 \dot{\boldsymbol{\omega}}_{i,k} + \alpha_z\tau_{0,i}\boldsymbol{\omega}_{i,k} - 2\alpha_z\beta_z \log(\mathbf{g}_o * \overline{\mathbf{q}}_{i,k}))^{\mathrm{T}}$. The rows of matrix $\mathbf{X}_i \in \mathbb{R}^{T_i\times N}$ are calculated using Gaussian DMP kernels $\mathbf{x}$ at phases $s_{i,k}$, i.e. $\mathbf{X}_i = [\mathbf{x}(s_{i,1}), \dots, \mathbf{x}(s_{i,T_i})]^{\mathrm{T}}$ [14]. We selected the tricube kernel [27] for $\mathrm{K}(\mathbf{c}, \mathbf{c}_i)$, which is defined as

$$\mathrm{K}(\mathbf{c}, \mathbf{c}_i) = \begin{cases} (1 - (||\mathbf{c} - \mathbf{c}_i||/h)^3)^3, & ||\mathbf{c} - \mathbf{c}_i||/h \le 1 \\ 0, & \text{otherwise} \end{cases}, \tag{14}$$

where h is a hyper-parameter that determines the range and importance of training data used for generalization.

Since the temporal scaling constants and goal position and orientation are measured directly, i.e. $\tau_{0,i} = t_{i,T_i}$, $\mathbf{g}_{p,i} = \mathbf{p}_{i,T_i}$, $\mathbf{g}_{o,i} = \mathbf{q}_{i,T_i}$, their generalization by LWR is easier. They are computed as follows

$$\tau_0 = \frac{\sum_{i=1}^m \mathrm{K}(\mathbf{c}, \mathbf{c}_i)\tau_{0,i}}{\sum_{i=1}^m \mathrm{K}(\mathbf{c}, \mathbf{c}_i)}, \tag{15}$$

$$\mathbf{g}_p = \frac{\sum_{i=1}^m \mathrm{K}(\mathbf{c}, \mathbf{c}_i)\mathbf{g}_{p,i}}{\sum_{i=1}^m \mathrm{K}(\mathbf{c}, \mathbf{c}_i)}, \tag{16}$$

$$\mathbf{g}_o = \frac{\tilde{\mathbf{g}}_o}{\|\tilde{\mathbf{g}}_o\|},\ \tilde{\mathbf{g}}_o = \frac{\sum_{i=1}^m \mathrm{K}(\mathbf{c}, \mathbf{c}_i)\mathbf{g}_{o,i}}{\sum_{i=1}^m \mathrm{K}(\mathbf{c}, \mathbf{c}_i)}. \tag{17}$$

VI. Experimental evaluation on peg-in-hole task

The proposed framework was experimentally verified on a peg in hole task (PiH), which is a typical assembly operation. For this purpose, we performed square peg insertion in *Cranfield Benchmark* [28], which is a standardized tool that encompasses a typical level of complexity for an industrial assembly task. The initial PiH policy was obtained with kinesthetic guidance. In our experiments, we focused on a typical source of failure in automated assembly, i.e. bad pose estimation of the assembly part, which causes either imperfect grasping or non-adequate insertion policy (or both of it). We used the collaborative robot arm *Franka Emika Panda* in all experiments. Integrated joint-torque sensors were used to estimate the Cartesian forces and torques.

Our first goal was to evaluate whether the proposed estimation of the context during the failure is appropriate to generate query points that are suitable for statistical learning. We started by evaluating the case where the robot grasped the assembly part at wrong angles. The experiment was repeated for 8 equally spaced angles around zero (which was the correct angle for the learnt control policy) with a spacing of 3 degrees, as illustrated in Fig. 3.

Due to the offset in the grasping angle, the robot failed to insert the peg and stopped the execution as it exceeded the force threshold, which was set to $10N$ in z direction. At that moment we recorded the forces and torques. Due to the different metric, we scaled the torque data by a factor of 10. We performed PCA on this set of data. Only the first singular value of the matrix $\mathbf{\Sigma}$ deviated from the others, which were practically equal to zero, meaning that our context is one dimensional. Figure 4 shows the evolution of this context depending on the offset in the grasping angle. Note that the estimated function is almost linear and, most of all, monotonically increasing, which makes it a perfect query for statistical learning. If we compare the estimated context to the forces and torques of our data set, we can see that it is very similar to the torque around x axis. This is exactly what an experienced robot operator would intuitively choose as a query. However, the proposed algorithm learns this without any human intervention.

Next, we evaluated the effectiveness of context determination with offsets in grasping position along y axis. Similar as in the previous case, we applied 8 equally spaced values with a spacing of 2 mm and recorded forces and torques during the attempts of insertion. Fig. 5 shows that the estimated context function is again monotonic, which is essential for query points.

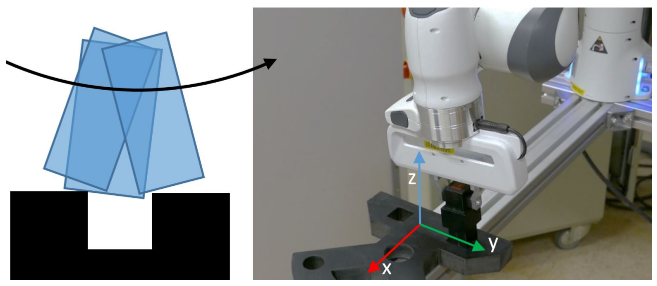

Fig. 3. Square PiH with Franka-Emika Panda robot where the part is grasped at different offsets from the ideal grasping position.

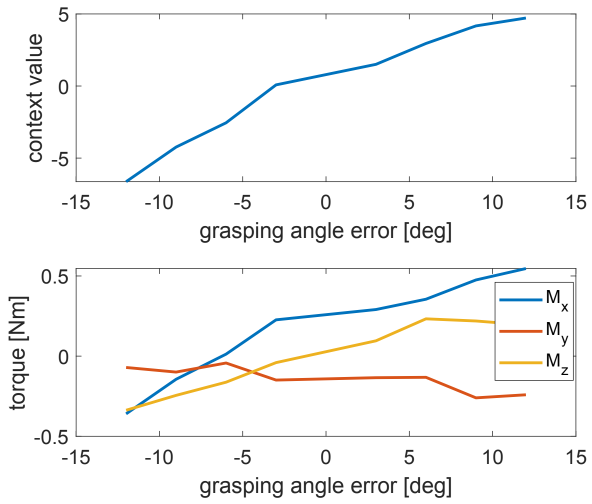

Fig. 4. Above: Context as a function of grasping angle offset. Below: The measured torques.

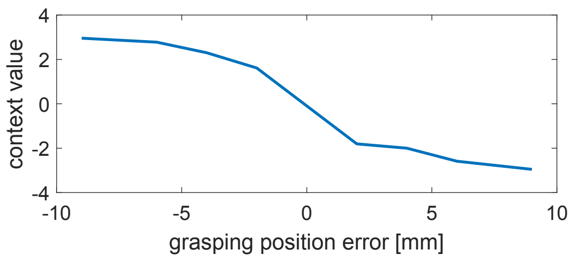

Fig. 5. Context as a function of grasping position offsets.

Finally, we evaluated our framework as a whole. Again we tested the influence of grasping offset in the y direction. We generated 4 grasps with equally spaced offsets with spacing of 3 mm around the correct grasp, i.e. the grasp applied during the initial demonstration. Because of this offset, the robot failed to insert the peg and stopped the execution due to excessive forces in the z direction. We captured the forces and torques, calculated the context value and demonstrated the alternative policy for each case as explained in Section III. These data were used to generate the initial database of exception strategies (11) for generalization. The original peg insertion policy, the four demonstrated exception policies, and one of the generalized exception startegies are shown in Fig. 6. The success rate in 50 experiments was 82%. Note

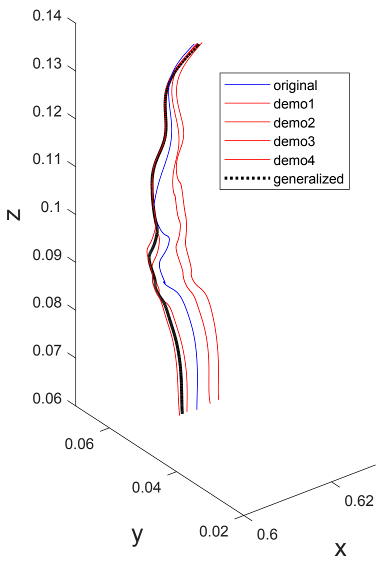

Fig. 6. The original policy, the four demonstrated exception policies and one of the generalized exception policies. Due to the offset in the grasping position, the final positions of the robot's end-effector are different although the final position of the peg is the same in all cases.

that the robot was stiff during the insertion. By exploiting the robot's compliance and also by making use of a larger database of pretrained exception policies for generalization, the success rate could be improved.

VII. Conclusions

In this work we proposed an integrated framework for learning exception strategies as they arise. It integrates learning by demonstration, PCA-based classification of failures using the resulting force-torque data, and the generation of exception strategies by statistical learning. The main novelties are 1) determination of the exception context using PCA and the 2) the application of the determined context for statistical learning of exception strategies. We also improved our previously developed method for the adaption of policies [24] by replacing batch regression with recursive regression. Finally, the proposed method for generalization of orientation DMPs by minimizing (13) is new. The result is a novel framework for learning and adaptation of exception strategies. The proposed approach was evaluated on the peg-in-hole task where we demonstrated the effectiveness of our approach.

In the current implementation, the context for the database query was calculated based on a single measurement of forces and torques. In general, however, forces and torques need to be taken into account as a time function. We can easily do this by encoding them as RBFs and expanding the measurements $\mathbf{h}$ with weights of RBFs. More challenging remains, how to include also other sensors such as vision to calculate the failure context. Our future work will involve integration of such multi modal data and direct estimation of low-dimensional context using deep auto-encoders.

References

[1] A. F. Fudzin and M. A. Majid, "Reliability and availability analysis for robot subsystem in automotive assembly plant: a case study," in *3rd International Conference of Mechanical Engineering Research (ICMER 2015)*, 2015.

[2] B. Argall, S. Chernova, M. M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, pp. 469–483, 2009.

[3] F. Abu-Dakka, B. Nemec, A. Kramberger, A. Glent Buch, N. Krüger, and A. Ude, "Solving peg-in-hole tasks by human demonstration and exception strategies," *Industrial Robot: An International Journal*, vol. 41, pp. 575–584, 10 2014.

[4] J. A. Marvel, R. Bostelman, and J. Falco, "Multi-robot assembly strategies and metrics," *ACM Computing Surveys*, vol. 51, no. 1, 2018.

[5] M. N. Nicolescu and M. J. Mataric, "Learning and interacting in human-robot domains," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 31, no. 5, pp. 419–430, 2001.

[6] M. N. Nicolescu and M. J. Mataric, "Natural methods for robot task learning: Instructive demonstrations, generalization and practice," in *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003, pp. 241–248.

[7] B. D. Argall, B. Browning, and M. Veloso, "Learning robot motion control with demonstration and advice-operators," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 399–404.

[8] J. Alonso, B. Saha, J. Romero, and D. Ortega-Aranda, "Bayesian network classifier with efficient statistical time-series features for the classification of robot execution failures," *SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE)*, vol. 3, pp. 80–89, 11 2016.

[9] A. Diryag, M. Mitic, and Z. Miljkovic, "Neural networks for prediction of robot failures," *Journal of Mechanical Engineering Science*, vol. 228, no. 8, pp. 1444–1458, 2014.

[10] C. N. Cho, J. T. Hong, and H. J. Kim, "Neural network based adaptive actuator fault detection algorithm for robot manipulators," *Journal of Intelligent & Robotic Systems*, 2018.

[11] S. Karapinar and S. Sariel, "Cognitive robots learning failure contexts through real-world experimentation," *Autonomous Robots*, vol. 39, no. 4, pp. 469–485, 2015.

[12] D. Altan and S. Sariel, "Probabilistic failure isolation for cognitive robots," in *Proceedings of the 27th International Florida Artificial Intelligence Research Society Conference, FLAIRS 2014*, 01 2014, pp. 370–375.

[13] K. Kronander and A. Billard, "Learning Compliant Manipulation through Kinesthetic and Tactile Human-Robot Interaction," *IEEE Transactions on Haptics*, vol. 7, no. 3, pp. 367 – 380, 2014.

[14] A. Ude, B. Nemec, T. Petrič, and J. Morimoto, "Orientation in cartesian space dynamic movement primitives," in *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014, pp. 2997–3004.

[15] S. Calinon, *Robot Learning with Task-Parameterized Generative Models*. Cham: Springer International Publishing, 2018, pp. 111–126.

[16] A. Paraschos, E. Rueckert, J. Peters, and G. Neumann, "Model-Free Probabilistic Movement Primitives for Physical Interaction," *IROS*, pp. 2860–2866, 2015.

[17] D. A. Bristow and M. Tharayil, "A Survey of Iterative Learning Control - A learning-based method for high-performance tracking control," *IEEE Control Systems Magazine*, no. June, pp. 96–114, 2006.

[18] J. Peters, K. Mülling, and J. Kober, "Towards motor skill learning for robotics," *Robotics Research*, pp. 1–14, 2011.

[19] D. Lee and C. Ott, "Incremental Motion Primitive Learning by Physical Coaching Using Impedance Control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 4133–4140.

[20] B. Nemec, L. Zlajpah, S. Slajpah, J. Piskur, and A. Ude, "An Efficient PbD Framework for Fast Deployment of Bi-manual Assembly Tasks," in *18th IEEERAS International Conference on Humanoid Robots*, 2018.

[21] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Real-time robot learning with locally weighted statistical learning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation*, vol. 1, 2000, pp. 288–293.

[22] T. Petrič, A. Gams, L. Colasanto, A. J. Ijspeert, and A. Ude, "Accelerated sensorimotor learning of compliant movement primitives," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1636–1642, 2018.

[23] R. Ravani and A. Meghdari, "Velocity distribution profile for robot arm motion using rational Frenet-Serret curves," *Informatica*, vol. 17, no. 1, pp. 69–84, 2006.

[24] B. Nemec, N. Likar, A. Gams, and A. Ude, "Human robot cooperation with compliance adaptation along the motion trajectory," *Autonomous Robots*, vol. 42, no. 5, pp. 1023–1035, 2018.

[25] F. Stulp and O. Sigaud, "Many regression algorithms, one unified model: A review." *Neural networks : the official journal of the International Neural Network Society*, vol. 69, pp. 60–79, 2015.

[26] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, oct 2010.

[27] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally Weighted Learning," *Artificial Intelligence Review*, vol. 11, no. 1-5, pp. 11–73, 1997.

[28] K. Collins, A. J. Palmer, and K. Rathmill, "The development of a European benchmark for the comparison of assembly robot programming systems," in *Robot technology and applications*, K. Rathmill, P. MacConail, S. O?leary, and J. Browne, Eds. New York: Springer, 1985, pp. 187–199.

[29] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–73, 2013.

[30] B. Nemec, A. Gams, and A. Ude, "Velocity adaptation for self-improvement of skills learned from user demonstrations," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Atlanta, USA, 2013, pp. 423–428.