Learning Augmented Joint-Space Task-Oriented Dynamical Systems: A Linear Parameter Varying and Synergetic Control Approach

Yonadav Shavit, Nadia Figueroa, Seyed Sina Mirrazavi Salehian, and Aude Billard

***Abstract*—In this letter, we propose an asymptotically stable joint-space dynamical system (DS) that captures desired behaviors in joint-space while converging toward a task-space attractor in both position and orientation. To encode joint-space behaviors while meeting the stability criteria, we propose a DS constructed as a linear parameter varying system combining different behavior synergies and provide a method for learning these synergy matrices from demonstrations. Specifically, we use dimensionality reduction to find a low-dimensional embedding space for modulating joint synergies, and then estimate the parameters of the corresponding synergies by solving a convex semidefinite optimization problem that minimizes the joint velocity prediction error from the demonstrations. Our proposed approach is empirically validated on a variety of motions that reach a target in position and orientation, while following a desired joint-space behavior.**

***Index Terms*—Motion control, learning from demonstration, kinematics, Gaussian mixture models, joint-space control.**

I. Introduction

ROBOT motion planning in joint-space has long been a major field of study [1]. For manipulation problems with an objective defined in task-space, we can often find a myriad of joint-space trajectories to achieve the same task-space goal. In many cases, however, certain joint-space trajectories are favored over others; for example, when we expect the robot to follow a desired joint-space behavior or "style", as illustrated in Fig. 1. In this paper, we will explore the problem of learning a preferred joint-space behavior from previously demonstrated trajectories while still accomplishing a task-space goal, through the Learning from Demonstrations (LfD) paradigm [2], [3]. Most LfD approaches learn motions either in joint space [4], [5] or in task space [6], [7] via probabilistic models. To ensure that the learnt models reach the desired task-space target, Dynamical System (DS) formulations have been used to ensure stability and convergence [8], [9].

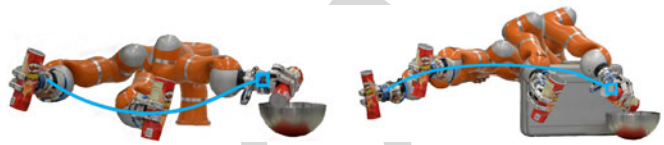

Fig. 1. Two robot motions in **joint-space** accomplishing *similar* behavior in **task-space** (pouring chips in a bowl). The left example avoids a known obstacle in its workspace, while the right one does not.

In many cases, however, one may want to specify a joint-space behavior that the physical robot should follow while simultaneously reaching a task-space objective. For example, learned joint-space behaviors are helpful in playing ping-pong [10], grasping [11], and avoiding self-collisions of bi-manual manipulators [12]. For a visual example of a task-space problem in which joint behavior is essential, see Fig. 1. Moreover, learning a motion in joint space allows us to avoid inverse kinematic (IK) approximations. Task-space motion generators learned from demonstrations all rely on projecting the desired task-space velocity into joint-space via Jacobian Pseudo-Inverse IK approximations and variants thereof [1]. When the main focus is on executing a specific task-space behavior, regardless of a joint-space constraint, this approach is sufficient [13], [14]. However, for other applications, such an approach yields significant problems [15]. Mainly, when the Jacobian matrix cannot be inverted (i.e., when the robot is near a singularity) its behavior becomes erratic, requiring layers of additional engineering to generate smooth trajectories and ensure the desired task-space behavior. This encapsulates the main source of inaccuracies in tasks that require fast dynamical motions, such as catching/reaching for moving objects [16], [17].

Finally, by learning a behavior in joint-space we can inherently reach a task-space target, not only in the Cartesian 3-D space $\mathbb{R}^3$, but also in the space of orientations $SO(3)$. Planning or learning the rotational component of motion is a challenging problem. Representing an orientation as a vector in Euclidean space may lead to inaccurate and unstable motions, due to its directional nature and vulnerability to singularities. Several works have proposed tailor-made learning approaches that consider the non-Euclidean geometry of the $SO(3)$ space to generate rotational motion [18] [19] [20]. These approaches, however, require an explicit coupling between position and orientation, that might cause discontinuities in the resulting motion. Such coupling is not necessary if the motion is encoded in joint-space.

Manuscript received November 21, 2017; accepted April 21, 2018. This letter was recommended for publication by Associate Editor F. Ficuciello and Editor H. Ding upon evaluation of the reviewers' comments. This work was supported by the EU project Cogimon H2020–ICT–23–2014. *(Yonadav Shavit, Nadia Figueroa, and Seyed Sina Mirrazavi Salehian contributed equally to this work). (Corresponding author: Nadia Figueroa.)*

Y. Shavit is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: yonadav.shavit@gmail.com).

N. Figueroa, S. S. M. Salehian, and A. Billard are with the Swiss Federal Institute of Technology (EPFL), Lausanne 1015, Switzerland (e-mail: nadia.figueroafernandez@epfl.ch; sina.mirrazavi@epfl.ch; aude.billard@epfl.ch).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LRA.2018.2833497

Several approaches have tackled the problem of learning joint-space behaviors with task-space objectives. [21] notably attempted to address this problem by learning separate motion policies in task space and the null space of the Jacobian (which would not affect task-space position), and driving the robot with a weighted sum of the two. However, this approach does not seek convergence to the desired task-space target and is still reliant on computing the pseudo-inverse Jacobian. [11] and [12] expand on this approach by projecting task-space constraints into joint space using IK, and then learning a joint-space policy that incorporates both task and joint space constraints, but this similarly relies on IK approximations and does not ensure convergence to an attractor. [22] proposed an approach with similar properties to our desiderata, where two concurrent DS, one in task-space and one in joint-space, are modulated by enforcing kinematic coherence constraints to avoid singularities. The resulting DS avoids singularities through generalization of the pseudo-inverse approximations. However, because the two DS have their own unique attractors, the non-linear interaction between them imposed by the kinematic constraints does not ensure that the combined DS has a unique attractor. This gives rise to spurious attractors or cycles, and thus requires careful tuning to avoid them.

In this work, we seek to devise an augmented **J**oint-space **T**ask-oriented **D**ynamical **S**ystem (JT-DS) that not only incorporates task-space attractors, but also avoids the problems generated by pseudo-inverse approximations. To approximate this DS, we further propose an algorithm to learn a set of *behavior synergies*, each of which corresponds to a different stable behavior in joint space, and modulate the use of these synergies throughout joint-space using a learned Linear Parameter Varying (LPV) system. We determine the scheduling parameters for the LPV by finding a *lower-dimensional embedding* of the joint space, which accounts for the variation in the demonstrated motions. We then learn a policy in embedding space for combining our behavior synergies to accurately reconstruct the demonstrated trajectories. Hence, our dynamical system:

1) computes a **motion in joint-space** that provably and asymptotically converges to a **task-space** target.
2) is formulated such that **joint-space behaviors** can be **learned** from demonstrations as **synergies**.
3) can transit through kinematic **singularities**.

The most similar approach to our proposed DS is the Jacobian transpose (JT) control method [23]. The JT control is an IK method that yields a dynamical system in joint space which converges stably over time to a desired end-effector target, without the need for pseudo-inverse computations. It shares some of our approach's advantages: fast computation and provable task-space stability. However, despite some previous work designing velocity adjustments by hand [24], this is, to the best of our knowledge the first work to employ a JT system to learn behaviors from demonstrations. Furthermore, by formulating our LPV system on a latent embedding (via dimensionality reduction schemes), we are able to discover meaningful local behavior synergy regions, while being robust to outliers, noise and redundancies that might arise from raw demonstrations. This leads to a compelling improvement in generalization of the demonstrated behavior, as opposed to learning the LPV system solely in joint space.

This paper is organized as follows. Section II formalizes the problem. The proposed dynamical system is introduced in Section III. In Section IV, a probabilistic model is introduced to approximate the parameters of the dynamical system. In addition, a convex optimization problem is formalized to estimate these parameters. In Section V we provide a thorough validation of our proposed DS and learning approach. We finalize with a discussion in Section VI.

II. Problem Statement

Consider a robotic system with d task-space dimensions and m degrees of freedom. The system is directed via a joint-position or joint-velocity controller. We are further provided with a set of N demonstrated joint-space trajectories $D = \{\{q_{t,n}, \dot{q}_{t,n}\}_{t=1,\dots,T_n}\}_{n=1,\dots,N}$, where T_n is the number of the sample points of the nth demonstration. We refer to the system's joint-space position as $q = \begin{bmatrix} q^1 & \dots & q^m \end{bmatrix}^T \in \mathbb{R}^m$, and to its task-space target as $x \in \mathbb{R}^d$.[1] The kinematics of the robot are assumed to be known, hence, the robot's forward kinematics are indicated by $x = H(q)$ and its Jacobian is $J(q) = \frac{dx}{dq} \in \mathbb{R}^{d \times m}$. We wish to formulate a DS $\dot{q} = f(q)$ which satisfies the following two criteria:

I) The DS must be asymptotically stable[2] with respect to a fixed task-space target x^*. This can be expressed by ensuring that the following Lyapunov function

$$V(q) = (H(q) - x^*)^T (H(q) - x^*) \tag{1}$$

is stable; i.e., $\dot{V}(q) < 0 \, \forall q \in Q$ and $V(q) = 0 \, \forall q \in Q^*$ where $Q^* = \{q | H(q) = x^* \wedge q \in Q\}$ and $Q = \{q | q \notin Q^*, J(q) is \, full \, rank\}$. $V(\cdot)$ can be thought of as a metric for the task-space distance-to-go.

II) The DS should encapsulate the desired joint-space behaviors such that the following metric is minimized

$$e_{total} = \frac{1}{NT_n} \sum_{n=1}^{N} \sum_{t=0}^{T_n} \| \dot{q}_{d;t,n} - f(q_{t,n}) \| \tag{2}$$

where $\dot{q}_d$ are the "true" velocities from the demonstrations, and $f(\cdot)$ is the motion generation policy.

The error metric (2) is advantageous in that it mimics the magnitude and direction of the demonstrated motions. This makes the reproduced motion visually most similar to the human definition of "joint motion style" [25].

III. Augmented Joint-Space Task-Oriented Dynamical System

To achieve the two criteria presented in (1) and (2), we propose the following DS, which we refer to as the augmented Joint-space Task-oriented DS (JT-DS):

$$\dot{q} = f(q) = -\mathcal{A}(q) J^T(q) (H(q) - x^*) \tag{3}$$

where $\mathcal{A}(q) \in \mathbb{R}^{m \times m}$ is constructed using the LPV system paradigm [16], [26]. $\mathcal{A}(q) = \sum_{k=1}^{K} \theta_k(\cdot) A_k$ is a linear combination of time-invariant linear matrices $A_k \in \mathbb{R}^{m \times m}$, each

[1] For pure position targets we consider $d = 3$, for position and orientation $d = 9$, where the first 3 dimensions correspond to Cartesian position and the remaining 6 correspond to the first and second columns of a rotation matrix $R \in SO(3)$.

[2] Unless otherwise specified, "stability" in this paper always refers to asymptotic stability within the workspace of the robot (in the regions where the Jacobian is full rank), and as such assumes no joint limits. We make no claim to proving global asymptotic stability, which is in fact impossible to achieve in a joint-constrained kinematic system.

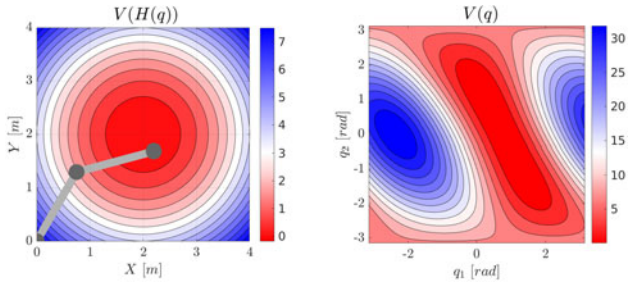

Fig. 2. Illustration of our task-space and proposed joint-space Lyapunov function. On the left we show the task-space error we are trying to minimize, represented by a potential function for a 2-DOF robot with target $x^* = [2, 2]^T$. On the right we show the corresponding Lyapunov function in joint space defined in (1). The colors on each plot correspond to each other. One can see how the error, and consequently the *attractive* regions in task-space, have been warped into attractive regions in joint-space.

of which encodes a "local joint-space behavior synergy" that shapes the motion in joint-space. Specifically, the "local synergy matrices" A_k are joint-space transformations that bias the resulting motion to use particular joints. By modulating each local synergy in time as well as space through the activation functions $\theta_k(\cdot)$ we can generate the desired non-linear joint-space behaviors. Note that in the LPV paradigm $\theta_k(\cdot)$ can be a function of time t, "joint-posture" q, an external signal $d(t)$ or a lower-dimensional representation of the joint posture $\phi(q)$. Because we seek a time-invariant controller and know that "joint-postures" can be accurately represented in a lower-dimensional space, in this work we chose to parametrize $\theta_k(\cdot)$ with $\phi(q)$.

Before proving that (3) satisfies all of the criteria, one can intuitively understand the control law as follows: $(H(q) - x^*)$ denotes the task-space error wrt. the target[3]. We derive the task-space velocity of a proportional controller minimizing that error by multiplying by -1. Then, by multiplying this task-space velocity by the transposed Jacobian $J^T(q)$, it is transformed into a joint-space velocity vector correlated with the error (similar to Jacobian transpose control [23], [27]), see Fig. 2. The positive definite matrix $\mathcal{A}(q)$ warps the resulting joint-space velocity; Fig. 3 illustrates the effects of $\mathcal{A}(q)$ on the generated motion. Thus the controller can be thought of as a proportional controller in joint space. Lastly, we refer to $\mathcal{A}(q)$ as the **joint augmentation matrix** as it augments the outputted joint velocities.

Proposition 1. The DS in (3) accomplishes criteria (I) if $\forall k \in \{1, \dots, K\}$ the following constraints are met.

$$\begin{cases} A_k \succ 0, \quad \theta_k(\cdot) \geq 0 \end{cases} \tag{4}$$

Proof: See Appendix A. ■

Criterion (II) (i.e., encoding specific joint-space behaviors) is achieved by embedding the desired joint-space behavior in the matrices $A_k(q)\,\forall k \in \{1, \dots, K\}$. We describe in the next section an approach to automatically learn the number of matrices K, their values and their corresponding activation functions $\theta_k(\cdot)$ from demonstrated data.

[3]If the target defines an orientation, this is the Euclidean distance between the two 9-dimensional position-and-orientation vectors.

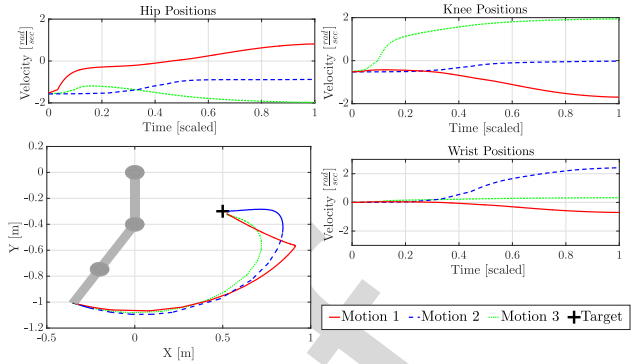

Fig. 3. Three example 3-DOF motions (A, B, C), each with a different constant joint augmentation matrix $\mathcal{A}(q)$ (emphasizing the hip, knee, and ankle respectively), A: $\mathcal{A}(q) = diag(5, 1, 1)$, B: $\mathcal{A}(q) = diag(1, 5, 1)$, and C: $\mathcal{A}(q) = diag(1, 1, 5)$. On the left, the task-space traces of each motion. On the right, the time-scaled joint positions of each joint. Each motion tends to use its "primary" joint most and uses the other available joints to compensate for what the primary joint cannot do.

IV. Learning Joint-Space Task-Oriented Dynamical Systems in Synergy Space

The behavior of the JT-DS algorithm can be best understood through the lens of synergy control [28]. In robotic synergy control, a robot's movements can be decomposed into a small number of synergies: principal components of the joint-space that are sufficient to accurately recreate the desired robotic behaviors. In our case, the synergies are represented by the matrices $A_1, A_2, \dots, A_k$, and $\mathcal{A}(q)$ represents the resulting motion constructed from a superposition of different synergies (through (3)).

A central question arises: how do we modulate the synergies in different regions to yield our desired behavior? First, we assume that our desired behavior can be efficiently described[4] using a sub-manifold of the joint-space, called the *embedding space*. We would like to define the robot's policy in embedding space such that in different regions of the space, we will prioritize different synergies. We are thus left with three problems: (I) finding an underlying synergy-space Z in which the behavior can be accurately controlled (defined by a mapping $\phi : Q \to Z$), (II) finding a policy for modulating the synergies in different regions of the synergy space (defined by $\theta_k(\phi(q))\,\forall k \in \{1, \dots, K\}$), and (III) finding parameters for the synergies themselves (defined by $A_k\,\forall k \in \{1, \dots, K\}$). Our choice of parameters must also obey the constraints laid out in (4). We thus propose the following 3-step learning procedure:

I) We first construct our embedding, which provides us a lower-dimensional manifold through which to control the robot, by projecting the demonstration data (i.e., collections of joint positions $q \in \mathbb{R}^m$) into a lower-dimensional embedding $\phi(q) \in \mathbb{R}^{p \leq m}$. In Section V we evaluated Principal Component Analysis (PCA) [29] or Kernel PCA (KPCA) with RBF kernel [30].

II) We then jointly estimate the optimal number K of "local synergy regions" and the parameters of the scalar

[4]Given some original space A and some behavior policy $\pi_A(a)$ in A, we say that an embedding $\phi(\cdot)$ and embedding space $B : \{b = \phi(a) | a \in A\}$ "efficiently describe" π_A if there exists some policy $\pi_B(b)$ in B such that we can deterministically reconstruct $\pi_A(a)$ given $\pi_B(\phi(a))$.

functions that determine the activation parameters $\theta_k(\phi(q))$ for weighting these synergies, by fitting a Gaussian Mixture Model (GMM) on the projected joint positions $\phi(q)$ seen in the demonstrations.

III) Once the local synergy regions have been found (described by each of the Gaussian distributions $\theta_k(\phi(q))$), we compute the corresponding joint synergy matrices $A_k\ \forall k \in \{1,\dots,K\}$ for each region by formulating a convex optimization problem that finds the optimal set of A_k's that minimize the overall velocity error with respect to the demonstrations (2).

A. Embedding Joint Configurations in Low-Dim. Space

The search for a lower-dimensional embedding of the joint space stems from the desire to identify a simplified coordinate system in which each principal component corresponds to an important source of variation in the demonstrated trajectories, and which is thus suitable for parameterizing the demonstrated behavior. Motor control studies have postulated that human arm motions like reaching or following straight/curved line trajectories, rather than utilizing the full joint space $|q|$, are the result of compromising between planning a straight line in the task space and a straight line in the joint space [31], [32]. This suggests that human arm motion in general tends to move on a plane, and thus can be represented in such a lower-dimensional space.

In this work, we assume that configurations that are nearby in joint-space should exhibit similar behaviors, and thus it is natural that we choose a low-dimensional embedding that preserves the variance in our demonstrations. To this end, we construct a single global embedding $\phi(q)$ by training dimensionality reduction techniques on the demonstrated trajectories. The learned embedding $\phi(\cdot)$ maps a joint configuration $q \in \mathbb{R}^m$ into a lower-dimensional configuration $z \in \mathbb{R}^p$, where $p < m$. For example, if the shoulder and arm joints are coupled throughout the motion, and $\phi(q)$ were a matrix multiplication (i.e., $\phi(q) = A_p \times q$ for $A_p \in \mathbb{R}^{p\times m}$), it could map the "shoulder" and "arm" components of q into a single "shoulder-arm" component in $\phi(q)$.

B. Discovering Local Behavior Synergies

The next step is identifying the regions of space in which to activate different synergies. Given the set of projected joint position trajectories $D = \{\{\phi(q_{t,n})\}_{t=1,\dots,T_n}\}_{n=1,\dots,N}$ where $\phi(q_{t,n})$ is the lower-dimensional embedding of $q_{t,n}$, t is the time-step and N is the number of demonstrations, we seek to learn a set of regions of distinct local synergies, each defined by their corresponding activation function $\theta_k(\phi(q))$. Moreover, we would like for $\theta_k(\phi(q))$ to have the following properties: (i) $\theta_k(\phi(\cdot)) > 0$ and (ii) $\sum_{k=1}^{K} \theta'_k(\phi(q)) = 1$. Such activation functions, also known as scheduling parameters for LPV systems, have been modeled in previous work as probability distributions [16], [17]. Intuitively, we search for a probabilistic model that "explains" the variance in the demonstrated trajectories, and treat each cluster as expressing a *local behavior*, which the synergy will then approximate. In this work, we adopt this approach and use a GMM to estimate the joint distribution over the projected joint positions[5], $p(\phi(q)) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\phi(q); \mu_k, \Sigma_k)$, where π_k are the prior probabilities and $\{\mu_k, \Sigma_k\}$ are the mean and covariance matrices that parametrize the k-th multivariate Gaussian distribution. Each distribution represents a local region of projected joint positions $\phi(q)$, and will be used to construct the activation functions θ_k of the kth synergy matrix. We define $\theta_k(\phi(q))$ as $p(k|\phi(q))$:

$$\theta_k(\phi(q)) = \frac{\pi_k \mathcal{N}(\phi(q); \mu_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\phi(q); \mu_k, \Sigma_k)} \tag{5}$$

which is the probability of the projected joint-position $\phi(q)$ belonging to the k-th local synergy region. Therefore, each synergy region is associated with a Gaussian component of the GMM, cumulatively describing all the synergy regions of the dynamical system. We use the standard Expectation Maximization (EM) training algorithm to estimate the parameters of the GMM [33]. We follow a model selection approach described in Section V to find the optimal K.

C. Estimating the Synergy Matrices

Given the parameters of $\theta_k(\phi(q))\ \forall k \in \{k = 1,\dots,K\}$, one can construct $\mathcal{A}(q)$ as a linear combination of local A_k synergy matrices as follows:

$$\mathcal{A}(q) = \frac{\sum_{k=1}^{K} A_k \pi_k \mathcal{N}(\phi(q); \mu_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\phi(q); \mu_k, \Sigma_k)}. \tag{6}$$

Notice the resemblance of (6) to the Nadaraya-Watson kernel estimator [34], [35][6] with a Gaussian pdf as its kernel function. Hence (6) can be considered a type of kernel estimator, with the key distinction that the weighting functions $\theta_k(\phi(q))$ are not determined by individual points (as in the original Nadaraya-Watson kernel estimator) but by the components of a GMM, similar to the weighting functions derived in Gaussian Mixture Regression (GMR).

Intuitively, each "synergy region" is defined by a Gaussian distribution in the lower-dimensional space. The closer the robot is to a region, the more that region's synergy (A_k) influences the robot's current joint-space motions. Finding the appropriate synergy matrices A_k to accurately reproduce the observed demonstrations can be reduced to a semidefinite program with the goal of minimizing (2). The resulting optimization minimizes the mean squared joint velocity error from (2), resulting in a convex semidefinite optimization with a quadratic objective, as follows:

$$\min_{A_1,\dots,A_K} \sum_{n=1}^{N} \sum_{t=0}^{T_n} \|\dot{q}_{d;t,n} - f(q_{t,n})\|$$

subject to

$$0 \prec A_k,\ \forall k \in \{1,\dots,K\}. \tag{7}$$

where $f(q_{t,n})$ is calculated by (3) with (6), and x_n^* is defined as the endpoint of the nth demonstrated trajectory. This optimization is always feasible; the only constraint is that the A_ks be independently PSD.

[5]It must be noted that, although we present GMM as the approach to estimate the activation parameters, alternative algorithms can be used.

[6]The Nadaraya-Watson kernel estimator is used to estimate an unknown regressive function $m(x) = \mathbb{E}\{Y|X\}$, which takes the general form of $\widehat{m}(x) = \frac{\sum_{i=1}^{n} y_i \mathcal{K}(x,x_i)}{\sum_{i=1}^{n} \mathcal{K}(x,x_i)}$ where $\mathcal{K}(x, x_i)$ is a kernel function denoting the distance or similarity of x_i to the given location x. [34], [35].

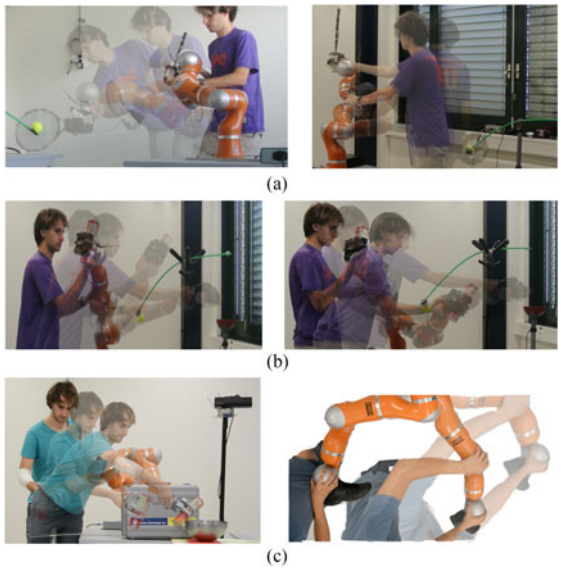

Fig. 4. Demonstrated joint-space behaviors with task-space targets. (a) Forward/backward reaching (1-2). (b) Pouring motions (3-4). (c) Pouring motion (5) and foot-step (6).

V. Experimental Validation

A. JT-DS Learning Performance Evaluation

To evaluate the proposed JT-DS learning algorithm we collected kinesthetic demonstrations on a 7-DOF KUKA LWR 4+ robot arm, for 7 different tasks (see Table I and Fig. 4) which require mimicking the demonstrated joint-space behavior while reaching for a single target in task space:

- *(1-2) Forward/Backward reaching:* We guide the robot in forward and backward reaching motions towards a specific target in task-space with joint-space trajectories mimicking forehand/backhand strokes as shown in Fig. 4(a).
- *(3-5) Pouring motions:* We guide the robot to emulate pouring motions with three different environmental constraints, leading to different joint-space trajectories. The first is without an obstacle in the workspace. The second is with an obstacle in the upper hemisphere of the workspace, where the human is seen folding the elbow and lowering the shoulder of the robot arm to avoid the obstacle. The third instance is with an obstacle in the lower hemisphere, and now the human raises the shoulder to avoid it.
- *(6) Footstep-like motion:* We demonstrate a footstep motion which begins with a straight leg, moving through a singularity, and finally bringing the knee up.
- *(7) Singularity motions:* Movement was constrained to the boundary of the workspace by fixing $q^i = 0 \, \forall i \in \{3, \dots, 7\}$, the second joint was fixed to $q^2 \in \{10^\circ, 20^\circ, \dots, 100^\circ\}$, and only the first joint was moved by a human. This restricted the motion to a series of arcs of different radii along the robot's motion boundary.

The recordings are collected at a rate of 500 Hz. However, our learning approach does not require such a dense representation of the trajectories, hence we down-sample them to 25 Hz. By performing our evaluation on all these datasets, we seek to:

I) Verify that the learned *behavior synergies* are mimicking the demonstrated joint-space behaviors and generalize them to new initial joint configurations.

II) Analyze the role of dimensionality reduction in our proposed approach and further find the method which yields the best trade-off between accuracy and model complexity; i.e., the least number of dimensions p to represent our activation function $\theta(\phi(q))$ and the least number of local behavior synergies K.

We thus evaluate three learning approaches with different dimensionality reduction algorithms: (1) None, (2) PCA and (3) K-PCA with RBF kernel. We perform 10-fold cross-validation on all datasets, for each learning approach, with a training/testing ratio of 60%. To evaluate performance, we take the square root of the MSE defined in (2), which we refer to as the joint-velocity RMSE.

Performing such cross-validation in our case is not trivial, as our $\mathcal{A}(q)$ formulation has several hyper-parameters: the dimensionality p of our lower-dimensional embedding $\phi(q)$ and the number of local behavior synergies K. Moreover, when using K-PCA (with RBF kernel $k(q, q') = \exp(-\frac{||q-q'||^2}{2\sigma^2})$) we must also find the optimal width σ. Thus, for each fold and each learning approach we find the optimal hyper-parameters p, K and σ (when applicable), as follows:

I) *p for PCA:* We choose p such that the projection is capable of explaining 95% of the variance in the data.

II) *p and σ for K-PCA:* The interaction of these two parameters plays a major roll in the resulting projection obtained from K-PCA. Hence, we do a grid search on a log-spaced range of $\sigma_{range} = [\sigma_{\min} : \sigma_{\max}]$ values, where $\sigma_{\min} = \frac{1}{\kappa\sqrt{2}} \max_{i \in M, j \in M} \{||q_i - q_j||^2\}$ and $\sigma_{\max} = \frac{2\kappa}{\sqrt{2}} \max_{i \in M, j \in M} \{||q_i - q_j||^2\}$ for $\kappa = cte$. This yields a feasible range for σ that is guided by the pairwise Euclidean distances between all points in the dataset. Moreover, K-PCA is not limited to providing a $p \ll m$, in fact it can generate $p \leq M$. This is a nuisance as in our datasets $M \approx 1000$. To alleviate this we truncate σ_{range} by computing the explained variance of the eigenvectors in feature space for different values of $\sigma \in \sigma_{range}$. We then remove σ values from σ_{range} whose number of *optimal* eigenvectors $p > m$ and resample it. By doing this procedure, we can ensure that for all values in truncated $\bar{\sigma}_{range}$ we will obtain $p \leq m$. We then choose σ_{opt} by running 10-fold cross-validation for all $\sigma \in \bar{\sigma}_{range}$.

III) *K for GMM:* We choose the optimal number of components K, by evaluating and selecting the best resulting model using the Bayesian Information Criterion (BIC) [36]. Typically, one chooses the optimal K manually, by visually identifying the point at which the BIC curve produced from K_{range} stops changing or plateaus. To automate this process, we devised an approach which selects the optimal K as the one which yields the highest inflection point on the second order derivative of the BIC curve.

Once these optimal hyper-parameters are estimated for each fold, we solve for the convex optimization problem in order to find our synergy matrices A's which best minimize the objective function (7). We then select the initial joint configuration q_0 and target in task-space x^* from each training/testing dataset, simulate the joint-space trajectories $\{\dot{q}_1, \dots, \dot{q}_T\}$, with our learned $\mathcal{A}(q)$, and compute the joint-velocity RMSE between these simulations and the training/testing trajectories, as reported in Table I.

TABLE I
PERFORMANCE COMPARISON OF LEARNING APPROACH WITH DIFFERENT DIM. RED. SCHEMES

Behavior Dataset	Dim. Red. Approach	Optimal Parameters — Joint Velocity RMSE $[rad/s]$			
		Optimal p	Optimal K	RMSE Train	RMSE Test
(1) Forward Reaching ($N = 10, M = 1424$)	None	7	5 ± (2.90)	**0.234 ± (0.027)**	0.633 ± (0.458)
	PCA	3.1 ± (0.316)	3.8 ± (1.549)	0.247 ± (0.017)	0.484 ± (0.263)
	K-PCA ($\sigma_{opt} = 5.111$)	**3 ± (0)**	**3.5 ± (1.204)**	0.2532 ± (0.019)	0.385 ± (0.098)
(2) Backward Reaching ($N = 11, M = 1223$)	None	7	5.8 ± (2.097)	**0.281 ± (0.015)**	0.647 ± (0.354)
	PCA	4.4 ± (0.516)	6 ± (3.091)	0.282 ± (0.044)	0.466 ± (0.102)
	K-PCA ($\sigma_{opt} = 6.238$)	**3.5 ± (0.5)**	**3.1 ± (0.3)**	0.319 ± (0.025)	0.482 ± (0.114)
(3) Pouring - Free ($N = 9, M = 1032$)	None	7	4.6 ± (1.897)	**0.181 ± (0.019)**	1.099 ± (1.140)
	PCA	2.6 ± (0.5164)	4 ± (1.054)	0.186 ± (0.026)	0.419 ± (0.119)
	K-PCA ($\sigma_{opt} = 3.92$)	**2**	**3.5 ± (0.5)**	0.183 ± (0.0204)	0.397 ± (0.105)
(4) Pouring - Obstacle 1 ($N = 11, M = 1232$)	None	7	7 ± (2.357)	**0.296 ± (0.029)**	0.984 ± (0.55)
	PCA	3.9 ± (0.316)	3.6 ± (0.699)	0.321 ± (0.028)	0.402 ± (0.0587)
	K-PCA ($\sigma_{opt} = 7.695$)	**3**	**3.2 ± (0.4)**	0.311 ± (0.011)	0.388 ± (0.040)
(5) Pouring - Obstacle 2 ($N = 7, M = 1406$)	None	7	4.1 ± (1.7288)	**0.1242 ± (0.0169)**	1.0907 ± (1.0686)
	PCA	**2.8 ± 0.4216**	3.6 ± (1.8974)	0.1345 ± (0.0218)	0.693 ± (1.0541)
	K-PCA ($\sigma_{opt} = 2.86$)	3.4 ± (0.4899)	**3.1 ± (0.3)**	0.1345 ± (0.0182)	0.4028 ± (0.1377)
(6) Foot Step ($N = 8, M = 1058$)	None	7	4.2 ± (1.4757)	**0.1396 ± (0.0252)**	1.0697 ± (0.6574)
	PCA	**1**	3.1 ± (0.3162)	0.1494 ± (0.0143)	0.2578 ± (0.0795)
	K-PCA ($\sigma_{opt} = 1.513$)	2	**3**	0.1557 ± (0.0116)	0.2271 ± (0.0466)
(7) Singularity Motions ($N = 10, M = 1467$)	None	7	6.4 ± (1.5055)	**0.048 ± (0.0158)**	0.2365 ± (0.0768)
	PCA	**1.9 ± (0.3162)**	5.7 ± (1.6364)	0.0503 ± (0.0153)	0.1802 ± (0.0748)
	K-PCA ($\sigma_{opt} = 1.769$)	3.9 ± (0.3)	**5.2 ± (2.1817)**	0.0593 ± (0.0122)	0.1276 ± (0.0591)

We present the `mean(std)` for the optimal p, K and joint-velocity RMSE on training and testing set, found for every learning scheme over 10 runs. $M = \sum_{n=1}^{N} T_n$.

As can be seen, for all datasets there is a significant increase in performance on the testing sets when using either dimensionality reduction (DR) approaches. This suggests that using DR to encode our activation functions θ_k in a lower-dimensional space $\phi(q)$ yields better generalization capabilities than encoding the behaviors in using the original q. This is most notable for the three pouring motions, where the joint-velocity RMSE testing error for a JT-DS model learned without DR is an order of magnitude higher than with DR. Such an error indicates that the demonstrated joint-behavior was over-fitted on the training set, which is also exhibited in the higher number of K needed to represent the motion without DR. For all datasets, the DR methods provided $p < m/2$, either comparable or less number of local behaviors synergies K and better RMSE errors on testing sets as opposed to no DR. By finding a lower-dimensional manifold to represent the joint trajectories, we are getting rid of outliers, noise and redundancies that might arise from the raw joint demonstrations. Hence, through DR we are capable of robustly extracting the local behavior synergies from raw demonstrations.

Both PCA and K-PCA yield comparable results, with K-PCA providing a slight improvement on some datasets. This suggests that perhaps a linear DR method might be sufficient for such tasks. However, if we seek to maximize accuracy a non-linear DR method should be employed. One of the drawbacks of K-PCA is its computational complexity for out-of-sample evaluations, which involves computing the kernel function between the new data-point and all the samples M. This is, however, not so taxing for our method as our datasets range in $M \approx 1000$ and in previous work [37] we have experimentally found that evaluating <3000 RBF kernel computations (on a 3.4-GHz i7 PC with 8GB RAM)) in a control-loop with 2ms rate is feasible. If one requires extremely fast computation; i.e., a control-loop rate of <2 ms then PCA should be used as opposed to K-PCA.

Implementation Details: The learning pipeline is implemented in MATLAB. We used the Matlab Toolbox for Dimensionality Reduction [38] for implementations of PCA, K-PCA and its out-of-sample extension. The YALMIP framework [39] was used to solve the definite convex optimization problem. Source code for the entire learning pipeline and simulated behavior on multi-DOF robot arms can be found in: https://github.com/epfl-lasa/JT-DS-Learning. For execution of the learnt JT-DS models on a real 7-DOF the KUKA LWR 4+ robot arm, we provide the following code in C++: https://github.com/epfl-lasa/JT-DS-lib. The robot is controlled on the joint position level (linearly interpolating from joint velocities computed from JT-DS) at a rate of 500 Hz. The resultant joint angles are filtered by a critically damped filter to avoid high torques.

B. JT-DS Execution Performance Evaluation

We now seek to elucidate the distinctive properties of our JT-DS model by comparing its performance to a DS-based Cartesian motion generator + IK solver approach for behaviors (5)–(7). For the Cartesian motion generator we use the SEDS (Stable Estimator of Dynamical Systems) approach [8] which learns an asymptotically stable DS in Cartesian space from demonstrations. We then generate joint trajectories through a damped least square IK solver. From hereon we refer to this approach as SEDS+IK.

1) Following Desired Joint and Task-Based Behaviors: We compare tracking capabilities of our JT-DS method with those of SEDS+IK for behaviors (5) and (6). In Fig. 5(a) and (b) we can qualitatively see the difference between these two approaches for behavior (5). The JT-DS algorithm mimicked the joint-space behavior of the demonstration (e.g., folding the elbow, raising the shoulder), successfully avoiding the obstacle while still converging to the desired Cartesian position. Meanwhile, SEDS+IK only learned the demonstrated behavior in task-space; it is incapable of constraining motion in joint-space. This ultimately led to one of its joints colliding with the obstacle. It should be noted that the JT-DS motion did not follow the demonstrations in task-space very closely (as expected), but did ultimately converge to its target position. For the foot-step behavior *(6)* JT-DS followed the demonstrations closely, while SEDS became unstable in the singularity (Figs. 6 and 7). This demonstrates the JT-DS algorithm's ability to move cleanly in and out of singularities.

2) Transiting through singular configurations: One of the main advantages of the proposed DS is its ability to generate

Fig. 5. Snapshots of the robot experiments. A corresponding video is available on-line https://youtu.be/mv9u5DgIEtw[https://youtu.be/mv9u5DgIEtw]. (a) Execution of pour obstacle 2 learned through JT-DS. (b) Execution of pour obstacle 2 learned through SEDS.

Fig. 6. Execution of *problematic* joint-space behaviors learned through JT-DS: (left) Foot step-like motion and (right) singularity motions.

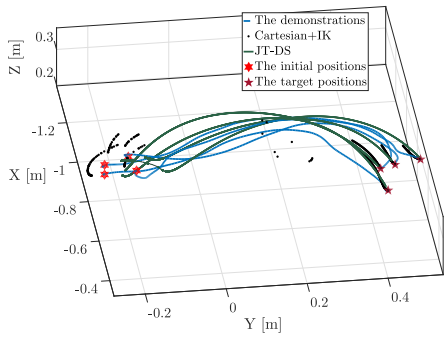

Fig. 7. End-effector trajectories for the footstep motion in Cartesian space. The JT-DS motion moves smoothly closely resembles the demonstrated trajectories. On the other hand, the SEDS-based Cartesian motion generator (whose values here are simulated because they can not physically executable) quickly becomes unstable, as evidenced by the dotted paths starting on the left side and abruptly disappearing.

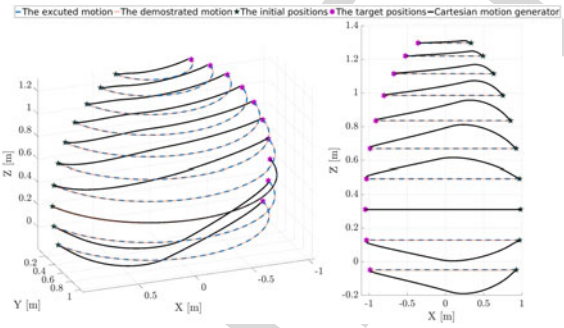

Fig. 8. The experiment generated $K = 3$ Gaussian components. As the first joint is the only joint which was not fixed during the demonstration, the learned augmentation matrices had only one nonzero entry $A_k(1,1) \neq 0 \, \forall k \in \{1,2,3\}$. In (a), the end-effector positions for the demonstrations and executed motions are plotted in Cartesian space. The JT-DS trajectory was generated closed-loop, while the SEDS trajectory was generated open-loop (otherwise it would be unstable).

accurate paths in classical singular configurations. To evaluate this we generated behavior *(7)*; i.e., joint-trajectories that transit entirely within a classic kinematic singularity. Fig. 8 shows the *demonstrated* motions and the motion *generated* by JT-DS (3). The algorithm never requires the pseudo-inverse of the Jacobian matrix, so the generated motion perfectly follows the demonstrations throughout the workspace boundary. In Fig. 6 we show the learnt singular motion successfully executed on the real robot.

VI. Discussion

In this paper, we have presented a dynamical system in joint space that is provably asymptotically stable in task space to a fixed target while replicating demonstrated joint-space behaviors. The desired motions are fast to compute, and smoothly handle singularities by avoiding the pseudo-inverse Jacobian. We showed the system's ability to learn different joint-space behaviors on a robotic platform.

One of the most important points when validating a learning from demonstration method is to evaluate the system's behavior away from demonstrations. When the current joint configuration is far from any of the local synergy regions, computing the activation parameters $\theta_k(\cdot)$ becomes numerically infeasible (all the Gaussians in (6) $\rightarrow 0$), and so we default to $\mathcal{A} = \frac{1}{K}\sum_{k=1}^{K} A_k$, which still guarantees that the robot moves towards the target.

Since we learn a lower-dimensional embedding space $Z := \{\phi(q) \,|\, q \in Q\}$, which we suggest provides a better representation for the learned joint behaviors, a careful reader might wonder: why not define a motion policy π_Z directly in Z-space and then map the learned policy back out into joint space using the inverse embedding $\phi^{-1}(z)$? The answer is that we would lose any guarantee of stability with respect to a target attractor x_t^*. By construction, ϕ maps from a joint-position vector of size d to a low-dimensional vector of size $u < d$. Thus the inverse ϕ^{-1} must be of less than full rank, so the resulting $\mathcal{A}$ matrix (derived as in (3)) will also be of less than full rank, and thus no longer positive definite. This means that our controller no longer provably converges to the attractor (since our convergence proof no longer holds). Intuitively, this is because any policy defined in a lower-dimensional space than the actuation space will forfeit certain degrees of freedom, and thus may not be able to span the configuration space. Instead, we use the embedding space to modulate synergies defined in joint space, which we can guarantee will be positive definite and lead us to convergence to the target.

Finally, we are currently working on improving the performance of the proposed DS, by learning its parameters wrt. the kinematic constraints. In this way, we can ensure that the performance of the DS is kinematically feasible for the robot to follow.

Appendix A
Proving Stability of the Dynamical System

JT-DS (3) is asymptotically stable with respect to the Lyapunov candidate

$$V(q) = \frac{1}{2}(H(q) - x^*)^T (H(q) - x^*)$$

That is, $0 \prec V(q), \forall q \neq q^*$ and $V(q^*) = 0$, where q^* is any joint configuration such that $H(q^*) = x^*$. The derivative of V wrt. time is:

$$\begin{aligned}\frac{dV(q)}{dt} &= (H(q) - x^*)^T J(q)\dot{q} \\ &= -(H(q) - x^*)^T J(q)\mathcal{A}(q)J^T(q)(H(q) - x^*) \\ &= -(H(q) - x^*)^T J(q) \sum_{k=1}^{K} \underbrace{\theta_k(\phi(q))}_{>0} \\ &\quad \underbrace{A_k}_{\succ 0} J^T(q)(H(q) - x^*) \leq 0 \end{aligned} \tag{8}$$

When $J(q)$ is full-rank, the above inequality can be tightened to $\frac{dV(q)}{dt} < 0$. Therefore, as $q = q^*$ is the largest invariance in $\mathbb{R}^m$, JT-DS (3) is stable with respect to a task-space attractor; i.e., $\lim_{t\to\infty} \|H(q) - x^*\| = 0$, and asymptotically stable for all regions in which the end-effector is fully manipulable ($J(q)$ is full rank).

ACKNOWLEDGMENT

The authors would like to thank A. Karimi for his insightful comments about formulating the convex optimization problem.

REFERENCES

[1] R. Kelly, V. S. Davila, and J. A. L. Perez, *Control of Robot Manipulators in Joint Space*. Berlin, Germany: Springer, 2006.

[2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*. New York, NY, USA: Springer, 2008, pp. 1371–1394.

[3] B. Argall, S. Chernova, M. M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[4] J. Garrido, W. Yu, and A. Soria, "Human behavior learning for robot in joint space," *Neurocomputing*, vol. 155, pp. 22–31, 2015.

[5] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real time robot learning," *Appl. Intell.*, vol. 17, no. 1, pp. 49–60, 2002.

[6] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 763–768.

[7] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intell. Service Robot.*, vol. 9, pp. 1–29, 2015.

[8] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, Oct. 2011.

[9] K. Neumann and J. J. Steil, "Learning robot motions with stable dynamical systems under diffeomorphic transformations," *Robot. Auton. Syst.*, vol. 70, no. C, pp. 1–15, 2015.

[10] Y. Huang, D. Büchler, O. Koç, B. Schölkopf, and J. Peters, "Jointly learning trajectory generation and hitting point prediction in robot table tennis," in *Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots,* 2016, pp. 650–655.

[11] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation," *IEEE Robot. Autom. Mag.*, vol. 17, no. 2, pp. 44–54, Jun. 2010.

[12] J. Silvério, S. Calinon, L. Rozo, and D. G. Caldwell, "Learning competing constraints and task priorities from demonstrations of bimanual skills," arXiv:1707.06791, to be published.

[13] N. Figueroa, A. L. P. Ureche, and A. Billard, "Learning complex sequential tasks from demonstration: A pizza dough rolling case study," in *Proc. 11th ACM/IEEE Int. Conf. Human Robot Interaction*, Mar. 2016, pp. 611–612.

[14] A. Ureche, K. Umezawa, Y. Nakamura, and A. Billard, "Task parameterization using continuous constraints extracted from human demonstrations," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1458–1471, Dec. 2015.

[15] S. R. Buss, "Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods," *IEEE J. Robot. Autom.*, vol. 17, no. 1, p. 16, May 2004.

[16] S. S. M. Salehian, M. Khoramshahi, and A. Billard, "A dynamical system approach for softly catching a flying object: Theory and experiment," *IEEE Trans. Robot.*, vol. 32, no. 2, pp. 462–471, Apr. 2016.

[17] S. S. M. Salehian, N. Figueroa, and A. Billard, "Coordinated multi-arm motion planning: Reaching for moving objects in the face of uncertainty," in *Proc. Robot.: Sci. Syst.*, Ann Arbor, MI, USA, Jun. 2016.

[18] S. Kim, R. Haschke, and H. Ritter, "Gaussian mixture model for 3-~~def~~ orientations," *Robot. Auton. Syst.*, vol. 87, pp. 28–37, 2017.

[19] A. Ude, B. Nemec, T. Petri, and J. Morimoto, "Orientation in ~~cartesian~~ space dynamic movement primitives," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2014, pp. 2997–3004.

[20] M. J. A. Zeestraten, I. Havoutis, J. Silvrio, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on Riemannian manifolds," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1240–1247, Jul. 2017.

[21] S. Calinon and A. Billard, "A probabilistic programming by demonstration framework handling constraints in joint space and task space," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 367–372.

[22] M. Hersch and A. Billard, "Reaching with concurrent dynamical systems," *Auton. Robots*, vol. 25, pp. 71–83, 2008.

[23] W. A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," in *Proc. 23rd IEEE Conf. Decis. Control*, 1984, pp. 1359–1363.

[24] Z. Shi, X. Huang, T. Hu, Q. Tan, and Y. Hou, "Weighted augmented ~~jacobian~~ matrix with a variable coefficient method for kinematics mapping of space teleoperation based on human-robot motion similarity," *Adv. Space Res.*, vol. 58, pp. 1401–1416, 2016.

[25] M. J. Gielniak, C. K. Liu, and A. L. Thomaz, "Stylized motion generalization through adaptation of velocity profiles," in *Proc. 19th Int. Symp. Robot Human Interactive Commun.*, 2010, pp. 304–309.

[26] Z. Emedi and A. Karimi, "Fixed-structure LPV discrete-time controller design with induced l2-norm and h 2 performance," *Int. J. Control*, vol. 89, no. 3, pp. 494–505, 2016.

[27] L. Sciavicco and B. Siciliano, "A solution algorithm to the inverse kinematic problem for redundant manipulators," *IEEE J. Robot. Autom.*, vol. 4, no. 4, pp. 403–410, Aug. 1988.

[28] M. Hayashibe and S. Shimoda, "Synergetic learning control paradigm for redundant robot to enhance error-energy index," *IEEE Trans. Cogn. Develop. Syst.*, to be published.

[29] I. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer-Verlag, 1986.

[30] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Proc. Int. Conf. Artif. Neural Netw.*, 1997, pp. 583–588.

[31] H. Cruse and M. Brüwer, "The human arm as a redundant manipulator: The control of path and joint angles," *Biol. Cybern.*, vol. 57, no. 1, pp. 137–144, 1987.

[32] T. Okadome and M. Honda, "Kinematic construction of the trajectory of sequential arm movements," *Biol. Cybern.*, vol. 80, no. 3, pp. 157–169, 1999.

[33] J. Bilmes, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden Markov models," *Int. Comput. Sci. Inst.*, Berkeley, CA, USA, Tech. Rep. TR-97-021, 1998.

[34] E. Nadaraya, "On estimating regression," *Theory Probob. Appl.*, vol. 9, pp. 141–142, 1964.

[35] G. S. Watson, "Smooth regression analysis," *Sankhyā: Indian J. Statist., Ser. A*, vol. 26, pp. 359–372, 1964.

[36] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics). Secaucus, NJ, USA: Springer-Verlag, 2006.

[37] S. S. M. Salehian, N. Figueroa, and A. Billard, "A unified framework for coordinated multi-arm motion planning," *Int. J. Robot. Res.*, 2018.

[38] L. van der Maaten, E. O. Postma, and H. J. van den Herik, "Dimensionality reduction: A comparative review," , Tilburg Univ., Tilburg, The Netherlands, Tech. Rep. TiCC TR 2009005, 2008.

[39] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *Proc. IEEE Int. Symp. Comput. Aided Control Syst. Des.*, 2004, pp. 284–289.

GENERAL INSTRUCTIONS

- Authors: When accessing and uploading your corrections at the Author Gateway, please note we cannot accept new source files as corrections for your letter. Do not send new Latex, Word, or PDF files, as we cannot simply "overwrite" your letter. Please submit your corrections as an annotated PDF or as clearly written list of corrections, with location in letter. You can also upload revised graphics to the Author Gateway.
- Authors: If you need an invoice or have any other billing questions, please contact reprints@ieee.org as they handle these requests.

QUERIES

Q1. Author: Please verify that the funding information is correct.
Q2. Author: There was a discrepancy between the source file and the pdf. We have followed the source file. Please check.
Q3. Author: Please update Refs. [12] and [28].
Q4. Author: Please provide full page range in Refs. [15] and [17].
Q5. Author: Please provide volume number and page range in Ref. [37].