Incremental Kinesthetic Teaching of End-Effector and Null-Space Motion Primitives

Matteo Saveriano, Sang-ik An and Dongheui Lee

Abstract— **In this paper, we propose a unified approach to teach and iteratively refine both end-effector and null-space movements. Hence, the robot can be taught to make use of all its degrees-of-freedom (DoF) to adapt its behavior to new dynamic scenarios. In order to achieve this goal we propose an incremental learning approach in a framework of kinesthetic teaching based on a multi-priority kinematic controller, the so-called Task Transition Control (TTC). The learning algorithm is responsible for skill acquisition and their incremental update. On the real-time level, end-effector and null-space motion primitives, as well as the physical guidance are considered as prioritized tasks. The transitions among these tasks and their insertion and removal are managed by the TTC according to the specified transition parameters. This allows to introduce a customized task which guarantees a proper and smooth response to the applied external forces during the kinesthetic teaching. Experimental results on a 7 DoF KUKA lightweight manipulator show the effectiveness of the proposed approach.**

I. Introduction

In real applications, the robot is required to execute complex tasks and to adapt its behavior to dynamic environments. Skills acquisition and their compact representation as motion primitives, as well as on-line adaptation of those skills to new scenarios, are of importance in everyday scenarios. The skills acquisition procedure has to be simple, natural and intuitive. In this way, new skills can be easily learned and also non-expert users can teach the robot how to accomplish a task.

A natural and intuitive way that humans use to teach new skills is kinesthetic teaching, i.e. manual guiding of the partner (robot) during the task execution. This is a well established concept in the Programming by Demonstration (PbD) literature [1]. Several approaches have been proposed to represent the skill in a compact form reducing the amount of data to store, for example, stochastic regression based approaches [2], [3] and attractor based ones [4].

In dynamic environments, the possibility of modifying the learned motion primitives is also required. Incremental refinement of the learned motion has been investigated in [5], [6]. In particular, in [5], the problem is handled of how motion primitives can be incrementally refined by physical coaching. The key idea in this work is to physically coach the robot during the execution. In this way, natural coordinated movements can be learned.

Nevertheless, when the robot is redundant, i.e. the robot has more DoF than the required DoF for task accomplishment, one should consider how the redundant DoF can be used in a fruitful manner. As known from the robot control literature, in fact, redundant DoF can be used to execute multiple tasks at the same time. Null-space projection techniques are usually adopted to avoid that lower priority tasks affect the execution of the higher priority ones [7], [8], [9].

The tasks to execute can be learned by PbD. In [10], a *recurrent neural network* is used to learn the inverse kinematic mapping and null-space constraints from recorded end-effector positions and joint angles. Training data are collected in two steps. Firstly, the user guides the robot towards the task execution. Then, with the end-effector fixed in the desired goal pose, the user teaches some local null-space configurations. End-effector and null-space tasks are then executed using a multi-priority controller. In [11], null-space policies are learned from observations considering the first priority task as a set of constraints on the null-space policy. The null-space policy is then estimated by solving an optimization problem, trying to minimize the inconsistency among the observations. However, these approaches do not allow the online adaptation of null-space motion primitives.

We propose a new method to incrementally learn end-effector and null-space motions via kinesthetic teaching. New training sequences are acquired during the motion execution by physically coaching the robot. Hence, the user can decide where the motion primitive has to be refined, leaving the rest of the motion unchanged. Our main contribution consists in combining incremental learning algorithms with a customized multi-priority kinematic controller that guarantees a smooth human-robot interaction. The proposed approach has the following advantages.

- The user can refine multiple motion primitives (e.g., end-effector and elbow motions) naturally and intuitively.
- Null-space motions can be learned without modifying the end-effector task execution by using a kinematic controller with varying task priorities.
- Being the teaching executed while the robot moves, the user can figure out if the desired null-space space motion is compatible with the end-effector task. He/she simply has to check if the end-effector deviates from the desired path.

The rest of the paper is organized as follows. In Section II we present an overview of the proposed approach. Section III describes how motion primitives are learned and incrementally refined. In Section IV the proposed interaction control for incremental learning approach is presented. Section V presents the experimental results. Section VI states the conclusions and the future work.

Authors are with Chair of Automatic Control Engineering, Technische Universität München, Munich, Germany. {matteo.saveriano, sangik.an, dhlee}@tum.de

II. Overview of the Proposed Approach

In this section we give an overview about the integration of human-robot interaction, incremental learning and multi-priority control in our system. The complete procedure is shown in Fig. 1. As a first step, motion primitives are learned in batch mode from human demonstrations. We decided to collect these demonstrations using kinesthetic teaching. Other techniques can also be used.

In case the learned motion primitive has to be refined online, new training data are collected during the motion execution. While a customized impedance controller was proposed for kinesthetic teaching in [5], in this paper we adopt a control method by varying priorities between the multiple tasks and by allowing smooth task transitions. This choice gives us the possibility to kinesthetically teach null-space tasks without affecting the end-effector task execution.

The motion refinement proceeds in this way. A multi-priority kinematic controller, namely the *Task Transition Controller (TTC)*, starts to execute a motion primitive. If the user wants to modify this motion primitive (s)he starts to physically guide the robot. This occurrence is detected by using an estimation of the forces applied on the robot (Sec. IV). Then, a new task is generated by transforming interaction forces into desired velocities $\dot{\boldsymbol{x}}_{ic}$.

The interaction task is initially inserted with the lowest priority. Hence, the robot tries to execute this task using only its redundant DoF, accurately executing the end-effector task. If both the tasks can be executed at the same time, the learned behavior, i.e. the desired velocity of the contact point, can be considered as a null-space motion primitive.

The priority of the interaction task can be increased when it is not correctly executed. To decide when tasks priorities has to be changed, we follow this simple idea. If the user perceives that the robot is not accomplishing its guidance, then he simply increases the applied force. Hence, a threshold on the external force is used to decide the tasks priority. This is the case of the end-effector motion primitives refinement. When the user guides the robot touching the end-effector, one between the interaction task and the end-effector motion primitive cannot be executed. At the beginning, the robot keeps following its original motion. Then, the user increases the contact force, the tasks are smoothly switched and the robot can accomplish the physical guidance and refine its behavior. The proposed switching mechanism is also useful to understand if the tasks are in conflict. The user has to simply check if the end-effector is significantly deviating from the desired path during the interaction.

III. Incremental Learning of Motion Primitives

In this section, we describe the main features of the adopted Hidden Markov Models (HMM) incremental learning approach and refer to [5] for further details.

A. Motion Learning and Generation

An HMM represents a Markov chain in which L states are hidden and it is described by the set of parameters $\boldsymbol{\lambda} = \{\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{c}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, where $\boldsymbol{\pi}$ is the initial state probability,

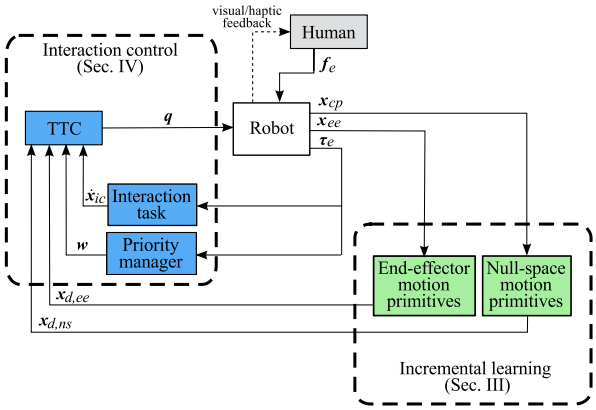


Fig. 1. System overview.

$\boldsymbol{A}$ is the state transition probability matrix. The mixture coefficients $\boldsymbol{c}$, the means $\boldsymbol{\mu}$ and the covariance matrices $\boldsymbol{\Sigma}$ represent the continuous observation probability distribution as a mixture of M Gaussians [12]. HMM parameters $\boldsymbol{\lambda}$ are usually learned from a time series of spatial data (Cartesian trajectories in this work) ${}^{s}\boldsymbol{O} = \{{}^{s}\boldsymbol{o}(t)\}$, using the EM algorithm [13].

To generate a smooth motion sequence from an HMM, a normalized time variable, linearly varying from 0 to 1, is introduced for each state. Firstly, the correlation between temporal and spatial data (${}^{t}\boldsymbol{\mu}$, ${}^{tt}\boldsymbol{\Sigma}$ and ${}^{ts}\boldsymbol{\Sigma}$) is learned from the motion ${}^{s}\boldsymbol{O}$ and the optimal $\boldsymbol{Q}^{\star}$ state sequences [13].

Then, a state sequence is generated deterministically, considering as initial state $q(1) = arg\max_i \boldsymbol{\pi}$. For each state i, the expected duration to stay at i is calculated as $d = 1/(1 - a_{ij})$, where a_{ij} is the probability to transit from state i to $j \neq i$. After staying in state i for d steps, the next state is chosen as $q(t+1) = arg\max_j a_{q(t)j}$.

From the generated state sequence $\boldsymbol{Q}$, the relative temporal sequence ${}^{t}\boldsymbol{O}$ is calculated. For this temporal sequence, the *responsibility* $\gamma_i(t)$ for each state i is calculated by using the HMM forward and backward variables [13] for the temporal sequence.

A sequence of spatial data is then calculated from $\boldsymbol{Q}$ and ${}^{t}\boldsymbol{O}$, considering for each time step t a mixture of K Gaussians in the state $i = q(t)$. Finally, a smooth trajectory ${}^{s}\boldsymbol{o}(t)$ is generated using GMR [12].

B. Incremental Motion Refinement

The goal of incremental learning of motion primitives is to update the previous knowledge of motion primitives as new demonstrations are provided, without keeping all the training data in the dataset. To avoid that the learning algorithm becomes insensitive to new data when the data set becomes large a forgetting factor is used.

The main idea is to use just two demonstrations (${}^{s}\boldsymbol{O}^{d}$, $d = 1, 2$): one is the new demonstration provided by the user, the other is the smooth trajectory generated from the current motion primitive. For the new demonstration the weighting term $w^d = \eta$, where η is the forgetting factor, is given. For the generated motion trajectory is $w^d = 1 - \eta$. The new

HMM parameters $\hat{\boldsymbol{\lambda}}$ are updated using the old ones $\boldsymbol{\lambda}$ and the demonstrations.

IV. Human-Robot Interaction Control

In order to realize the incremental motion refinement by physical contact, we propose a new kinesthetic teaching control framework, based on the so-called task transition control (TTC).

A. Task transition control

A task T of a robot can be defined by a tuple $(\dot{\mathbf{x}}, \dot{\boldsymbol{x}}_d)$ in the velocity level where $\dot{\boldsymbol{x}} \triangleq \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$ is the task variable, $\dot{\boldsymbol{x}}_d(\boldsymbol{q}, t)$ is the desired trajectory of $\dot{\boldsymbol{x}}$, $\boldsymbol{q}$ is the generalized coordinate of the robot, and $\boldsymbol{J}(\boldsymbol{q})$ is the Jacobian of the forward kinematics, then the inverse kinematics is to find $\dot{\boldsymbol{q}}$ that minimizes the task error $\|\dot{\mathbf{e}}\| \triangleq \|\dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}}\|$. In general, the definition of the task is not fixed for whole operation time and multiple tasks $T_1, \cdots, T_k$ can exist at the same time. For the clearness of the discussion, we introduce a mathematical convention of a set of tasks, such that $(T_i, T_j),\ i, j = 1, \ldots, k,\ i \neq j$ represents an unprioritized accumulation of two tasks and $[T_i, T_j]$ represents a prioritized accumulation in which T_i has priority.

Given a set of accumulated tasks $[T^1, \ldots, T^l]$, the inverse kinematic solution $\dot{\boldsymbol{q}}^i$ for each task T^i is calculated independently [9] and the resulting joint trajectories $\dot{\boldsymbol{q}}^i$ are directly interpolated by using the barycentric coordinate. When the task definitions need to be changed during operations, smooth transitions of the barycentric coordinates guarantee tasks switch without jumps in the desired joint trajectory.

Definition 1 (Barycentric Coordinates)**.** The barycentric coordinates of $\dot{\boldsymbol{q}}$ with respect to $\mathcal{Q} \triangleq \{\dot{\boldsymbol{q}}^1, \cdots, \dot{\boldsymbol{q}}^l\}$ is defined as any set of real coefficients $w^1, \cdots, w^l$ depending on $(\boldsymbol{q}, t)$, such that all the following properties hold:

- Nonnegativity: $w^i \geq 0$,
- Linearity: $\dot{\boldsymbol{q}} = \sum_{i=1}^{l} w^i \dot{\boldsymbol{q}}^i$ with $\sum_{i=1}^{l} w^i = 1$, and
- Smoothness: $w^i \in C^s(\boldsymbol{q}, t)$

where $s \in \mathbb{N}^{\geq 0}$ depends on the degree of smoothness needed.

Theorem 1 (Task Transition Control)**.** *The TTC given by* (1) $\sim$ (5) *provides smooth and arbitrary task transitions within* $\mathcal{T} = \{T^1, \cdots, T^l\}$*, as well as bounds the inverse solutions such that* $\|\dot{\boldsymbol{q}}\| \leq \max\{\|\dot{\boldsymbol{q}}^1\|, \cdots, \|\dot{\boldsymbol{q}}^l\|\}$

$$\dot{\boldsymbol{q}} = \boldsymbol{q}\boldsymbol{w} \tag{1}$$

$$\boldsymbol{w}^{(s+1)} = -\sum_{j=1}^{s} k_j \boldsymbol{w}^{(j)} + k_0(\boldsymbol{w}_d - \boldsymbol{w}) \tag{2}$$

$$\boldsymbol{w}_d(\boldsymbol{q}, t) \in \{\hat{\mathbf{e}}^1, \cdots, \hat{\mathbf{e}}^l\} \subset \mathbb{R}^l \tag{3}$$

$$\boldsymbol{w}(t_i) \in \{\boldsymbol{a} \in \mathbb{R}^n : \mathbf{1}^{\mathbf{T}}\boldsymbol{a} = \mathbf{1},\ \boldsymbol{a} \geq \mathbf{0}\} \tag{4}$$

$$\boldsymbol{w}^{(j)}(t_i) = \mathbf{0},\ \forall j \in \mathbb{N}_{\leq k} \tag{5}$$

where $\boldsymbol{q} \triangleq [\dot{\boldsymbol{q}}^1\ \cdots\ \dot{\boldsymbol{q}}^l] \in \mathbb{R}^{n \times l}$*,* $\boldsymbol{w} \triangleq [w^1\ \cdots\ w^l]^T \in \mathbb{R}^l$*,* $\boldsymbol{w}^{(j)} \triangleq d^j\boldsymbol{w}/dt^j$*,* $s \in \mathbb{N}^{\geq 0}$*,* $\mathbf{1} \triangleq [1\ \cdots\ 1]^T \in \mathbb{R}^l$*,* $\{\hat{\boldsymbol{e}}^1, \cdots, \hat{\boldsymbol{e}}^l\}$ *is a set of the standard basis in* $\mathbb{R}^l$*, and* $\{k_0, \cdots, k_k\} \subset \mathbb{R}$ *are stabilizing control gains that don't generate overshoot of the* $(s+1)$*-th order linear system.*

The proof of theorem 1, as well as further details on the TTC, can be found in [14].

B. Tasks definition

The robot is required to execute the motion primitives and, at the same time, to be compliant in case of physical interaction. We define two different tasks: 1) end-effector task T_{ee} and 2) interaction control task T_{ic}. Given T_{ee} and T_{ic}, the required task can be changed in each time by the strength of the human intervention (the norm of the external force $f = \|\boldsymbol{f}_e\|$):

- $T^1 = T_{ee}$: If there is not human intervention ($f < f_i$), only T_{ee} exists.
- $T^2 = [T_{ee}, T_{ic}]$: If there is weak human intervention ($f_i \leq f < f_s$), both tasks exist and T_{ee} has priority.
- $T^3 = [T_{ic}, T_{ee}]$: If there is strong human intervention ($f_s \leq f$), both tasks exist and T_{ic} has priority.

The threshold f_i is used to decide to insert the interaction control task as the lower priority task, generating extra null-space motions. Another threshold f_s is used to switch priority between two tasks as shown in Tab. I. In other words, the robot firstly tries to project T_{ic} in its null-space. If the human perceives that the task is not correctly executed, (s)he simply applies a bigger force until T_{ic} becomes the first priority task.

TABLE I
Task Transition Rule

$f < f_i$	$f_i \leq f < f_s$	$f_s \leq f$
T_{ee}	$[T_{ee}, T_{ic}]$	$[T_{ic}, T_{ee}]$

The interaction task is an admittance control that transforms the external Cartesian force in a desired velocity using the relationship:

$$\dot{\boldsymbol{x}}_{ic} = c\boldsymbol{f}_e\ , \tag{6}$$

where $\dot{\boldsymbol{x}}_{ic}$ is the interaction task desired velocity, $\boldsymbol{f}_e$ is the applied external force and c is a tunable gain.

Hence, an estimation of the external Cartesian force applied to the robot is needed. The estimation of the external force requires two steps. Firstly, the external torque $\boldsymbol{\tau}_e$, applied in each joint, must be estimated. Secondly, given the contact point C, the external force $\boldsymbol{f}_e$ is computed by inverting the well-known equation $\boldsymbol{\tau}_e = \boldsymbol{J}_C^T\boldsymbol{f}_e$, where $\boldsymbol{J}_C$ is Jacobian of the contact point.

Approaches have been developed to estimate the external torque and the link where the contact occurs. For example, in [15], a momentum based disturbance observer is used for collision detection and reaction. For robots that do not have torque sensors in each joint, external torque can be estimated using only the encoders, as proposed in [16]. For the KUKA LWR manipulator, an estimation of the external joint torque is provided up to 1 KHz through the Fast Research Interface [17]. Being the provided torque ideally zero for all the joints located after the contact point, the touched link can be easily found using a thresholding method.

Precise contact point estimation becomes a challenging problem without the usage of an exteroceptive sensor (e.g., cameras). Since the estimation of the contact point is beyond the scopes of this work, we simply assume that the contact always occurs at the end of the contact link.

V. Experimental Results

A. Pick-and-Place learning and refinement

In this section we show how end-effector and null-space motion primitives are incrementally refined in our framework. The end-effector task is the pick-and-place (point-to-point) motion in Fig. 2, originally learned in batch mode (Sec. III-A) from 5 demonstrations. The number of hidden states in the HMM[1], as well as the number of Gaussians in each state, are empirically chosen to 7 and 1 respectively. The end-effector orientation is kept constant during the execution.

Grasping and release positions are also learned from physical interaction. If the grasping position is not set yet, the robot waits until its end-effector is touched. Then, the current position is saved as grasping position and the motion primitive executed. A similar approach is used for the release position[2]. The gain $c = 0.005\ m/Ns$ is used to transform the external forces $\boldsymbol{f}_e$ into velocities (6).

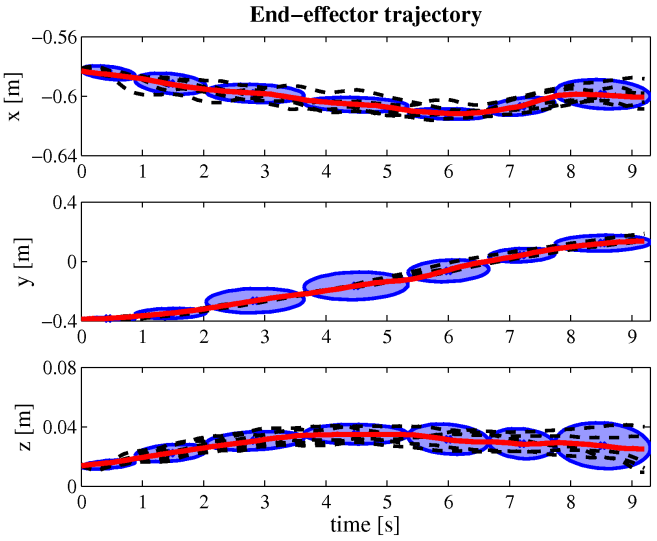


Fig. 2. Batch learning of the pick-and-place motion primitive using an HMM with 7 states and 1 Gaussian for each state. Five demonstrations (black dashed line) are used. The blue ellipses represents the learned Gaussian in each state. The red solid line is the retrieved smooth trajectory (Sec. III-A).

1) End-effector motion refinement: In this experiment, the original target position, located at $\boldsymbol{p} = [-0.6\ 0.14\ 0.027]$, is incrementally refined to reach the new target $\bar{\boldsymbol{p}} = [-0.5\ 0.006\ 0.027]$. During the execution, the user corrects the trajectory by physically guiding the robot toward the new goal.

[1]In this paper we always use *left-to-rigth* HHM [13], being this structure the most suitable for representing point-to-point motions.

[2]A video with the experiments is included.

At the beginning of the execution, the end-effector motion is the unique task in the stack. When the user starts to interact with the robot ($\|\boldsymbol{f}_e\| > 5\ N$), the TTC generates the new task $[T_{ee}, T_{ic}]$. Being the tasks $[T_{ee}]$ and $[T_{ee}, T_{ic}]$ in conflict, the robot cannot accomplish the physical guidance. Hence, the user applies a bigger force and the tasks priorities are smoothly switched ($\|\boldsymbol{f}_e\| > 15\ N$). After three repetitions, the robot is able to reach the new target position $\bar{\boldsymbol{p}}$, as shown in Fig. 3. The interaction starts at about 3 seconds after starting the execution. Hence, the first 3 seconds of the motion primitive (before the interaction) are left unchanged.

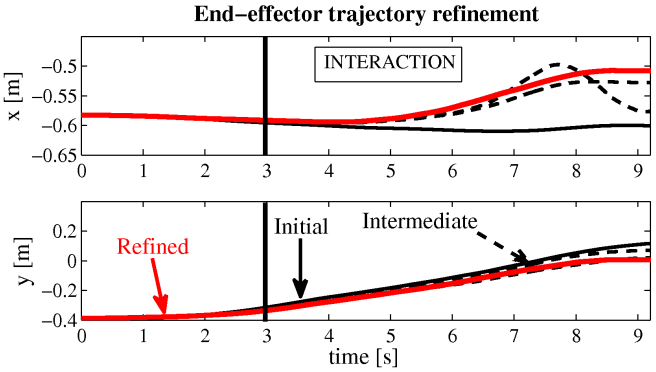


Fig. 3. Incremental motion refinement results. The robot is able to reach the new goal position $\bar{\boldsymbol{p}} = [-0.5\ 0.006\ 0.027]$ after three demonstrations. The motion in the z direction is not showed because it is not updated during the kinestethic teaching.

2) Elbow motion refinement: In this experiment, an unforeseen obstacle is put along the trajectory of the robot elbow (Fig. 5(a)). The end-effector motion is the pick-and-place task in Fig. 2. During the task execution the user perceives the robot is going too close to the obstacle and he physically teaches a collision-free motion for the robot's elbow (Fig. 5(b)). After three iterations, the robot is able to execute the end-effector task avoiding the collision with the learned null-space motion (Fig. 5(c)). The results of the incremental learning after three iterations are summarized in Fig. 4. The interaction starts at about 5.4 seconds after starting the execution. Hence, the first 5.4 seconds of the motion primitive (before the interaction) are left unchanged. The thresholds used to insert T_{ic} and to switch the priority are chosen as $f_i = 5\ N$ and $f_s = 15\ N$ respectively. Note that, as for the end-effector, the learned null-space primitive can be further refined in an online fashion.

To underline the role of f_i and f_s in the proposed teaching approach, we perform three experiments varying the value of the threshold f_s used to switch the priority between the end-effector task T_{ee} and the interaction task T_{ic}. The threshold used to insert T_{ic} is kept constant at $f_i = 5\ N$.

i) $f_i = f_s = 5\ N$: With this choice the interaction task T_{ic} is directly inserted as the first priority task, as shown in Fig. 7(a). In this case the physical guidance is easier because the robot can use more DoF to accomplish the task. Nevertheless, if the tasks are in conflict, errors in the motion primitive execution are accumulated. In this particular case T_{ee} and

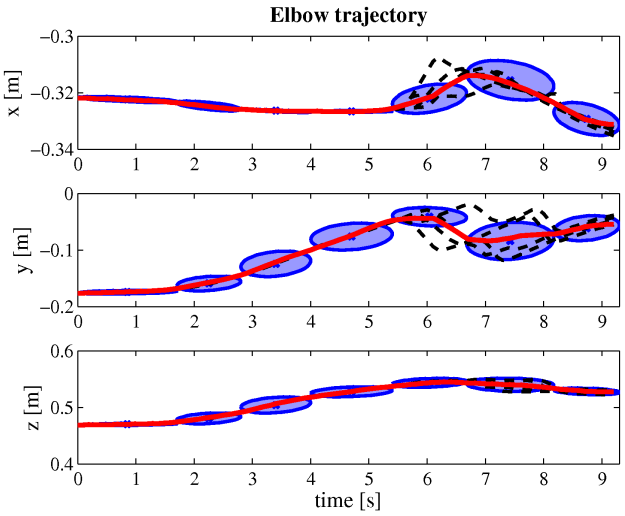
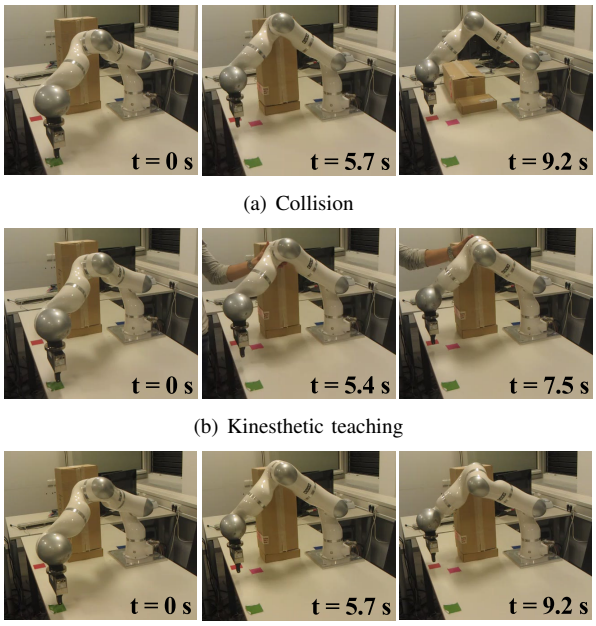


Fig. 4. Elbow motion learning using an HMM with 7 states and 1 Gaussian for each state. Three demonstrations (black dashed line) are incrementally provided. The blue ellipses represents the learned Gaussian in each state. The red solid line is the retrieved smooth trajectory (Sec. III-A).



(a) Collision

(b) Kinesthetic teaching

(c) Autonomous execution

Fig. 5. Snapshots of the elbow motion refinement procedure. (a) An unforeseen obstacle is placed on the elbow trajectory. (b) To avoid collisions, a collision-free elbow trajectory is learned by kinesthetic teaching. (c) After three iterations, the robot is able to execute both the end-effector and null-space tasks.

T_{ic} are not in conflict, since the end-effector position tracking error (Fig. 7(a)) is relatively small (less than 3 mm).

ii) $f_i = 5\ N,\ f_s = 30\ N$*:* With this choice T_{ic} is always executed in the null-space of T_{ee} (Fig. 7(b)) using only the redundant DoF. If the robot can accomplish the physical guidance, then the task are not in conflict and the new motion primitive can be effectively executed in the null-space of T_{ee}.

iii) $f_i = 5\ N,\ f_s = 15\ N$*:* With this choice transitions occur between T_{ic} and T_{ee} (Fig. 7(c)). This transitions are needed, for example, to refine end-effector motion primitives, being in this case T_{ee} and T_{ic} in conflict. An intermediate value for f_s is always suggested, since it is unknow a priori if the physical guidance is in conflict with the end-effector motion. During the kinesthetic teaching the user can figure out if the end-effector deviates significantly from the desired path and, if it happens, the user can reconsider the initial order of priorities. The end-effector errors in Fig. 7 show that the value of f_s has no practical effects on the system if T_{ic} and T_{ee} are not in conflict.

In all the previous cases the barycentric coordinates (Fig. 7) show several transitions between $[T_{ee}]$ and other tasks. This is basically due to the way of teaching. The user, in fact, moves the elbow far away from the obstacle a first time. While the execution proceeds other corrections may be needed, and so on until T_{ee} is completed.

The interaction controller (Sec. IV) combines the tracking performances of position controllers with an increased flexibility. As an example, Fig. 6 shows the end-effector position tracking errors for position and impedance control. The position controller accurately tracks the motion also during the interaction, but it makes impossible the teaching. On the other hand, to allow physical guidance with impedance control, one has to set low impedance gains[3] penalizing the end-effector task execution also without contacts.

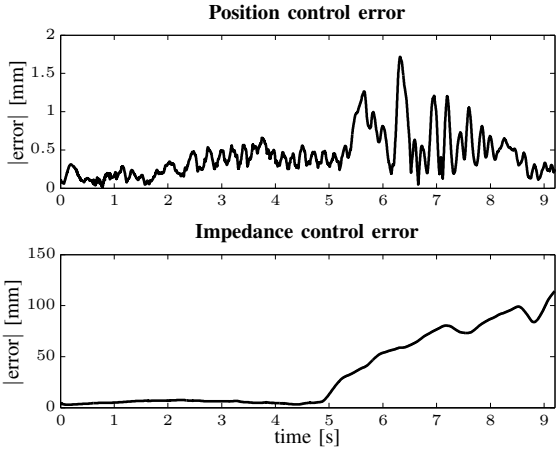
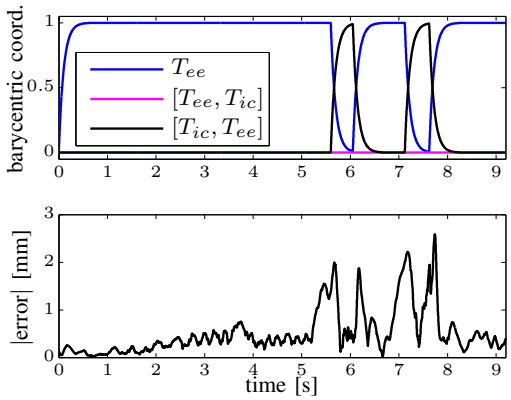


Fig. 6. End-effector position tracking error with position and impedance control. The interaction starts around 5 ms in both cases.

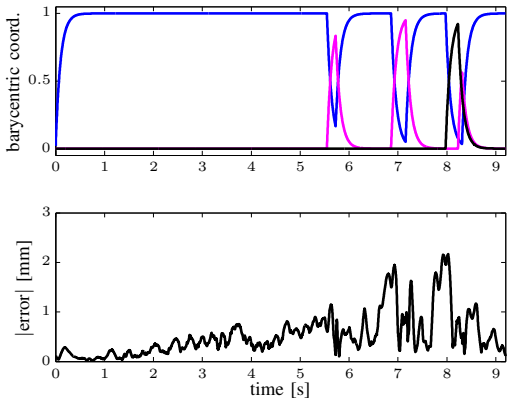
VI. Conclusion and Future Work

We presented an approach for learning and incremental refinement of both end-effector and null-space motion primitives, useful to adapt the robot behavior in unforeseen scenarios. The entire procedure is natural and intuitive. Firstly, a motion primitive for the end-effector is learned in batch mode. Demonstration can be provided, as in this case, by kinesthetic teaching, or by human imitation, as in [5]. Secondly, during the task execution, the user can modify the learned primitives by physically guiding the robot. A

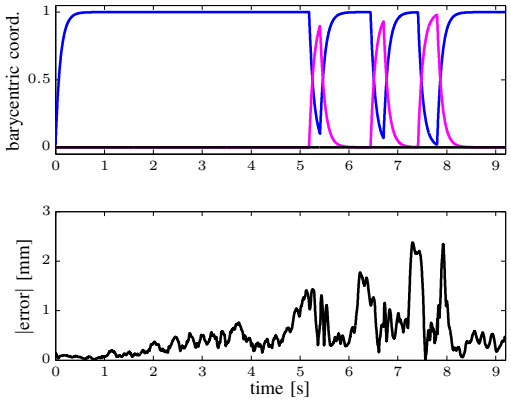
[3]We choose the stiffness matrix as $\boldsymbol{S} = 200\boldsymbol{I}$ and the damping matrix as $\boldsymbol{D} = 2d\sqrt{200}\boldsymbol{I}$, where $d = 0.7$ to avoid overshoot.



(a) $f_i = f_s = 5\ N$



(b) $f_i = 5\ N,\ f_s = 30\ N$



(c) $f_i = 5\ N,\ f_s = 15\ N$

Fig. 7. Barycentric coordinates and end-effector position tracking error for different values of f_i and f_s.

customized kinematic controller manages the tasks execution and the human-robot interaction, in order to guarantee safe and smooth responses of the manipulator. The controller is able to dynamically switch multiple tasks priority. Hence we can teach a robot motion, which consists of multiple tasks, by adding a new task without disturbing the already learned tasks much.

The threshold f_i is used to insert the interaction task and this value depends on the robot configuration when the interaction starts and also on the touched link. As a future work we will investigate the possibility of automatic selecting the value of f_i for different tasks and different touched links. Moreover, we plan to realize a usability study similar to [18] to better understand the advantages and disadvantages of the proposed approach when non-expert users are asked to teach the robot.

ACKNOWLEDGEMENTS

This work has been partially supported by the European Community within the FP7 ICT-287513 SAPHARI project and Technical University of Munich, Institute for Advanced Study, funded by the German Excellence Initiative.

REFERENCES

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., 2008, pp. 1371–1394.

[2] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 37, no. 2, pp. 286–298, 2007.

[3] D. Lee and Y. Nakamura, "Mimesis model from partial observations for a humanoid robot," *The International Journal of Robotics Research*, vol. 29, no. 1, pp. 60–80, 2010.

[4] A. Ijspeert, J. Nakanishi, P. Pastor, H. Hoffmann, and S. Schaal, "Dynamical Movement Primitives: learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[5] 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.

[6] S. M. Khansari-Zadeh and A. Billard, "Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014.

[7] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *International Journal of Robotic Research*, vol. 6, no. 2, 1987.

[8] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Transaction on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997.

[9] S. An and D. Lee, "Prioritized inverse kinematics using qr and cholesky decompositions," in *IEEE International Conference on Robotics and Automation*, 2014.

[10] A. Nordmann, C. Emmerich, S. Rüther, A. Lemme, S. Wrede, and J. J. Steil, "Teaching nullspace constraints in physical human-robot interaction using reservoir computing," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 1868–1875.

[11] C. Towell, M. Howard, and S. Vijayakumar, "Learning nullspace policies," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 241–248.

[12] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, vol. 4, no. 1, pp. 129–145, 1996.

[13] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, 1989, pp. 257–286.

[14] S. An and D. Lee, "Prioritized inverse kinematics with multiple task definitions," in *IEEE International Conference on Robotics and Automation*, 2015.

[15] S. Haddadin, A. Albu-Schäffer, A. De Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3356–3363.

[16] H. Sadeghian, L. Villani, M. Keshmiri, and B. Siciliano, "Task-space control of robot manipulators with null-space compliance," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 493–506, 2014.

[17] G. Schreiber, A. Stemmer, and R. Bischoff, "The fast research interface for the kuka lightweight robot," in *Proc. of the IEEE ICRA Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications*, 2010, pp. 15–21.

[18] S. Wrede, C. Emmerich, R. Grnberg, A. Nordmann, A. Swadzba, and J. Steil, "A user study on kinesthetic teaching of redundant robots in task and configuration space," *Journal of Human-Robot Interaction*, vol. 2, no. 1, pp. 56–81, 2013.