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

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

***Abstract*— In this paper, we propose an asymptotically stable joint-space dynamical system that captures desired behaviors in joint-space while stably converging towards a its task-space attractor. To encode joint-space behaviors while meeting the stability criteria, the dynamical system is constructed as a Linear Parameter Varying (LPV) system combining different motor synergies, and we 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 semi-definite optimization problem that minimizes the joint velocity prediction error from the demonstrations. Our proposed approach is empirically validated on a variety of motions.**

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 the 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].

Many LfD approaches learn motions in either joint space [4], [5] or task space [6], [7]. These approaches generally learn probabilistic model of the demonstrated motions. Both spaces are independently useful: task-space behaviors let us control the most task-relevant component of the robot's behavior. In particular, a number of approaches have pioneered stable convergence to an attractor in task space [8], [9]as a desirable capability for a dynamical system.

In many cases, one may want to specify a joint-space behavior to directly define the behavior of the physical robot while simultaneously fulfilling 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.

*These authors contributed equally to this work.

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), 1015 Lausanne, Switzerland.
E-mail: {nadia.figueroafernandez,sina.mirrazavi,aude.billard}@epfl.ch.

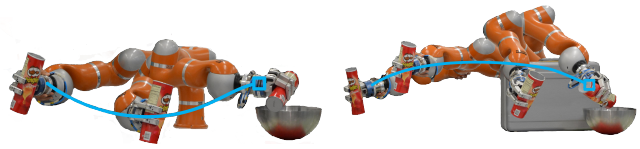


Fig. 1: Two robot motions in **joint-space** accomplishing *similar* behavior in **task-space** (pouring chips in a bowl). The left example avoids a known obstacle in its workspace, while the right one does not.

Furthermore, learning a motion in joint space allows us to avoid inverse kinematic (IK) approximations. The previously discussed task-space dynamical systems 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 non-linear task-space behavior, regardless of a specific joint-space constraint, this approach has been deemed sufficient [13], [14]. However, for other applications, such an approach yields significant problems [15]. First of all, finding the pseudo-inverse is computationally taxing. Moreover, 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. These problems encapsulate the main source of inaccuracies in tasks that require generating fast dynamical motions, such as catching or reaching for moving objects [16], [17]. By defining a controller in joint space, one can avoid IK and all its associated drawbacks.

Several approaches have tackled the problem of learning joint-space behaviors with task-space objectives. [18] 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. [19] learns from demonstration in both joint and task space within the "thin-plate spline" trajectory warping framework. [20] proposed an approach with similar properties to this 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 must be carefully tuned in order 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 provably stable behavior in joint space, and modulates 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, and then learning 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 in **synergy space**.
3) can transit through kinematic **singularities**.

The most similar approach to our proposed DS is the Jacobian transpose (JT) control method [21] . 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 position, 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 [22], 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 capable of discovering 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. We direct the system using a joint position or joint velocity controller, which can have joint position limits and a maximum joint velocity. 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 n^{th} 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 position as $x \in \mathbb{R}^d$.[1] The kinematics of the robot are assumed to be known, hence, the robot's forward kinematics is indicated by $x = H(q)$ and its Jacobian is $J(q) = \frac{dx}{dq} \in \mathbb{R}^{m \times d}$.
We wish to formulate a dynamical system $\dot{q} = f(q)$ which satisfies the following two criteria:

(I) The dynamical system 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. $\dot{V}(q) \leq 0 \; \forall q \in Q$ and $V(q) = 0 \; \forall q \in Q^*$. Where $Q = \{q | q^i_{min} < q^i < q^i_{max}, \; \forall i \in \{1, \dots, d\}\}$ and $Q^* = \{q | H(q) = x^* \wedge q \in Q\}$. $P \in \mathbb{R}^{d \times d}$ is a symmetric and positive definite matrix. $V(.)$ can be thought of as a metric for the task-space distance-to-go.

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

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

where $\dot{q}_d$ is the desired "true" velocity, and $f(\cdot)$ is the motion generation policy.

The error metric (2) is advantageous in a way that it mimics the magnitude and direction of the demonstrated motions. Hence, as suggested by [23], it is visually most similar to the human definition of "joint motion style".

III. Augmented Joint-Space Task-oriented Dynamical System

We propose the following augmented Joint-space Task-oriented Dynamical System (JTDS) to achieve the two criteria presented in (1) and (2).

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

where $\mathcal{A}(q) \in \mathbb{R}^{m \times m}$ is constructed using the LPV system paradigm [24], [16], where the overall $\mathcal{A}(q)$ is a changing linear combination of constant matrices, each of which encodes a local joint-space synergy.

$$\mathcal{A}(q) = \sum_{k=1}^{K} \theta_k(q) A_k \tag{4}$$

where $A_k \in \mathbb{R}^{m \times m}$ are the different synergies and $\theta_k(q) \in \mathbb{R}^1 \; \forall k \in \{1, \dots, K\}$ are the scheduling parameters[3] modulating each local synergy through time as well as space.

[1] For sake of brevity and simplicity, the time index, t, is dropped throughout the paper.

[2] Unless otherwise specified,"stability" in this paper always refers to asymptotic stability within the workspace of the robot, and 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.

[3] It is noteworthy that the presented stability proof can easily be extended for $\theta_k(t, q(t), d(t))$, where $d(t)$ is an external signal.

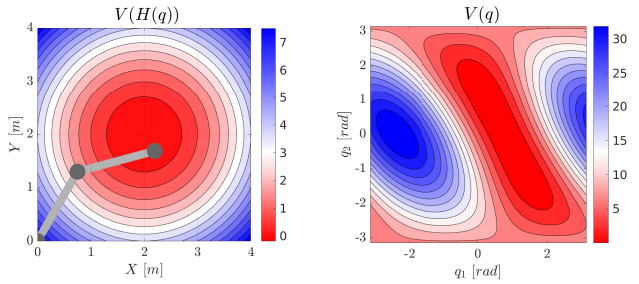


Fig. 2: Illustration of our proposed Lyapunov function and error projection. On the left we show the task-space potential function that we are trying to descend on for a 2-DOF robot with target $x^* = [2, 2]^T$. On the right we show the corresponding Lyapunuv function in joint space (1). The colors on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

Based on (3), one can utilize different synergies in different regions, and compose them to create a more non-linear multi-behavior motion.

Before proving that the proposed dynamical system satisfies all of the criteria, let us establish an intuitive understanding of the components of the system. While it may seem daunting at first, each of the elements has a straightforward explanation. One can intuitively understand the control law as follows: $(H(q) - x^*)$ denotes the position error w.r.t. the task target and by multiplying that error by the transposed Jacobian $J^T(q)$, the error is projected into joint space (similar to Jacobian transpose control [21], [25]), 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 flow of motion generated by the dynamical system (3) accomplishes criteria (I) if $\forall k \in \{1, \ldots, K\}$ (3) meets the following constraints.

$$\begin{cases} 0 \prec A_k, & 0 \leq \theta_k(\cdot) \end{cases} \tag{5}$$

Proof: See Appendix I. ■

Criterion (II) (i.e. encoding specific joint-space behaviors) is achieved by embedding the desired dynamics in the matrices $A_k(q)\ \forall k \in \{1, \ldots, K\}$. We describe in the next section an approach to automatically learn these matrices from demonstrated data.

IV. Learning Joint-Space Task-oriented Dynamical System in Synergy Space

The behavior of the JT-DS algorithm can be best understood through the lens of synergy control [26]. 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 $A_1, A_2, \ldots, A_k$, and $\mathcal{A}(q)$ represents the resulting motion constructed from a superposition of different synergies (through (3)).

A central question becomes, how to modulate the synergies in different regions to yield our desired behavior?

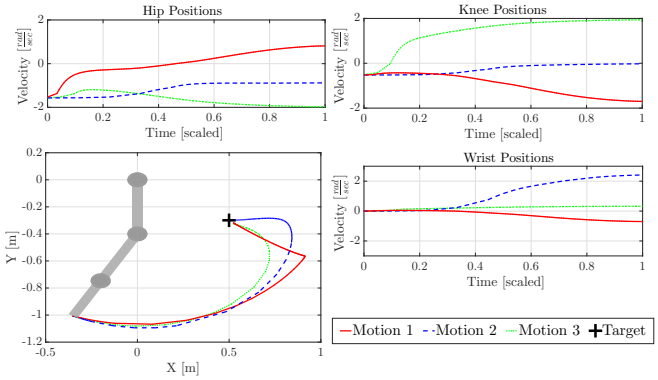


Fig. 3: Three example 3-DOF motions (A, B, C), each with a different constant joint augmentation matrix $\mathcal{A}(q)$ (emphasizing the hip, knee, and ankle respectively), A: $\mathcal{A}(q) = diag(5, 1, 1)$, B: $\mathcal{A}(q) = diag(1, 5, 1)$, and C: $\mathcal{A}(q) = diag(1, 1, 5)$. On the left, 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.

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 $\phi : Q \rightarrow Z$), (II) finding a policy for modulating the synergies in different regions of the synergy space (defined by $\theta_k(q)\ \forall k \in \{1, \ldots, K\}$), and (III) finding parameters for the synergies themselves (defined by $A_k\ \forall k \in \{1, \ldots, K\}$). Our choice of parameters must also obey the constraints laid out in (5). 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) into a lower-dimensional embedding $\phi(q) \in \mathbb{R}^{\delta \leq d}$. In each experiment, we used either Principal Component Analysis (PCA) [27] or Kernel PCA (KPCA) with RBF kernel [28].

(II) We then jointly estimate the optimal number K of synergy "regions" and the parameters of the scalar functions that determine the scheduling parameters $\theta_k(q)$ for weighting these synergies, by fitting a Gaussian Mixture Model (GMM) on the projected joint positions $\phi(q)$ seen in the demonstrations.

(III) Finally, once the local synergy regions have been found (described by each of the Gaussian distributions $\theta_k(q)$), we compute the corresponding joint synergy matrices $A_k\ \forall k \in \{1, \ldots, K\}$ 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).

[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 $\pi_B(\phi(a))$

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 corresponded to an important source of variation in the demonstrated trajectories, and which is thus suitable for learning 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 [29], [30]. 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 an embedding $\phi(q)$ by training dimensionality reduction techniques on the demonstrated trajectories. The learned embedding $\phi(\cdot)$ maps a joint configuration $q \in \mathbb{R}^d$ into a lower-dimensional configuration $z \in \mathbb{R}^\delta$, where $\delta < d$.

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 $D = \{\{\phi(q_{t,n})\}_{t=1,\dots,T}\}_{n=1,\dots,N}$ 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 scheduling parameter $\theta_k(q) = \theta'_k(\phi(q))$. Moreover, we would like for our scheduling parameters $\theta'_k(\phi(q))$ to have the following properties: (i) $0 \prec \theta'_k(\phi(q))$ and (ii) $\sum_{k=1}^{K} \theta'_k(\phi(q)) = 1$.

Scheduling parameters for LPV systems with such properties 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 scheduling parameter θ'_k to the kth synergy of (4). We define $\theta'_k(\phi(q))$ as $p(k|\phi(q))$:

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

which is the probability of the projected joint-position $\phi(q)$ belonging to the k-th synergy. 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.

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

C. Estimating the Synergy Matrices

Given the parameters of $\theta'_k(\phi(q))\ \forall k \in \{k = 1, \dots, K\}$, from (4) one can construct $\mathcal{A}(q)$ as a linear combination of local A_k synergy matrices weighted by their scheduling parameters $\theta'_k(\phi(q))$ as follows:

$$\mathcal{A}(q) = \frac{\sum_{k=1}^{K} A_k \pi_k \mathcal{N}(\phi(q); \mu_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\phi(q); \mu_k, \Sigma_k)}. \tag{7}$$

Notice the resemblance of (7) to the Nadaraya-Watson kernel estimator [36], [37][6] with a Gaussian pdf as its kernel function. Hence (7) 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" in synergy space is defined by a Gaussian distribution, and 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 behaviors can be reduced to a convex semidefinite optimization, with the goal of minimizing (2). To achieve this, the following optimization is proposed which uses mean square error as a means to minimize the joint velocity error from (2) as follows:

$$\begin{aligned} &\min_{A_1,\dots,A_K} \sum_{n=1}^{N} \sum_{t=0}^{T_n} \|\dot{q}_{t,n} - f(q_{t,n})\| \\ &\text{subject to} \\ &0 \prec A_k,\ \forall k \in \{1, \dots, K\}. \end{aligned} \tag{8}$$

where $f(q_{t,n})$ is calculated by (3) with (7), and x^*_n is defined as the endpoint of the n^{th} demonstrated trajectory.

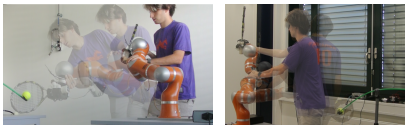
V. Experimental Validation

A. JT-DS Learning Performance Evaluation

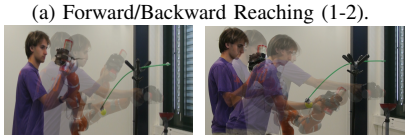
To evaluate the proposed JT-DS learning algorithm we collected kinesthetic demonstrations on a 7-DOF KUKA LWR 4+ robot arm, for 7 different tasks (see Table I and Fig. 5) which require mimicking the demonstrated joint-space behavior while reaching for a single target in task space:

- *(1-2) Forward/Backward reaching:* We guide the robot to forward and backward reaching motions towards a specific target in task-space with joint-space trajectories mimicking forehand/backhand strokes as shown in Fig. 4a.

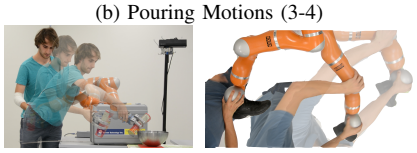
[6]The Nadaraya-Watson kernel estimator is used to estimate an unknown regressive function $m(x) = \mathbb{E}\{Y|X\}$, which takes the general form of $\widehat{m}(x) = \frac{\sum_{i=1}^{n} y_i \mathcal{K}(x, x_i)}{\sum_{i=1}^{n} \mathcal{K}(x, x_i)}$ where $\mathcal{K}(x, x_i)$ is a kernel function denoting the distance or similarity of x_i to the given location x. [36], [37]



(a) Forward/Backward Reaching (1-2).



(b) Pouring Motions (3-4)



(c) Pouring Motion (5) and Foot-step (6).

Fig. 5: Demonstrated Joint-Space Behaviors with Task-Space Targets. [videos]

Behavior Dataset	Dim. Red. Approach	Optimal Parameters — Joint Velocity RMSE $[rad/s]$ Optimal δ	Optimal K	RMSE Train	RMSE Test
(1) Forward Reaching ($N = 10, M = 1424$)	None	7	5.5 ± (2.99)	**0.271 ± (0.038)**	0.827 ± (0.697)
	PCA	4.1 ± (0.316)	4.8 ± (2.347)	0.276 ± (0.033)	0.589 ± (0.367)
	K-PCA ($\sigma_{opt} = 5.488$)	**3.9 ± (0.316)**	**3.2 ± (0.421)**	0.305 ± (0.021)	0.433 ± (0.104)
(2) Backward Reaching ($N = 11, M = 1223$)	None	7	5.1 ± (2.643)	**0.192 ± (0.033)**	0.728 ± (0.709)
	PCA	3.1 ± (0.316)	3.5 ± (0.971)	0.216 ± (0.021)	0.469 ± (0.164)
	K-PCA ($\sigma_{opt} = 6.238$)	**3.1 ± (0.316)**	**3.1 ± (0.316)**	0.219 ± (0.013)	0.355 ± (0.068)
(3) Pouring - Free ($N = 9, M = 1032$)	None	7	**4.1 ± (1.286)**	0.163 ± (0.026)	7.609 ± (18.99)
	PCA	**2.3 ± (0.483)**	4.8 ± (2.044)	**0.161 ± (0.034)**	0.623 ± (0.301)
	K-PCA ($\sigma_{opt} = 3.92$)	3	4.3 ± (1.767)	0.171 ± (0.106)	0.511 ± (0.305)
(4) Pouring - Obstacle 1 ($N = 11, M = 1232$)	None	7	6.2 ± (2.573)	**0.247 ± (0.039)**	3.049 ± (3.113)
	PCA	3.8 ± (0.421)	3.8 ± (1.549)	0.292 ± (0.018)	0.486 ± (0.125)
	K-PCA ($\sigma_{opt} = 7.695$)	**3.1 ± (0.421)**	**3.1 ± (0.316)**	0.288 ± (0.020)	0.482 ± (0.078)
(5) Pouring - Obstacle 2 ($N = 7, M = 1406$)	None	7	4.3 ± (1.418)	**0.122 ± (0.014)**	3.095 ± (3.406)
	PCA	**2.9 ± 0.316**	3.7 ± (2.213)	0.126 ± (0.02)	0.825 ± (0.764)
	K-PCA ($\sigma_{opt} = 2.86$)	3.9 ± (0.316)	**3**	0.129 ± (0.013)	0.738 ± (0.507)
(6) Foot Step ($N = 8, M = 1058$)	None	7	4.4 ± (2.366)	**0.080 ± (0.008)**	0.368 ± (0.265)
	PCA	**1**	3.2 ± (0.421)	0.084 ± (0.004)	0.123 ± (0.011)
	K-PCA ($\sigma_{opt} = 1.513$)	3.2 ± (0.421)	**3**	0.091 ± (0.007)	0.115 ± (0.015)
(7) Singularity Motions ($N = 10, M = 1467$)	None	7	6.2 ± (2.898)	0.054 ± (0.009)	0.137 ± (0.098)
	PCA	**2**	6 ± (2.357)	**0.053 ± (0.010)**	0.105 ± (0.046)
	K-PCA ($\sigma_{opt} = 1.769$)	3.6 ± (0.516)	**5.3 ± (2.452)**	0.055 ± (0.018)	0.071 ± (0.010)

TABLE I: Performance Comparison of Learning Approach with different Dim. Red. Schemes. We present the `mean(std)` for the optimal δ, K and joint-velocity RMSE on training and testing set, found for every learning scheme over 10 runs. $M = \sum_{n=1}^{N} T_n$.

- *(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 $q^2 \in \{10°, 20°, \dots, 100°\}$, 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 δ to represent our activation function $\theta(q)$ and the least number of local behavior synergies K.

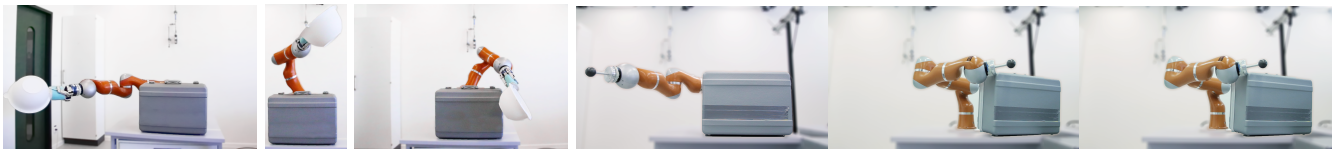
We thus evaluate three learning approaches with different dimensionality reduction algorithms: (1) None, (2) PCA and (3) K-PCA with RBF kernel. We perform 10-fold cross-validation on all datasets, for each learning approach, with a training/testing ratio of 60%. To evaluate performance, we take the square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

Performing such cross-validation in our case, is however not trivial, as our $\mathcal{A}(q)$ formulation has several hyper-parameters: the dimensionality δ 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 δ, K and σ (when applicable), as follows:

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

(II) *δ 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 $\delta \ll d$, in fact it can generate $\delta \leq M$. 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 $\delta > d$ and resample it. By doing this procedure, we can ensure that for all values in truncated $\bar{\sigma}_{range}$ we will obtain $\delta \leq d$. 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) [35]. 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



(a) Execution of Pour Obstacle 2 learned through JT-DS.

(b) Execution of Pour Obstacle 2 learned through SEDS.

Fig. 6: Snapshots of the robot experiments. A corresponding video is available on-line [video link].

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 (8). 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 $\{\dot{q}_1, \ldots, \dot{q}_T\}$, 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

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 represent our activation functions $\theta(q)$ yields better generalization capabilities than the contrary. This is most notable for the three pouring motions, where the joint-velocity RMSE testing error for a JT-DS model learnt 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 number of K needed to represent the motion. For all datasets, the DR methods provided $\delta < d/2$, 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 [31] 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 < 2ms 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 [34] for implementations of PCA, K-PCA and its out-of-sample extension. The

Fig. 7: Execution of *problematic* joint-space behaviors learned through JT-DS: (left) Foot step-like motion and (right) Singularity motions.

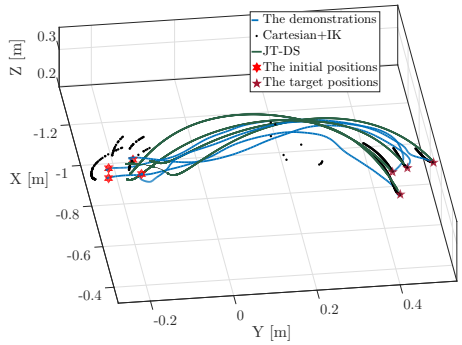


Fig. 8: End-effector trajectories for the footstep motion in Cartesian space. The JT-DS motion moves smoothly closely resembles the demonstrated trajectories. On the other hand, the SEDS-based Cartesian motion generator (whose values here are simulated because they can not physically executable) quickly becomes unstable, as evidenced by the dotted paths starting on the left side and abruptly disappearing.

YALMIP framework [33] was used to solve the definite convex optimization problem. Source code for the entire learning pipeline and cross-validation with simulation of 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 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 herein 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

	Norm. converg. [s/m]	Comp. time [ms]	Track. error [m]
SEDS + IK	10.6566 ± 2.1364	71.7752 ± 4.9189	0.0163 ± 0.0063
JT-DS	14.1926 ± 5.26453	12.1736 ± 0.8573	0.0 ± 0.0

TABLE II: Each simulated trajectory is initialized at $q = [0 \ \dots \ 0]^T$. The convergence duration is the time required to move within 0.001m of the target. The normalized convergence duration is the convergence duration divided by the distance between the initial and target positions.

those of SEDS+IK for behaviors *(5)* and *(6)*. In Fig. 6a and 6b 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 does not have the capabilities 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 (Fig. 7 and 8). This demonstrates the JT-DS algorithm's ability to move cleanly in and out of singularities.

2) Systematic Assessment: To systematically determine the performance properties (tracking error, computation time, and convergence time) of JT-DS (3) and compare it to a SEDS+IK, we simulated 400 simple motions on each system. The systems were tasked with moving to a fixed target randomly chosen from the region $\begin{bmatrix}-0.0081 \pm 0.3 & -0.0188 \pm 0.3 & 0.4974 \pm 0.21\end{bmatrix}^T$. The Cartesian motion generator was SEDS-based, and mapping from Cartesian motions to joint-space motions was done using a damped least-squares IK solver. To save on computation time, both the JT-DS and SEDS algorithms were taught behaviors defined by a single synergy, i.e. uniform behaviors. The results, summarized in Table II, indicate that the computation time of the proposed approach is significantly faster than the Cartesian-based DS, because the algorithm does not require calculating the Jacobian pseudo-inverse. As the generated joint motion by (3) is directly transmitted to the robot, the tracking error is zero. Nevertheless, the convergence time of (3) is slightly higher than SEDS+IK.

3) Transiting through singular configurations: One of the main advantages of the proposed DS is its ability to generate 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. 9 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. 7 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 Lyapunov-stable in task space

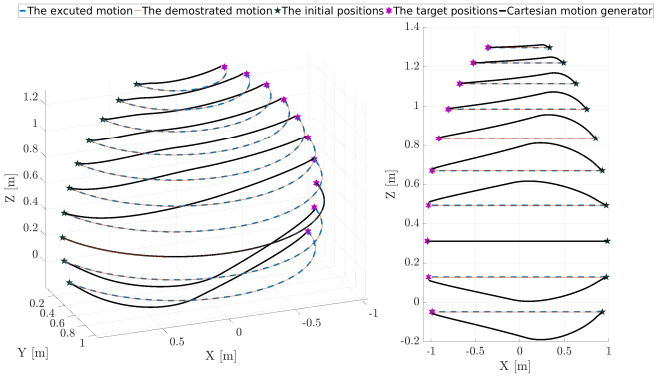


Fig. 9: The experiment generated $K = 3$ Gaussian components. As the first joint is the only joint which was not fixed during the demonstration, the learned augmentation matrices had only one nonzero entry $A_k(1,1) \neq 0 \ \forall k \in \{1,2,3\}$. In (a), the end-effector positions for the demonstrations 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).

and which replicates 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 redundant 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 scheduling parameters $\theta_k(\cdot)$ becomes numerically infeasible (all the Gaussians in (7) $\rightarrow 0$), and so $\frac{1}{K}\sum_{k=1}^{K} A_k$ is used to move the robot, which is still guaranteed to move towards the target. Moreover, in overlapping local regions where multiple A_k's might be in conflict, the presented system compromises between them while still stably converging to the target. The reason for this is that rather than "determining" the velocity of the system, our $\mathcal{A}$ matrix only warps it. This means that adding multiple "conflicting" synergies amounts to nothing more than repeatedly warping the velocity, and still maintains the original stability property.

Since we learn a lower-dimensional embedding space $Z := \{\phi(q) \mid q \in Q\}$, 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, and which we can thus guarantee will be positive definite and lead us to convergence to the target.

Finally, as the proposed DS might get stuck at its kinematic joint limits, 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 I
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:

$$\begin{aligned}\frac{dV(q)}{dt} &= (H(q) - x^*)^T J(q)\dot{q} \\ &= -(H(q) - x^*)^T J(q)\mathcal{A}(q)J^T(q)(H(q) - x^*) \\ &= -(H(q) - x^*)^T J(q) \sum_{k=1}^{K} \underbrace{\theta_k(q)}_{>0} \underbrace{A_k}_{\succ 0} J^T(q)(H(q) - x^*) \leq 0\end{aligned} \tag{9}$$

Therefore, JT-DS (3) is stable and as $q = q^*$ is the largest invariance in $\mathbb{R}^m$, it is globally asymptotically stable with respect to a task-space attractor; i.e. $\lim_{t\to\infty} \|H(q) - x^*\| = 0$.

ACKNOWLEDGMENT

This work was supported by EU project Cogimon H2020–ICT –23–2014. 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*. Springer Science & Business Media, 2006.

[2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer handbook of robotics*. Springer, 2008, pp. 1371–1394.

[3] B. Argall, S. Chernova, M. M. Veloso, and B. Browning, "A survey of robot learning from demonstration." *Robotics and Autonomous Systems*, 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," *Applied Intelligence*, vol. 17, no. 1, pp. 49–60, Jun. 2002.

[6] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2009, pp. 763–768.

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

[8] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 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, Aug. 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 *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*. IEEE, 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 Robotics & Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.

[12] J. Silvério, S. Calinon, L. Rozo, and D. G. Caldwell, "Learning competing constraints and task priorities from demonstrations of bimanual skills," *arXiv preprint arXiv:1707.06791*, 2017.

[13] N. Figueroa, A. L. P. Ureche, and A. Billard, "Learning complex sequential tasks from demonstration: A pizza dough rolling case study," in *2016 11th ACM/IEEE HRI Conference*, March 2016, pp. 611–612.

[14] A. Ureche, K. Umezawa, Y. Nakamura, and A. Billard, "Task parameterization using continuous constraints extracted from human demonstrations," *Robotics, IEEE Transactions on*, 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 Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 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 Transactions on Robotics*, vol. 32, no. 2, pp. 462–471, April 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 *Proceedings of Robotics: Science and Systems*, Ann-Arbor, Michigan, June 2016.

[18] S. Calinon and A. Billard, "A probabilistic programming by demonstration framework handling constraints in joint space and task space," in *IROS 2008*, 2008, pp. 367–372.

[19] A. X. Lee, S. H. Huang, D. Hadfield-Menell, E. Tzeng, and P. Abbeel, "Unifying scene registration and trajectory optimization for learning from demonstrations with application to manipulation of deformable objects," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4402–4407.

[20] M. Hersch and A. Billard, "Reaching with concurrent dynamical systems," *Autonomous robots*, 2008.

[21] W. A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," in *Decision and Control, 1984. The 23rd IEEE Conference on*. IEEE, 1984, pp. 1359–1363.

[22] 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 humanrobot motion similarity," *Advances in Space Research*, pp. –, 2016.

[23] M. J. Gielniak, C. K. Liu, and A. L. Thomaz, "Stylized motion generalization through adaptation of velocity profiles," in *19th International Symposium in Robot and Human Interactive Communication*. IEEE, 2010, pp. 304–309.

[24] Z. Emedi and A. Karimi, "Fixed-structure lpv discrete-time controller design with induced l 2-norm and h 2 performance," *International Journal of Control*, vol. 89, no. 3, pp. 494–505, 2016.

[25] L. Sciavicco and B. Siciliano, "A solution algorithm to the inverse kinematic problem for redundant manipulators," *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 403–410, 1988.

[26] M. Hayashibe and S. Shimoda, "Synergetic learning control paradigm for redundant robot to enhance error-energy index," *IEEE Transactions on Cognitive and Developmental Systems*, vol. PP, no. 99, 2017.

[27] I. Jolliffe, *Principal Component Analysis*. Springer Verlag, 1986.

[28] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *International Conference on Artificial Neural Networks*. Springer, 1997, pp. 583–588.

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

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

[31] S. S. M. Salehian, N. Figueroa, and A. Billard, "A unified framework for coordinated multi-arm motion planning," *Int. J. Rob. Res. [Conditionally Accepted]*, 2017.

[32] S. Calinon and A. Billard, "Recognition and reproduction of gestures using a probabilistic framework combining pca, ica and hmm," in *Proceedings of the 22nd ICML Conference*, 2005, pp. 105–112.

[33] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, 2004, pp. 284–289.

[34] L. van der Maaten, E. O. Postma, and H. J. van den Herik, "Dimensionality reduction: A comparative review," 2008.

[35] C. M. Bishop, *Pattern Recognition and Machine Learning (Informa-*

tion Science and Statistics). Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
[36] E. Nadaraya, "On estimating regression," *Theory of Prob. and Appl.*, vol. 9, pp. 141–142, 1964.
[37] G. S. Watson, "Smooth regression analysis," *Sankhyā Ser.*, vol. 26, pp. 359–372, 1964.
[38] T. Petrič and L. Žlajpah, "Smooth continuous transition between tasks on a kinematic control level: Obstacle avoidance as a control problem," *Robotics and Autonomous Systems*, vol. 61, no. 9, pp. 948–959, 2013.