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*— Too long, re-write In this paper, we propose a stable joint-space dynamical system that captures desired behaviors in joint-space while converging towards a reachable target in task-space with respect to the robot's kinematic constraints. Our method is fast to compute and smoothly moves through classic kinematic singularities by avoiding the use of the pseudo-inverse Jacobian; moreover, we prove that it stably moves towards its target and is guaranteed to obey the robot's kinematic joint limits. To encode complex joint-space behaviors while meeting these stability criteria, the dynamical system is constructed as a Linear Parameter Varying (LPV) system, enabling us to encode complex joint-space behaviors. We propose a method to train such complex behaviors from kinesthetic demonstrations, by learning the parameters of the LPV system. We begin by applying a latent-embedding and a probabilistic approach to discover and represent a set of local behaviors from the demonstrated motions. Then, the corresponding local joint-space behaviors are estimated by solving a convex semi-definite optimization problem that minimizes the joint velocity error from the demonstrations. Our proposed approach is validated on a variety of motions for a 7-DOF KUKA LWR4+ robot arm.**

I. Introduction

Robot motion planning in joint-space has long been a major field of study [1]. For manipulation problems with an objective defined in task space (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.

Learning robot motion generators that follow desired joint and/or task-space behaviors is, in fact, the central focus of Programming by Demonstration (PbD) [2] [3]. Historically, much of the original work in PbD focused on learning motions solely in joint-space by parameterizing motions using GMMs [4], HMMs [5], GPs [6], or other probabilistic models [7], [8]. Concurrently, a significant body of work from the graphics community was targeted at learning joint-space motion "styles" from human motion capture data, through a variety of latent-space and stylistic probabilistic motion models [9], [10]. Yet, over time, research shifted to pure task-space learning [11], [12], [13], which has two main advantages over pure joint-space: *generalization* and *re-usability*. By learning motions in task-space, one can concisely encode the task-relevant features and generate new motions which meet these task-space criteria. Furthermore, a task-space trajectory is robot-independent, and thus can be reused across robots with differing morphologies.

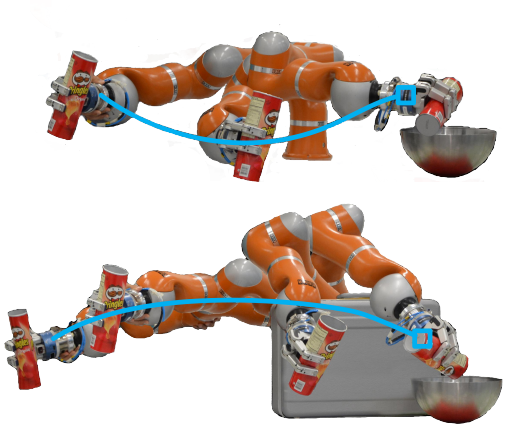


Fig. 1: Two robot motions in joint-space accomplishing the same behavior in task-space (pouring chips in a bowl). Though the **task-space trajectories** are *similar*, the **joint-space trajectories** are different; e.g. the bottom example moves in joint-space to avoid an obstacle that the top one does not.

A major drawback of these initial task-space methods was that despite learning motions similar to the demonstrations, there was no guarantee when applying the learned motion to a new problem (e.g. a different initial/target position) that the end-effector would stably converge to its ultimate task-space target. This led to the introduction and subsequent popularity of task-space learning techniques such as SEDS [14], τ-SEDS [15] and LMDS [16] which encode task-space behaviors as asymptotically stable dynamical systems (DS) guaranteed to converge towards a desired task-space target.

For all the previously mentioned task-space techniques, the robot is ultimately controlled by 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

*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@mit.edu.

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.

or reaching for moving objects [20], [21].

Although DS-based task-space learning techniques ensure target convergence and stability, an advantageous feat within the LfD paradigm, they still suffer from this reliance on IK methods, possibly generating erratic or discontinuous motions during execution. Further, for certain tasks, rather than blindly choosing the arbitrary joint-space trajectory returned by the IKs solver, it might be desirable to learn behaviors and "styles" in joint-space directly. For example, throwing a ball is a motion less characterized by task-space behaviors (the ball moving backwards and then forwards) and more characterized by the joint behaviors: the elbow folds, the shoulder moves back and then swings forward, followed by the elbow straightening and the wrist snapping down. For a visual example of the need for such combined task/joint-space behavior learning, see Fig. 1. This is precisely the shortcoming associated with LfD in task-space: it fails to learn the joint-space behavior of the system. The current formulation of the DS-based learning techniques introduced thus far (e.g. SEDS, τ-SEDS LMDS, etc.) do not allow for integrating combined task/joint-space behavior learning. [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 no longer guarantees convergence to the desired task-space target and is still reliant on computing the pseudo-inverse Jacobian.

In this work, we seek to devise an augmented **J**oint-space **T**ask-oriented **D**ynamical **S**ystem (JT-DS) that not only complies with the task-space constraints, but also avoids the problems generated by pseudo-inverse approximations and takes into consideration the robot's kinematic limits. [23] 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 and joint kinematic limits. The resulting DS avoids singularities through generalization of the pseudo-inverse approximations and joint limits via online modulation. 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 depart from any modulation or combination of DS and instead propose a novel dynamical system which:

1) computes a **complex motion in joint space** that provably and 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**.
4) is capable of complying with **kinematic joint limits**.

The most similar approach to our proposed DS is the Jacobian transpose (JT) control [24] method. JT control is an inverse kinematics method that yields a dynamical system in joint space $\dot{q} = f(q)$ which converges stably (in the sense of Lyapunov) 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 integrating discontinuous joint limits [25] and designing velocity adjustments by hand [26], this is the first known work to employ a JT system to learn behaviors from demonstrations while obeying the robots kinematic constraints.

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 criteria:

(I) The dynamical system must be Lyapunov 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. $\dot{V}(q) \leq 0 \; \forall q \in Q$ 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 must respect the kinematic joint constraints of the robot:

$$q(0) \in Q \rightarrow q(t) \in Q \quad \forall t > 0 \tag{2}$$

[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 global Lyapunov stability. We make no claim to proving global asymptotic stability, which is in fact impossible to achieve in a joint-constrained kinematic system. For example, any finite-size manipulator will always have regions beyond its reach that it cannot converge to.

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

$$e_{total} = \frac{1}{NT_n} \sum_{n=1}^{N} \sum_{t=0}^{T_n} \|\dot{q}_{d;t,n} - f(q_{t,n})\| \tag{3}$$

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 (3). 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 [9] suggests 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 three criteria presented in (I), (II), (III).

$$\dot{q} = f(q) = -\mathcal{A}(q) J^T(q) P(H(q) - x^*) \tag{4}$$

where

$$\mathcal{A}(q) = S(q) A(q) S^T(q). \tag{5}$$

$S(q) = diag(s^1(q), \dots, s^d(q))$ is a diagonal matrix and $s^i(\cdot) \in C^0 \ \forall i \in \{1, \dots d\}$. $A(q) \in \mathbb{R}^{d \times d}$ is constructed using the Linear Parameter Varying system paradigm [27], [20], where the overall $A(q)$ is a nonlinear combination of constant linear matrices, each of which encodes a local joint-space behavior.

$$A(q) = \sum_{k=1}^{K} \theta_k(q) A_k \tag{6}$$

where $A_k \in \mathbb{R}^{d \times d}$ and $\theta_k(q) \in \mathbb{R}^1 \ \forall k \in \{1, \dots, K\}$ are constant matrices and the scheduling parameters[3] representing each local behavior, respectively. Based on (4), one can enforce different local joint behaviors 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 and convergence proof can easily be extended for $\theta_k(t, q(t), d(t))$.

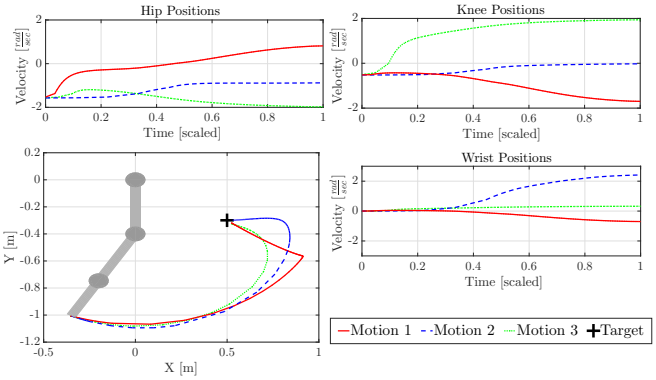


Fig. 2: Three example 3-DOF motions (A, B, C), each with a different constant behavior matrix $\mathcal{A}$ (emphasizing the hip, knee, and ankle respectively), A: $\mathcal{A}(q) = diag(5, 1, 1)$, B: $\mathcal{A}(q) = diag(1, 5, 1)$, and C: $\mathcal{A}(q) = diag(1, 1, 5)$. On the left, the task-space 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 [24], [25]). The positive semi-definite matrix $\mathcal{A}(q)$ warps the resulting joint-space velocity; Fig. 2 illustrates the effects of $A(q)$ on the generated motion. Thus the controller can be thought of as a proportional controller in joint space. We refer to the P matrix as the **task augmentation matrix** (because it augments the task error, and thus the direction of motion) and $A(q)$ as the **joint augmentation matrix** (because it augments the outputted joint velocities). Lastly, $S(q)$ intuitively enforces the joint constraints: as q^i nears q^i_{min}, the velocity in that direction goes to 0, and similarly when it nears q^i_{max} the velocity goes to 0.

Proposition 1: The flow of motion generated by the dynamical system (4) accomplishes criteria (I) and (II), if (4) meets the following constraints.

$$\begin{cases} 0 \prec P \\ 0 \preceq A_k \\ 0 \leq \theta_k(\cdot) \end{cases} \quad \forall k \in \{1, \dots, K\} \tag{7a}$$

$$\begin{cases} s^i(q^i_{min}) = 0 \\ s^i(q^i_{max}) = 0 \end{cases} \quad \forall i \in \{1, \dots, d\} \tag{7b}$$

Proof: See Appendix I. ■

It is worth mentioning that the constraints in (7a) do not ensure that JT-DS (4) asymptotically converges to the desired target position x^*, as $\exists q \in Q - Q^*$ 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 $S_{i,i}(q) = 0$ (the joint has reached its limit). Note that in practice, because $s(q)$ is continuous and designed based on (7b), executing JT-DS (4) will likely never cause the manipulator to reach the joint limit. Instead, $s(q)$ significantly shrinks the magnitude of a joint's motion when a joint nears its motion boundary. For a simple method of constructing $S(q)$, see Appendix II.

Criterion (III) (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

Talk about synergy space here, why we need it, etc.

JT-DS (4) is parametrized by $\theta_k(q)$ and $A_k\ \forall k \in \{1,\dots,K\}$. One is free to determine these parameters as long as that they satisfy constraints (7a). We propose a 3-step procedure as follows:

(I) We first project the demonstrated data (i.e. collections of joint positions q) into a lower-dimensional embedding $\phi(q) \in \mathbb{R}^{\delta \leq d}$ through Principal Component Analysis (PCA) [28], so as to reduce the dimensionality and extract correlations across similar joint behavior regions.

(II) We then jointly estimate the optimal number K of local behavior regions and the parameters of the scalar functions that determine the scheduling parameters $\theta_k(q)$, by fitting a Gaussian Mixture Model (GMM) on the projected joint positions $\phi(q)$.

(III) Finally, once the local behavior regions have been found (described by each of the Gaussian distributions $\theta_k(q)$), we compute the corresponding joint behavior 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 (3).

A. Embedding Joint Configurations in Latent Space

The search for a lower-dimensional representation of the joint positions stems from the desire to identify regions of *similar* "joint motion styles" (defined in Section I). It is important to understand that distances in joint space (defined as $\|\Delta q\|$) might not numerically reflect what a human intuitively perceives as the distance between joint configurations. Motor control studies have postulated that most human arm motions, be they reaching motions or following trajectories as straight/curved lines, 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 lower-dimensional space.

To this end, any type of topology-preserving latent-space embedding or dimensionality reduction approach can theoretically be used for $\phi(q)$. In this work, we choose to construct the projection $\phi(q)$ of q through a linear map $W : q \in \mathbb{R}^d \rightarrow \phi(q) \in \mathbb{R}^{\delta \leq d}$ given by $\phi(q) = Wq$, where the projection matrix $W \in \mathbb{R}^{\delta \times d}$ is computed through Principal Component Analysis (PCA). Although the projected dimensions extracted from PCA are linear combinations of the original variables, this has been proven to be sufficient to encapsulate the correlations in joint-space data [31].

B. Discovering Local Behavior Regions

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 behaviors, 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]. In this work, we adopt this approach and use a GMM to estimate the joint distribution over the projected joint positions[4], $p(\phi(q)) = \sum_{k=1}^K \pi_k \mathcal{N}(\phi(q); \mu_k, \Sigma_k)$, 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 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. We will define $\theta'_k(\phi(q))$ as $p(k|\phi(q))$:

$$\theta'_k(\phi(q)) = \frac{\pi_k \mathcal{N}(\phi(q); \mu_k, \Sigma_k)}{\sum_{k=1}^K \pi_k \mathcal{N}(\phi(q); \mu_k, \Sigma_k)} \tag{8}$$

which is the probability of the projected joint-position $\phi(q)$ belonging to the k-th behavior. Therefore, each joint-space behavior region is associated with a Gaussian component of the GMM, cumulatively describing all the behavior 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) [32].

C. Estimating the Joint Behavior Matrices

Given the parameters of $\theta'_k(\phi(q))\ \forall k \in \{k = 1,\dots,K\}$, from (6) one can construct $A(q)$ as a linear combination of local A_k matrices weighted by their scheduling parameters $\theta'_k(\phi(q))$ as follows:

$$A(q) = \frac{\sum_{k=1}^K A_k \pi_k \mathcal{N}(\phi(q); \mu_k, \Sigma_k)}{\sum_{k=1}^K \pi_k \mathcal{N}(\phi(q); \mu_k, \Sigma_k)}. \tag{9}$$

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

Notice the resemblance of (9) to the Nadaraya-Watson kernel estimator [33], [34][5] with a Gaussian pdf as its kernel function. Hence (9) 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) [35].

Intuitively, each "region of behavior" in joint/latent space is defined by a Gaussian distribution, and the closer the robot is to a region, the more that region's behavior (A_k) influences the robot's current joint-space motions. Finding the appropriate joint augmentation matrices A_k for the set of local behaviors can be reduced to a convex semidefinite optimization, with the goal of minimizing (3). To achieve this, the following optimization is proposed which uses mean square error as a means to minimize the joint velocity error from (3) as follows:

$$\begin{aligned} &\min_{A_1,\ldots,A_K} \sum_{n=1}^{N}\sum_{t=0}^{T_n} \|\dot{q}_{t,n} - f(q_{t,n})\| \\ &\text{subject to} \\ &0 \preceq A_k,\ \forall k \in \{1,\ldots,K\}. \end{aligned} \tag{10}$$

where $f(q_{t,n})$ is calculated by combining our dynamical system formulation (4) with (9), and x_n^* is defined as the endpoint of the n^{th} demonstrated trajectory.

V. Experimental Validation

A. JT-DS Learning Evaluation

The learning algorithm presented in the previous section, (10), was implemented in MATLAB. In order to solve the semidefinite convex optimization problem, the YALMIP framework [36] 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:
https://github.com/epfl-lasa/JT-DS-Learning

1. Introduce Datasets and metrics

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

[5]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. [33], [34]

TABLE I: Each simulated trajectory is initialized at $q = [0\ \ldots\ 0]^T$. 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 II: 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 (4) 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 Gaussian, i.e. uniform behaviors. The results, summarized in Table I, 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 (4) is directly transmitted to the robot, the tracking error is zero. Nevertheless, the convergence time of (4) 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 II) comparing the tracking capabilities of our JT-DS method with those of the Cartesian-space SEDS [14] 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 [14]. 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 (4). 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 system provably obeys the manipulator's kinematic joint limits, is fast to compute, and smoothly handles 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 behavior regions, computing the scheduling parameters $\theta_k(\cdot)$ becomes numerically infeasible (all the Gaussians in (9) $\to 0$), and so $\sum_{k=1}^{K} \frac{A_k}{K}$ is used to move the robot, which is still guaranteed

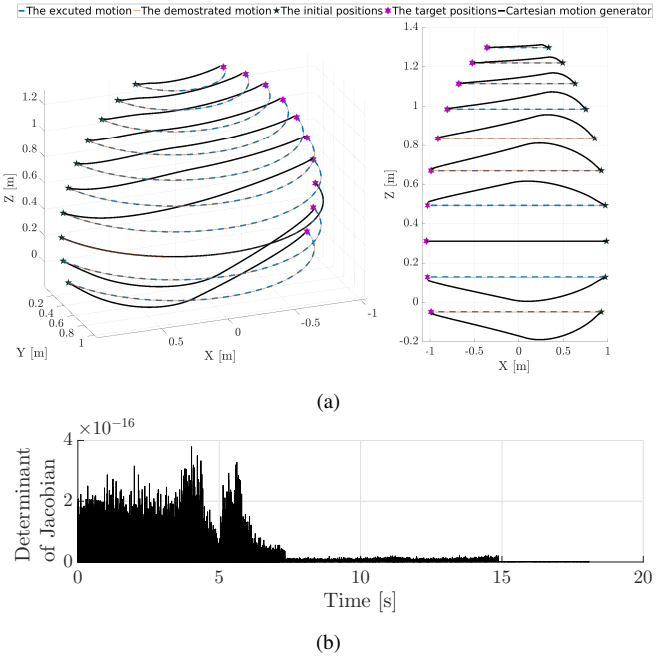


Fig. 3: The experiment generated $K = 3$ Gaussian components. As the first joint is the only joint which was not fixed during the demonstration, the learned augmentation matrices had only one nonzero entry $A_k(1,1) \neq 0 \; \forall k \in \{1, 2, 3\}$. In (a), the end-effector positions for the demonstrations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

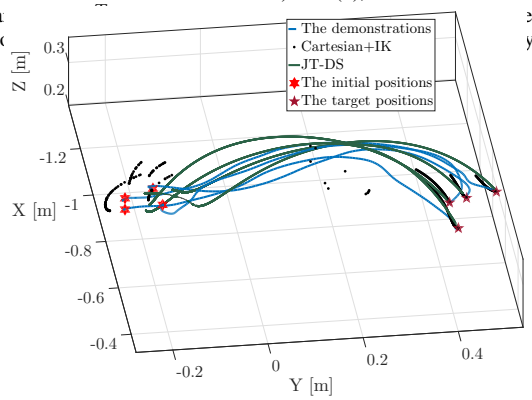


Fig. 4: A plot of the end-effector trajectories for the footstep motion in Cartesian space. The JT-DS motion moves smoothly closely resembles the demonstrated trajectories. On the other hand, the SEDS-based Cartesian motion generator (whose values here are simulated because they can not physically executable)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

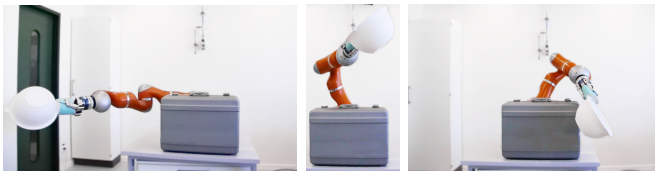
to move towards the target. Moreover, in overlapping local regions where multiple A_k's might be in conflict, the presented system compromises between them while still stably moving to the target. The reason for this is that rather than "determining" the velocity of the system, our A matrix only warps it. This means that adding multiple "conflicting" behaviors 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.

The JTDS algorithm can also be expanded to accommodate joint velocity limits (in addition to joint position limits) by scaling the controller's velocity down such that the highest-velocity component is still within the velocity limits. Specifically, given some set of velocity constraints for each joint i s.t. $\dot{q}_i < \dot{q}_{i,max}$, we can design a new joint augmentation matrix $\mathcal{A}' = r\mathcal{A}$ where $r = \min\left(1, \max_i\left(\frac{\dot{q}_{i,max}}{\dot{q}_i}\right)\right)$. This velocity scalar would slow down the joint trajectory when the robot dynamics could not execute them, but would not change the position profile of the joint motion, preserving a major part of the joint behavior.

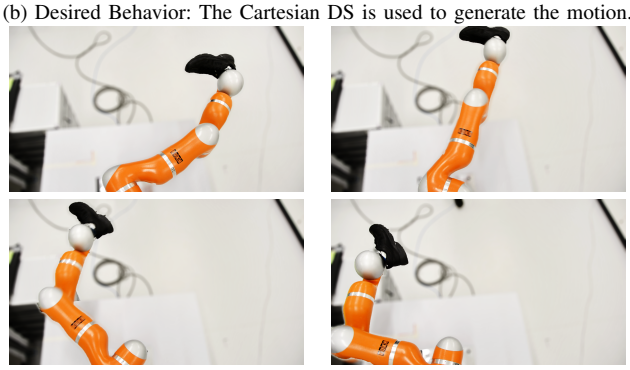
A drawback of the approach is that the controller is not guaranteed to converge to the task-space target x^* even if a path to a valid configuration q^* exists. As mentioned in Sec. III, the controller will always minimize (or keep even) the warped distance to goal $(H(q) - x^*)^T P(H(q) - x^*)$, and 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.

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 [25], [37]. 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.



(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.

Finally, we are currently working on improving the performance of this method (4) 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,.

APPENDIX I
PROVING STABILITY OF THE DYNAMICAL SYSTEM

We wish to prove Proposition 1, that is, that JT-DS (4) (reproduced below)

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

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

Theorem 1.1 (Proof of Lyapunov Stability): JT-DS (4) is stable in the sense of Lyapunov with respect to the Lyapunov candidate

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

That is,

$$0 \prec V(q) \quad V(q^*) = 0 \quad \forall q \neq q^* \tag{11}$$

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 $\frac{dV}{dt}$:

$$\begin{aligned}\frac{dV(q)}{dt} &= (H(q) - x^*)^T P J(q)\dot{q} \\ &= -(H(q) - x^*)^T P J(q)\mathcal{A}(q)J^T(q)P(H(q) - x^*) \\ &= -(H(q) - x^*)^T P J(q)S(q)\sum_{k=1}^{K}\underbrace{\theta_k(q)}_{0\leq}\underbrace{A_k}_{0\preceq} \\ &\quad \cdot S^T(q)J^T(q)P(H(q) - x^*) \leq 0\end{aligned} \tag{12}$$

By observation, each and every term in the final expression is multiplied by its transpose (creating a square) except for A, which is positive semi-definite. This means that the expression is guaranteed to be negative semi-definite. Therefore, JT-DS (4) is globally Lyapunov stable; i.e. $\|H(q) - x^*\|$ and $\dot{q}$ are bounded.

To prove the fulfillment of criterion (II), assume the contrapositive, that there exists some time $0 < \tau$ such that $q^i_{max} < q^i(\tau)$. Because $\dot{q}^i$ given by (4) with respect to (7a) and (7b) is both bounded and continuous, we can invoke the intermediate value theorem to show that there must have been some $0 < c < \tau$ s.t. $q^i(c) = q^i_{max}$. According to (7b), $s^i(q^i(c)) = 0$, which ensures that $\dot{q}^i(c) = f(q^i(c)) = 0$ and will be forever (by induction), which contradicts the assumption. The same proof holds for a $q^i(\tau) < q^i_{min}$. ■

APPENDIX II
ENFORCING JOINT LIMITS

We wish to find a joint-position-dependent diagonal matrix $S(q)$ which is bounded and satisfies:

$$\begin{cases} s^i(q^i_{max}) = 0 \\ s^i(q^i_{min}) = 0 \end{cases} \qquad \forall i \in 1, \ldots, m \tag{13}$$

as previously described in Section III. We propose the using the following functions:

$$s^i(q) = 1 - \left(2\frac{q^i - q^i_{min}}{q^i_{max} - q^i_{min}} - 1\right)^{2d} \quad \forall i \in 1, \ldots, m \tag{14}$$

where d is a positive integer. This function is continuous and satisfies all of the above criteria. We can understand it holistically as follows: the ratio $\frac{q(v) - q_{min}}{q_{max} - q_{min}}$ maps the set of valid q values to the range between 0 and 1. The rest maps these values to a parabola-like polynomial, such that at $q^i = q^i_{max}$ or q^i_{min} the value is 0, and elsewhere the value is positive (with the maximum value, $s^i = 1$, at $q^i = \frac{q_{max} - q_{min}}{2}$). Note as d gets bigger, $S(q)$ is flatter away from the joint limits and drops off more sharply near them, which allows us to preserve a wider range of motion. We found $d = 2$ to be an effective choice.

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] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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

[14] 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.

[15] 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.

[16] 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.

[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] M. Hersch and A. Billard, "Reaching with concurrent dynamical systems," *Autonomous robots*, 2008.
[24] 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.
[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] 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] 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.
[28] I. Jolliffe, *Principal Component Analysis*. Springer Verlag, 1986.
[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. 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.
[32] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
[33] E. Nadaraya, "On estimating regression," *Theory of Prob. and Appl.*, vol. 9, pp. 141–142, 1964.
[34] G. S. Watson, "Smooth regression analysis," *Sankhyā Ser.*, vol. 26, pp. 359–372, 1964.
[35] H. G. Sung, "Gaussian mixture regression and classification," Ph.D. dissertation, Rice University, Houston, Texas, 2004.
[36] 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.
[37] 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.