Transfer of Assembly Operations to New Workpiece Poses by Adaptation to the Desired Force Profile

Bojan Nemec, Fares J. Abu-Dakka, Barry Ridge, and Aleš Ude
Humanoid and Cognitive Robotics Lab
Jozef Stefan Institute, Ljubljana, Slovenia
email: bojan.nemec@ijs.si

Jimmy A. Jørgensen, Thiusius Rajeeth Savarimuthu, Jerome Jouffroy, Henrik G. Petersen, and Nobert Krüger
Cognitive Vision Lab
Maersk Mc-Kinney Moller Institute
University of Southern Denmark

***Abstract*—In this paper we propose a new algorithm that can be used for adaptation of robot trajectories in automated assembly tasks. Initial trajectories and forces are obtained by demonstration and iteratively adapted to specific environment configurations. The algorithm adapts Cartesian space trajectories to match the forces recorded during the human demonstration. Experimentally we show the effectiveness of our approach on learning of Peg-in-Hole (PiH) task. We performed our experiments on two different robotic platforms with workpieces of different shapes.**

I. INTRODUCTION

Many industrial manufacturing applications include assembly operations. One of the typical operations needed for the automatic assembly is peg insertion, often referred to as Peg-in-Hole task (PiH). Positioning inaccuracies and tight tolerances between the objects involved in PiH operations require some level of on-line adaptation of the programmed trajectories. In the literature, a number of approaches to solve the PiH problem were proposed. They were based on different strategies that used force feedback control [1], [2], [3], [4]. In general, the appropriate strategy depends on the workpiece geometry. This often requires an engineer to hard-code a different strategy for every new workpiece geometry [5], [6]. In this paper we propose a new approach to solve the PiH problem using Learning by Demonstration (LbD) paradigm [7]. Unlike standard LbD approaches, we acquire not only trajectories but also forces and torques arising during the task demonstration. This paradigm was for example utilized in [8], [9], where a haptic interface was used to demonstrate the initial trajectories. The appropriate force profile can also be learned by means of reinforcement learning [10]. Here we focus on teaching peg insertion tasks regardless of a workpiece geometry. We start with a single human demonstration where both the Cartesian space trajectory and the associated force / torque profile of the human execution of the peg-in-hole operation are recorded, see Fig. 1. When applying the recorded PiH operation in a new situation, e. g. when a workpiece is translated and rotated, the robot cannot simply replay the recorded trajectory due to noise in the estimated workpiece pose, uncertainties in the posture of the peg in the gripper, and due to a different joint space configuration of the robot arm. Since humans are very good at performing assembly tasks that require compliance and force control, we use human demonstrated force / torque profiles as reference. Hence, we are not trying to improve the human policy but rather match it with the robot. The developed approach enables the robot to adjust the demonstrated trajectory in few iterations so that the arising forces and torques during the execution of the PiH operation are similar to the ones recorded during human demonstration. Our experimental results show that the proposed approach is effective for learning of PiH operations for workpieces with different geometry.

The paper consists of five sections. In Section II we briefly present the data acquisition procedure used for acquiring trajectories and forces for an assembly operation. In Section III we present the policy learning and adaptation algorithm which uses the dynamic movement primitives (DMP) framework as the underlying representation of peg insertion trajectories. We propose an approach to iteratively adapt the learned trajectory to improve the task performance. This procedure exploits the properties of DMPs. Results of the experimental evaluation on two different platforms are given in Section IV, followed by final remarks and conclusion.

II. LEARNING OF THE PIH TASK BY HUMAN DEMONSTRATION

In this section we briefly describe the basic procedure for learning of PiH operations by human demonstration. The human demonstrations are performed in two different ways:

- Tele-operation using a TrackSTAR 6D pose tracker from Ascension attached to the object (Fig. 1 a).
- Kinesthetic guiding using the Kuka LWR arm in gravity compensation mode (Fig. 1 b).

In the first setup we use a magnetic tracker attached to the peg, which is held by a human demonstrator, to track the peg's position and orientation. The demonstrator executes the peg-in-hole task as usually. The measured poses are used to tele-operate the robot after being translated and rotated in such a way that the peg held by the robot is in the hole when the peg held by a human is in the hole. This is a constant transformation, which is estimated by the magnetic tracker in an initial calibration phase. The developed demonstration system is precise enough that the tele-operated robot successfully executes the task together with a human. The second setup uses the robot gravity compensation mode to implement kinesthetic guiding, where the human guides

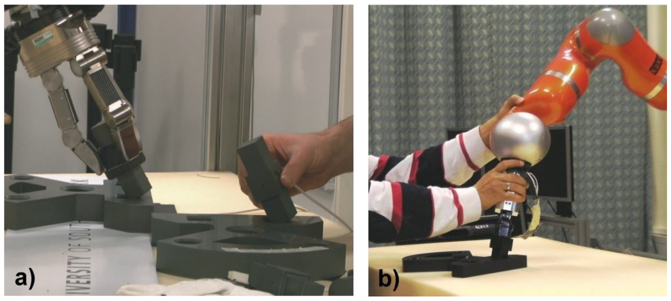

Fig. 1. a) The tele-operation setup with the position tracker inserted into the peg and the Universal Robot arm. b) Kinesthetic guiding using Kuka LWR arm in gravity compensation mode.

the robot's tool center point along the desired trajectory, for example so that the peg insertion task is successfully executed. We measure the Cartesian space trajectory by proprioception during the execution. In Kuka LWR arm, the force sensors are located in the robot joints and consequently the forces exerted from the human operator during the demonstration affect the measured forces and torques. Therefore, to get the net forces and torques at the robot tool center point, we repeat the measured trajectory with the robot, record the resulting joint torques, and transform them into the corresponding tool center point forces and torques. We take care that the workspace configuration does not change during this process and since our robot can accurately track the joint space trajectories, the measured forces and torques provide a good reference for adaptation later. The data acquired by both of our systems can further be improved by means of reinforcement learning [10], where a suitable cost function should be minimized. The cost function should relate to the desired properties of the task, e. g. smoothness of execution, execution time, etc. In our system the initial trajectories were good enough for successful peg insertion, thus refinement through reinforcement learning was not necessary.

With each successful execution of the PiH task, the system acquires the Cartesian space tool center point positions and orientations (represented as quaternions), velocities, and accelerations

$$\mathcal{M} = \{\mathbf{p}_j, \mathbf{q}_j, \dot{\mathbf{p}}_j, \omega_j, \ddot{\mathbf{p}}_j, \dot{\omega}_j\}, \tag{1}$$

and the associated forces and torques

$$\mathcal{F} = \{\mathbf{F}_j, \mathbf{M}_j\}, \tag{2}$$

all acquired at times t_j, $j = 0, \ldots, T$. Positions and orientations are represented as a frame $\mathcal{Q} = \{\mathbf{p}, \mathbf{q}\}$, where $\mathbf{p} = [x, y, z]$ is a 3-dimensional vector and $\mathbf{q} = (v, \mathbf{u})$ is a unit quaternion, respectively.

III. POLICY LEARNING AND ADAPTATION

The demonstrated Cartesian space trajectories that result in a successful execution of the PiH task are encoded by DMPs. A DMP for a single degree of freedom trajectory y is defined by the following set of nonlinear differential equations [11]

$$\begin{aligned} \tau \dot{z} &= \alpha_z(\beta_z(g - y) - z) + f(x), & (3)\\ \tau \dot{y} &= z, & (4)\\ \tau \dot{x} &= -\alpha_x x, & (5) \end{aligned}$$

where x is the phase variable and z is an auxiliary variable. Parameters α_z and β_z define the behavior of the second order system described by (3) and (4). With the choice $\tau = t_T > 0$, $\alpha_z = 4\beta_z > 0$ and $\alpha_x > 0$, the convergence of the underlying dynamic system to a unique attractor point at $y = g$, $z = 0$ is ensured [11]. $f(x)$ is defined as a linear combination of N nonlinear radial basis functions, which enables the robot to follow any smooth trajectory from the initial position y_0 to the final configuration g

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \tag{6}$$

$$\Psi_i(x) = \exp\left(-h_i\,(x - c_i)^2\right), \tag{7}$$

where c_i are the centers of radial basis functions distributed along the trajectory and h_i are their widths. For each Cartesian degree of freedom, the weights w_i and the goal g are estimated from the measured data (1) using regression in such a way that the resulting DMPs encode the desired peg insertion trajectory. Each position / orientation dimension is encoded as a single DMP with a common phase variable x. Since quaternions are calculated by integrating the above system, the integrated quaternion has to be normalized after each integration step. The exact solution is proposed in [12], but since we confirmed experimentally that the differences between both approaches are small, we use Eq. (3) – (4) also for quaternion integration. Such an approach simplifies implementation.

Since forces and toques are used only as desired variables along the trajectory and not as robot control variables, they do not need to be encoded by DMPs. Instead we use a linear combination of radial basis functions

$$F_{d,j}(x) = \frac{\sum_i w_i^{F,j} \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \tag{8}$$

$$M_{d,j}(x) = \frac{\sum_i w_i^{M,j} \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \tag{9}$$

$j = 1, \ldots, 3$, to approximate the desired forces and torques along the phase $x_i = x(t_i)$. We denote $\mathbf{F}_d = [F_{d,1}, F_{d,2}, F_{d,3}]^T$ and $\mathbf{M}_d = [M_{d,1}, M_{d,2}, M_{d,3}]^T$. Linear systems similar to (21) need to be solved in order to estimate $\mathbf{F}_d$ and $\mathbf{M}_d$ from the measured force / torque data (2).

When the robot executes the demonstrated trajectory, the resulting forces and torques differ from the ones that were measured during human demonstration. This happens due to small displacements arising from inaccurate pose estimation, reduced accuracy of the robot tracking control due to changes in the robot configuration (note that the original trajectory must be moved to a new configuration defined by a workpiece), and uncertainties in the placement of the peg in the gripper. This could worsen or even prevent the successful execution of the PiH task. In order to adapt to a new situation, we propose to modify the demonstrated trajectory according to the admittance (indirect compliance) control law [13]

$$\begin{aligned} \mathbf{p}_r(x) &= \mathbf{p}'_{DMP}(x) + \mathbf{K}_{s1}\mathbf{e}_p(x) + \o_p(x), & (10)\\ \mathbf{q}_r(x) &= \chi(\mathbf{K}_{s2}\mathbf{e}_q(x)) * \o_q(x) * \mathbf{q}'_{DMP}(x), & (11) \end{aligned}$$

where $\mathbf{p}_r(x)$ is the position vector fed to the robot controller, $(\mathbf{p}'_{DMP}, \mathbf{q}'_{DMP})$ is the displaced demonstrated trajectory computed by integrating the learned DMPs $\mathbf{p}_{DMP}$, $\mathbf{q}_{DMP}$ and applying the workpiece displacement $(\Delta\mathbf{t}_w, \Delta\mathbf{q}_w)$,

$$\begin{aligned}(0, \mathbf{p}'_{DMP}(x)) &= \Delta\mathbf{q}_w * (0, \mathbf{p}_{DMP}(x)) * \overline{\Delta\mathbf{q}}_w + (0, \Delta\mathbf{t}_w),\\ \mathbf{q}'_{DMP}(x) &= \Delta\mathbf{q}_w * \mathbf{q}_{DMP}(x).\end{aligned}$$

$\mathbf{K}_{s1}$ and $\mathbf{K}_{s2}$ are 3×3 diagonal matrices, $\mathbf{q}_r(x)$ is the orientation quaternion fed to the robot controller, and $\mathbf{q}_{DMP}(x)$ is the reference quaternion obtained by integrating the learned DMP. The quaternion product $*$, defined as $\mathbf{q}_1 * \mathbf{q}_2 = (v_1, \mathbf{u}_1) * (v_2, \mathbf{u}_2) = (v_1 v_2, v_2\mathbf{u}_1 + v_1\mathbf{u}_2 + \mathbf{u}_1 \times \mathbf{u}_2)$, has been used in (11). The feedback error term, i. e. $\mathbf{K}_{s1}\mathbf{e}_p(x)$ and $\chi(\mathbf{K}_{s2}\mathbf{e}_q(x))$, respectively, provides force / torque feedback control. Vectors $\boldsymbol{\o}_p(x)$ and $\boldsymbol{\o}_q(x)$ denote additional displacements and rotations, respectively, which are learned on line (see Section III-A). The idea is to move as much of the feedback error as possible to these displacement vectors. Initially, they are set to $\boldsymbol{\o}_p = [0,0,0]^{\mathrm{T}}$ and $\boldsymbol{\o}_q = (1,0,0,0)$. The errors $\mathbf{e}_p(x)$ and $\mathbf{e}_q(x)$ are defined as

$$(0, \mathbf{e}_p(x)) = \mathbf{q}(x) * (0, \mathbf{F}_d(x) - \mathbf{F}) * \overline{\mathbf{q}(x)}, \quad (12)$$

$$(0, \mathbf{e}_q(x)) = \mathbf{q}(x) * (0, \mathbf{M}_d(x) - \mathbf{M}) * \overline{\mathbf{q}(x)}, \quad (13)$$

where $\mathbf{F}_d(x)$ is the desired reference force at phase x as acquired from human demonstration, $\mathbf{F}$ the current measured force, $\mathbf{M}_d(x)$ is the desired reference torque, $\mathbf{M}$ the current measured torque, and $\mathbf{q}(x)$ the unit quaternion specifying the current tool orientation. χ denotes the transformation which maps an angular velocity to a unit quaternion describing the resulting rotation within the sampling time $\triangle t$

$$\chi(\omega) = \left(cos\frac{\|\omega\|\triangle t}{2}, \frac{\omega}{\|\omega\|} sin\frac{\|\omega\|\triangle t}{2} \right). \quad (14)$$

The proposed controller tracks simultaneously the desired position and orientations and forces and torques. Force / torque adaptation requires low gains (matrices $\mathbf{K}_s$) for stable and robust operation. Thus, force adaptation is usually slow. In order to effectively minimize the force / torque error, we propose slowing down the trajectory execution using DMP slow-down feedback. For DMP phase stopping [11], the original equation for phase (5) is replaced with

$$\tau\dot{x} = -\frac{\alpha_x x}{1 + \alpha_{px}\|\mathbf{e}\|}, \quad (15)$$

where $\mathbf{e}$ is the combined force / torque error

$$\|\mathbf{e}\| = \|[\mathbf{e}_p^T, \mathbf{e}_q^T]\|. \quad (16)$$

Note that in case of large forces or torques, the error $\|\mathbf{e}\|$ becomes large which in turn makes the phase change $\dot{x}$ small. Thus the phase evolution is stopped until the robot reduces the force / torque error. During the execution of the demonstrated trajectory, the resulting positions and orientations offsets are captured depending on the phase variable x, which ensures that the sampling is independent of the trajectory duration. Thus, the trajectory is sampled exactly the same number of times as during the human demonstration, even when the phase is slowed down during the execution according to (15).

A. *Offset Learning*

The goal of learning is to iteratively modify in sequential steps the positional and orientational part of the demonstrated trajectory so that the transformed trajectory results in similar forces and torques as during the human demonstration. The offset trajectory is updated at each iteration step as follows

$$\mathbf{u}^p_{j,l+1} = \boldsymbol{\o}_{p,l}(x_j) + \mathbf{K}_{s1}\mathbf{e}_p(x_j), \quad (17)$$

$$\mathbf{u}^q_{j,l+1} = \chi(\mathbf{K}_{s2}\mathbf{e}_q(x_j)) * \boldsymbol{\o}_{q,l}(x_j). \quad (18)$$

The updates are actually part of Eq. (10) and (11). Index l denotes the learning trial. Each component $\o_k$ of the offsets $\boldsymbol{\o}_{p,l}$ and $\boldsymbol{\o}_{q,l}$ of this newly sampled offset trajectory is represented as a linear combination of M radial basis functions (similarly as for DMPs in Eq. (6))

$$\o_k(x) = \frac{\sum_{i=1}^M w_{i,k}\Psi_i(x)}{\sum_{i=1}^M \Psi_i(x)} x. \quad (19)$$

The new data points $\{u^k_{j,l+1}\}$, $j = 0,\ldots,T$, are obtained from the k-th components of the offsets trajectories, where $k = 1,2,3$ denote the components of the positional part $\mathbf{u}^p_{j,l+1}$ and $k = 4,5,6,7$ the components of the quaternion part $\mathbf{u}^q_{j,l+1}$. The aim of optimization is to find weights $\{w_{i,k}\}$ that minimize the quadratic cost function

$$\sum_{j=0}^T (\o_k(x_j) - u^k_{j,l+1})^2. \quad (20)$$

The optimal weights $\mathbf{w}_k$ are then computed by solving the following linear system of equations

$$\mathbf{A}\mathbf{w}_k = \boldsymbol{\o}_k, \quad (21)$$

$$\mathbf{w}_k = \begin{bmatrix} w_{1,k} \\ \vdots \\ w_{M,k} \end{bmatrix}, \quad \boldsymbol{\o}_k = \begin{bmatrix} \o_k(x_0) \\ \vdots \\ \o_k(x_T) \end{bmatrix},$$

$$\mathbf{A} = \begin{bmatrix} \frac{\psi_1(x_0)x_0}{\sum_{i=1}^M \psi_i(x_0)} & \cdots & \frac{\psi_M(x_0)x_0}{\sum_{i=1}^M \psi_i(x_0)} \\ \vdots & \ddots & \vdots \\ \frac{\psi_1(x_T)x_T}{\sum_{i=1}^M \psi_i(x_T)} & \cdots & \frac{\psi_M(x_T)x_T}{\sum_{i=1}^M \psi_i(x_T)} \end{bmatrix}.$$

ψ_i are the Gaussian kernel functions from (7). It is also possible to compute a solution to (21) recursively by incrementally updating the following quantities

$$\mathbf{P}_j = \mathbf{P}_{j-1} - \frac{\mathbf{P}_{j-1}\mathbf{a}_j\mathbf{a}_j^{\mathbf{T}}\mathbf{P}_{j-1}}{1 + \mathbf{a}_j^{\mathbf{T}}\mathbf{P}_{j-1}\mathbf{a}_j}, \quad (22)$$

$$\mathbf{w}_j = \mathbf{w}_{j-1} + (f(t_j) - \mathbf{a}_j^{\mathbf{T}}\mathbf{w}_{j-1})\mathbf{P}_j\mathbf{a}_j, \quad (23)$$

where $\mathbf{a}_j$ is the $T+1$ dimensional column vector associated with the corresponding row of the matrix $\mathbf{A}$ and the optimal weights are $\mathbf{w} = \mathbf{w}_{T+1}$. Index j denotes the j-th phase sample. In the latter case we avoid saving the complete offset trajectory during the adaptation. Note that – similar to the DMP integration – the quaternion part of the offset trajectory $\boldsymbol{\o}_q$ has to be normalized before being used in Eq. (11).

B. *Control Scheme*

Fig. 2 shows the implementation scheme for the proposed learning algorithm. In this scheme, $\mathcal{M}_d$ denotes the original demonstrated trajectory, which is encoded by DMPs. $\mathcal{Q}_d$ refers to the output signal obtained by integrating (3) – (5) and applying the workpiece displacement $(\Delta\mathbf{t}_w, \Delta\mathbf{q}_w)$ as estimated by vision, which occurs in block $\mathbf{T}_1$. $\mathcal{F}_d$ denotes the forces and torques captured during the execution of the demonstrated trajectory and $\mathcal{F}_d$ calculates the output by evaluating (8) and (9). Due to the noise induced by the errors in the vision system, robot tracking errors, and imperfect grasping, the measured forces/torques $\mathbf{F}, \mathbf{M}$ differ from the desired demonstrated forces/torques $\mathcal{F}_d$. Our goal is to minimize this difference. During the execution, the measured forces/torques $\mathbf{F}, \mathbf{M}$ are compared to the desired demonstrated forces/torques $\mathcal{F}_d$. In $\mathbf{T}_2$ this error is transformed into robot base coordinates. Using admittance control (10) – (11), the position / orientation offset is calculated and added to the existing offset from the previous iteration step. This offset is computed using function approximation in the block $\o(x) = \{\o_p(x), \o_q(x)\}$. Motor commands $\mathcal{Q}_c$ are given as the aggregation of the DMP generated trajectory, force feedback (10) – (11) and the offset learned in several iterations (17) – (18). The whole procedure is repeated until the measured forces match the desired forces. This algorithm belongs to a class of Iterative Learning Control, where we applied current iteration causal learning [14], [15].

Special attention should be paid to the proper selection of the gain $\mathbf{K}_{s1}$ and $\mathbf{K}_{s2}$. The admittance control law as defined by (10) and (11) modifies positions / orientations according to the force error signal and the selected gains $\mathbf{K}_{s1}$ and $\mathbf{K}_{s2}$. Higher gains result in faster updates, but cause chattering around the set point $\mathbf{e}_p = 0$ and $\mathbf{e}_q = 0$. We propose to apply nonlinear gains in order to combine the benefits of faster updating and prevent the chattering around the set point

$$\mathbf{K}_{s1} = K_{s1_0}\left(1 - \exp\left(-\frac{\|\mathbf{e}_p\|^2}{\sigma_1}\right)\right)\mathbf{I}, \tag{24}$$

$$\mathbf{K}_{s2} = K_{s2_0}\left(1 - \exp\left(-\frac{\|\mathbf{e}_q\|^2}{\sigma_2}\right)\right)\mathbf{I}. \tag{25}$$

Parameters K_{s1_0} and K_{s2_0} are appropriately chosen for stable and repulsive behavior at large force / torque error signals. Parameters σ_1 and σ_2 define the range, where the gains drop to zero in order to prevent chattering. In our approach, the originally demonstrated trajectory is always preserved and we learn offsets to this trajectory rather than modifying the trajectory itself. In such a case, the offsets can be easily reset when we encounter new situations. Another benefit is the improved stability of the overall learning control scheme. Namely, it was found that the repeated learning and execution of the same signals with a DMP is subject to an exponentially growing bias, which is caused by the discrete implementation of the DMP integration. In our algorithm, the learned offset is encoded as a linear combination of Gaussian kernel functions, which does not involve the integration and therefore is not subject to this bias.

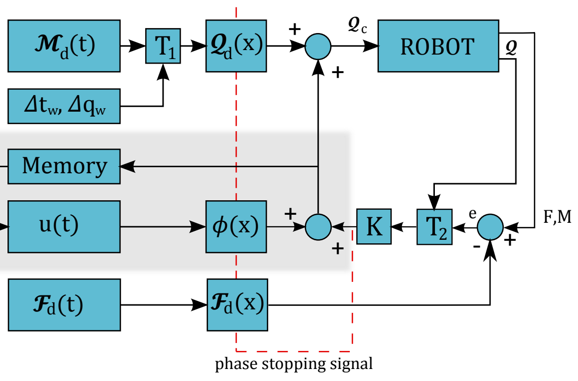

Fig. 2. The learning scheme

IV. EXPERIMENTAL EVALUATION

The proposed LbD procedure was implemented on two different robot platform shown in Fig. 1 a and b, respectively:

- A 6 D.O.F Universal Robot arm - type UR5 - equipped with a SCHUNK SDH gripper and a force/torque wrist sensor. This robot uses high gain non-compliant controller. Therefore, we implemented admittance force control law (Eq. 10 and 11).
- A 7 D.O.F. Kuka LWR robot equipped with a two-finger gripper. This robot is capable of measuring joint torques and can switch to Cartesian compliance control.

The task was the insertion of the square and round peg into a hole of the Cranfield benchmark. Trajectories were obtained using tele-operation and kinesthetic guiding as described in Section. II. We executed the trajectory for different randomly selected positions of the base plate. The measured trajectory data was translated and rotated into new configurations using the pose of the base plate as estimated by vision. In this scenario the uncertainties come from vision. Therefore, the resulting forces during the execution of the demonstrated trajectory may exceed the expected ones.

A. Experimenting with UR5 platform

The learning procedure was implemented in C++ and linked to the robot using ROS [16]. The results for square peg insertion are shown in Fig. 3 – 5. Fig. 3 shows the learned offsets of the positional part of the trajectory in six consecutive

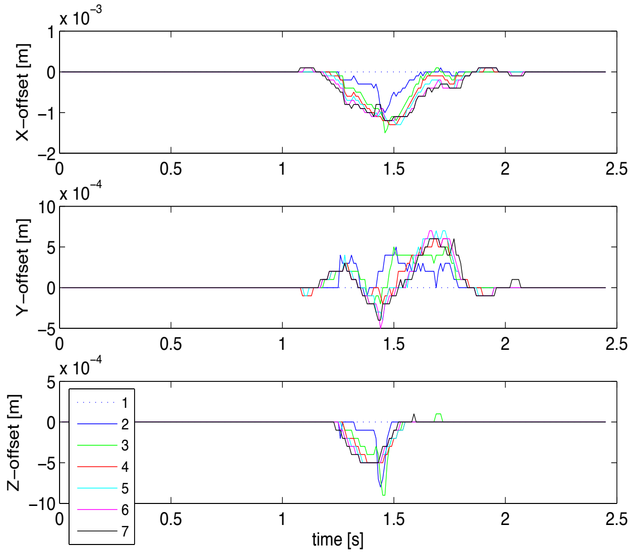

Fig. 3. Positional part of the learned offset in 6 consecutive cycles. Cycle 1 denotes the execution of the demonstrated trajectory (zero offset).

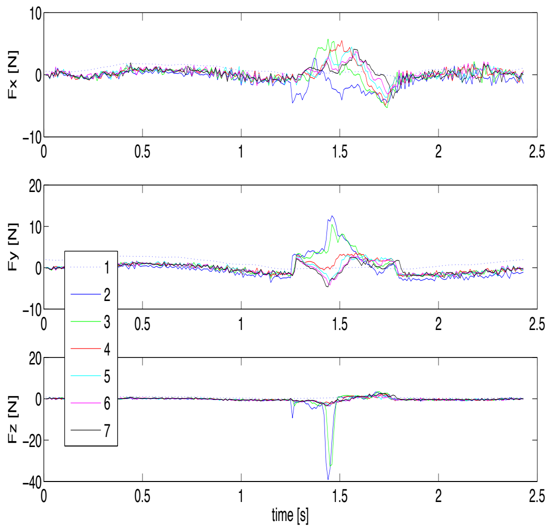

Fig. 4. Forces measured during the execution of the square PiH task. Dashed lines (cycle 1) correspond to the original DMP trajectory and solid lines to the adjusted trajectories in 6 consecutive learning cycles.

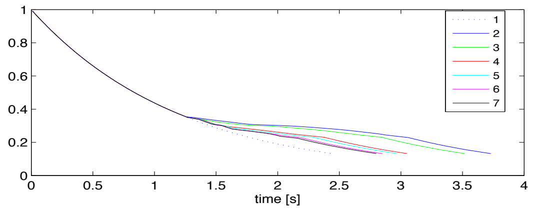

Fig. 5. Phase evolution during the execution of the square PiH task. Dashed line corresponds to an ideal phase without DMP stopping actions and solid lines to the actual phase in 6 consecutive cycles.

cycles. Fig. 4 shows the forces and torques that result from the execution of the original (demonstrated) trajectories and the adjusted (learned) trajectories. Note that the adjusted trajectory results in generally lower forces and torques. The constant offset in the force profile is due to imperfect force sensor calibration. Fig. 5 shows the phase evolution during learning. The phase measured during human demonstration is shown as dotted line. Whenever the difference between the measured and the desired forces became large, the phase was slowed down due to the force / torque error in Eq. (15) until the force controller sufficiently reduced it. Our learning procedure reduces the differences between the desired and measured forces and torques. This makes phase stopping less frequent and consequently the execution time decreases in each learning cycle as seen in Fig. 5. Note that in all plots time is normalized to the equal duration in order to compare the adaptation of the proposed algorithm trough iterations. Next experiment was similar to the previous one, except that we used another demonstrated trajectory and that the base plate was translated and rotated to another position. As can be seen in Fig. 6 – 8, our algorithm can deal with such changes. Due to the vision errors – especially at the beginning of learning – the execution was much slower, but after learning it becomes comparable to the first experiment where the base plate was not moved. In

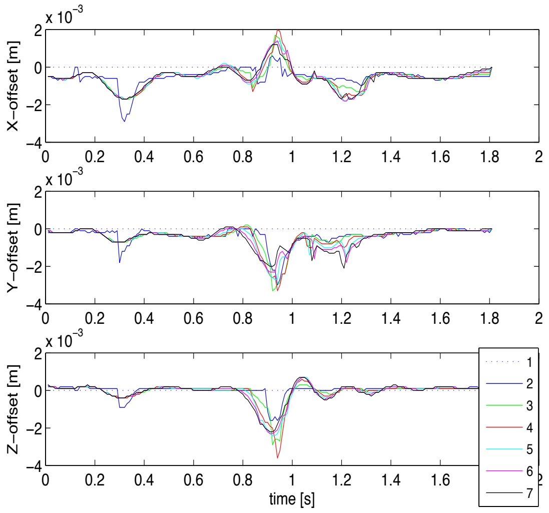

Fig. 6. The learned offset in 6 consecutive cycles. Cycle 1 denotes zero offset.

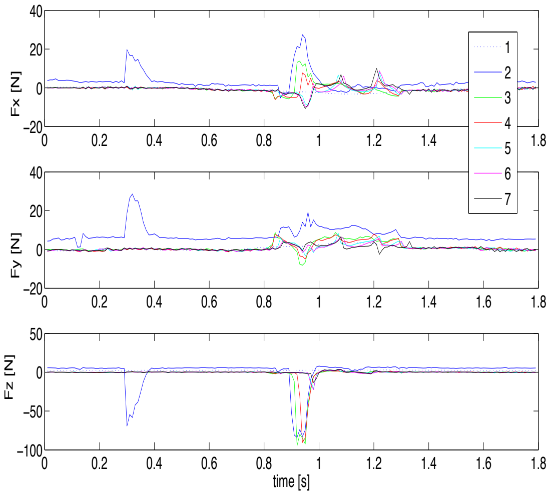

Fig. 7. Forces measured during the execution of the square PiH task in 6 consecutive learning cycles.

order to demonstrate that our algorithm is not dependent on the shape of the workpiece, we repeated the same experiment using round peg. The results shown in Figs. 9 – 11 demonstrate that the algorithm works equally well for round peg and is therefore suitable for different peg shapes.

B. *Experimenting with Kuka*

Additional experiments were conducted on Kuka LWR arms. The learning procedure was implemented in Matlab, which communicated with the Kuka LWR controller using Fast Research Interface [17]. Figs. 12 – 14 show the experimental results of the algorithm on the Kuka robot. The obtained results

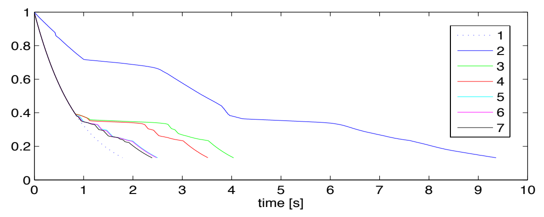

Fig. 8. Phase evolution during the execution of the square PiH task 6 consecutive cycles.

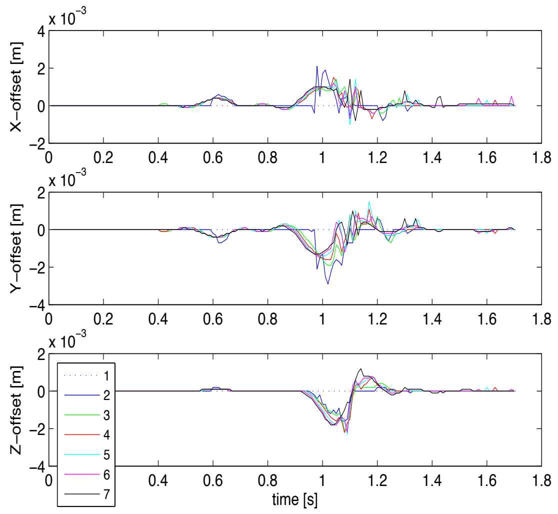

Fig. 9. Positional part of the learned offset in 6 consecutive cycles.

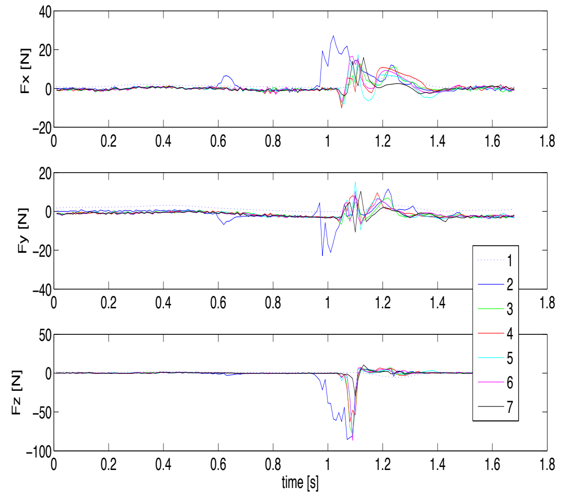

Fig. 10. Forces measured during the execution of the round PiH task in 6 consecutive learning cycles.

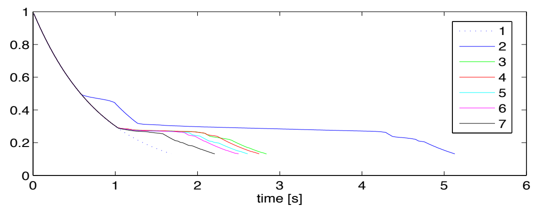

Fig. 11. Phase evolution during the execution of the round PiH task in 6 consecutive cycles.

are comparable with the results obtained with the UR5, but the insertion is generally smoother and more robust due to the Cartesian impedance control.

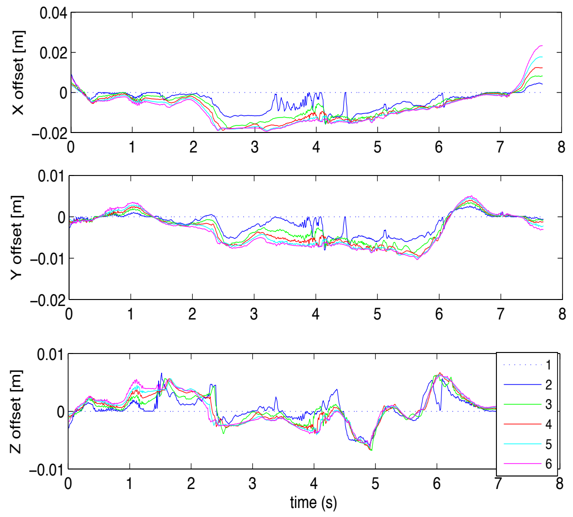

Fig. 12. Positional part of the learned offset in 5 consecutive cycles. Cycle 1 denotes the execution of the demonstrated trajectory (zero offset).

V. CONCLUSION

We proposed a new approach for learning by demonstration, which considers both positions and orientations and forces and torques arising during a human demonstration. We focused on tasks that require accurate force control, such as for example peg-in-hole insertion. Direct replication of the demonstrated trajectories leads to suboptimal performance due to misalignments caused by noise, which can arise due to the inaccuracies of the vision system, due to uncertainties in the gripping pose, and due to the Cartesian space robot tracking error caused by a different joint space configuration. Since humans are very good at performing tasks that require compliance and force control, we utilize human demonstration to obtain reference force / torque profiles for the PiH task. During task execution, the robot attempts to replicate the learned forces and torques rather than positions and orientations. Our learning procedure modifies the demonstrated trajectory in such a way that the resulting forces and torques match the demonstrated ones. The adaptation to the desired forces can be accomplished using either admittance or impedance control law, where the robot motion is modified to reduce the force / torque error. Since force adaptation is usually slow (due to

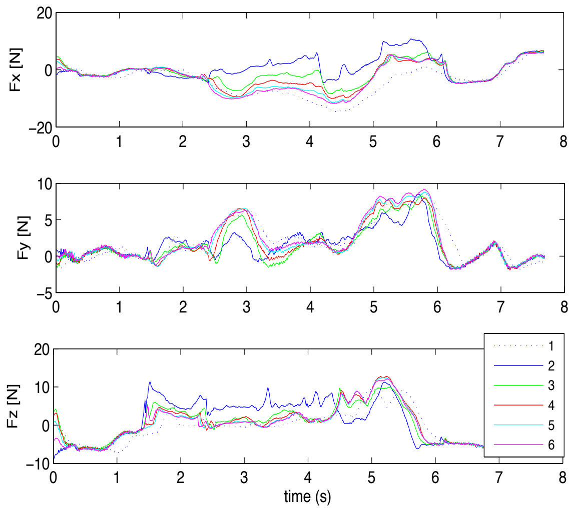

Fig. 13. Forces measured during the execution of the square PiH task. Dashed lines (cycle 1) correspond to the original DMP trajectory and solid lines to the adjusted trajectories in 5 consecutive learning cycles.

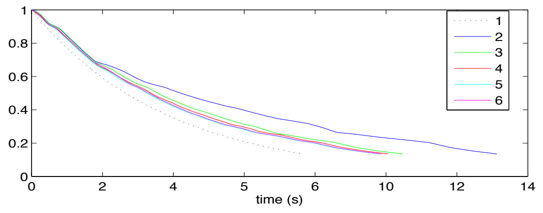

Fig. 14. Phase evolution during the execution of the square PiH task. Dashed line corresponds to an ideal phase without DMP stopping actions and solid lines to the actual phase in 5 consecutive cycles.

the low force / torque gains, which are required for stable operation), we slow down the trajectory evolution whenever the force / torque error becomes large. This is accomplished using the DMP stopping technique. During the execution we capture the offset to the demonstrated trajectory caused by the feedback controller and this offset is added to the demonstrated trajectory in the next learning cycle. In most cases we can effectively execute the task after a few iterations. The offset trajectory is encoded with Gaussian kernel functions and estimated as described in Section III-A. Experimental results demonstrate robustness and fast adaptation capability of the proposed algorithm on two different robot platforms. One of the major advantages of the proposed approach is that it is independent on the workpiece geometry, which is not the case in many industrial implementations of the PiH task.

Video of our experiments is available at `http://www.ijs.si/usr/nemec/Video-ICAR.m4v` The first part of the video shows a human demonstration of the peg-in-hole operation in the tele-operation setup. The second part shows the round and square peg insertion on both robots. Note that the insertion time decreases significantly after learning.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 269959, IntellAct.

References

[1] S.-K. Yun, "Compliant manipulation for peg-in-hole: Is passive compliance a key to learn contact motion?" in *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, California, 2008, pp. 1647–1652.

[2] K. Hirana, T. Suzuki, and S. Okuma, "Optimal motion planning for assembly skill based on mixed logical dynamical system," in *7th International Workshop on Advanced Motion Control*, Maribor, Slovenia, 2002, pp. 359–364.

[3] P. R. Giordano, A. Stemmer, K. Arbter, and A. Albu-Schaffer, "Robotic assembly of complex planar parts: An experimental evaluation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, 2008, pp. 3775–3782.

[4] W. S. Newman, M. S. Branicky, H. A. Podgurski, S. Chhatpar, L. Huang, J. Swaminathan, and H. Zhang, "Force-responsive robotic assembly of transmission components," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, Detroit, Michigan, 1999, pp. 2096–2102.

[5] H. Bruyninckx, S. Dutre, and J. De Schutter, "Peg-on-hole: a model based solution to peg and hole alignment," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, Nagoya, Japan, 1995, pp. 1919–1924.

[6] J. Xiao, "Goal-contact relaxation graphs for contact-based fine motion planning," in *IEEE International Symposium on Assembly and Task Planning (ISATP)*, Marina del Rey, California, 1997, pp. 25–30.

[7] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, 2004.

[8] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Advanced Robotics*, vol. 25, no. 5, pp. 581–603, 2011.

[9] L. Rozo, P. Jiménez, and C. Torras, "A robot learning from demonstration framework to perform force-based manipulation tasks," *Inteligent Service Robotics*, vol. 6, pp. 33–51, 2013.

[10] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning force control policies for compliant manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, 2011, pp. 4639–4644.

[11] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control – a unifying view," *Progress in Brain Research*, vol. 165, no. 6, pp. 425–445, 2007.

[12] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, California, 2011, pp. 365–371.

[13] L. Villani and J. De Schutter, "Force control," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2008, pp. 161–185.

[14] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *Control Systems, IEEE*, vol. 26, no. 3, pp. 96 – 114, june 2006.

[15] K. Moore, Y. Chen, and H.-S. Ahn, "Iterative learning control: A tutorial and big picture view," in *Decision and Control, 2006 45th IEEE Conference on*, dec. 2006, pp. 2352 –2357.

[16] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibsz, E. Bergery, R. Wheeler, and A. Ng, "**ROS:** an open-source robot operating system," in *ICRA Workshop on Open Source Software*, Kobe, Japan, 2009.

[17] G. Schreiber, A. Stemmer, and R. Bischoff, "The fast research interface for the KUKA lightweight robot," in *ICRA Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications – How to Modify and Enhance Commercial Controllers*, Anchorage, Alaska, 2010.