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

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

***Abstract*— In this paper, we propose an asymptotically stable joint-space dynamical system that captures desired behaviors in joint-space while converging towards a reachable target in task-space. Our method is fast to compute and smoothly moves through classic kinematic singularities by avoiding the use of the pseudo-inverse Jacobian; moreover, it stably converges towards its task-space attractor. To encode complex joint-space behaviors while meeting these 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 validated on a variety of motions for a 7-DOF KUKA LWR4+ robot arm.**

I. Introduction

A. *Motivation for learning joint+task-space behaviors*

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 (i.e. target or desired trajectory), 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], [6], [7] or task space [8], [9], [10]. These approaches generally learn probabilistic model of the demonstrated motions, and construct a dynamical system that chooses behaviors according to the learned model. Both joint and task space are independently useful: joint-space behaviors let us directly influence the behavior of the physical robot, while 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 [11], [12], [13] as a desirable capability for a dynamical system.

*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.

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

In many cases, one may want to specify a joint-space behavior while simultaneously fulfilling a task-space objective. For example, throwing a ball is a motion characterized both by a task-space behavior (the ball moves backwards and then forwards) and also more obviously by its joint-space behavior (the elbow folding, the shoulder moving back and then swinging forward, and then the elbow straightening and the wrist snapping down). Similarly, learned joint-space behaviors are helpful in playing ping pong [14], grasping [15], and avoiding self-collisions of bimanual manipulators[16]. For a visual example of a task-space problem in which joint behavior is essential, see Fig. 1.

A second reason for learning a motion in joint space is that this 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 based Inverse-Kinematic (IK) approximations and variants thereof [1]. When the main focus is on executing a complex task-space behavior, regardless of a specific joint-space constraint, this approach has been deemed sufficient [17], [18]. However, for other applications, such an approach yields significant problems [19]. 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 [20], [21]. By defining a controller in joint space directly, one can avoid IK and all its associated drawbacks.

B. *Prior Work on Combined Joint and Task Space Control*

[22] 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. [15] and [16] 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.

[23] learns from demonstration in both joint and task space

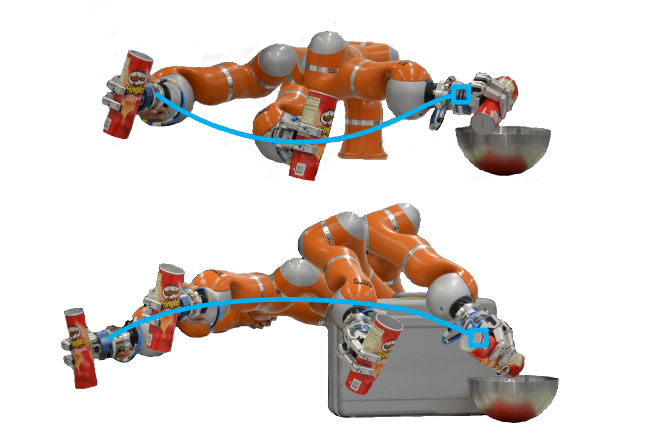


Fig. 1: Two robot motions in **joint-space** accomplishing *similar* behavior in **task-space** (pouring chips in a bowl). The bottom example avoids a known obstacle in its workspace, while the top one does not.

within the "thin-plate spline" trajectory warping framework, but does not propose a dynamical system for generating motions. [14] learns a controller for playing table tennis by combining joint-space Dynamic Movement Primitives [**?**] to mimic a desired task-space racket swing, but does not learn joint-space behaviors.

[24] 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.

C. Our Proposed Approach

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. We specifically chose to focus on linear parameter-varying dynamical systems because they are fast to compute and thus allow a controller to quickly react in a changing environment, a capability that is further improved by our not needing to compute pseudo-inverses. Our dynamical system:

1) computes a **complex motion in joint space** that provably and asymptotically stably converges to a **fixed task-space target**.
2) **avoids** the need for computing **pseudo-inverses**, and cleanly moves through kinematic "singularities".
3) is formulated such that complex **joint-space behaviors** can be **learned** from demonstrations in **synergy space**.

In brief, our algorithm learns a set of *behavior synergies*, each of which corresponds to a different provably stable behaviors in joint space, and modulates the use of these synergies throughout joint-space using a learned 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.

The most similar approach to our proposed DS is the Jacobian transpose (JT) control [25] method. JT control is an inverse kinematics method that yields a dynamical system in joint space $\dot{q} = f(q)$ which converges stably over time to a desired end-effector position x^*, 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 [26], this is to the best of our knowledge the first work to employ a JT system to learn behaviors from demonstrations.

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. The effectiveness of the proposed method is shown through simulations and experiments on a 7-DOF robot-arm in Section V. The paper is finalized with a discussion over our method and results 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

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

is stable; i.e. $\dot{V}(q) < 0 \; \forall q \in Q$ and $V(q) = 0 \; \forall q \in Q^*$. Where $Q = \{q | q^i_{min} < q^i < q^i_{max}, \; \forall i \in \{1, \dots, d\}\}$

[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.

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

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

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

We make two decisions by the choice of behavior error metric (2). First, the metric implicitly suggests that behaviors are defined by expert motions of the behavior, and that fulfilling a behavior means moving similarly to the experts' motion. Second, the type of feature of the trajectory that is being minimized is important. If instead the error feature were joint position distance $\sum_i \|q_{d,i} - q_i\|$ where q_d is a vector of demonstrated positions, then executing a "behavior" would imply mimicking position, but not velocity. Alternatively, if the error feature were joint velocity direction $\sum_i \left|\frac{\dot{q}_{d,i}}{\|\dot{q}_{d,i}\|} - \frac{\dot{q}_i}{\|\dot{q}_i\|}\right|$ mimicking a "behavior" would involve following the motion profile, but not the motion's speed (so for example a slap and a push would exhibit the same behavior). By choosing the combined direction and magnitude of the joint velocity as the error, we are choosing to mimic the magnitude and direction of the motion, which [27] suggests is visually most similar to the human definition of "joint motion style".

III. Augmented Joint-Space Task-oriented Dynamical System

Change semi-definite to definite everywhere
Introduce 'behavior synergies' here

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

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

where $P \in \mathbb{R}^{d\times d}$, and $\mathcal{A}(q) \in \mathbb{R}^{m\times m}$ is constructed using the Linear Parameter Varying system paradigm [28], [20], where the overall $\mathcal{A}(q)$ is a changing linear combination of constant matrices, each of which encodes a local joint-space synergy.

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

where $A_k \in \mathbb{R}^{m\times m}$ are the different synergies and $\theta_k(q) \in \mathbb{R}^1\ \forall k \in \{1,\ldots,K\}$ are the scheduling parameters[3] modulating each local synergy through time as well as space. Based on (3), one can utilize different synergies in different regions, and compose them to create a more complex multi-behavior motion.

[3]The scheduling parameters can be a function of time t, the states of the system q or external signals $d(t)$, i.e. $\theta_k(t, q(t), d(t))$. In this paper, we only consider it as a function of the states of the system. It is noteworthy that the presented stability proof can easily be extended for $\theta_k(t, q(t), d(t))$.

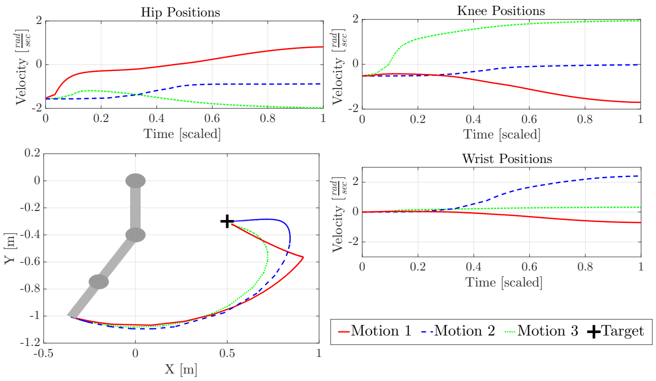


Fig. 2: Three example 3-DOF motions (A, B, C), each with a different constant joint augmentation matrix $\mathcal{A}(q)$ (emphasizing the hip, knee, and ankle respectively), A: $\mathcal{A}(q) = diag(5,1,1)$, B: $\mathcal{A}(q) = diag(1,5,1)$, and C: $\mathcal{A}(q) = diag(1,1,5)$. On the left, the task-space 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.

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 (warped according to P), and by multiplying that error by the transposed Jacobian $J^T(q)$, the error is projected into joint space (similar to Jacobian transpose control [25], [29]). The positive definite matrix $\mathcal{A}(q)$ warps the resulting joint-space velocity; Fig. 2 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 the P matrix as the **task augmentation matrix** (because it augments the task error, and thus the direction of motion) and $\mathcal{A}(q)$ as the **joint augmentation matrix** (because it augments the outputted joint velocities).

Proposition 1: The flow of motion generated by the dynamical system (3) accomplishes criteria (I) if (3) meets the following constraints.

$$\begin{cases} 0 \prec P \\ 0 \prec A_k \\ 0 \leq \theta_k(\cdot) \end{cases} \quad \forall k \in \{1,\ldots,K\} \quad (5a)$$

Proof: See Appendix I. ■

It is worth mentioning that the constraints in (5a) only ensure that JT-DS (3) asymptotically converges to the desired target position x^* if the robot does not have kinematic joint limits. If we introduce kinematic joint limits, then it's possible we may find $\exists q \in Q - Q^*$ such that both $\dot{V}(q) = 0$ and $H(q) \neq x^*$. This happens if the target position is kinematically unreachable; i.e. for each task-space axis i, $\left(J^T(q)P(H(q) - x^*)\right)_i = 0$ (meaning that moving joints increases the value of (1)), or the joint has reached a kinematic limit. In this case, the proof in Appendix I shows instead that the system converges in the sense of

Lyapunov, meaning that its distance to the attractor is always monotonically decreasing.

Criterion (II) (i.e. encoding specific joint-space behaviors) is achieved by embedding the desired dynamics in the matrices $A_k(q)\ \forall k \in \{1, \dots, K\}$. 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 [**?**]. 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 $A_1, A_2, \dots, A_k$, 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. First, we assume that our desired behavior can be efficiently described[4] using some 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: finding an underlying synergy-space Z in which the behavior can be accurately controlled (defined by a mapping $\phi : Q \to Z$), finding a policy for modulating the synergies in different regions of the synergy space (defined by $\theta_k(q)\ \forall k \in \{1, \dots, K\}$), and 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 (5a). We propose the following 3-step 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) [30] or Kernel PCA (KPCA) [31].

(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 $A_k\ \forall k \in \{1, \dots, K\}$ 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 Lower-Dimensional 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. For example, 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 [32], [33]. 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 $z \in \mathbb{R}^u$, where $u < d$. Specifically, we experimented with Principal Component Analysis (PCA) and Kernel PCA [31]. In previous work, PCA has shown to be sufficient to encapsulate the correlations in joint-space data [34], and this work further explores the usefulness of joint-space dimensionality reduction.

B. Discovering Local Behavior Synergies

The next step is identifying the regions of space in which to activate different synergies (though the synergies' parameters have not yet been specified). Specifically, 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 [20], [21]. 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

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

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

are the prior probabilities and $\{\mu_k, \Sigma_k\}$ are the mean and covariance matrices that parametrize the k-th multivariate Gaussian distribution. Each such distribution represents a local region of projected joint positions $\phi(q)$, and will be used to construct the scheduling parameter θ'_k of our LPV system relating to the kth synergy. We will 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 and choose the optimal number of components K, by evaluating and selecting the best resulting model using the Bayesian Information Criterion (BIC) [45].

C. Estimating the Synergy Matrices

Given the parameters of $\theta'_k(\phi(q))$ $\forall k \in \{k = 1, \ldots, K\}$, 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 [50], [51][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) [52].

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:

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

[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. [50], [51]

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

TABLE I: Datasets used to Evaluate JT-DS learning approach

Task Behavior Dataset	# Demos	# Samples (25Hz)
Backhand Stroke	11	1223
Forehand Stroke	10	1424
Pouring - Free	9	1032
Pouring - Obstacle	11	1232
Foot Swing	8	1058
Singularity Motions	10	1467

V. EXPERIMENTAL VALIDATION

A. JT-DS Learning Evaluation

The learning algorithm presented in the previous section, (8), was implemented in MATLAB. In order to solve the semidefinite convex optimization problem, the YALMIP framework [43] was used. In all the experiments, the optimization is initialized multiple times, and the best resulting run is used for performance analysis. The source code for learning (MATLAB) and execution (C++) of the Augmented Joint-Space Task-Oriented Dynamical System together with data generated in these experiments are available on-line:

Learning: https://github.com/epfl-lasa/JT-DS-Learning

Execution: https://github.com/epfl-lasa/JT-DS-lib

To evaluate the proposed JT-DS learning algorithm we collected demonstrations from 6 different tasks (see Table I) which require mimicking the demonstrated joint-space behavior while reaching for a single target in task space. These M demonstrations consist of (14-D) time-series $\boldsymbol{\xi}^{(i)} = \{\xi_1^{(i)}, \dots, \xi_T^{(i)}\}$ for $i = \{1, \dots, M\}$. The 14 dimensions in $\xi_t = [q_t, \dot{q}_t]^T$ correspond to joints-space positions and velocities.

2. Table with Evaluation of Learning Schemes for JTDS Models on Different Datasets and discussion of results

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.

	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 III: Parameters and Performance of Comparative Experiments This table is wrong.. re-do

	No. Dem.	No. K	$\phi(q)$ dim.	e_{totol} [rad/s]
Pour Obst. (JT-DS)	7	4	4	0.0137
Foot (JT-DS)	3	1	1	0.01344
Pour Obst. (SEDS)	7	3	-	-
Foot (SEDS)	3	1	-	-

B. JT-DS Performance Evaluation

The performance of the proposed framework was evaluated on a 7-DOF robot arm, the KUKA LWR 4+. The robot is controlled on the joint position level (linearly interpolating from joint velocities) at a rate of 500 Hz. The resultant joint angles are filtered by a critically damped filter to avoid high torques. Our empirical validation consists of three sections, each highlighting a different advantage of the proposed dynamical system: (i) moving in singular configurations, (ii) following desired behaviors in joint and task spaces simultaneously, and (iii) fast computation/convergence time.

1) Systematic Assessment: To systematically determine the performance properties (tracking error, computation time, and convergence time) of JT-DS (3) and compare it to a Cartesian motion generator, 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. Furthermore, 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 the Cartesian motion generator.

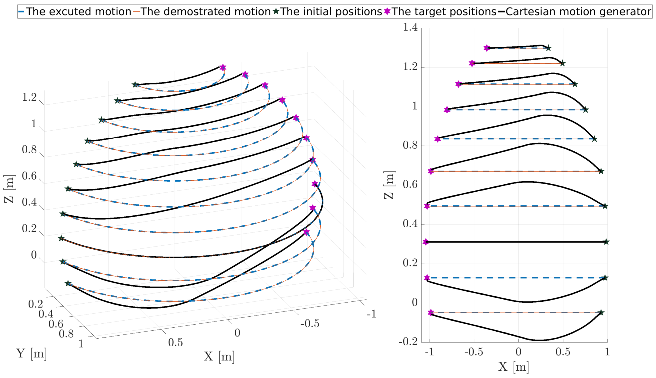
2) Following Desired Joint Behaviors: To evaluate the system's ability to track learned joint behaviors, we devised two experiments (summarized in Table III) comparing the tracking capabilities of our JT-DS method with those of the Cartesian-space SEDS [11] algorithm using damped least square IK. In the first experiment, the robot was guided through a complex motion task, moving an object through an environment with an obstacle. Human experts guided the robot through a series of demonstrations, ensuring that neither the robot's joints nor the held object intersected the obstacle. The results can be seen in Fig. 5a and Fig. 5b. The JT-DS algorithm mimicked the joint-space behaviors of the demonstration (e.g. folding the elbow, lowering the shoulder), and therefore successfully avoided the obstacle while still converging to the desired Cartesian position. Meanwhile, SEDS only learned the demonstrated behavior in task-space (the position of the end-effector is signified by a black ball) meaning that the robot was not constrained 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 we would expect), but did ultimately converge to its target position. In the second experiment the robot was taught a footstep-like motion, beginning with a straight leg, moving through a singularity, and finally bringing the knee up (Fig. 5c). JT-DS followed the demonstrations closely, while SEDS became unstable in the singularity (Fig. 4). This demonstrates the JT-DS algorithm's ability to move cleanly in and out of singularities, overcoming the Achilles heel of equivalent Cartesian-based motion generators.

Further inspection reveals another advantage of JT-DS over stable Cartesian motion generators: not needing to preprocess demonstrations. In Cartesian-space dynamical systems, in order to guarantee stability, every demonstration must have the same target position. As a result, all demonstrated trajectories are warped to end at a single position, introducing a non-negligible difference between the original demonstrations and those used to train the DS. JT-DS, on the other hand, does not require any demonstration warping to learn. Rather than shifting demonstrations, the algorithm learns behavior regions with respect to the base frame of the manipulator. This effect can be seen in the diagrams in Fig. 3a, which show a comparison between JT-DS and a Cartesian-space DS trained through SEDS [11]. The Cartesian-space system warps the demonstrations and as a result fails to track them, while JT-DS does not modify the demonstrations and thus succeeds.

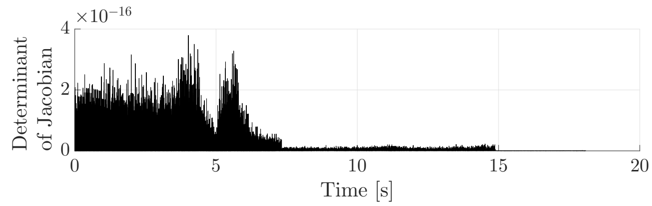
3) Moving in singular configurations: One of the main advantages of the proposed dynamical system is its ability to generate accurate paths in classical singular configurations. The first scenario, shown in Fig. 3a, was designed to assess this capability. The 10 training demonstrations were constructed as follows: 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 demonstrator back-driving it. This restricted the motion to a series of arcs of different radii along the robot's motion boundary. In other words, the demonstrated motions were entirely within a classic kinematic singularity. Fig. 3a 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.

VI. Discussion and Future Work

In this paper, we have presented a dynamical system in joint space that is provably Lyapunov-stable in task space and which replicates demonstrated joint-space behaviors. The



(a)



(b)

Fig. 3: 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). In (b), the determinants of the Jacobia ... ed. Due to sens ... y close.

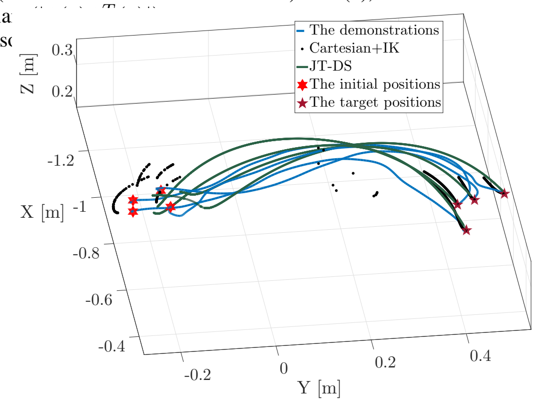


Fig. 4: A plot of the end-effector trajectories for the footstep motion in Cartesian space. The JT-DS motion moves smoothly closely resembles the demonstrated trajectories. On the other hand, the SEDS-based Cartesian motion generator (whose values here are simulated because they can not physically executable) quickly becomes unstable, as evidenced by the dotted paths starting on the left side and abruptly disappearing due to becoming too large.

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.

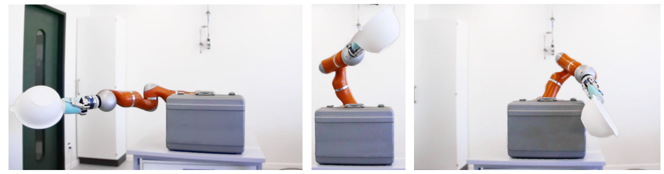
One of the most important points when validating a learning from demonstration method is to evaluate the system's behavior away from demonstrations. When the current joint configuration is far from any of the local synergy regions, computing the scheduling parameters $\theta_k(\cdot)$ becomes numerically infeasible (all the Gaussians in (7) $\rightarrow 0$), and so $\sum_{k=1}^{K} \frac{A_k}{K}$ 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 together amounts to nothing more than repeatedly warping the velocity, and still maintains the original stability property.

One of the major advantages of the proposed dynamical system is that it avoids the undesirable effects of traditional Inverse Kinematic (IK) solvers. Fast dynamical motions require not only fast and accurate motion planning, but also precise IK mapping, which is not practically feasible. By utilizing the JT-DS method, even a fast system could reactively generate stable motions to its target.

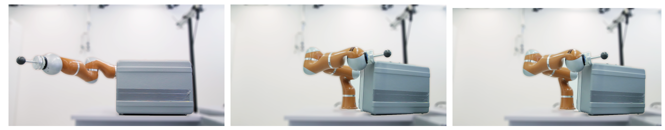
Since we learn a lower-dimensional embedding space $Z := \{\phi(q) \mid q \in Q\}$, which we suggest provides a better representation for the learned joint behaviors, a careful reader might wonder: why not define a motion policy π_Z directly in Z-space and then map the learned policy back out into joint space using the inverse embedding[7] $\phi^{-1}(z)$? 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.

A drawback of the approach is that the controller is not guaranteed to converge to the task-space target x^* even if a path to a valid configuration q^* exists. As mentioned in Sec. III, the controller will always minimize (or keep even) the warped distance to goal $(H(q) - x^*)^T P(H(q) - x^*)$, and in the process may get stuck at its kinematic joint limits. If the only valid trajectory to reach the goal requires that we temporarily move away from the target (and increase the warped distance-to-goal), this controller will not find that trajectory. There is a simple fix, however: rather than providing a single task-space objective x^*, provide a series of sequential task-space objectives $\mathbf{x}^* = \{x_0^*, x_1^*, \ldots, x_n^*\}$ and have the controller move to each in succession. The smaller subtrajectories give the robot less of an opportunity to deviate from the learned (valid) joint behavior and thus help it avoid being caught by the robot's joint limits. Further, specifying a sequence of task-space targets provides greater control over the robot's task-space trajectory while still exhibiting the learned joint-space behaviors.

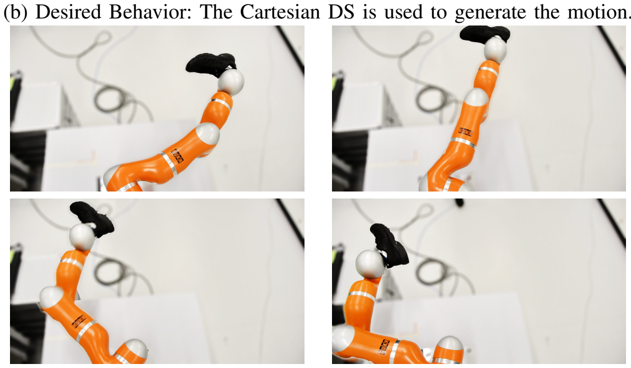
[7]Such an inverse is trivial to extract in the case where ϕ is obtained through PCA.



(a) Desired Joint Behavior: obstacle avoidance motion. The luggage is the obstacle.



(b) Desired Behavior: The Cartesian DS is used to generate the motion.



(c) Desired Joint Behavior: Foot step-like motion.

Fig. 5: Snapshots of the robot experiments. A corresponding video is available on-line https://youtu.be/1dgKfmN1UgE.

One application that we have discussed for this algorithm is obstacle avoidance. In an environment with static obstacles, the system would, through demonstration, learn joint-space behaviors that avoid those obstacles. One can compare this form of joint-space obstacle avoidance with other methods from the literature [29], [53]. The main difference is that these previous methods require explicit knowledge of the obstacles' position, whereas our algorithm learns the obstacle positions only implicitly by learning motions that avoid those positions. This lack of explicit obstacle location can be an advantage: encoding obstacles' geometry can be expensive and cumbersome. However, it is also a drawback: without knowing the exact location of obstacles, the algorithm cannot be guaranteed to avoid collisions, especially away from demonstrations.

Finally, we are currently working on improving the performance of this method (3) by considering a the second order dynamical system. This would provide numerous improvements: a guaranteed smoothness in the velocity profile, the ability to integrate critically damped filter, and the versatility to learn more complex behaviors. Are we actually still doing this?

Appendix I
Proving Stability of the Dynamical System

We wish to prove Proposition 1, that is, that JT-DS (3) (reproduced below)

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

accomplishes criterion (I), where $\mathcal{A}(q)$ is positive definite, and P is positive definite.

Theorem 1.1 (Proof of Lyapunov Stability): JT-DS (3) is asymptotically stable with respect to the Lyapunov candidate

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

That is,

$$0 \prec V(q) \quad V(q^*) = 0 \quad \forall q \neq q^* \tag{9}$$

where q^* is any joint configuration such that $H(q^*) = x^*$. The first two statements are trivially true because P is positive definite and the rest of V is a square that is only 0 when $H(q) = x^*$. To prove the last statement, we find $\frac{dV}{dt}$:

$$\begin{aligned} \frac{dV(q)}{dt} &= (H(q) - x^*)^T P J(q) \dot{q} \\ &= -(H(q) - x^*)^T P J(q) \mathcal{A}(q) J^T(q) P (H(q) - x^*) \\ &= -(H(q) - x^*)^T P J(q) \sum_{k=1}^{K} \underbrace{\theta_k(q)}_{0\prec} \underbrace{A_k}_{0\prec} \\ &\quad \cdot J^T(q) P (H(q) - x^*) < 0 \end{aligned} \tag{10}$$

By observation, each and every term in the final expression is multiplied by its transpose (creating a square) except for $\mathcal{A}(q)$, which is positive definite. This means that the expression is guaranteed to be negative definite. Therefore, JT-DS (3) is globally asymptotically stable with respect to a task-space attractor; i.e. $\|H(q) - x^*\|$ and $\dot{q}$ are bounded.

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] S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot," in *ACM/IEEE HRI*, 2007, pp. 255–262.

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

[6] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Control, planning, learning, and imitation with dynamic movement primitives," in *IROS*, Las Vegas, NV, Oct. 27-31, 2003.

[7] 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.

[8] 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.

[9] E. Gribovskaya, S. Khansari-Zadeh, and A. Billard, "Learning non-linear multivariate dynamics of motion in robotic manipulators," *Int. J. Rob. Res.*, vol. 30, no. 1, pp. 80–117, Jan. 2011.

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

[11] 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.

[12] 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.

[13] K. Kronander, M. Khansari, and A. Billard, "Incremental motion learning with locally modulated dynamical systems," *Robot. Auton. Syst.*, vol. 70, no. C, pp. 52–62, Aug. 2015.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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 *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 4402–4407.

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

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

[32] 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.

[33] T. Okadome and M. Honda, "Kinematic construction of the trajectory

of sequential arm movements," *Biological Cybernetics*, vol. 80, no. 3, pp. 157–169, 1999.

[34] 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.

[35] K. Guglielmo and N. Sadegh, "Theory and implementation of a repetitive robot controller with cartesian trajectory description," *Journal of Dynamic Systems, Measurement, and Control*, vol. 118, no. 1, pp. 15–21, 1996.

[36] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," in *ACM transactions on graphics (TOG)*, vol. 23, no. 3. ACM, 2004, pp. 522–531.

[37] M. Brand and A. Hertzmann, "Style machines," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 183–192.

[38] C. K. Liu, A. Hertzmann, and Z. Popović, "Learning physics-based motion style with nonlinear inverse optimization," *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 1071–1081, 2005.

[39] A. Colomé, "Smooth inverse kinematics algorithms for serial redundant robots," Ph.D. dissertation, 2011.

[40] 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.

[41] A. P. Shon, K. Grochow, and R. P. N. Rao, "Robotic imitation from human motion capture using gaussian processes." in *Humanoids*. IEEE, 2005, pp. 129–134.

[42] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011.

[43] 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.

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

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

[46] M. M. Fateh and H. Farhangfard, "On the transforming of control space by manipulator jacobian," *International Journal of Control, Automation, and Systems*, vol. 6, no. 1, pp. 101–108, 2008.

[47] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1049–1065, 2014.

[48] 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, 2002.

[49] A. Colom and C. Torras, "Closed-loop inverse kinematics for redundant robots: Comparative assessment and two enhancements," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 944–955, April 2015.

[50] E. Nadaraya, "On estimating regression," *Theory of Prob. and Appl.*, vol. 9, pp. 141–142, 1964.

[51] G. S. Watson, "Smooth regression analysis," *Sankhyā Ser.*, vol. 26, pp. 359–372, 1964.

[52] H. G. Sung, "Gaussian mixture regression and classification," Ph.D. dissertation, Rice University, Houston, Texas, 2004.

[53] 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.

[54] C. Alessandro, I. Delis, F. Nori, S. Panzeri, and B. Berret, "Muscle synergies in neuroscience and robotics: from input-space to task-space perspectives," *Frontiers in computational neuroscience*, vol. 7, 2013.