2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

A Shared Control Method for Online Human-in-the-Loop Robot Learning Based on Locally Weighted Regression

Luka Peternel[a,c], Erhan Oztop[b] and Jan Babič[c]

Abstract— **We propose a novel method that arbitrates the control between the human and the robot actors in a teaching-by-demonstration setting to form synergy between the two and facilitate effective skill synthesis on the robot. We employed the human-in-the-loop teaching paradigm to teleoperate and demonstrate a complex task execution to the robot in real-time. As the human guides the robot to perform the task, the robot obtains the skill online during the demonstration. To encode the robotic skill we employed Locally Weighted Regression that fits local models to specific state region of the task based on the human demonstration. If the robot is in the state region where no local models exist, the control over the robotic mechanism is given to the human to perform the teaching. When local models are gradually obtained in that region, the control is given to the robot so that the human can examine its performance already during the demonstration stage, and take actions accordingly. This enables a co-adaptation between the agents and contributes to a faster and more efficient teaching. As a proof-of-concept, we realised the proposed robot teaching system on a haptic robot with the task of generation of a desired vertical force on a horizontal plane with unknown stiffness properties.**

I. Introduction

Robot learning can be a good alternative to time-consuming classical manual programming that requires experts and expert time. One such example is reinforcement learning where the robot autonomously learns the task through the trial-and-error principle [1]. Another example is learning-by-demonstration where the human directly demonstrates the robot how to perform a given task [2]–[9].

Robot learning-by-demonstration can be generally divided into offline and online type of learning [10]. In offline learning the training data is acquired in the demonstration stage, and subsequently the training data is used in combination with a machine-learning method to form a robotic skill, which can be later used by the robot in the autonomous stage [2]–[4], [11]. The advantage of this approach is that the demonstrator can select the best data or adjust it in order to achieve the best learning performance. The key drawback of pure offline learning in the teaching-by-demonstration framework is that the human demonstrator does not have any feedback about the robot learning *during* the demonstration stage. The actual performance of the obtained robotic skill by using the data collected during demonstration can only be observed later in the autonomous stage. If the autonomous performance is unsatisfactory then the whole procedure has to be repeated.

Above-mentioned drawback can partially be addressed by multiple demonstrations to obtain some variety that can benefit the obtained robotic skill during the autonomous stage [11]. Alternatively, the skill obtained during the demonstration stage can be corrected by the human in the autonomous stage through social interaction [12]–[14]. However, this method requires relatively high level of robot intelligence to recognise the human advice and devise a strategy to correct its performance on sensorimotor level. If the task is complex, the correction procedure may be time-consuming and in some cases unsuccessful. Therefore, another direct sensorimotor-level demonstration from the human could be faster and more efficient.

Contrary to offline learning, in online learning the robotic skill is formed gradually already during the demonstration stage [10]. One of the advantages of this is that the transition between demonstration stage and autonomous stage can be direct and automatic [7]–[9]. Importantly, online learning also allows the demonstrator to get useful feedback about the performance of the robotic skill being constructed during the demonstration [7], [15], [16]. In this case, the correction of the obtained skill can be performed online and directly on sensorimotor level [15], [16], and it can therefore be faster and less complicated from the user perspective.

In [15], the authors proposed a kinaesthetic teaching method that allows the demonstrator to incrementally improve the the currently leant robotic skill. The skill refinement is done by physical interaction at low robot stiffness. However, in kinaesthetic teaching the immersion of demonstrator is limited, especially with regards to tactile sensing. In addition, simultaneous online demonstration of impedance [9], [17] can be difficult. These drawbacks can be addressed by teleoperation-based teaching [3], [7], [9]. However, in this approach the control commands for the robotic mechanism are simultaneously generated by both the human demonstrator and by the robot control policy (represented as a non-linear function approximator) being learnt [7]. Therefore, a mechanism that arbitrates the control responsibility over the target robotic platform is required to enable the co-adaptation between the involved agents.

II. Related Work

While shared control is a well-studied subject in teleoperation [18], [19] and assistive robotics [16], [19]–[21], there are only a few studies that investigated shared control on

[a]HRI[2] Lab, Dept. of Advanced Robotics, Istituto Italiano di Tecnologia, Genoa, Italy, Email: luka.peternel@iit.it
[b]Robotics Laboratory, Özyeğin University, Istanbul, Turkey
[c]Dept. of Automation Biocybernetics and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia

This work was supported by the EC Framework Programme 7 through the CoDyCo project (#600716) and the Converge project (#321700).

a sensorimotor level in the teleoperation-based human-in-the-loop robot teaching. Study in [7] proposed a method to share the control between the demonstrator and robotic skill in an online robot learning scenario based on Locally Weighted Regression (LWR) [22], [23] machine learning. The control-sharing algorithm defined the amount of control responsibility of each agent as a ratio based on the prediction error, i.e. the average error between the human generated motor output and LWR output. The reasoning was that when the machine-learning algorithm learns to replicate human generated motor commands reliably then the control could be shifted to the machine control. This works very well when the demonstrator quickly becomes an expert in performing the task at hand [7].

Study in [24] developed a variant of the approach proposed in [7] by designing a new strategy for determining the control arbitration. Instead of using prediction error to adjust the control sharing, they used a local success measure of the task execution, which does not directly depend on the prediction error of the machine-learning module. This is beneficial for tasks where the human demonstrator needs time to adapt to task to perform well; because this does not allow the machine to learn a bad policy by simply becoming a very good imitator of a bad demonstrator. The local success measure is somewhat analogous to introducing virtual rewards to speed-up learning in the reinforcement-learning framework. The disadvantage of this approach is the additional complexity and task dependency. For a useful local success measure the task must be well defined and should naturally facilitate local success definition, by for example, allowing state space decomposition into subtasks. Naturally, some tasks do not meet these conditions, and for those that meet, the one-time expert effort might be non-negligible.

The two approaches in [7] and [24] described above both advocate a control sharing mechanism between the agents. This means the performance of the robot during demonstration is a mixture of human and machine control. Here we have two potentials difficulties for the human demonstrator. (1) The credit assignment problem: it may be difficult to determine whether the current unsatisfactory performance of the robot is due to the control commands from the demonstrator or the machine-learning module. (2) The non-stationarity problem: how the robot responds to the human control commands that change with time and in a complex way; this must be an initial burden for human demonstrators [24].

In this paper we propose and investigate a new shared-control approach for teleoperation-based human-in-the-loop online robot teaching that addresses the two difficulties indicated above. Contrary to [7] and [24], we adopt a binary control responsibility scheme where the control is either fully delegated to the demonstrator or to the machine-learning module. This allows the demonstrator to clearly know who is responsible for the ongoing task execution performance. In addition, the approach retains the simplicity from the user perspective [7] as it does not require the task and reward function to be defined in advance. In [7] the control is

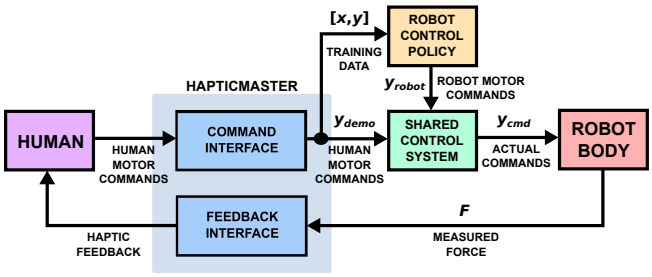

Fig. 1. The proposed human-in-the-loop teaching approach with shared control between the actions of demonstrator (y_{demo}) and robotic skill (y_{robot}). HapticMaster robot serves as an interface between the demonstrator and the controlled robotic manipulator.

delegated gradually based on the average error between the human and robotic skill performance over the entire state space up until the current observation time. The disadvantage of this is that human cannot efficiently inspect the robotic skill performance locally in a specific state region. Contrary to that, in the proposed method the control is delegated based on the existence of local models in specific state regions. This gives the demonstrator an option to freely inspect the performance of the robotic skill in any specific state region (and if necessary to correct it).

III. Methods

In the proposed approach, the control sharing is based on a mechanism that exploits the machine-learning method (LWR) adopted [22], [23]. LWR builds a non-linear model by creating local linear models over non-linear basis functions that have local span (i.e. receptive fields) over the states and are incrementally generated during task demonstration. The existence or nonexistence of local models (in the LWR module) in a given state region is used to arbitrate control: when the robot finds itself in a state region where no local models exist the control is given to the human demonstrator so that new models can be learnt. If the current state of the robot is within the proximity of a local model (i.e. it activates the receptive field of some model) then the control is given to the machine-learning module. This allows the human to examine the performance of the current robotic skill during the demonstration stage. If the performance is unsatisfactory the teaching can be repeated and models can be updated/replaced in the given state region.

To demonstrate and validate the concept of the proposed method we performed experiments on a Moog HapticMaster (HM) robot. The task of the robotic manipulator was to produce a reference force between its end-effector and unknown objects in the environment. The surfaces of the objects had different stiffness. The task of the human demonstrator was to teach the robotic manipulator how to adjust the commanded (reference) position in order to maintain the predefined force over the entire operational space (plane).

The human demonstrator was included into the robot control loop by a human-robot interface as shown in Fig. 1. The HM robot served as the necessary control interface for the virtual robotic manipulator that needs to be manoeuvred

to achieve a constant vertical force feedback over an environment that has non-homogenous stiffness. While operating the robot trough this teleoperation setup, the human demonstrator could easily gain the skill to perform the task through the robot. After an initial practice session, the skill demonstration stage was started where the human demonstrator taught the robot how to perform the desired task. Simultaneously, the robot acquired the skill incrementally in real-time as detailed below. The proposed control arbitration mechanism delegated the control to the demonstrator or to the so-far-built robot control policy (robotic skill). The control policy learning was inhibited for the state regions where sufficient level of learning was already attained, and the robot was commanded using this learnt policy. In contrast, when the robot entered a state region where sufficient learning did not take place, the control was granted to the demonstrator and the learning resumed.

A. Locally Weighted Regression

The LWR method[1] is an incremental (online) function approximator that can be used to represent *output* as a function of *input* given a set of [input, output] pairs, i.e. the training data points. To do this, it forms local expert models that are responsible for particular input regions. Each local expert is a linear model and is assumed to represent its responsibility area. The output prediction $\hat{\boldsymbol{y}}(\boldsymbol{x})$, for some new input $\boldsymbol{x}$, is given by the sum of contributions of all local models [22]

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \frac{\sum_{i=1}^{N} w_i \boldsymbol{y}_i^{model}}{\sum_{i=1}^{N} w_i}, \tag{1}$$

where $\boldsymbol{y}_i^{model}$ is the prediction of the i^{th} local model, and w_i its weight. Weights determine the contribution of each local model in the final prediction, $\hat{\boldsymbol{y}}$ given input $\boldsymbol{x}$. Weights are computed by the activation of models' receptive fields defined by Gaussian kernel functions [22]

$$w_i = \exp\big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{c}_i)^T \boldsymbol{D}_i (\boldsymbol{x}-\boldsymbol{c}_i)\big), \tag{2}$$

where $\boldsymbol{D}_i$ is positive definite matrix that contains information about the size of receptive field. Parameter $\boldsymbol{c}_i$ represents the centre of the receptive field and its local model. By observing (2) one can notice that the influence (weight) of a local model $\boldsymbol{y}_i^{model}$ decreases with the distance of input $\boldsymbol{x}$ from the centre $\boldsymbol{c}_i$ of that model.

Local linear models are defined as [22]

$$\boldsymbol{y}_i^{model} = (\boldsymbol{x}-\boldsymbol{c}_i)^T \boldsymbol{m}_i + m_{0,i} = \tilde{\boldsymbol{x}}_i^T \boldsymbol{M}_i, \tag{3}$$

$$\tilde{\boldsymbol{x}} = [(\boldsymbol{x}-\boldsymbol{c}_i)^T, 1]^T, \tag{4}$$

where $\boldsymbol{y}_i^{model}$ is the i-th model prediction given the input $\boldsymbol{x}$, while $\boldsymbol{c}_i$ is the centre of the i^{th} model. Vector $\boldsymbol{M}_i$ contains the parameters that describes the local model and $\tilde{\boldsymbol{x}}$ is the distance from the centre of the model.

[1]LWR is used here for generality. Locally Weight Projection Regression [23] can be used in our method instead to cope with higher dimensionality.

The learning process needs to determine model parameters $\boldsymbol{M}_i$, centre parameters $\boldsymbol{c}_i$ and distance metrics $\boldsymbol{D}_i$ that determines the size of the model's receptive field. The learning is done incrementally while the training data are fed to the learning system. Each training data point $[\boldsymbol{x}, \boldsymbol{y}]$ updates the existing models via recursive least-square method [22]

$$\boldsymbol{M}(k+1) = \boldsymbol{M}(k) + w\boldsymbol{P}(k+1)\tilde{\boldsymbol{x}}\boldsymbol{e}_{lwr}^T, \tag{5}$$

$$\boldsymbol{e}_{lwr} = (\boldsymbol{y} - \boldsymbol{M}(k)^T \tilde{\boldsymbol{x}}), \tag{6}$$

$$\boldsymbol{P}(k+1) = \frac{1}{\lambda}\left(\boldsymbol{P}(k) - \frac{\boldsymbol{P}(k)\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T\boldsymbol{P}(k)}{\frac{\lambda}{w} + \tilde{\boldsymbol{x}}^T\boldsymbol{P}(k)\tilde{\boldsymbol{x}}}\right), \tag{7}$$

where λ forgetting factor that allows the system to gradually forget the previously demonstrated training data points.

Adjustment of model's receptive field size is performed by a modified leave-one-out cross-validation method [22]. As new training data points are used for the LWR update, new local models are created - and in same cases removed, to facilitate a parsimonious distribution of the local models across the input space. A new model is created when a training point $[\boldsymbol{x}, \boldsymbol{y}]$ does not active any of the receptive fields more than a threshold w_{gen} [22], and the centre of the newly created model is set to the position of the new training point ($\boldsymbol{c} = \boldsymbol{x}$). When a training point activates two receptive fields simultaneously for more than a predetermined threshold w_{prune}, then the model with the larger covariance matrix is removed [22]. This prevents unnecessary overlap of the models that would lead to computational inefficiency.

B. Shared-Control Approach

We delegated the control over the robotic mechanism between the human and robotic skill by the following expression [7] (see also Fig. 1)

$$\boldsymbol{y}_{cmd} = C \cdot \boldsymbol{y}_{demo} + (1-C) \cdot \boldsymbol{y}_{robot}, \tag{8}$$

where $\boldsymbol{y}_{cmd}$ is a vector of commands sent to the robotic mechanism, $\boldsymbol{y}_{demo}$ is a vector of commands given by the human demonstrator, $\boldsymbol{y}_{robot}$ is a vector of commands produced by the current robotic skill and $C \in [0,1]$ is a weight factor that determines the influence of each agent. Pure human control is achieved when $C = 1$, and pure machine control is achieved when $C = 0$.

In our method, we used the information about the current state of the local models in LWR to determine the value of the factor C. When the demonstrator is performing the task in a state (i.e. position on the plane) where a feasible local model exists the task execution is purely based on the current robot control policy that is represented by (1), and learning is inhibited. If no local model is present in the current robot state then the control of the task execution is given to the demonstrator, and teaching is resumed. To determine the proximity to the local model we used the activation w_{max} of the model that provides the maximum activation for the current position of the input vector $\boldsymbol{x}$. We defined factor C

as

$$C = \begin{cases} 1 & \text{if } w_{max} < (w_{th} - \frac{d}{2}) \\ \frac{1+\cos\left(\pi \frac{w_{max} - (w_{th} - \frac{d}{2})}{d}\right)}{2} & \text{if } (w_{th} - \frac{d}{2}) \leq w_{max} \leq (w_{th} + \frac{d}{2}), \\ 0 & \text{if } w_{max} > (w_{th} + \frac{d}{2}) \end{cases} \tag{9}$$

where w_{max} is the strongest activation of the model receptive fields [22] and w_{th} is an activation threshold that we introduced to determine the model-proximity based shared control. Although, the main regime for C is binary, we used a cosine based function to prevent sudden jumps in net control output and to facilitate smooth transitions between human control and machine control modes of operation. Here the parameter d defines the width of switching function, and thus should be assigned a small value to obtain the desired binary regime.

During the demonstration stage the training data is collected and accumulated for a predefined sample length before it is fed to the LWR-based learning system. Instead of feeding the data to LWR in the order it is received, a random order can improve the learning at lower λ. Data accumulation takes place only when the human demonstrator has control over the robot (i.e. control factor $C = 1$). When appropriate LWR models exist in a given state region, the robot behaves autonomously by following the control policy dictated by the active LWR models (i.e. control factor $C = 0$). The data accumulation is paused in this case, until when the user gains full control again. The described approach can be described as

$$\boldsymbol{A}_{new} = \begin{cases} \boldsymbol{A} & \text{if } C \neq 1 \\ [\boldsymbol{A}; [\boldsymbol{x}, \boldsymbol{y}]] & \text{if } C = 1 \ \& \ \text{length}(\boldsymbol{A}) < N_{acc}, \\ [\,] & \text{if } \text{length}(\boldsymbol{A}) = N_{acc} \end{cases} \tag{10}$$

$$\boldsymbol{A}_{in} = \boldsymbol{A}\big(\text{randperm}(N_{acc}), :\big), \tag{11}$$

where $\boldsymbol{A}$ is the matrix (accumulation buffer) where the training data is accumulated, and the notation $\boldsymbol{A}_{new}$ indicates that $\boldsymbol{A}$ is updated at each iteration. N_{acc} is the predefined accumulation sample length that determines the amount of data that is fed to LWR in each update epoch. As mentioned above, a shuffling is applied before the data is fed to LWR, so instead of $\boldsymbol{A}$, $\boldsymbol{A}_{in}$, the random shuffled version of it is used. If we want to accumulate a large data set before each update then it is reasonable to reduce its size by also applying down-sampling before the shuffling.

The final transition from the demonstration stage to the fully autonomous stage (human demonstrator disconnected from the robot control loop) can be either done automatically or manually. The automatic switch can be realised based on the threshold percentage of the given task space being covered by the local models.

IV. Experimental Setup

In the experiments we used HM robot as an interface between the human demonstrator and the robotic manipulator (see Fig. 2). Please refer to the supplementary multimedia file

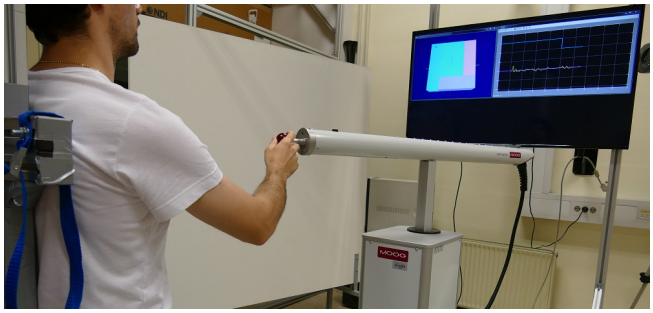

Fig. 2. Robot manipulator control interface consisting of HapticMaster robot and monitor for providing human with visual feedback.

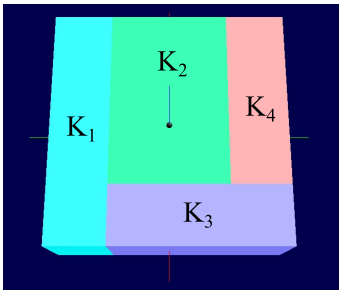

Fig. 3. Virtual environment. The colours represent different objects on the surface. Each object has a different stiffness (K_1, K_2, K_3 and K_4). The robotic manipulator's actual end-effector position is displayed by the black ball in the centre.

for a video of the setup/experiment. HM has a robotic arm with 3 active degrees of freedom that can render a haptic environment at the end-effector where the human holds and moves it. At the same time HM provides the measured position of the end-effector that can be used to command the position of the teleoperated robotic manipulator. The physical aspects of the robotic manipulator and the environment were simulated in MATLAB/Simulink, while the visual aspects were simulated by the OpenGL Utility Toolkit (GLUT) (see Fig. 3). The robot was positioned in front of the demonstrator who sat on a chair. We mounted a monitor in front of the demonstrator where we displayed the virtual environment and feedback related to the task execution. The machine-learning algorithm was based on the toolbox provided by [25]. The learning and shared-control algorithm operated at 100 Hz.

The HM robot operated in two modes. The first mode was admittance control where the measured interaction force between the HM end-effector and human hand produced the desired motion. This mode was used in case when the demonstrator had the control over the virtual robotic manipulator ($C = 1$). The actual position of the HM end-effector (human demonstrator's arm end-effector) was used as the commanded (reference) position for the virtual robotic manipulator. The actual position of the manipulator was dependant on the surface in the virtual environment. We assisted the demonstrator in precise maintaining the desired commanded position in z-axis by a virtual spring-damper system. The second mode was position control using a proportional controller realised by a virtual spring between the reference and actual HM end-effector position in z-axis.

This mode was used in case when the robotic skill had the control over the virtual robotic manipulator ($C = 0$).

One experienced human demonstrator participated in the experiments. The task of the robotic manipulator was to produce a reference interaction force on the unknown objects in the environment. The interaction force had to be produced perpendicularly to objects (in z-axis). Each object had different stiffness properties. To maintain the desired force the demonstrator had to adjust the commanded (reference) end-effector position in z-axis of the robotic manipulator based on the position in x-y plane. Please note that variables x and y now correspond to the position of robotic manipulator's end-effector. The following equation describes the necessary force control policy

$$F_z = K(x,y)\big(z_r - z_a\big), \tag{12}$$

where F_z is the interaction force acting between the manipulator and the environment in z-axis, $K(x,y)$ is the stiffness of the object in z-axis, z_a is actual and z_r is reference position of the manipulator's end-effector in z-axis. The stiffness $K(x,y)$ depends on the object (i.e. on the position in x-y plane). Actual position z_a depends on the environment too as the object blocks it from reaching the reference position inside the object. The higher the desired force is at a given stiffness of the object the larger the difference between commanded and actual position must be (i.e. reference must be deeper inside the object). For producing the same force, the difference between the commanded and actual position must be lower for objects with higher stiffness and vice-versa.

We constructed a 0.4 by 0.4 meter surface from four different objects. The configuration of surface is shown in Fig. 3. Each object had a different stiffness: $k_1 = 100$ N/m, $k_2 = 150$ N/m, $k_3 = 300$ N/m and $k_4 = 500$ N/m. The goal was to teach the robotic manipulator how to produce a reference force $F_z = 10$ N (LWR output) perpendicularly to the surface regardless of the position on the x-y plane (LWR input). The human demonstrator had to move the manipulator's position on x-y plane across various parts of the surface and the commanded position in z-axis had to be adjusted so that the reference force was maintained according to (12). While doing this, the learning system gradually acquired the skill how to produce the task autonomously. If the models existed in the given state region of x-y plane the shared-control system delegated the control responsibility over the manipulator to the robotic skill. Otherwise, the demonstrator had the control in order to continue the teaching process.

We determined parameters for the LWR-based learning system experimentally. The parameter related to the size of the model was set to $\boldsymbol{D} = \begin{bmatrix} 400 & 0 \\ 0 & 400 \end{bmatrix}$. The values of diagonal elements correspond to each input. Since in our case the inputs are the same physical quantity (position in x and y-axis), we set both diagonal elements to equal value. In case the inputs are different physical quantities then the values should be appropriately set for each input separately. Parameters related to the generation and removal of the models were set to $w_{gen} = 0.8$, $w_{prune} = 0.9$. For these proof-of-concept experiments selected $\lambda = 1$ since we did not want to forget the old training data. If forgetting of older data is preferred then $\lambda < 1$ should be used.

The parameters for shared-control system were also set experimentally. The threshold of model's receptive field activation that determines whether the control is given to the demonstrator or robotic skill was set to $w_{th} = 0.5$. We recommend following $w_{th} \leq w_{gen}$ in order not to create new models while the demonstrator does not have the control responsibility. The parameter related to smooth transition of control responsibility between the human and robotic skill was set to $d = 0.05$. If slower transition phase is preferred then this parameter should be increased. Accumulation buffer length was set to $N_{acc} = 1000$. This parameter should be lower if more frequent updates of robotic skill are preferred.

V. Results

The results of the robot learning during the experiment are shown in Fig. 4. Each column represents the different stage of training data update. The time stamps of the update application are displayed on the top. The top row shows the acquired models (robotic skill) for manipulator's displacement of reference position from the actual position in z-axis. The middle row shows the force prediction error of the models at each stage. The bottom row shows the motion of the robot manipulator (thick black and green line) on the object's surface (thin black rectangle). The black line shows the trajectory when the demonstrator had the control over the robotic manipulator's force production task. The green line shows the trajectory when robotic skill had the control over the robotic manipulator's force production task. The red crosses show the centres while blue ellipses show the threshold activation ranges w_{th} of the currently available local models that are part of robotic skill.

From the graphs we can see how the overall model was gradually updated when the demonstrator was teaching at various sections of the surface. In the first stage the local models are only created around the state region where the demonstration was initially performed. In the other sections of the surface the prediction error is naturally high due to non-existence of local models. When the demonstration progressed additional models were generated in the other sections. In the last training data update the local models fully covered the space of the given task and the robotic manipulator entered the fully autonomous stage (i.e. autonomously produced the desired force based on the position on the x-y plane).

Additional results are shown in Fig. 5. The interaction force control as a result of adjusted commanded (reference) position along z-axis is shown in the top graph. We can see that the desired force was maintained relatively well by both the demonstrator and the obtained robotic skill. The black dots in the top show the time stamps when the training data was updated and correspond to the update times in Fig. 4.

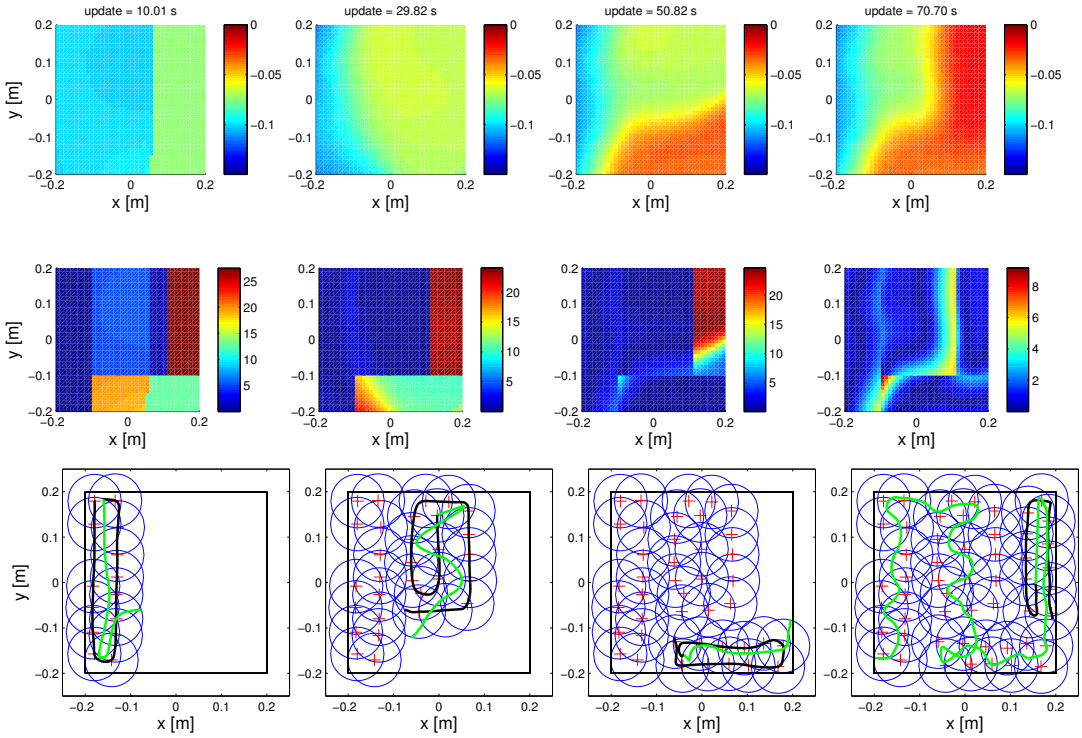

Fig. 4. Results of the experiments - learning. The top row graphs show prediction of robotic manipulator's commanded (reference) position displacement from the actual position in z-axis. The middle row graphs show the error of predicted interaction force in z-axis. The bottom row graphs show the motion trajectory in x-y plane (thick black line corresponds to $C = 1$, thick green line corresponds to $C = 0$) and distribution of local models (red crosses are centres and blue ellipses are activation ranges of w_{th}). Each column represents the state of robotic skill after training data update (update time stamps are on the top).

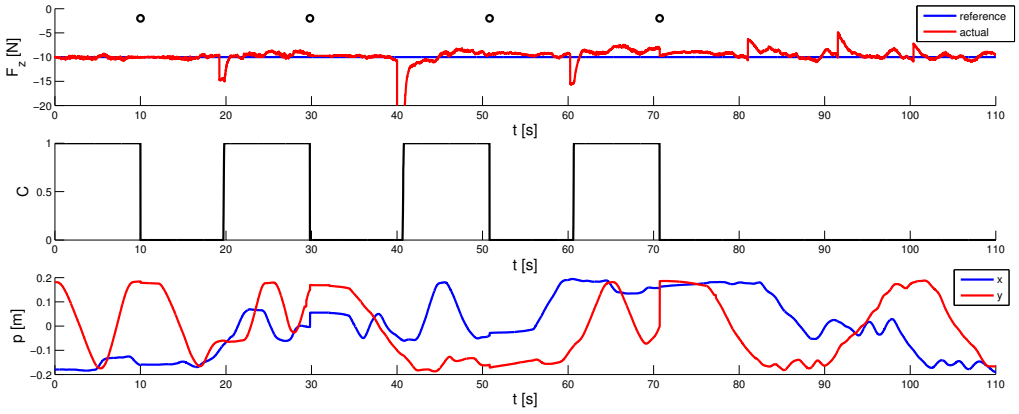

Fig. 5. Results of the experiments - task execution and shared control. The top graph shows the reference and actual force in z-axis. The middle graph shows the control responsibility factor C (1 = demonstrator, 0 = robotic skill). The bottom graph shows the position of the manipulator in x-y plane. Training data updates are indicated by black dots on the top and correspond to the time stamps at the top of Fig. 4.

The control responsibility factor C is shown in the middle graph. The position of the manipulator in x and y-axis is shown in the bottom graph. Initially, C was equal to one as there were no models present. After the prescribed amount of training data N_{acc} was accumulated the robotic skill was updated with the learnt models in the demonstrated state region. Demonstrator then remained in that state region to examine the performance of the obtained robotic skill. Consequently C became zero to give the control responsibility to the robotic skill. This procedure then repeated for the remaining three state regions of the task defined by different stiffness of the plane.

By observing the control responsibility we can see how it switched from demonstrator to the currently available local models. After each model update the control responsibility was shifted from the demonstrator to the robotic skill as the current position of robotic manipulator was within the predefined range of the existing models ($w_{max} > w_{th}$). The

demonstrator could then move within the area where the teaching was already done to observe how the obtained robotic skill performs the task autonomously (see thick green lines in bottom graph of Fig. 4). When the validation was done, the demonstrator moved to the unexplored area to perform the teaching there (see thick black lines in the bottom graph of Fig. 4).

VI. Discussion

The aim of this paper was to propose a new shared-control approach for human-in-the-loop robot teaching, and realise it for a non-trivial task as a proof-of-concept. In the hardware experiments conducted, the demonstrator could examine, online, the performance of the robotic skill being built during the demonstration stage, and gradually transform the robot to be an expert in the full allowed workspace. In the current implementation, we did not implement the ability of the demonstrator to remove already built local models. This was not needed in the current experiments as the demonstrator was expert and did not teach *bad* policies. This feature can be incorporated into the current system as a manual interface (e.g. pushing a button) or an automatic mechanism that detects the demonstrator's effort to correct a bad policy.

In future we will introduce an upgrade to the proposed framework where the human demonstrator will be able to remove the models, if the robotic skill performance is unsatisfactory after online examination. This will clear the undesired models in designated state region and allow corrections by re-teaching. Parameter λ could be used in this case to forget the undesired models. In addition, we will experimentally explore the influence of system parameters on the performance of the proposed method.

By observing the obtained models in Fig. 4, we can see that there is a considerable prediction error on the borders between two objects of different stiffness. This can be attributed to the difficulty of regression method to model larger discrete transitions within the training data output.

Potential disadvantage of the proposed method compared to [7] and [24] is that the demonstrator has to manually inspect and determine the quality of the robotic skill. However, this gives the demonstrator more control over the learning. In addition, no extra cost functions are required to determine the quality automatically, as they can be hard to acquire and can produce unreliable results in complex tasks.

References

[1] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robots*, vol. 33, no. 4, pp. 361–379, Apr. 2012.

[2] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *Robotics, IEEE Transactions on*, vol. 24, no. 6, pp. 1463–1467, 2008.

[3] P. Evrard, E. Gribovskaya, S. Calinon, A. Billard, and A. Kheddar, "Teaching physical collaborative tasks: object-lifting case study with a humanoid," in *IEEE-RAS International Conference on Humanoid Robots*, 2009, pp. 399–404.

[4] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *Robotics, IEEE Transactions on*, vol. 26, no. 5, pp. 800–815, Oct 2010.

[5] H. Ben Amor, E. Berger, D. Vogt, and B. Jung, "Kinesthetic bootstrapping: teaching motor skills to humanoid robots through physical interaction," in *Proceedings of the 32nd annual German conference on Advances in artificial intelligence*, 2009, pp. 492–499.

[6] D. Nguyen-Tuong and J. Peters, "Local gaussian process regression for real-time model-based robot control," in *Intelligent Robots and Systems (IROS), 2008 IEEE/RSJ International Conference on*, Sept 2008, pp. 380–385.

[7] L. Peternel and J. Babič, "Humanoid robot posture-control learning in real-time based on human sensorimotor learning ability," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 2013, pp. 5309–5314.

[8] A. Gams, A. J. Ijspeert, S. Schaal, and J. Lenarčič, "On-line learning and modulation of periodic movements with nonlinear dynamical systems," *Auton. Robots*, vol. 27, no. 1, pp. 3–23, Jul. 2009.

[9] L. Peternel, T. Petrič, E. Oztop, and J. Babič, "Teaching robots to cooperate with humans in dynamic manipulation tasks based on multi-modal human-in-the-loop approach," *Autonomous robots*, vol. 36, no. 1-2, pp. 123–136, Jan 2014.

[10] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey." *Cognitive Science*, vol. 12, no. 4, pp. 319–40, 2011.

[11] M. Ewerton, G. Neumann, R. Lioutikov, H. B. Amor, J. Peters, and G. Maeda, "Learning multiple collaborative tasks with a mixture of interaction primitives," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 1535–1542.

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

[13] A. Lockerd and C. Breazeal, "Tutelage and socially guided robot learning," in *Intelligent Robots and Systems (IROS), 2004 IEEE/RSJ International Conference on*, vol. 4, Sept 2004, pp. 3475–3480.

[14] B. Argall, B. Browning, and M. Veloso, "Learning by demonstration with critique from a human teacher," in *Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on*, March 2007, pp. 57–64.

[15] D. Lee and C. Ott, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Autonomous Robots*, vol. 31, no. 2, pp. 115–131, 2011.

[16] L. Peternel, T. Noda, T. Petrič, A. Ude, J. Morimoto, and J. Babič, "Adaptive control of exoskeleton robots for periodic assistive behaviours based on EMG feedback minimisation," *PLoS ONE*, vol. 11, no. 2, p. e0148942, Feb 2016.

[17] L. Peternel, T. Petrič, and J. Babič, "Human-in-the-loop approach for teaching robot assembly tasks using impedance control interface," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 1497–1502.

[18] G. Niemeyer, C. Preusche, and G. Hirzinger, "Telerobotics," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer-Verlag Berlin Heidelberg, 2008, pp. 741–757.

[19] A. D. Dragan and S. S. Srinivasa, "A policy-blending formalism for shared control," *Int. J. Rob. Res.*, vol. 32, no. 7, pp. 790–805, Jun. 2013.

[20] M. O'Malley, A. Gupta, M. Gen, and Y. Li, "Shared control in haptic systems for performance enhancement and training," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 128, no. 1, pp. 75 – 85, 2006.

[21] S. Jain, A. Farshchiansadegh, A. Broad, F. Abdollahi, F. Mussa-Ivaldi, and B. Argall, "Assistive robotic manipulation through shared autonomy and a body-machine interface," in *Rehabilitation Robotics (ICORR), 2015 IEEE International Conference on*, Aug 2015, pp. 526–531.

[22] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural Comput.*, vol. 10, no. 8, pp. 2047–2084, Nov. 1998.

[23] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Comput.*, vol. 17, no. 12, pp. 2602–2634, Dec. 2005.

[24] M. Zamani and E. Oztop, "Simultaneous human-robot adaptation for effective skill transfer," in *Advanced Robotics (ICAR), 2015 International Conference on*, July 2015, pp. 78–84.

[25] S. Klanke, S. Vijayakumar, and S. Schaal, "A library for locally weighted projection regression," *J. Mach. Learn. Res.*, vol. 9, pp. 623–626, Jun. 2008.