Autonomous Robots (2019) 43:713–726
https://doi.org/10.1007/s10514-018-9745-2

Learning from demonstration for semi-autonomous teleoperation

Ioannis Havoutis[1] · **Sylvain Calinon**[2]

Received: 23 March 2017 / Accepted: 2 April 2018 / Published online: 25 April 2018

Abstract
Teleoperation in domains such as deep-sea or space often requires the completion of a set of recurrent tasks. We present a framework that uses a probabilistic approach to learn from demonstration models of manipulation tasks. We show how such a framework can be used in an underwater ROV teleoperation context to assist the operator. The learned representation can be used to resolve inconsistencies between the operator's and the robot's space in a structured manner, and as a fall-back system to perform previously learned tasks autonomously when teleoperation is not possible. We evaluate our framework with a realistic ROV task on a teleoperation mock-up with a group of volunteers, showing a significant decrease in time to complete the task when our approach is used. In addition, we illustrate how the system can execute previously learned tasks autonomously when the communication with the operator is lost.

Keywords Learning from demonstration · Teleoperation · Shared autonomy · Manipulation in ROVs

1 Introduction

Manipulator arms are often used as a proxy for human manual labour in environments that are hostile. Such tasks might involve manipulating equipment in deep underwater facilities or handling materials that can be dangerous to handle by hand. Often such systems are controlled directly by a human operator who receives visual feedback from the teleoperated manipulator workspace and accordingly controls the degrees of freedom (DoFs) of the arm.

Such direct teleoperation can be cognitively very demanding for the operator. Teleoperation tasks are often very structured, but the robot operating environment seldom changes dramatically. Most variability in such tasks comes from the change of the pose of the manipulators and the pose of the items that are to be manipulated. For example, imagine an underwater ROV that needs to connect a set of hydraulic hoses to a distribution panel at a deep-sea facility. The task of connecting each hose is intrinsically the same, but the parametrization of each task instance changes. This way, for each connection the ROV might assume a different position while the connector for each hose would be at a different pose on the distribution panel.

Long range teleoperation is often the only solution for most dangerous environments, including space and deep-sea. In such cases the communication bandwidth imposes a hard constraint on the efficiency of the system. For example, streaming live video feeds from the teleoperated manipulator workspace imposes large delays to the completion of the set of tasks at hand.

In this paper, we present a probabilistic method that can assist an operator in performing a teleoperation task efficiently, through learning a representation of the task by demonstration. We present a structured way to deal with discrepancies between operator space and robot space and a way of autonomously executing the task in case of communication break-down (see Figs. 1 and 2 for an overview). We use a *task-parametrized Hidden semi-Markov model* (TP-HSMM) to learn task representations, and generate motions by sampling the model in combination with an optimal control formulation, namely with *linear quadratic tracking*

This work was in part supported by the DexROV project through the EC Horizon 2020 programme (Grant #635491).

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10514-018-9745-2) contains supplementary material, which is available to authorized users.

✉ Ioannis Havoutis
ioannis@robots.ox.ac.uk

Sylvain Calinon
sylvain.calinon@idiap.ch

[1] Oxford Robotics Institute, Department of Engineering Science, University of Oxford, 17 Parks Road, OX1 3PJ Oxford, UK

[2] Idiap Research Institute, 1920 Martigny, Switzerland

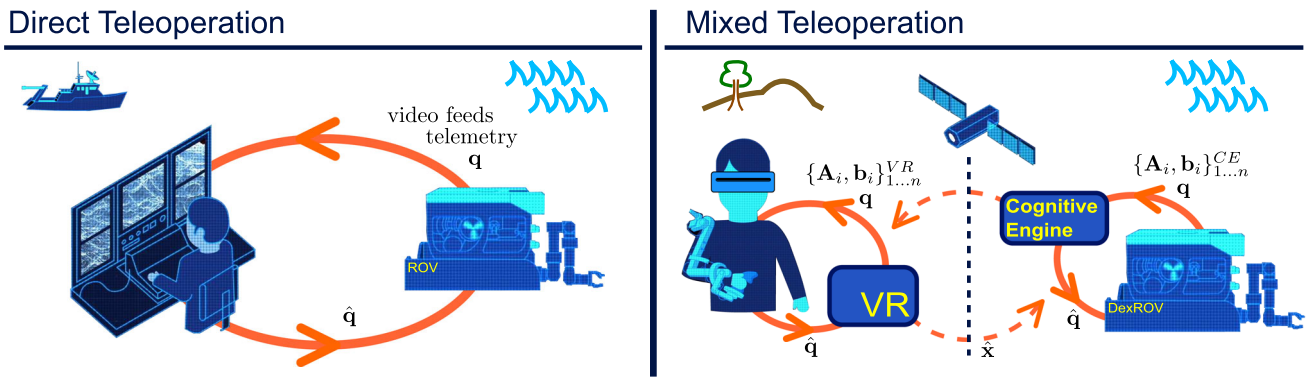

Fig. 1 Overview of the approach. The left side presents a traditional teleoperation scenario where the operator is situated on the vessel that supports the ROV at sea. The operator receives a set of video feeds from the cameras of the ROV, as well as the ROV's telemetry and joint configuration data. Based on this, the operator then directly commands the desired configuration for the robotic manipulator. The right side presents our new paradigm of mixed teleoperation. Here the operator commands a virtual ROV in a virtual environment (onshore), set up according to the model of the environment that the ROV will operate in, at sea (offshore). The arms of the ROV are controlled locally by a *cognitive engine* that does not require any large data (video feeds, telemetry) to be communicated. Both systems, onshore and offshore, are locally controlled while both sides have a local parametrization of the corresponding variables. Only minimal communication between the two sides is now required to adjust the local control loops. (i.e. operator state and coordinate system importances, in contrast to video feeds, telemetry, joint states, etc.)

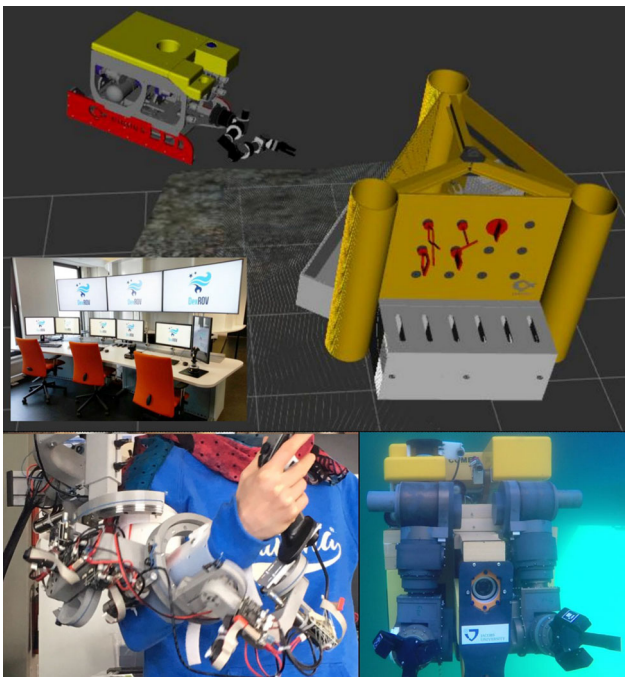

Fig. 2 Current stage of the DexROV project. Top: On the left, the simulated DexROV, equipped with manipulator and gripper. On the right, the simulated underwater testing rig, with a set of standardized ROV handles. The inset image shows the DexROV project onshore control center. Bottom: Left, the prototype exoskeleton that will be used as an input interface for the VR system. Right, the developed ROV with prototype manipulators and prototype underwater vision system visible between the arms

(LQT) with a double integrator system. Our approach leverages the key advantages of two models, the flexibility of a task-parametrized mixture model encoding (Calinon 2016) alongside with the ability of the HSMM to accurately capture motion duration and generate novel trajectories.

Throughout this manuscript we will use the term *mixed teleoperation* to refer to the behavior of the system. We chose this term as, we believe, it accurately describes that the behavior on the robot's side is a combination of the operator's input and the learned task model. Mixed teleoperation corresponds to the similar terms, often found in the literature, of assisted teleoperation, semi-autonomous teleoperation or shared autonomy.

Our contribution in this work is twofold. First, we present the task-parametrized extension of our previous work (Havoutis et al. 2016), that allows learned movements to be used in novel and changing situations. Second, we show how such a probabilistic model, learned by demonstration, can be used in a teleoperation context to disambiguate between different task parametrizations, making the operators' work easier and faster, and how the system can be used as a fallback in case of communication break-down.

Underwater ROVs capable of teleoperated manipulation are typically tethered to the support vessel. Consequently, the communication bottleneck occurs between the support vessel and the onshore control center, typically connected over a satellite link.

Current field ROVs that target teleoperated manipulation use simple extending gripper arms to secure the ROV on a designated rail before the operator begins the manipulation phase. This way, the operators do not need to constantly adjust the control of the ROV body and can focus on the manipulation task at hand. The underwater vehicle dynamics, external disturbances of the underwater environment or the control of the ROV body are beyond the scope of this paper.

The rest of the paper is structured as follows. We discuss previous research in Sect. 2 and we present our approach in Sect. 3. In Sect. 4 we present two illustrative planar examples with 2 and 3 frames to aid in the reader's understanding of the approach. Section 5 describes the learning from demonstration procedure and the evaluation trials with a group of volunteers. Last Sect. 6 summarizes this work and presents directions for future work.

2 Related work

The use of learning techniques in the context of ROV teleoperation is novel (Palomeras et al. 2016). The traditional teleoperation approach requires the operator to directly command the different actuators of the ROV. The automation efforts in this field have largely been concerned with the control of the body of the ROV, e.g., how to home to a desired position or how to counter the effect of currents, while ROV manipulators are directly teleoperated.

In contrast, developing learning approaches to model and reproduce skills given motion examples has been a central topic of robotics research. A number of such *learning from demonstration* (LfD) approaches exist to date. One of the most widely used approaches is using *Dynamic Movement Primitives* (DMPs) (Ijspeert et al. 2013). DMPs are dynamical systems with simple convergence behaviour that are coupled with a learned non-linear function that modulates their output. This way, DMPs can provide adaptive motion representations that are easy to implement. For example, the approach in Palomeras et al. (2016) uses DMPs to model and synthesise a valve-turning motion for a 4-DoF arm. One drawback of standard DMPs is that a sequence of radial basis activation functions needs to be allocated manually (usually spread equally in time with a shared variance), and that each DoF of the system is separately described (synchronized by a phase variable), sometimes leading to sets of DMPs that have difficulty in capturing joint synergies when few basis functions are used.

A number of extensions to the DMP framework have been proposed in the current literature. Gams et al. (2014) extends the DMP framework by adding a modulation term that allows interaction with objects and the environment. Learning of this coupling term is performed with an iterative learning control algorithm. Other extensions include the work of Ude et al. (2014) and Pastor et al. (2011), with the former encoding orientation trajectories as DMPs, and the latter using a model of previous sensory information to modulate the target of learned DMPs.

An alternative LfD approach is to encode sets of motions as a Gaussian mixture model (GMM) and use Gaussian Mixture Regression (GMR) to regenerate motion. It was shown in Calinon et al. (2012) that a DMP can be recast as a GMR problem for a GMM with diagonal covariance, and that its natural extension to a GMM with full covariances can retrieve local coordination patterns. With this probabilistic form of DMP, the basis functions can also be learned from the demonstrations. GMM/GMR have been successful in representing sets of demonstrated examples that are time-indexed, this way using time as the Gaussian conditioning variable to perform the regression, see Calinon (2016) for an overview. Such systems are often used for learning models from a set of demonstrations but are somewhat restrictive in their generalization capability. An extension to the traditional GMM-based learning approach is the parametrization of the problem with a set of different coordinate systems (Calinon 2016). In this setting, learning is performed in multiple coordinate systems, whose information is fused through products of Gaussians used to generate the final motion. Our work in Zeestraten et al. (2017) shows how one can learn position and orientation trajectories of bimanual tasks and how the resulting task-parametrized GMM can generalize to different orientations of objects or tools.

Hidden Markov Models (HMMs) have been used in robotics in a number of approaches. For example, Lee and Ott proposed to combine HMM with GMR to cope with the poor duration modeling properties of standard HMMs (Lee and Ott 2010; Lee et al. 2010). Similarly, Chan et al. (2013) used GMR and HMM as part of a constrained manipulator visual servoing controller. Bowen and Alterovitz presented the combination of an HMM (for task representation) with a sampling-based motion planner to produce (asymptotically) optimal plans (Bowen and Alterovitz 2014). Kulic et al. (2008) used HMMs to incrementally group together human whole-body motions, using hierarchical agglomerative clustering, based on their relative distances in HMM space.

Often, the use of HMM-based approaches in robotics applications is limited by the simplistic state duration modeling that HMMs provide. Other signal processing related disciplines, such as speech synthesis, have developed a number of models that seek to model state duration information more explicitly (for an overview see Rabiner 1989). One such model is the Hidden Semi-Markov Model (HSMM), see Yu and Kobayashi (2006) for a review. Recently we experimented with the use of HSMM in robot applications, by contrasting it to a set of different graphical model based approaches (Calinon et al. 2011). HSMMs are relevant for robot motion generation because they model the transitions and the durations of state activations, thus providing a relative time instead of a global time representation. In Tanwani and Calinon (2016), we exploited this local duration representation for autonomously learning and reproducing in new situations the tasks of opening a valve and moving objects while avoiding obstacles. In Havoutis and Calinon (2016), we showed how a similar task representation can be used to learn assistive behaviours of recurrent tasks. In Havoutis

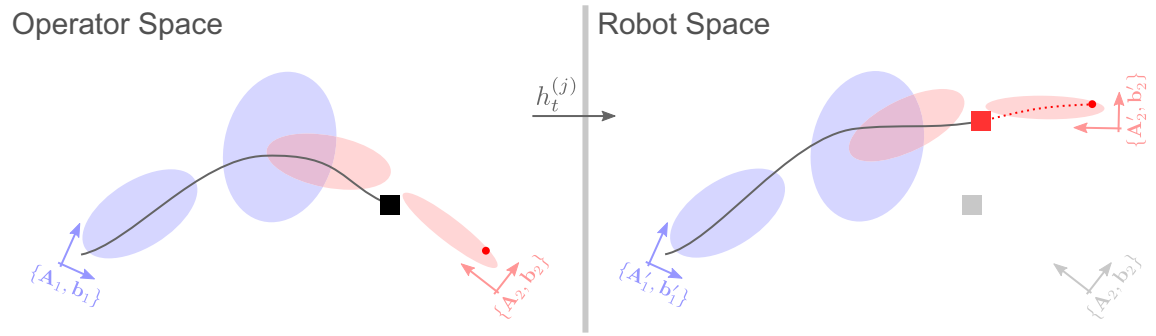

Fig. 3 Sketch of our approach in a 2-dimensional case with 2 coordinate systems and two Gaussians per coordinate system. The red dot represents the endpoint of the motion. On the left, the setup that is available to the operator. The solid line is the motion performed up to now (until time t) and the black square is the current state of the operator's end-effector. On the right, the (remote) space where the robot performs the task. Note that the coordinate system parametrization between the two spaces is different. The solid line represents the motion performed until time t while the red square represents the current state or the robot's end-effector. The coordinate system importance, $h_t^{(j)}$, is computed on the operator space and passed to the robot space. The red dotted line is the motion predicted by the model from the current robot state. The gray square in the robot space represents what the robot end-effector state would be with a direct teleoperation behavior, which would only poorly match the current situation in the robot space (Color figure online)

et al. (2016) and Havoutis and Calinon (2017), we showed how such a model can be learned in an online manner, and be used in a teleoperation scenario with failing communication, to semi-autonomously perform an ROV task (*hot-stabbing*) using an MPC formulation for motion generation.

This paper is extending this line of research by exploiting the strengths of the task-parametrized representation in a mixed teleoperation setting. While in Havoutis and Calinon (2017), we focused on learning a TP-HSMM online in a supervisory teleoperation setting, we present here a different but complementary approach in a shared control setting. By relying on task-adaptive statistical models of movements and a batch training with an infinite horizon LQT formulation, we show how such a teleoperation strategy can be used to resolve differences between local and remote environment configurations. With such decoupling of the operator's and the robot's spaces in the statistical representation, we demonstrate that the proposed approach can replace the conventional use of video streams in teleoperation, with a minimal exchange of activation weights as communication overhead. We also extend our analysis with a user study, highlighting the benefit of using our model both in terms of time to complete the task and resulting trajectory quality.

3 Approach

Here we present the TP-HSMM representation used throughout this work. This model is an extension of a TP-GMM in which the HSMM also represents the temporal evolution of the motion. Our model is trained from a set of demonstrations with varying task parameters.

The task parameters can be regarded as coordinate systems relevant to the task at hand. For example, consider the task of pushing a button with a manipulator. One task parameter can be the pose of the manipulator base in a globally defined coordinate system, while a second task parameter can be the pose of the button in the global coordinate system. Varying the pose of the button and/or the pose of the base of the manipulator would result in a different adaptation of the motion to push the button. Task parameters in our model are represented by P coordinate systems, defined at time step t by $\{\boldsymbol{b}_{t,j}, \boldsymbol{A}_{t,j}\}_{j=1}^{P}$, representing the transformation matrix for each coordinate system (see toy example in Fig. 3).

Each demonstration $m \in \{1, \dots, M\}$ contains T_m datapoints forming a dataset of N datapoints $\{\boldsymbol{\xi}_t\}_{t=1}^{N}$ with $N = \sum_m^M T_m$. The demonstrations $\boldsymbol{\xi} \in \mathbb{R}^{D \times N}$ are observed from the perspective of each of the different coordinate systems, forming P trajectory samples $\boldsymbol{X}^{(j)} \in \mathbb{R}^{D \times N}$. These samples can be collected from each coordinate system individually or can be computed with

$$\boldsymbol{X}_t^{(j)} = \boldsymbol{A}_{t,j}^{-1}(\boldsymbol{\xi}_t - \boldsymbol{b}_{t,j}). \tag{1}$$

For example, a demonstrated motion is provided in the global coordinate system and is later projected to the local coordinate system of the manipulator base and the coordinate system of the button.

The parameters of a TP-HSMM with K components are defined by

$$\left\{\{\pi_i, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^{P}, \boldsymbol{\mu}_i^{\mathcal{D}}, \boldsymbol{\Sigma}_i^{\mathcal{D}}\}_{i=1}^{K}, a_{i,j}\right\},$$

where π_i are the mixing coefficients, $\boldsymbol{\mu}_i^{(j)}$ and $\boldsymbol{\Sigma}_i^{(j)}$ are the center and covariance matrix of the ith Gaussian component in coordinate system j, $\mu_i^{\mathcal{D}}$ and $\Sigma_i^{\mathcal{D}}$ are univariate Gaussian distributions that directly model the duration of each state, and $a_{i,j}$ the transition probabilities between states.

3.1 Parameter estimation

Learning of the TP-HSMM parameters is performed in two steps. First the parameters of the spatial part of the model are estimated, and second the encoding of the temporal aspect of the data is performed.

First we estimate the priors, centers and covariances, $\{\pi_i, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^{P}, \}_{i=1}^{K}$, of the model with an *expectation-maximization* (EM) algorithm (Dempster et al. 1977) that are iteratively computed until model convergence, see "Appendix I".

For the duration aspect of TP-HSMM, we need to estimate the center and covariance for each state duration as a distribution with parameters $\{\boldsymbol{\mu}_i^{\mathcal{D}}, \boldsymbol{\Sigma}_i^{\mathcal{D}}\}_{i=1}^{K}$, as well as the transition probabilities. $a_{i,j}$ can be arranged as a $K \times K$ transition probability matrix, where each element represents the probability to move to state q_j, while currently being in state q_i. For all demonstrations, given the spatial model that was learned above, we can compute the state probabilities of every datapoint. This way, for each datapoint $\boldsymbol{\xi}_t$ we can estimate the state q_j and the previous state q_i. To build up the transition probabilities, $a_{i,j}$, we keep a log of $c_{i,j} \in \mathbb{R}^{K \times K}$, counting the number of state transitions that are not self-transitory, by performing a pass through each demonstration, namely

$$\forall \{\boldsymbol{\xi}_{t-1}, \boldsymbol{\xi}_t\} \Rightarrow c_{i,j} = c_{i,j} + 1, \qquad i \neq j$$
$$a_{i,j} = \frac{c_{i,j}}{\sum_{j=1}^{K} c_{i,j}}.$$

When computing the transition probabilities, we only need to keep track of the non self-transitory instances, as we are modeling the relative time during which the system will stay at each state. For simplicity, we use a univariate Gaussian distribution $\mathcal{N}(\mu_i^{\mathcal{D}}, \Sigma_i^{\mathcal{D}})$ to model this duration, but other distributions that would better model durations are possible. Hence we bypass the computationally expensive HSMM batch training procedure and replace it with an iterative approach keeping statistics over the state transitions, which showed in practice to be a reasonable approximation in our experiments. This way, as we add each demonstration, we keep track of each state duration and accordingly update the statistics of each state. This is done using a running statistics method to compute the mean and variance for each state duration. This requires that we only keep track of the total number of samples while we incrementally add new observations.

The total computation time for learning a model with 2 coordinate systems, by providing 6 demonstrations would amount to approximately 3 s on commodity hardware. Sampling as detailed below is much faster and can be computed online (below 1 ms).

3.1.1 Orientation data

We use unit quaternions to represent orientation data, as in Silvério et al. (2015). We chose this representation as it is singularity-free and uses only 4 parameters. We define a unit quaternion as $\boldsymbol{\epsilon} = [v, \ \boldsymbol{u}^\top]^\top$, with $v \in \mathbb{R}$ the scalar and $\boldsymbol{u} \in \mathbb{R}^3$ the vector part of the representation. Accordingly, the conjugate quaternion is defined as $\bar{\boldsymbol{\epsilon}} = [v, \ -\boldsymbol{u}^\top]^\top$. To describe the orientation of one coordinate system with respect to another we use the quaternion product which is defined as

$$\boldsymbol{\epsilon}_1 * \boldsymbol{\epsilon}_2 = \begin{bmatrix} v_1 v_2 - \boldsymbol{u}_1^\top \boldsymbol{u}_2 \\ v_1 \boldsymbol{u}_2 + v_2 \boldsymbol{u}_1 + \boldsymbol{u}_1 \times \boldsymbol{u}_2 \end{bmatrix}. \tag{2}$$

The quaternion product can also be implemented in matrix-vector multiplication form as $\boldsymbol{\epsilon} = \boldsymbol{\mathcal{E}}_1 \boldsymbol{\epsilon}_2$, where

$$\boldsymbol{\mathcal{E}}_1 = \begin{bmatrix} v_1 & -u_{1,1} & -u_{1,2} & -u_{1,3} \\ u_{1,1} & v_1 & -u_{1,3} & u_{1,2} \\ u_{1,2} & u_{1,3} & v_1 & -u_{1,1} \\ u_{1,3} & -u_{1,2} & u_{1,1} & v_1 \end{bmatrix}, \tag{3}$$

is a quaternion matrix built from the elements of the quaternion $\boldsymbol{\epsilon}_1$. This way we can directly use this representation of orientations under the TP-HSMM formulation by setting $\boldsymbol{\xi}_n = \hat{\boldsymbol{\epsilon}}_n$, $\boldsymbol{b}_{n,j} = 0$ and $\boldsymbol{A}_{n,j} = \boldsymbol{\mathcal{E}}_{n,j}$ for the orientation part of the task parameters, see Silvério et al. (2015) for details. The projection of Eq. (1) then becomes $\boldsymbol{\mathcal{E}}_{n,j}^{-1} \hat{\boldsymbol{\epsilon}}_n$ and maps the reference orientation (demonstration) $\hat{\boldsymbol{\epsilon}}_n$ to coordinate system j. This allows us to seamlessly integrate orientation data to our learning approach and encode both position and orientation of demonstrated motions.

3.2 Reproduction

The learned TP-HSMM is used to generate motions given a new set of coordinate system parameters $\{\hat{\boldsymbol{b}}_{t,j}, \hat{\boldsymbol{A}}_{t,j}\}_{j=1}^{P}$. This is done in two steps. First we generate a GMM with parameters $\{\pi_i, \hat{\boldsymbol{\mu}}_{t,i}, \hat{\boldsymbol{\Sigma}}_{t,i}\}_{i=1}^{K}$ where

$$\mathcal{N}\left(\hat{\boldsymbol{\mu}}_{t,i}, \hat{\boldsymbol{\Sigma}}_{t,i}\right) \propto \prod_{j=1}^{P} \mathcal{N}\left(\hat{\boldsymbol{\mu}}_{t,i}^{(j)}, \ \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)}\right), \text{ with}$$
$$\hat{\boldsymbol{\mu}}_{t,i}^{(j)} = \boldsymbol{A}_{t,j} \boldsymbol{\mu}_i^{(j)} + \boldsymbol{b}_{t,j}, \quad \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)} = \boldsymbol{A}_{t,j} \boldsymbol{\Sigma}_i^{(j)} \boldsymbol{A}_{t,j}^\top, \tag{4}$$

where the result of the Gaussian product is given by

$$\hat{\boldsymbol{\Sigma}}_{t,i} = \left(\sum_{j=1}^{P} \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)^{-1}} \right)^{-1}, \quad \hat{\boldsymbol{\mu}}_{t,i} = \hat{\boldsymbol{\Sigma}}_{t,i} \sum_{j=1}^{P} \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)^{-1}} \hat{\boldsymbol{\mu}}_{t,i}^{(j)}. \tag{5}$$

Next we generate the state sequence by recursively computing the probability of the datapoint $\boldsymbol{\xi}_t$ to be in state i at time step t, given the partial observation $\{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_t\}$, using the forward variable $\alpha_{i,t}^{\text{HSMM}}$ as in Rabiner (1989) with

$$\alpha_{i,t}^{\text{HSMM}} = \sum_{j=1}^{K} \sum_{d=1}^{d^{\max}} \alpha_{j,t-d}^{\text{HSMM}}\, a_{j,i}\, \mathcal{N}_{d,i}^{\mathcal{D}} \prod_{s=t-d+1}^{t} \mathcal{N}_{s,i}, \quad \text{where}$$
$$\mathcal{N}_{d,i}^{\mathcal{D}} = \mathcal{N}(d|\mu_i^{\mathcal{D}}, \Sigma_i^{\mathcal{D}}) \text{ and } \mathcal{N}_{s,i} = \mathcal{N}\big(\boldsymbol{\xi}_s|\, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\big),$$

that is computed with an iterative procedure, described in "Appendix II". With this representation, a generative process can be constructed by setting equal observation probability $\mathcal{N}_{s,i} = 1\ \forall i$, i.e. the spatial part of the motion representation. This yields a step-wise state reference that we use in the subsequent optimal control formulation.

3.3 Optimal control trajectory generation

We formulate the trajectory generation step as an optimal control problem. We use a double integrator system as a virtual unit mass attached to the end-effector of the controlled arm. This system has the form

$$\dot{\boldsymbol{\xi}}_{t+1} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \boldsymbol{\xi}_t + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{u}_t. \tag{6}$$

Using an infinite-horizon linear-quadratic tracking (LQT) controller, we can generate a trajectory to smoothly track the previously computed stepwise reference state sequence, $q_1 \ldots q_T$.[1] The step-wise reference trajectory $\mathcal{N}(\hat{\boldsymbol{\mu}}_{q_t}, \hat{\boldsymbol{\Sigma}}_{q_t})$ is used to set up a cost function that trades-off accuracy to control effort according to the demonstrated motions (Calinon et al. 2014). The cost function to minimize at each time step t is given by

$$c(\boldsymbol{\xi}_t, \boldsymbol{u}_t) = \sum_{t=t_0}^{\infty} (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_{q_t})^{\top} \boldsymbol{Q}_t (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_{q_t}) + \boldsymbol{u}_t^{\top} \boldsymbol{R} \boldsymbol{u}_t, \tag{7}$$

where $\boldsymbol{u}_t \in \mathbb{R}^m$ is the control input of the system. Setting $\boldsymbol{Q}_t = \hat{\boldsymbol{\Sigma}}_{q_t}^{-1}$, $\boldsymbol{R} \succ 0$, $\boldsymbol{\xi}_t = [\boldsymbol{x}_t^{\top}, \dot{\boldsymbol{x}}_t^{\top}]^{\top}$, $\hat{\boldsymbol{\mu}}_{q_t} = [\hat{\boldsymbol{\mu}}_t^{x^{\top}}, \hat{\boldsymbol{\mu}}_t^{\dot{x}^{\top}}]^{\top}$ with $\boldsymbol{x}, \dot{\boldsymbol{x}}$ representing the position and velocity of the system, the optimal control input $\boldsymbol{u}_t^*$ is obtained by solving the algebraic Riccati equation (see "Appendix III" for details), yielding full stiffness and damping matrices that are regulated in accordance to the precision required by the task, as learned by the demonstrations.

[1] A model predictive control (MPC) formulation can alternatively be employed as in Zeestraten et al. (2016), generating the state sequence in a receding horizon manner.

3.4 Arm control

For controlling the arm, we chose a torque control approach. This allows us to naturally regulate the compliance of the controlled arm, according to the full stiffness and damping terms computed by the above LQT solution. The torque command that the robot low-level controller tracks is

$$\boldsymbol{\tau} = \boldsymbol{J}^{\top} \boldsymbol{f} + \boldsymbol{\tau}_G, \tag{8}$$

where $\boldsymbol{J}$ is the arm Jacobian, $\boldsymbol{f} = [\boldsymbol{f}_p^{\top}, \boldsymbol{f}_o^{\top}]^{\top}$ is a wrench and $\boldsymbol{\tau}_G$ are the gravity compensation terms computed based on the manipulator model and state. Given the desired pose of the arm end-effector, $\hat{\boldsymbol{\xi}}_t = [\hat{\boldsymbol{x}}, \hat{\boldsymbol{\epsilon}}]^{\top}$,

$$\boldsymbol{f}_p = \boldsymbol{K}_t^{\mathcal{P},x} (\hat{\boldsymbol{x}}_t - \boldsymbol{x}) - \boldsymbol{K}_t^{\mathcal{V},x} \dot{\boldsymbol{x}}, \tag{9}$$
$$\boldsymbol{f}_o = \boldsymbol{K}_t^{\mathcal{P},\epsilon} 2 \log(\hat{\boldsymbol{\epsilon}}_t * \boldsymbol{\epsilon}) - \boldsymbol{K}_t^{\mathcal{V},\epsilon} \boldsymbol{\omega}, \tag{10}$$

where $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity of the end-effector. The quaternion product, $\hat{\boldsymbol{\epsilon}}_t * \boldsymbol{\epsilon}$, computes the orientation error between desired and current orientation, and the logarithmic map, $\log(\cdot)$, converts the quaternion error to an axis-angle difference. As outlined above, the computed LQT gains can be used when tracking a generated trajectory. When the system is at a mixed teleoperation mode, the stiffness and damping terms are heuristically set to a desired manipulator compliance.

3.5 Task model based discrepancy resolution

The ROVs operate in an environment that is often dynamic. In most operation scenarios a model of the main structures of the environment, where the ROV will be operating at, is known a priori. Nonetheless, such models are often only partially accurate.

A naive (VR) teleoperation approach would require real-time knowledge of all the task-relevant components. This way the ROV would either need to locally recognize the relevant components or send back a video feed for this operation to be carried on-shore. As the relevant information is processed, all coordinate systems will have to be readjusted so that the operator's space (VR) parametrization provides a consistent match with the real-world configuration that the ROV is operating in (robot space).

Our approach decouples the configuration of the operator's (VR) space and the configuration of the real environment (robot space), where the ROV is operating at. This way we can locally resolve discrepancies/inconsistencies between the parametrizations of coordinate systems, without the need to transmit such information or readjust the operator's (VR) environment. Hence the operator's space (VR environment) keeps only a local configuration of the environment and the operator is performing the task in this setup. Here we make

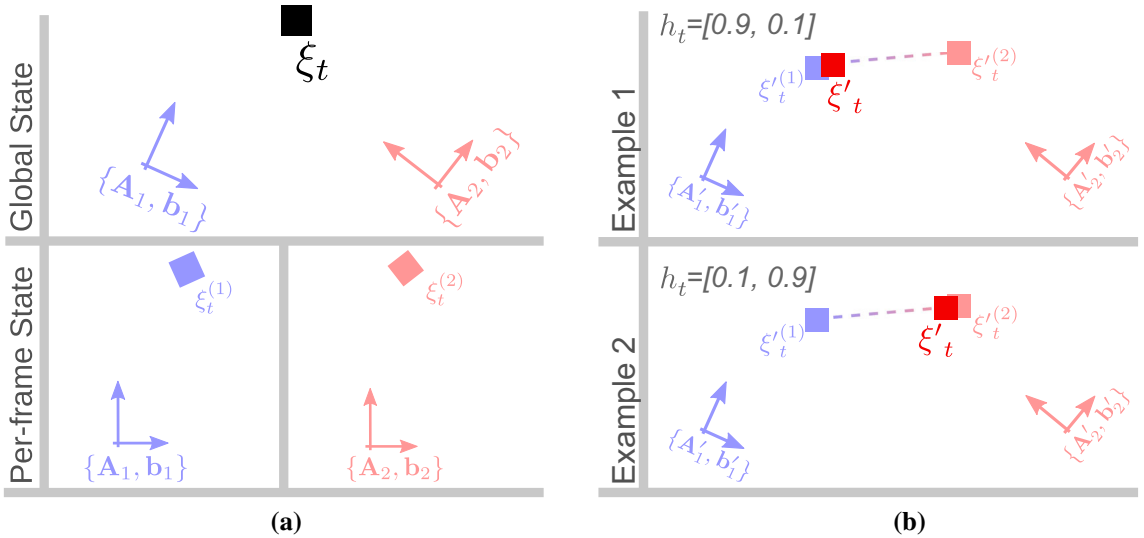

Fig. 4 **a** The state on the operator's space: (top row) considered globally and (bottom row) as seen from each of the 2 frames in the example. **b** Two examples of different frame importances in the robot space. In example 1 (top row), frame 1 is weighted more, while in example 2 (bottom row) frame 2 is considered more important. The global prediction in the robot side (red square) is the importance-weighted average of the per-frame predictions (Color figure online)

the implicit assumption that the same set of reference frames is available on both sides, a feature that is trivial to enforce programmatically given the learned task model. The ROV operates in its local configuration while we resolve the difference between the two spaces using the learned task model.

We do this by computing the probability of the state of the system to belong to each coordinate system in the model, on the operator's side. We calculate the probability of the current state, projected on the different coordinate systems, to belong to the model of each coordinate system. This is computed as

$$P(\boldsymbol{\xi}_t, j) = \sum_{i=1}^{K} \pi_i \, \mathcal{N}\left(\boldsymbol{\xi}_t^{(j)} \middle| \, \boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\right), \tag{11}$$

$$h_t^{(j)} = \frac{P(\boldsymbol{\xi}_t, j)}{\sum_{j=1}^{P} P(\boldsymbol{\xi}_t, j)}, \tag{12}$$

which can be interpreted as the relevance of each coordinate system in the current position. By treating these coordinate system probabilities as a measure of importance, we can then look at the related state of the system on the operator's side, and reconstruct the global state in the ROV space according to this information.

In steps, this is done by first computing the frame-local state of the global state on the operator's side, using the frames' parameters as

$$\boldsymbol{\xi}_t^{(j)} = \boldsymbol{A}_t^{(j)\top}(\boldsymbol{\xi}_t - \boldsymbol{b}_t^{(j)}). \tag{13}$$

for each frame $j = 1, \ldots, P$. Next, we compute a global robot state, $\boldsymbol{\xi'}_t^{(j)}$, for each of the robot frame's with

$$\boldsymbol{\xi'}_t^{(j)} = \boldsymbol{A'}_t^{(j)} \boldsymbol{\xi}_t^{(j)} + \boldsymbol{b'}_t^{(j)} \tag{14}$$

where $[\,]'$ denotes the parameters of the robot side frames that can differ from the operator's side. This is the per-frame estimate of the global (desired) robot state. To arrive at the final prediction we compute a weighted sum of all frames' estimates, in the form

$$\boldsymbol{\xi'}_t = \sum_{j=1}^{P} h_t^{(j)} \boldsymbol{\xi'}_t^{(j)} \tag{15}$$

We thus transform each coordinate system state representation to the global coordinate system and compute an average, weighted by the coordinate system importance that is communicated to the robot side. Figure 4 helps explain the relevant variables and gives an intuitive presentation of the above steps.

4 Planar examples

We use two planar examples to illustrate our approach. The task in both cases is to start from the "U" shape on the left of the panel and reach the "U" shape on the right side. In the case of 3 frames, the path needs to pass from the area between the two lines in the middle, representing a cross-section of a cylinder, as an additional constraint to the task. In both example, each shape represents a frame. The operator

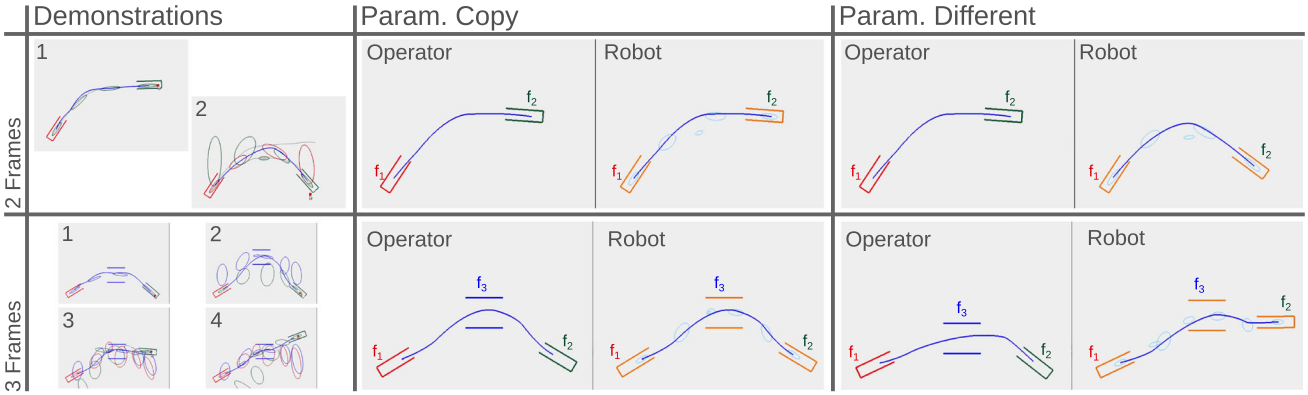

Fig. 5 Planar examples of our developed mixed-teleoperation framework with tasks involving 2 frames (top row) and 3 frames (bottom row). All frames' parameters can vary. In particular, the parameters of frame 2 differ in the first example (top row), while in the second example (bottom row) both frame 2 and 3 are in different configurations with respect to the operator's space. Please see Sect. 4 for details. The accompanying video presents the full sequences for both examples

uses a mouse cursor to interact with the local side and provide motion demonstrations. Snapshots are presented in Fig. 5 and the accompanying video presents the full sequence.

We begin by learning the task in the local side, where the operator demonstrates a number of motions that are encoded online. In the 2-frame case 2 demonstrations are provided, while for the 3-frame case the operator demonstrates 3 motions. The models are being built online and the blue line represents the current motion prediction of the system.

The middle column of Fig. 5 shows the prediction of the model in the operator (local) and robot (robot) space while the parametrization of the frames remains identical. This way as the operator executes the motion, the robot on its side follows the same path (red square in the accompanying video). Next, we change the parametrization on the robot's space to demonstrate the behavior of the system in a scenario where the parametrization of the corresponding frames differs. In the 2-frame example we change the configuration of frame 2 in the robot's space, while in the 3-frame example we want to highlight that all frame parametrizations can change and we change both frame 2 and 3 in the robot space. In this example, the blue line represents the predicted motion in the now different parametrization. As the operator performs her version of the task on the local panel, the system can correctly compensate for the inconsistency. This way, on the 2-frame example, the state can successfully reach the (differently configured) target and in the 3-frame case, the remote state correctly passes through the cylinder and reaches the target, even if the configurations between the two spaces differ.

5 Experimental evaluation

This section describes the learning of a task model and presents two examples of how such a model can be used

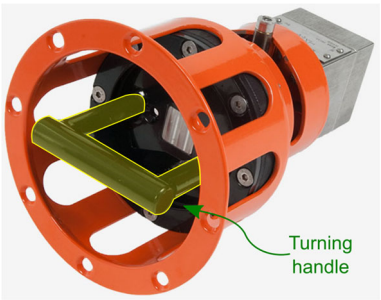

Fig. 6 Example of standardised ROV-operated rotary valve. Valve handle highlighted in yellow (Color figure online)

in the context of underwater teleoperation. We show how a model learned with our approach can be used to generate novel motions in unseen situations whenever we want the ROV to perform the task autonomously or when communication with the operator breaks down. In addition we show how the learned model can be used to map the control of the operator to the real environment parametrization of the ROV. This way we can decouple the two control spaces and perform control only locally with minimal communication overhead (i.e. operator state and coordinate system importances, in contrast to video feeds, telemetry, joint states, etc.).

5.1 Use case

As a use case we have selected an experimental setup that resembles an ROV rotary control valve, see Fig. 6. ROVs often need to operate such control valves, that are crucial for sealing or pressurizing parts of hydraulic circuits. As we do not have access to a real ROV valve, we use a symmetric handle as a mock-up (Fig. 7). The task in this setup is to reach the control valve (handle), placed on the robot side (right arm of the robot), using the left arm of the robot as the input device

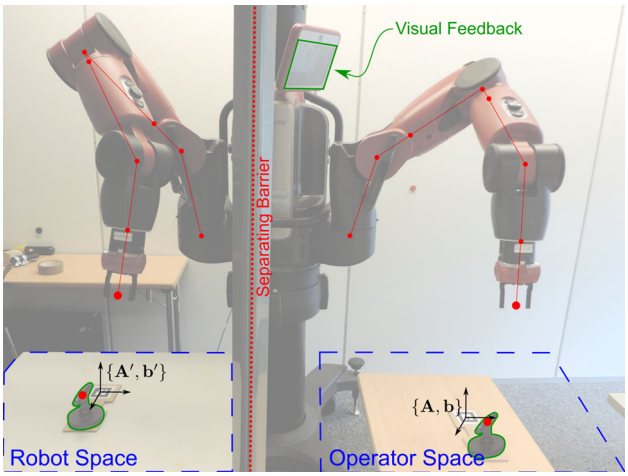

Fig. 7 Experimental setup as a teleoperation mock-up using the Baxter robot. In mixed teleoperation mode the pose of the coordinate systems in operator and robot space are decoupled

(master arm) and optionally turn it 90° clockwise. The valves on the operator's space and the robot's space can move freely. For the direct teleoperation case, the camera mounted in the teleoperated arm's wrist is used to provide visual feedback to the operator.

5.2 Model learning

We use kinesthetic demonstrations to collect a set of trajectories with an operator performing a valve reaching and turning task. The trajectories consist of end-effector poses, containing a Cartesian position and a quaternion orientation part. Accordingly, two coordinate systems are set: the base of the arm and the valve. The coordinate system of the arm is set to the base of the operator's arm with an orientation equal to the orientation of the global coordinate system. The second coordinate system, attached to the valve, is free-floating in the operator's space and is being visually tracked using a fiducial marker[2] (Fig. 7), by using the camera embedded within the robot's hand. This camera has wide lens and produces a stream at 25 frames per second. It is placed to look over the two fingers of the standard Baxter gripper, i.e. the gripper fingers are always visible.

We use a torque controller compensating for the effect of gravity on the arm to let the operator directly guide the arm to the position and orientation of the valve (handle). We collected 7 movements of the operator performing the task and used 5 demonstrations to learn a model with the procedure described in Sect. 3. The remaining 2 trajectories are kept for model testing and evaluation.

5.3 Model evaluation

We learn a TP-HSMM for the valve task, setting the number of states $K = 5$ empirically. A number of techniques can alternatively be used for model selection, for example with a Bayesian information criterion (Schwarz 1978) or a Dirichlet process (Rasmussen 2000). The learned TP-HSMM for the valve task is shown in Fig. 8, where both the Gaussian states per coordinate system and the state connectivity are depicted.

We first use the learned TP-HSMM with the same set of task parameters as in the demonstrations to generate a trajectory. We compare the trajectories that our model generates against the demonstrated "ground truth", for both the training samples and the test samples that were not used in learning. The RMSE for the training set is 0.047 m and 0.31 rad, while the RMSE for the test set is 0.053 m and 0.38 rad, for position and orientation accordingly.

Figure 9 shows positions and orientation trajectories for all demonstrations in black lines, one of the test trajectories in red and a trajectory generated by the TP-HSMM with the same coordinate system parametrization in blue. This figure highlights that the learned model can generalize to novel parametrizations (not "seen" during learning) and successfully perform the encoded task in an unseen situation.

5.4 Mixed teleoperation evaluation trials

To directly demonstrate the benefit of using our approach we organized evaluation trials with a set of 5 volunteers. The volunteers have some experience using the robot arm but have no specific prior experience of the proposed teleoperation setup.

We distinguish two cases; first *direct teleoperation* of the robot arm using visual feedback, second *mixed teleoperation* where the operator performs the task locally (in the operator space) and the correspondence between operator space and robot space parametrization is handled by the proposed approach.

5.4.1 Protocol

All the volunteers start with 1 minute of direct teleoperation to familiarize with the use of the robot arm under gravity compensation. Next the volunteers are asked to perform the valve task in direct teleoperation, where they can directly see the robot arm. Next, a separating barrier is placed between the operator's space and the robot's space. The volunteers are asked to perform the valve task with visual feedback from the wrist-mounted camera on the robot side. After these familiarization steps, we begin the main set of trials.

[2] Please note that fiducial markers are used here to simplify the perception part of the problem. These will be replaced at a later stage with a specialized model-based underwater perception solution, that is contributed by another partner within the DexROV consortium.

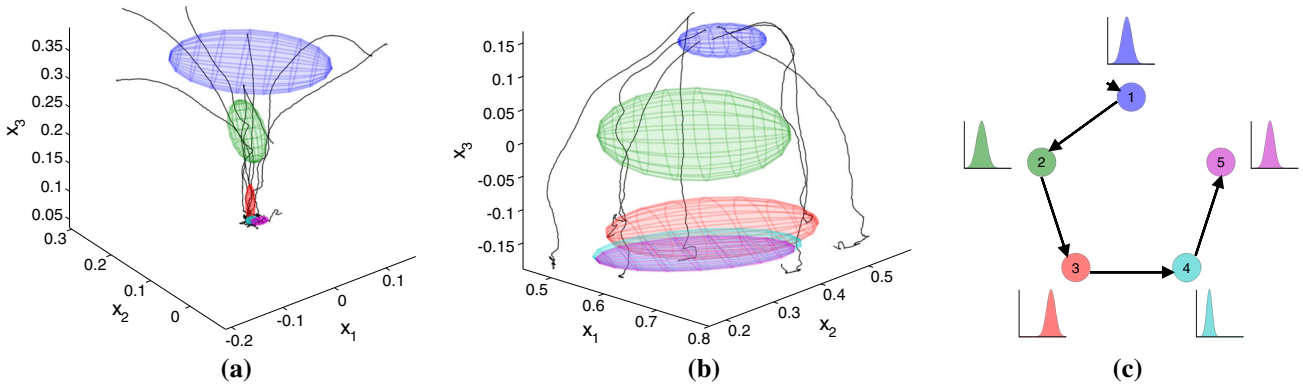

Fig. 8 The TP-HSMM representation learned from the set of demonstration trajectories. **a** The Gaussians represented in the coordinate system of the valve (isocontour of one standard deviation), where all demonstrations converge to the handle. **b** The Gaussians learned in the coordinate system of the robot arm base. Note that the variance in this coordinate system is higher, leading to larger ellipsoids. **c** The learned state transition graph with the corresponding state duration probabilities. Note that the state colors are consistent across the images, and that the training samples are displayed in both coordinate systems in black color

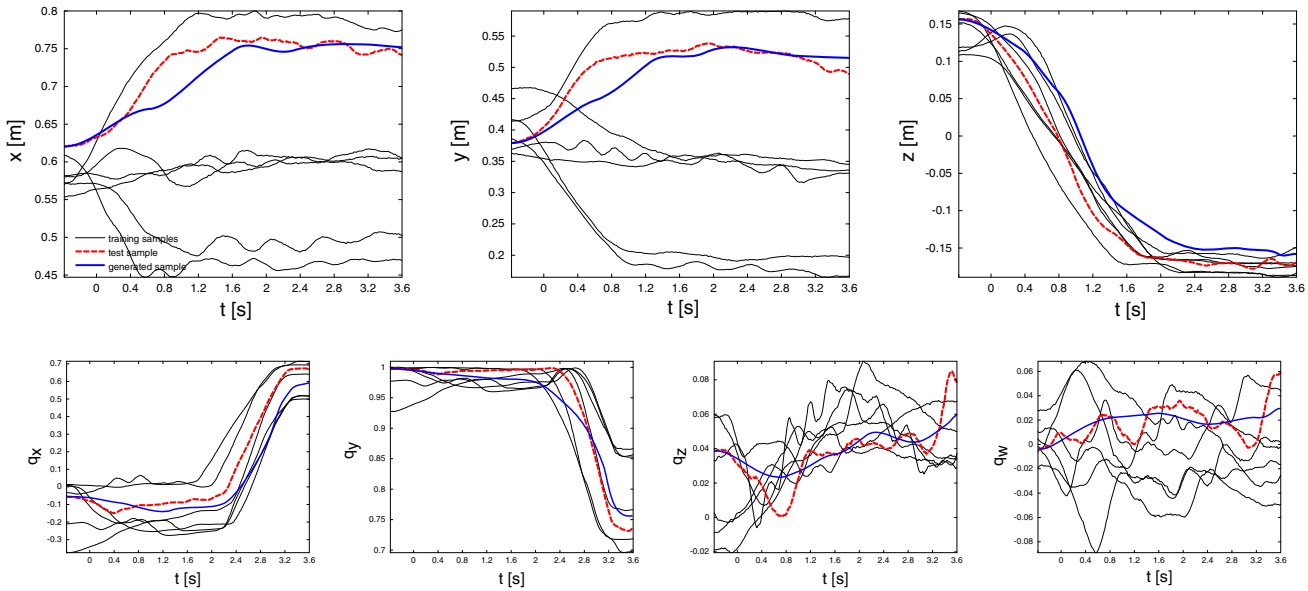

Fig. 9 Pose of the robot arm end-effector during autonomous reproduction of the learned task. Black lines are training samples, the red lines show one sample from the testing set, and the blue lines show the motion generated from the TP-HSMM given the same task parametrization as the test sample (Color figure online)

The volunteers are asked to perform the valve reaching task from an approximately similar starting pose, while only visual feedback from the wrist camera is available through the monitor of the robot. The valve (handle) is placed each time at a random position, within reachable limits, in the robot space. The trial is considered successful once the handle is placed within the distance between the robot fingers, signaled by the trials' supervisor. Note that no reference exists in the operator's space at this time. This direct teleoperation trial is repeated 3 times with different configurations, where both operator arm and robot arm end-effector poses are collected.

Next a valve reference (handle) is placed at a random position in the operator's space and the volunteers are asked to perform the valve task using this local task reference. Note that the pose of the local valve (operator space) does not correspond to the pose of the remote valve (robot space), i.e. the coordinate system parametrizations between the two spaces are different. During this period, our system uses the reference information in both spaces, the operator's input and the learned model to disambiguate the end-effector target pose

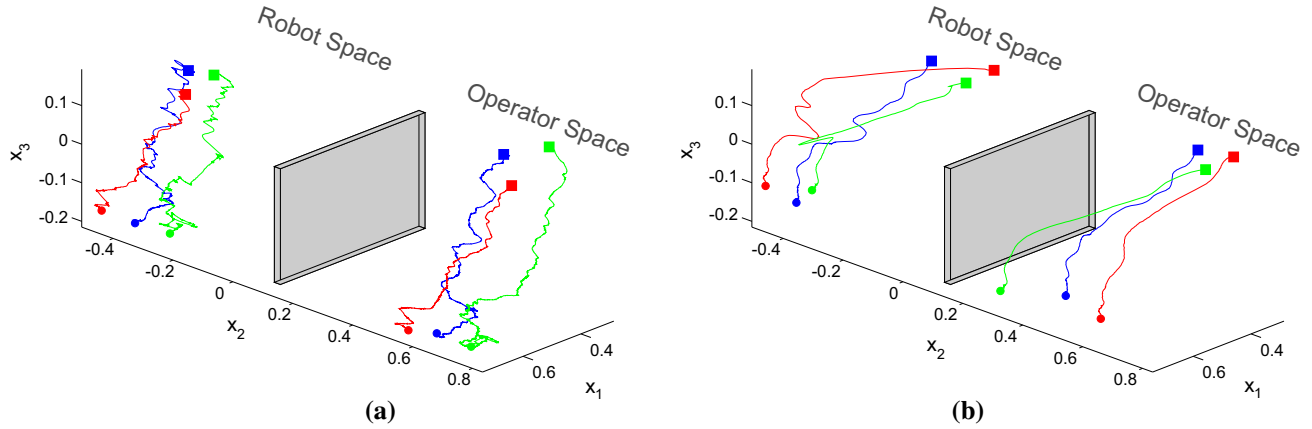

Fig. 10 Direct and mixed teleoperation trials from the evaluation study. **a** Direct teleoperation mode, the robot arm directly tracks the pose of the operator's arm. The operator receives visual feedback of the task execution from a camera in the robot space. **b** Mixed teleoperation mode where the operator performs a version of the task locally, and the learned task model is used to map his/her input to the control of the robot arm. Note that, in contrast to the direct teleoperation mode, the task coordinate systems in robot and operator spaces are not coupled. In principle the operator can be performing the exact same task locally while the robot on the remote side is successfully executing a different version of the task (i.e. with other task parametrization)

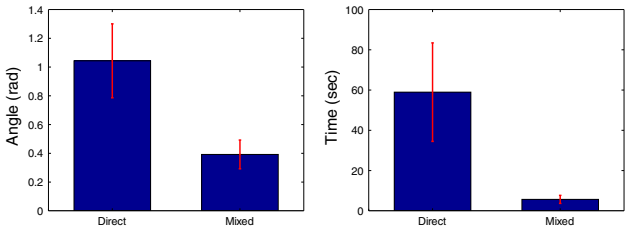

Fig. 11 Left: Average change in direction per performed motion, averaged over all trials per teleoperation mode. Right: Average time required to complete the valve task for each teleoperation mode. The red errorbars represent ± 1 standard deviation (Color figure online)

for the teleoperated arm. This way the operator is performing the task locally while the remote arm is performing a "different version" of the task successfully, using the approach described in Sect. 3.5. Again the end of each trial is signaled by the supervisor, as soon as the handle is placed between the robot fingers. As before, the mixed teleoperation trial is repeated 3 times in different configurations, where the end-effector poses of both arms are collected. During this, no visual feedback from the remote side is available to the operator, i.e. the monitor of the robot displays a black screen.

5.5 Results

The evaluation trials, as previously described, yield 30 teleoperation trajectories, 15 trajectories for each teleoperation mode. Figure 10 presents 3 samples from each mode. Qualitatively, the direct teleoperation motions appear coarse in comparison to the mixed teleoperation motions. This reflects the strategy that all volunteers used when performing the task, with small incremental movements where each action was planned accordingly to the visual feedback from the previous action. In contrast, in the mixed teleoperation samples, where the teleoperator had direct access to a local feedback, we observed that motions are consistently smooth and fluid for all samples. The average directional change per motion and the average over the set of trials for each teleoperation mode are computed. This provides a measure of smoothness of the performed motions, the results of which are summarized in Fig. 11a.

Assisting the operator by providing immediate local feedback about the task, by completing a local version of the task, resulted in a significant improvement of the time required to complete the valve reaching task.[3] The direct teleoperation trials required on average 1 min (58.9 ± 24.4 s) to complete the task. In contrast, the volunteers completed the task in the proposed mixed teleoperation setting in under 6 seconds on average and more consistently (5.69 ± 1.9 s). These results are summarized in Fig. 11b.

5.6 Communication disruption

In addition to the significant reduction of the task execution time, we can also use the learned task model to generate motions that achieve the task at hand when the communication is disrupted. An example of this situation is presented in Fig. 12. We consider the example in which the operator is performing the valve task in the mixed teleoperation mode when the communication between the two spaces is abruptly

[3] Note that ultimately, the target in the project is to replace the operator's space by an immersive VR setup rather than the teleoperation mock-up.

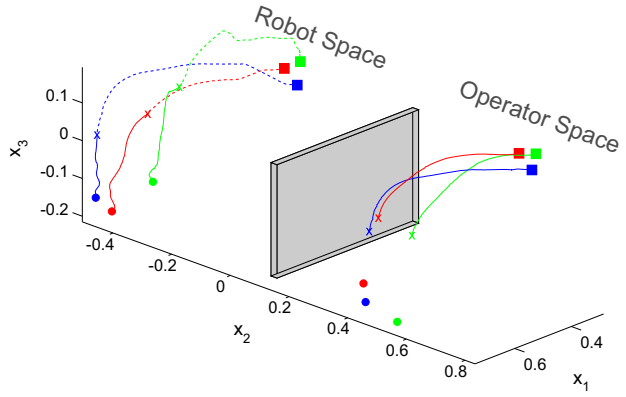

Fig. 12 Examples of mixed teleoperation trials where communication is disrupted. Dashed lines represent the robot end-effector motion until the communication is cut. Solid lines on the robot space are the motions that are generated from the TP-HSMM and executed by the robot autonomously

disrupted (see the × points on the trajectory in Fig. 12). In this situation, the robot arm on the remote side, having access to a local copy of the TP-HSMM, can still generate a motion from the current state and continue to perform the task autonomously. In turn, this feature can also be used for autonomous task execution, where the operator is presented with the generated motion and can select between teleoperation and autonomous task execution. The predicted motion can be presented either on the operator's monitor or in a VR system. The operator can then select whether she would like to have the robot execute the motion or proceed in a mixed teleoperation fashion. Execution can be triggered through the operator's interface, for example by confirming the action execution by clicking the appropriate button on the graphical user interface.

5.7 Discussion

In most mixed teleoperation trials, we observed that the system presents a transient behavior around the area where the probabilities of the coordinate systems are approximately equal. This is visible in Fig. 10b and is related to the shape of the Gaussians. Broader Gaussians (e.g. using a prior on minimal covariance used as regularization term) would result in smoother changes of coordinate system weighting, but might not be suitable for tasks requiring very high precision.

The reduction of task execution time based on the trials with volunteers highlights the support that our system can provide to non-experienced users. In a real world ROV teleoperation situation, an experienced ROV pilot operates the controls of the ROV and generally, the visual feedback is more extensive, i.e. multiple cameras from different viewpoints are traditionally used. Nonetheless such feedback can only be accessible to ROV operators off-shore, as communicating multiple video streams through a satellite link is—at least currently—infeasible. To perform the teleoperation from an on-shore site, in the context of the DexROV project, video streams are not available. In contrast, we showed how the use of a learning approach can make video streams redundant by decoupling the operator's and the robot's spaces with minimal communication overhead.

6 Conclusion

We presented a learning by demonstration approach for manipulation tasks that can be used in a teleoperation context to remotely operate underwater ROVs. We proposed a mechanism that can be used to disambiguate differences between the operator's and the robot's workspaces based on a learned task model. This way the operator is performing the task using local feedback while the robot performs a different version of the same task, based on the input of the operator. In addition, we showed how the system can be used as a fall-back in case of communication break-down—something common when operating through a satellite link—and proceed to execute the remainder of the task autonomously.

In future work, we aim to investigate ways of accommodating force and torque dimensions in our demonstrations, as such information is often important for successful interaction with the environment. Along the same line, we will investigate the role of force feedback to the operator and how such loop can be closed in cases where the operator and the robot spaces differ. Finally, we are currently exploring ways of inferring the operators intent while he/she is performing a task, and choosing which task to execute on the robot side from a set of tasks in a learned task library.

Acknowledgements Havoutis was supported by the UK EPSRC grants EP/M019918/1, EP/R026084/1 and EP/R026173/1.

Appendix I

The E and M steps have the following form.
E-step

$$h_{t,i} = \frac{\pi_i \prod_{j=1}^{P} \mathcal{N}\left(\boldsymbol{X}_t^{(j)} \middle| \boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\right)}{\sum_{k=1}^{K} \pi_k \prod_{j=1}^{P} \mathcal{N}\left(\boldsymbol{X}_t^{(j)} \middle| \boldsymbol{\mu}_k^{(j)}, \boldsymbol{\Sigma}_k^{(j)}\right)}. \tag{16}$$

M-step

$$\pi_i \leftarrow \frac{\sum_{t=1}^{N} h_{t,i}}{N}, \tag{17}$$

$$\boldsymbol{\mu}_i^{(j)} \leftarrow \frac{\sum_{t=1}^{N} h_{t,i}\, \boldsymbol{X}_t^{(j)}}{\sum_{t=1}^{N} h_{t,i}}, \tag{18}$$

$$\boldsymbol{\Sigma}_i^{(j)} \leftarrow \frac{\sum_{t=1}^{N} h_{t,i} \left(\boldsymbol{X}_t^{(j)} - \boldsymbol{\mu}_i^{(j)}\right)\left(\boldsymbol{X}_t^{(j)} - \boldsymbol{\mu}_i^{(j)}\right)^{\top}}{\sum_{t=1}^{N} h_{t,i}}. \tag{19}$$

These steps are the result of a log-likelihood maximization process subject to the constraint that the data in the different coordinate systems are drawn from the same source.

Appendix II

We compute the the forward variable, $\alpha_{i,t}^{\mathrm{HSMM}}$ as in Rabiner (1989) with

$$\alpha_{i,t}^{\mathrm{HSMM}} = \sum_{j=1}^{K} \sum_{d=1}^{d^{\max}} \alpha_{j,t-d}^{\mathrm{HSMM}}\, a_{j,i}\, \mathcal{N}_{d,i}^{\mathcal{D}} \prod_{s=t-d+1}^{t} \mathcal{N}_{s,i}, \quad \text{where}$$
$$\mathcal{N}_{d,i}^{\mathcal{D}} = \mathcal{N}(d|\mu_i^{\mathcal{D}}, \Sigma_i^{\mathcal{D}}) \text{ and } \mathcal{N}_{s,i} = \mathcal{N}\left(\boldsymbol{\xi}_s|\, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right),$$

with the initial iterations (for $t < d^{\max}$) given by

$$\alpha_{i,1}^{\mathrm{HSMM}} = \Pi_i \mathcal{N}_{1,i}^{\mathcal{D}}\, \mathcal{N}_{1,i},$$
$$\alpha_{i,2}^{\mathrm{HSMM}} = \Pi_i \mathcal{N}_{2,i}^{\mathcal{D}} \prod_{s=1}^{2} \mathcal{N}_{s,i} + \sum_{j=1}^{K} \alpha_{j,1}^{\mathrm{HSMM}}\, a_{j,i} \mathcal{N}_{1,i}^{\mathcal{D}} \mathcal{N}_{2,i},$$
$$\alpha_{i,3}^{\mathrm{HSMM}} = \Pi_i \mathcal{N}_{3,i}^{\mathcal{D}} \prod_{s=1}^{3} \mathcal{N}_{s,i} + \sum_{j=1}^{K} \sum_{d=1}^{2} \alpha_{j,3-d}^{\mathrm{HSMM}}\, a_{j,i} \mathcal{N}_{d,i}^{\mathcal{D}} \prod_{s=4-d}^{3} \mathcal{N}_{s,i},$$

etc., corresponding to the update rule

$$\alpha_{i,t}^{\mathrm{HSMM}} = \Pi_i \mathcal{N}_{t,i}^{\mathcal{D}} \prod_{s=1}^{t} \mathcal{N}_{s,i} + \sum_{j=1}^{K} \sum_{d=1}^{t-1} \alpha_{j,t-d}^{\mathrm{HSMM}} a_{j,i} \mathcal{N}_{d,i}^{\mathcal{D}} \prod_{s=t-d+1}^{t} \mathcal{N}_{s,i}. \tag{20}$$

Note that the iterations can be reformulated for efficient computation, see Yu and Kobayashi (2006) and Yu (2010) for details.

Appendix III

For the infinite horizon LQT formulation, the cost function in Eq. (7) can be minimized with the algebraic Riccati equation (ARE)

$$\boldsymbol{A}^{\top}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\top}\boldsymbol{P} + \boldsymbol{Q} = \boldsymbol{0}$$

$$\Leftrightarrow \begin{bmatrix}\boldsymbol{P} & -\boldsymbol{I}\end{bmatrix} \boldsymbol{H} \begin{bmatrix}\boldsymbol{I} \\ \boldsymbol{P}\end{bmatrix} = \boldsymbol{0}, \tag{21}$$

with the Hamiltonian matrix

$$\boldsymbol{H} = \begin{bmatrix}\boldsymbol{A} & -\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\top} \\ -\boldsymbol{Q} & -\boldsymbol{A}^{\top}\end{bmatrix}. \tag{22}$$

Under suitable hypotheses about symmetry, stabilizability and detectability, Eq. (21) has a unique positive semidefinite solution $\boldsymbol{P}$, which can be obtained by several methods, see Laub (1979) for details.

The key is to convert the problem to a stable invariant subspace problem of the Hamiltonian matrix, i.e., finding the invariant subspace corresponding to eigenvalues of $\boldsymbol{H}$ with negative real parts. This subspace can be found in several ways. With the cost $\boldsymbol{Q}$ evaluated from precision matrices, a simple solution is to use an eigendecomposition approach (with ordered eigencomponents) to decompose $\boldsymbol{H}$ as

$$\boldsymbol{H} = \boldsymbol{V} \begin{bmatrix}\boldsymbol{\lambda}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\lambda}_2\end{bmatrix} \boldsymbol{V}^{\top}, \quad \text{with} \quad \boldsymbol{V} = \begin{bmatrix}\boldsymbol{V}_1 & \boldsymbol{V}_{12} \\ \boldsymbol{V}_{21} & \boldsymbol{V}_2\end{bmatrix}, \tag{23}$$

where $\begin{bmatrix}\boldsymbol{V}_1 \\ \boldsymbol{V}_{21}\end{bmatrix}$ forms the stable eigenspace of $\boldsymbol{H}$. We have that $\boldsymbol{H} \in \mathbb{R}^{4D \times 4D}$ for a double integrator as in Eq. (6), which precisely has $2D$ eigenvalues with negative real parts. Together with (21), it provides the ARE solution

$$\boldsymbol{P} = \boldsymbol{V}_{21}\boldsymbol{V}_1^{-1}, \tag{24}$$

which is used at each iteration to compute the tracking gains.

References

Bowen, C., & Alterovitz, R. (2014). Closed-loop global motion planning for reactive execution of learned tasks. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Calinon, S. (2016). A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, *9*(1), 1–29. https://doi.org/10.1007/s11370-015-0187-9.

Calinon, S., Bruno, D., & Caldwell, D. G. (2014). A task-parameterized probabilistic model with minimal intervention control. In *Proceedings of the IEEE international conference on robotics and automation (ICRA), Hong Kong, China* (pp. 3339–3344).

Calinon, S., Li, Z., Alizadeh, T., Tsagarakis, N. G., & Caldwell, D. G. (2012). Statistical dynamical systems for skills acquisition in humanoids. In *Proceedings of the IEEE international conference on humanoid robots (Humanoids), Osaka, Japan* (pp. 323–329).

Calinon, S., Pistillo, A., & Caldwell, D. G. (2011). Encoding the time and space constraints of a task in explicit-duration hidden Markov model. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), San Francisco, CA, USA* (pp. 3413–3418).

Chan, A., Croft, E. A., Little, J. J. (2013). Modeling nonconvex workspace constraints from diverse demonstration sets for constrained manipulator visual servoing. In *Proceedings of the IEEE*

international conference on robotics and automation (ICRA) (pp. 3062–3068).
Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*(1), 1–38.
Gams, A., Nemec, B., Ijspeert, A., & Ude, A. (2014). Coupling movement primitives: Interaction with the environment and bimanual tasks. *IEEE Transactions on Robotics*, *30*(4), 816–830.
Havoutis, I., & Calinon, S. (2016). Learning assistive teleoperation behaviors from demonstration. In *Proceedings of the IEEE international symposium on safety, security and rescue robotics, Lausanne, Switzerland* (pp. 258–263).
Havoutis, I., & Calinon, S. (2017). Supervisory teleoperation with online learning and optimal control. In *Proceedings of the IEEE international conference on robotics and automation (ICRA), Singapore* (pp. 1534–1540).
Havoutis, I., Tanwani, A. K., & Calinon, S. (2016). Online incremental learning of manipulation tasks for semi-autonomous teleoperation. In *IEEE/RSJ international conference on intelligent robots and systems (IROS), workshop on closed-loop grasping and manipulation: Challenges and progress, Daejeon, Korea*.
Ijspeert, A., Nakanishi, J., Pastor, P., Hoffmann, H., & Schaal, S. (2013). Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, *25*(2), 328–373.
Kulic, D., Takano, W., & Nakamura, Y. (2008). Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden Markov chains. *The International Journal of Robotics Research*, *27*(7), 761–784.
Laub, A. J. (1979). A Schur method for solving algebraic Riccati equations. *IEEE Transactions on automatic control*, *24*(6), 913–921.
Lee, D., & Ott, C. (2010). Incremental motion primitive learning by physical coaching using impedance control. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), Taipei, Taiwan* (pp. 4133–4140).
Lee, D., Ott, C., & Nakamura, Y. (2010). Mimetic communication model with compliant physical contact in human–humanoid interaction. *The International Journal of Robotics Research*, *29*(13), 1684–1704.
Palomeras, N., Carrera, A., Hurtós, N., Karras, G. C., Bechlioulis, C. P., Cashmore, M., et al. (2016). Toward persistent autonomous intervention in a subsea panel. *Autonomous Robots*, *40*(7), 1279–1306.
Pastor, P., Righetti, L., Kalakrishnan, M., & Schaal, S. (2011). Online movement adaptation based on previous sensor experiences. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 365–371).
Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–285.
Rasmussen, C. E. (2000). The infinite Gaussian mixture model. In S. A. Solla, T. K. Leen, & K. R. Mueller (Eds.), *Advances in neural information processing systems (NIPS)* (pp. 554–560). Cambridge: MIT Press.
Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, *6*(2), 461–464.
Silvério, J., Rozo, L., Calinon, S., & Caldwell, D. G. (2015). Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), Hamburg, Germany* (pp. 464–470).
Tanwani, A., & Calinon, S. (2016). Learning robot manipulation tasks with task-parameterized semitied hidden semi-markov model. *IEEE Robotics and Automation Letters*, *1*(1), 235–242. https://doi.org/10.1109/LRA.2016.2517825.
Ude, A., Nemec, B., Petrič, T., & Morimoto, J. (2014). Orientation in cartesian space dynamic movement primitives. In *Proceedings of the IEEE international conference on robotics and automation (ICRA), Hong Kong, China* (pp. 2997–3004).
Yu, S. Z. (2010). Hidden semi-Markov models. *Artificial Intelligence*, *174*, 215–243.
Yu, S. Z., & Kobayashi, H. (2006). Practical implementation of an efficient forward–backward algorithm for an explicit-duration hidden Markov model. *IEEE Transactions on Signal Processing*, *54*(5), 1947–1951.
Zeestraten, M. J. A., Calinon, S., & Caldwell, D. G. (2016). Variable duration movement encoding with minimal intervention control. In *Proceedings of the IEEE international conference on robotics and automation (ICRA), Stockholm, Sweden* (pp. 497–503).
Zeestraten, M. J. A., Havoutis, I., Silvério, J., Calinon, S., & Caldwell, D. G. (2017). An approach for imitation learning on Riemannian manifolds. *IEEE Robotics and Automation Letters (RA-L)*, *2*(3), 1240–1247. https://doi.org/10.1109/LRA.2017.2657001.

Ioannis Havoutis is a Departmental Lecturer in Robotics at the Department of Engineering Science of the University of Oxford. He is part of the Oxford Robotics Institute (ORI) and a colead, of the Dynamic Robot Systems Group. He leads the research direction of robotic legged locomotion as his expertise lies in the design and implementation of algorithms that enable autonomous legged mobility. Previously, 2015-2017, he worked on learning complex skills by demonstration at the Idiap Research Institute, while before (2011-2015) he led the locomotion group within the Dynamic Legged Systems lab (HyQ team) at the Department of Advanced Robotics, Istituto Italiano di Tecnologia. His focus is on approaches for dynamic whole-body motion planning and control that allow robots with arms and legs to robustly operate in a variety of challenging domains. He holds a Ph.D. and M.Sc. with Distinction from the School of Informatics at the University of Edinburgh.

Sylvain Calinon is a permanent senior researcher at the Idiap Research Institute. He is also a lecturer at the Ecole Polytechnique Federale de Lausanne (EPFL), and an external collaborator at the Department of Advanced Robotics (ADVR), Italian Institute of Technology (IIT). From 2009 to 2014, he was a Team Leader at ADVR, IIT. From 2007 to 2009, he was a Postdoc at the Learning Algorithms and Systems Laboratory, EPFL, where he obtained his PhD in 2007. He is the author of 100+ publications and a book in the field of robot learning and human-robot interaction, with recognition including Best Paper Awards in the journal of Intelligent Service Robotics (2017) and at IEEE Ro-Man'2007, as well as Best Paper Award Finalist at ICRA'2016, ICIRA'2015, IROS'2013 and Humanoids'2009. He currently serves as Associate Editor in IEEE Transactions on Robotics (T-RO), IEEE Robotics and Automation Letters (RA-L), Intelligent Service Robotics (Springer), and Frontiers in Robotics and AI. Personal website: http://calinon.ch.