Autonomous Robots (2019) 43:897–912
https://doi.org/10.1007/s10514-018-9758-x

# Learning position and orientation dynamics from demonstrations via contraction analysis

**Harish chaandar Ravichandar**[1] · **Ashwin Dani**[2]

Received: 1 January 2017 / Accepted: 18 April 2018 / Published online: 2 May 2018


**Abstract**
This paper presents a unified framework of model-learning algorithms, called contracting dynamical system primitives (CDSP), that can be used to learn pose (i.e., position and orientation) dynamics of point-to-point motions from demonstrations. The position and the orientation (represented using quaternions) trajectories are modeled as two separate autonomous nonlinear dynamical systems. The special constraints of the $\mathbb{S}^3$ manifold are enforced in the formulation of the system that models the orientation dynamics. To capture the variability in the demonstrations, the dynamical systems are estimated using Gaussian mixture models (GMMs). The parameters of the GMMs are learned subject to the constraints derived using partial contraction analysis. The learned models' reproductions are shown to accurately reproduce the demonstrations and are guaranteed to converge to the desired goal location. Experimental results illustrate the CDSP algorithm's ability to accurately learn position and orientation dynamics and the utility of the learned models in path generation for a Baxter robot arm. The CDSP algorithm is evaluated on a publicly available dataset and a synthetic dataset, and is shown to have the lowest and comparable average reproduction errors when compared to state-of-the-art imitation learning algorithms.

**Keywords** Learning from demonstration · Gaussian mixture models · Contraction analysis · Model learning

## 1 Introduction

Learning from demonstration (LfD) (Wang et al. 2014; Billard and Matarić 2001; Khansari-Zadeh and Billard 2011; Gribovskaya et al. 2010; Schaal 1999; Ijspeert et al. 2002) is a paradigm for training robots using demonstrations of a task. For developing a robot assistant, the robot should be given an ability to learn motion plans from the task demonstrations shown by the user. For example, in a manufacturing context, a non-expert programmer should be able to program the robot by demonstrating the task to the robot (Rossano et al. 2013; Ravichandar and Dani 2015; Ravichandar et al. 2016, 2017; Koenig and Matarić 2016), or in the case of robots assisting the elderly, the user should be able to teach the robots various tasks without significant effort (Ravichandar and Dani 2016). In many tasks, such as carrying water in a bottle and pouring it into a cup, and part assembly in manufacturing, learning in both position and orientation space is necessary. Successful reproduction of such tasks require accurately reproducing end-effector pose (i.e., position and orientation) paths and achieving a desired end-effector goal pose. In this paper, a method called contracting dynamical system primitive (CDSP) is developed to learn position and the orientation dynamics of complex end-effector paths from demonstrations.

In this paper, the position trajectories are modeled using an autonomous dynamical system (DS) $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t))$, where $\boldsymbol{x}(t)$ denotes the position at time $t$, and $f(\cdot)$ is estimated using a Gaussian mixture model (GMM). The problem of learning position dynamics is formulated as a parameter learning problem with constraints derived from partial contraction analysis of nonlinear systems (Lohmiller and Slotine 1998; Wang and Slotine 2005). Details of contraction and

**Electronic supplementary material** The online version of this article (https://doi.org/10.1007/s10514-018-9758-x) contains supplementary material, which is available to authorized users.

✉ Ashwin Dani
ashwin.dani@uconn.edu

Harish chaandar Ravichandar
harish.ravichandar@uconn.edu

[1] Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269, USA

[2] Department of Electrical and Computer Engineering Management and Engineering for Manufacturing Program, University of Connecticut, Storrs, CT 06269, USA

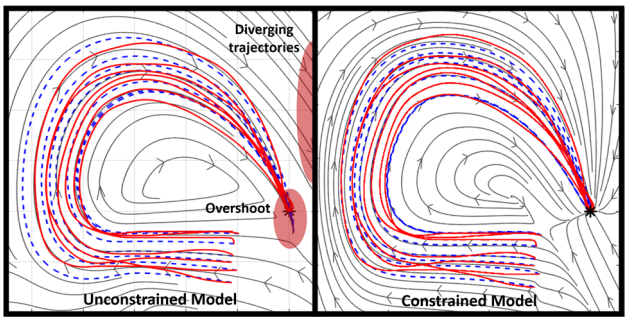


**Fig. 1** Left: Model learned without constraints could result in target overshoot and diverging trajectories. Right: Model learned under constraints derived based on partial contraction analysis provides guaranteed convergence to the goal location. For both models, demonstrations (solid red), reproductions (dashed blue), streamlines (solid gray), and the desired goal location (asterisk) are shown (Color figure online)

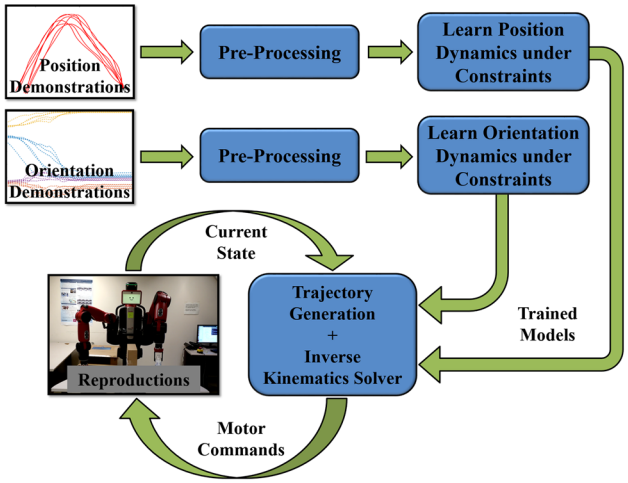


**Fig. 2** Block diagram representation of the CDSP algorithm for learning position and orientation dynamics of point-to-point motions from demonstrations

partial contraction analyses are provided in Sect. 2. Due to the constraints developed using partial contraction analysis, the CDSP algorithm is able to accurately reproduce the demonstrations and guarantee that the trajectories generated by the learned model converge to the goal location from any initial condition (see an example in Fig. 1). As shown in Fig. 1, learning motion models without constraints might result in target overshoots, and diverging trajectories. The CDSP algorithm can accurately model a wide class of motions including complex shapes, such as the *SharpC*, *L-shape*, *J-shape* and *Snake* from the LASA dataset (Khansari-Zadeh and Billard 2011).

For learning orientation dynamics quaternion parametrization is used. To guarantee that the generated quaternion trajectory retains its membership to $\mathbb{S}^3$, the angular velocity dynamics are learned from the demonstrations. The angular velocity trajectory generated from the learned model is used to recreate the quaternion trajectory based on standard quaternion dynamics: $\dot{\boldsymbol{q}} = \frac{1}{2}\Omega(\boldsymbol{\omega})\boldsymbol{q}$, where $\boldsymbol{q} \in \mathbb{S}^3$ is the quaternion orientation, $\boldsymbol{\Omega}(\boldsymbol{\omega}(t)) = \begin{bmatrix} -[\boldsymbol{\omega}(t)]_\times & \boldsymbol{\omega}(t) \\ -\boldsymbol{\omega}(t)^T & 0 \end{bmatrix}$, and $[\boldsymbol{\omega}(t)]_\times \in \mathbb{R}^{3\times 3}$ is a skew-symmetric matrix formed with the angular velocity vector $\boldsymbol{\omega}(t)$ at time $t$. The system modeling the angular velocity dynamics is a function of the current angular velocity, the current quaternion, and the goal quaternion. Similar to the learning algorithm for position dynamics, the parameter learning algorithm for orientation dynamics is subject to constraints derived using partial contraction analysis.

Constrained optimization problems are formulated to learn the parameters of the GMMs used to estimate the motion dynamics. The cost is chosen to be the mean squared error between the demonstrations and the reproductions generated by the trained models. Partial contraction analysis yields constraints that are matrix inequality conditions on the model parameters. The constrained optimization problems are solved using the Sequential Quadratic Programming (SQP) algorithm. Initial estimates of the parameters of the GMM are obtained using the expectation–maximization (E–M) algorithm (Dempster et al. 1977). Detailed experimental evaluations of the CDSP algorithm, illustrating its ability to accurately learn both position and orientation dynamics, are provided in Sect. 5. In Fig. 2, a block diagram describing the overall workflow of the CDSP algorithm is shown.

The contributions of this work are summarized below:

1. To the best of our knowledge, this is the first method that can learn a dynamical system in state-space from demonstrations subject to convergence constraints derived by using partial contraction analysis. The dynamical system is approximated by using a GMM and partial contraction theory is used to learn the GMM parameters, such that the reproductions, both, closely resemble the demonstrations, and are guaranteed to converge to the desired goal location from any initial condition.
2. The partial contraction-based model learning algorithm is derived to learn orientation dynamics. The orientation is parametrized using the quaternion representation. The learned model not only generates quaternion trajectories that faithfully reproduce the demonstrations, but also guarantee the retention of their membership to the $\mathbb{S}^3$ manifold.

Compared to our earlier work, presented in Ravichandar and Dani (2015) and Ravichandar et al. (2016), the current paper presents (1) a constrained learning algorithm that can learn a wider range of point-to-point motions, (2) a unified framework to learn pose (i.e., position and orientation) dynamics, (3) thorough experimental evaluations and comparisons with the state-of-the-art algorithms including statistical analyses, and (4) demonstration of three different

tasks on a Baxter robot using models learned from demonstrations collected from the robot.

### 1.1 Related work

LfD is a widely used approach for imitation learning, in which a new task or skill is learned from demonstrations provided by humans [see Argall et al. (2009) for a comprehensive review]. One way to accomplish this is to directly learn the control policy or the dynamics involved in the task of interest from the available demonstrations (Ijspeert et al. 2013; Khansari-Zadeh and Billard 2011; Calinon et al. 2014; Lemme et al. 2013; Saunders et al. 2006; Jenkins et al. 2000). Methods that use such a direct approach can further be categorized into: (a) methods that classify the current state of the robot into any one of the finite number of available control actions (Saunders et al. 2006; Jenkins et al. 2000; Langsfeld et al. 2014), and (b) methods that use regression to either approximate the mapping between the state space and the control action space or learn dynamical systems that represent task primitives (Ijspeert et al. 2013; Khansari-Zadeh and Billard 2011; Calinon et al. 2014; Lemme et al. 2013; Ravichandar and Dani 2015; Ravichandar et al. 2016).

Another popular technique for LfD is reinforcement learning (RL) (Sutton and Barto 1998; Peters and Schaal 2008; Kober et al. 2013). Methods based on RL maximize a predefined reward function to obtain the optimal control policy. A particular variant of the standard RL approach, in which the reward function is learned from the demonstrations, is called inverse optimal control (IOC) or inverse reinforcement learning (IRL) (Todorov and Jordan 2002; Abbeel and Ng 2004; Laumond et al. 2014; Priess 2014; Laumond et al. 2015). RL-based approaches typically require the system to explore the state space, which can be difficult for many applications. Methods that directly learn the control policy or dynamics from demonstrations are more practical in such settings. In Schaal et al. (2007), a comparison between dynamical system-based methods and optimal control methods is provided. Other categories of LfD methods include sampling-based methods (e.g., Bowen and Alterovitz 2014), optimization-based approaches (e.g., Dragan et al. 2015; Zucker et al. 2013), geometric methods (e.g., Ahmadzadeh et al. 2017), and key-frame-based methods (e.g., Akgun et al. 2012).

The algorithm proposed in this paper falls under the category of dynamical system-based LfD. Dynamical system (DS)-based LfD methods have received a lot of attention in the recent past. Methods using time-invariant stable dynamical systems, including CDSP, can instantaneously adapt to sudden spatial perturbations and changes in goal location during path generation without the need for re-planning. Furthermore, a single dynamical system can encode motions that converge to a single goal location, but represent different dynamics in different regions of the state-space. One of the first DS-based frameworks, called the dynamic movement primitives (DMPs) (Schaal 1999; Ijspeert et al. 2002; Kalakrishnan et al. 2012; Ijspeert et al. 2013; Rai 2014), uses a stable DS containing a linear proportional derivative (PD)-like term coupled with a nonlinear term for encoding a desired trajectory. These two terms are coupled through a so-called phase variable. Tasks, such as drumming (Ude et al. 2010), pouring (Nemec et al. 2009), and pick and place have been recreated using DMPs on various robotic platforms. The DMP formulation, however, models multi-dimensional systems by learning one DS for each dimension separately, thereby neglecting the combined effect of all the dimensions. The idea of movement primitives is extended to probabilistic framework in Paraschos (2013). In Calinon et al. (2014)), task-parametrized GMMs (TP-GMM) are used to design control policies for motion generation through generalization of available demonstration. However, the algorithm in Calinon et al. (2014) does not provide any stability guarantees for the learned DS. An algorithm to encode motion dynamics by using neurally imprinted vector fields (NiVF) is presented in Lemme et al. (2013). The stability of the learned model is verified using constraints derived through Lyapunov analysis. The NiVF algorithm restricts the neural network (NN) learning process to an approach called extreme learning machines (ELM). While NiVF is shown to be capable of learning a variety of motions, the stability property is restricted to finite regions of the state space.

An LfD method, called stable estimator of dynamical systems (SEDS), that learns globally stable DSs directly in higher dimensional state-space using GMMs is developed in Khansari-Zadeh and Billard (2010), Gribovskaya et al. (2010) and Khansari-Zadeh and Billard (2011). To keep the attractor properties of the synthesized DS, Lyapunov stability conditions are used to learn the parameters of GMMs, ensuring asymptotic stability of the goal location. An important limitation of SEDS is that a quadratic Lyapunov function ($V = x^T x$) is used in the development of stability constraints. Thus, SEDS can only model trajectories whose 2-norm distances to the target decrease monotonically in time. In Khansari-Zadeh and Billard (2014), an approach called Control Lyapunov Function-based Dynamic Movements (CLF-DM) is introduced. This approach parametrizes the energy function using either a weighted sum of asymmetric quadratic functions (WSAQF) or a neurally imprinted Lyapunov candidate (NILC). The parameters of the WSAQF or the NILC are learned from the demonstrations. The learned energy function is used during run time to ensure global stability of the generated trajectories. While it can encode a wide variety of motions, CLF-DM suffers from the disadvantage of the online correction signal potentially interfering with the learned DS. More specifically, as pointed out in Neumann and Steil (2015), CLF-DM has the disadvantage of solving

a separate optimization problem for parameter selection of the control Lyapunov function, which can lead to numerical stability issues in parameter selection. In Neumann and Steil (2015), an algorithm called $\tau$-SEDS is introduced, which uses diffeomorphic transformations along with the SEDS algorithm to generalize the class of motions that can be learned. The diffeomorphic transformation is used to transform the task space such that the transformed demonstrations are consistent with a quadratic Lyapunov function, and hence, SEDS can be used to learn the dynamics in the transformed space. The learned system is then back-transformed to the original task space.

In contrast, the CDSP algorithm is capable of encoding a wider class of motions whose 2-norm distances to the target do not necessarily decrease monotonically in time. This is achieved by enforcing novel partial contraction analysis-based constraints (Wang and Slotine 2005). The enforcement of these partial contraction constraints guarantees that the trajectories of the learned system converge to a specific trajectory, which is chosen as the equilibrium point in this paper. It is seen that the enforcement of the partial contraction constraints improves the learning accuracy of complex shapes, such as *SharpC*, *L-shape*, *J-shape,* and *Snake* from the LASA dataset (Khansari-Zadeh and Billard 2011).

While the SEDS, NILC, $\tau$-SEDS, and CLF-DM algorithms can learn a wide variety of position dynamics, they have not been demonstrated and tested for learning orientation dynamics. In contrast, a unified framework is presented in our paper to encode pose (i.e., position and orientation) dynamics. Indeed, encoding several tasks, such as wire insertion, parts assembly, complex object manipulation, and pouring liquids, requires both position and orientation dynamics of the end-effector in the task-space to be learned from demonstrations. Furthermore, a thorough evaluation and comparison of the CDSP algorithm with SEDS, CLF-DM, NiVF, and $\tau$-SEDS algorithms are presented in Sect. 5.1. The results indicate that the CDSP algorithm shows a statistically significantly better performance on the LASA handwriting dataset in terms of shape reproduction accuracy when compared to rest of the algorithms evaluated on the dataset.

State-of-the-art algorithms that encode orientation dynamics include Silvério et al. (2015), Pastor et al. (2009), and Ude et al. (2014). The algorithm presented in Silvério et al. (2015) learns the pose dynamics involved in bi-manual manipulation tasks in a task-parametrized manner. In Pastor et al. (2009), the DMP framework is extended to control the gripper orientation and finger position by encoding the dynamics of each degree-of-freedom using a separate DMP. While the algorithms in Silvério et al. (2015), and Pastor et al. (2009), are shown to be capable of encoding complex pose dynamics in the task-space, they do not take into account the special constraints of the $\mathbb{S}^3$ manifold in the learned model, and thus, require post-reproduction normalization steps. Similar to the CDSP algorithm, a modified DMP formulation, introduced in Ude et al. (2014), addresses this drawback by taking the constraints of the $\mathbb{S}^3$ manifold into consideration. A disadvantage of the framework in Ude et al. (2014) is that the parameters of the DMP have to be carefully tuned for each task to be learned. In contrast, the CDSP framework learns its parameters directly from the demonstrations. Experimental evaluations of the CDSP algorithm and its comparisons with the orientation-DMP (ODMP) (Ude et al. 2014), SEDS (Khansari-Zadeh and Billard 2011), and CLF-DM (Khansari-Zadeh and Billard 2014) algorithms, in terms of learning orientation dynamics, are presented in Sect. 5.2. The comparison results indicate that the CDSP algorithm results in the lowest average orientation reproduction error.

## 2 Preliminaries

### 2.1 Brief review of contraction analysis

In this section, contraction analysis (Lohmiller and Slotine 1998) for analyzing exponential stability of nonlinear systems is briefly reviewed. Consider a nonlinear, autonomous system of the form

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)) \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^d$ is a state vector and $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a continuously differentiable nonlinear function. With the assumed properties of (1), the exact relation $\delta\dot{\boldsymbol{x}} = \frac{\partial f(\boldsymbol{x}(t))}{\partial \boldsymbol{x}}\delta\boldsymbol{x}$ holds, where $\delta\boldsymbol{x}$ is an infinitesimal virtual displacement in fixed time. The squared virtual displacement between two trajectories of (1) in a symmetric, uniformly positive definite contraction metric $\boldsymbol{M}(\boldsymbol{x}) \in \mathbb{R}^{d\times d}$ is given by $\delta\boldsymbol{x}^T \boldsymbol{M}(\boldsymbol{x}(t))\,\delta\boldsymbol{x}$ and its time derivative by

$$\frac{d}{dt}\left(\delta\boldsymbol{x}^T \boldsymbol{M}(\boldsymbol{x}(t))\,\delta\boldsymbol{x}\right) = \delta\boldsymbol{x}^T \left(\frac{\partial f}{\partial \boldsymbol{x}}^T \boldsymbol{M}(\boldsymbol{x}(t)) + \dot{\boldsymbol{M}}(\boldsymbol{x}(t)) + \boldsymbol{M}(\boldsymbol{x}(t))\frac{\partial f}{\partial \boldsymbol{x}}\right)\delta\boldsymbol{x}. \tag{2}$$

**Definition 1** Given the autonomous DS $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t))$ and a contraction metric $\boldsymbol{M}(\boldsymbol{x}(t))$, that is a uniformly positive definite symmetric matrix, if the system satisfies the condition, $\frac{\partial f}{\partial \boldsymbol{x}}^T \boldsymbol{M}(\boldsymbol{x}(t)) + \dot{\boldsymbol{M}}(\boldsymbol{x}(t)) + \boldsymbol{M}(\boldsymbol{x}(t))\frac{\partial f}{\partial \boldsymbol{x}} \leq -\gamma_c \boldsymbol{M}(\boldsymbol{x}(t))$, $\forall \boldsymbol{x}$ for a strictly positive constant $\gamma_c$, then the system is said to be globally contracting with respect to $\boldsymbol{x}$ (Lohmiller and Slotine 1998).

**Theorem 1** *If the autonomous system $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t))$ is globally contracting with respect to $\boldsymbol{x}$, then all of its trajectories converge to each other exponentially.*

***Proof*** See Lohmiller and Slotine (1998, Theorem 2). □

### 2.2 Brief review of partial contraction analysis

Consider the nonlinear autonomous system[1]

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{x}(t)) \tag{3}$$

and an auxiliary system of the form

$$\dot{\boldsymbol{y}}(t) = f(\boldsymbol{x}(t), \boldsymbol{y}(t)) \tag{4}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^d$ and $\boldsymbol{y}(t) \in \mathbb{R}^d$ are the state vectors and $f : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a continuously differentiable nonlinear function. Note that for $\boldsymbol{y}(t) = \boldsymbol{x}(t)$, the auxiliary system in (4) reduces to the system in (3). Hence, $\boldsymbol{y}(t) = \boldsymbol{x}(t)$ is a particular solution of the auxiliary system in (4).

**Definition 2** If the auxiliary $\boldsymbol{y}$-system in (4) is contracting with respect to $\boldsymbol{y}$ according to Definition 1 (i.e., $\frac{\partial f}{\partial \boldsymbol{y}}^T \boldsymbol{M}(\boldsymbol{y}(t)) + \dot{\boldsymbol{M}}(\boldsymbol{y}(t)) + \boldsymbol{M}(\boldsymbol{y}(t)) \frac{\partial f}{\partial \boldsymbol{y}} \leq -\gamma_c \boldsymbol{M}(\boldsymbol{y}(t))$, $\forall \boldsymbol{y}$), the original $\boldsymbol{x}$-system in (3) is said to be partially contracting (Dani et al. 2015; Wang and Slotine 2005).

**Theorem 2** *If the auxiliary $\boldsymbol{y}$-system is contracting with respect to $\boldsymbol{y}$ and any of its particular solutions verifies a smooth specific property (for instance, convergence to an equilibrium point), then all the trajectories of the partially contracting original $\boldsymbol{x}$-system verify this property exponentially* (Wang and Slotine 2005, Theorem 1).

***Proof*** See Wang and Slotine (2005, Theorem 1). □

***Remark 1*** A smooth specific property of a trajectory may denote a trajectory converging to an equilibrium point or a manifold.

### 2.3 Brief review of quaternion parametrization on $\mathbb{S}^3$

In order to learn orientation dynamics, it is necessary to parametrize the SO(3) manifold. While there are many parametrizations, such as Euler angles and angle-axis, they suffer from the well-known disadvantage of singularity (Corke 2011). Quaternion parametrization, on the other hand, is a viable option that is non-minimal and singularity-free. Following the notation used in Ude et al. (2014), let the unit quaternion $\boldsymbol{q} \in \mathbb{S}^3$, where $\mathbb{S}^3$ is the unit hypershpere of $\mathbb{R}^4$, be given by

$$\boldsymbol{q} \triangleq \begin{bmatrix} v \\ \boldsymbol{u} \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\phi}{2}\right) \\ \sin\left(\frac{\phi}{2}\right)\boldsymbol{n} \end{bmatrix} \tag{5}$$

[1] Following the notation in partial contraction analysis literature (Wang and Slotine 2005), $\boldsymbol{x}(t)$ is written twice to represent the dependency of $\boldsymbol{x}(t)$ in multiple places in $f(\cdot)$.

where $v \in \mathbb{R}$, $\boldsymbol{u} \in \mathbb{R}^3$, $\phi$ and $\boldsymbol{n}$ are the angle and normalized axis of rotation in the angle-axis representation. Given a quaternion $\boldsymbol{q} = \left[v, \ \boldsymbol{u}^T\right]^T$, a conjugate quaternion is defined as $\bar{\boldsymbol{q}}(t) = \left[v, -\boldsymbol{u}^T\right]^T$. The product of two quaternions, $\boldsymbol{q}_1 * \boldsymbol{q}_2$, is seen as the coordinate frame whose orientation is described by $\boldsymbol{q}_2$ undergoing a rotation described by $\boldsymbol{q}_1$. The quaternion product is defined as

$$\boldsymbol{q}_1 * \boldsymbol{q}_2 \triangleq \begin{bmatrix} v_1 + v_2 - \boldsymbol{u}_1^T \boldsymbol{u}_2 \\ v_2\boldsymbol{u}_2 + v_2\boldsymbol{u}_1 + \boldsymbol{u}_1 \times \boldsymbol{u}_2 \end{bmatrix}. \tag{6}$$

### 2.4 Brief review of dynamics represented using Gaussian mixture models

Approximating a continuously differentiable function $f(\cdot)$, in an autonomous DS $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, using a GMM involves approximating the joint density of $\boldsymbol{x} \in \mathbb{R}^d$ and $\dot{\boldsymbol{x}} \in \mathbb{R}^d$ using a finite mixture of Gaussian functions. The parameters of the $k$th Gaussian of a GMM will include the prior $\pi^k \in [0, 1]$; the mean $\boldsymbol{\mu}^k = \left[\left(\boldsymbol{\mu}_{\boldsymbol{x}}^k\right)^T, \left(\boldsymbol{\mu}_{\dot{\boldsymbol{x}}}^k\right)^T\right]^T \in \mathbb{R}^{2d}$ where $\boldsymbol{\mu}_{\boldsymbol{x}}^k \in \mathbb{R}^d$ and $\boldsymbol{\mu}_{\dot{\boldsymbol{x}}}^k \in \mathbb{R}^d$ are the mean vectors associated with $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$, respectively; and the covariance $\boldsymbol{\Sigma}^k = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{x}}^k & \boldsymbol{\Sigma}_{\boldsymbol{x}\dot{\boldsymbol{x}}}^k \\ \boldsymbol{\Sigma}_{\dot{\boldsymbol{x}}\boldsymbol{x}}^k & \boldsymbol{\Sigma}_{\dot{\boldsymbol{x}}}^k \end{bmatrix} \in \mathbb{R}^{2d \times 2d}$ where $\boldsymbol{\Sigma}_{\boldsymbol{x}}^k \in \mathbb{R}^{d \times d}$ and $\boldsymbol{\Sigma}_{\boldsymbol{x}\dot{\boldsymbol{x}}}^k \in \mathbb{R}^{d \times d}$ are the covariance matrices associated with $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$, respectively, and $\boldsymbol{\Sigma}_{\boldsymbol{x}\dot{\boldsymbol{x}}}^k \in \mathbb{R}^{d \times d}$ is the cross-covariance matrix between $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$. Given a set of $N$ demonstrations, denoted by $\{\boldsymbol{x}_n(t), \dot{\boldsymbol{x}}_n(t)\}_{t=0}^{t=T_n}$, $\forall n = 1, 2, .., N$, each pair $\{\boldsymbol{x}_n(t), \dot{\boldsymbol{x}}_n(t)\}$ is assumed to be sampled from the following density function

$$\mathcal{P}(\boldsymbol{x}_n(t), \dot{\boldsymbol{x}}_n(t); \boldsymbol{\theta}) = \sum_{k=1}^{K} \mathcal{P}(k)\, \mathcal{P}(\boldsymbol{x}_n(t), \dot{\boldsymbol{x}}_n(t) | k)$$

for $t \in \{0, .., T_n\}$ and $n = 1, .., N$, where $K$ is the number of Gaussian functions, $\boldsymbol{\theta} = \{\boldsymbol{\mu}^1 \ldots \boldsymbol{\mu}^K, \boldsymbol{\Sigma}^1 \ldots \Sigma^K, \pi^1 \ldots \pi^K\}$ is the set of parameters of the GMM, $\mathcal{P}(k) = \pi^k$ denotes the prior and $\mathcal{P}(\boldsymbol{x}_n(t), \dot{\boldsymbol{x}}_n(t) | k)$ is the conditional joint density of $\boldsymbol{x}_n(t)$ and $\dot{\boldsymbol{x}}_n(t)$ given by

$$\mathcal{P}(\boldsymbol{x}_n(t), \dot{\boldsymbol{x}}_n(t) | k) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x}_n(t) \\ \dot{\boldsymbol{x}}_n(t) \end{bmatrix}; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k\right)$$

where $\mathcal{N}(\cdot)$ denotes the Gaussian function. Now, the estimate of the state derivative, $\dot{\boldsymbol{x}}$, is given by expected value of the posterior density $\mathcal{P}(\dot{\boldsymbol{x}}_n(t) | \boldsymbol{x}_n(t))$ as follows (Cohn et al. 1996; Khansari-Zadeh and Billard 2011)

$$\dot{\boldsymbol{x}} = \sum_{k=1}^{K} \frac{\mathcal{P}(k)\,\mathcal{P}(\boldsymbol{x}|k)}{\sum_{i=1}^{K}\mathcal{P}(i)\,\mathcal{P}(\boldsymbol{x}|i)} \Big( \boldsymbol{\mu}_{\dot{\boldsymbol{x}}}^{k} + \boldsymbol{\Sigma}_{\dot{\boldsymbol{x}}\boldsymbol{x}}^{k} \left(\boldsymbol{\Sigma}_{\boldsymbol{x}}^{k}\right)^{-1} \left(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}}^{k}\right) \Big).$$

## 3 Learning position dynamics from demonstrations

Consider a state variable $\boldsymbol{x}(t) \in \mathbb{R}^d$ at time $t$ that represents the position of a point in $d$-dimensions. Let a set of $N_p$ demonstrations of a point-to-point motion be solutions to the following DS

$$\dot{\boldsymbol{x}}(t) = f_p(\boldsymbol{x}(t), \boldsymbol{x}(t)) \tag{7}$$

where $f_p : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ is a nonlinear continuously differentiable autonomous function that models the position dynamics. Each demonstration corresponds to a reaching motion ending at $\boldsymbol{x}^* = \boldsymbol{g}_p$, where $\boldsymbol{g}_p \in \mathbb{R}^d$ is the goal location. The $n$th demonstration consists of the trajectories of the states $\{\boldsymbol{x}_n(t)\}_{t=0}^{t=T_n}$, and the trajectories of the state derivatives $\{\dot{\boldsymbol{x}}_n(t)\}_{t=0}^{t=T_n}$.

***Remark 2*** In the case of point-to-point motions, the position trajectories start from various initial locations and end at the final goal location. If the recorder trajectories do not end precisely at the goal location, they can be translated. Additionally, the velocity and acceleration are zero at the goal location.

### 3.1 Encoding position dynamics using contracting GMMs

Similar to Khansari-Zadeh and Billard (2014), the nonlinear function $f_p(\cdot)$ defined in (7) is modeled using a GMM and the resulting autonomous DS is given by[2]

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \sum_{k=1}^{K_p} h_p^k(\boldsymbol{x}(t)) \left(\boldsymbol{A}_p^k \boldsymbol{x}(t) + \boldsymbol{b}_p^k\right) \\ &= f_p(\boldsymbol{x}(t), \boldsymbol{x}(t))\end{aligned} \tag{8}$$

where $h_p^k(\boldsymbol{x}) = \frac{\mathcal{P}(k)\mathcal{P}(\boldsymbol{x}|k)}{\sum_{i=1}^{K_p}\mathcal{P}(i)\mathcal{P}(\boldsymbol{x}|i)}$ is the scalar weight associated with the $k$th Gaussian, such that $0 \le h_p^k(\boldsymbol{x}) \le 1$ and $\sum_{k=1}^{K_p} h_p^k(\boldsymbol{x}) = 1$, $\mathcal{P}(k) = \pi_p^k$ is the prior probability, $\boldsymbol{A}_p^k(x) = \boldsymbol{\Sigma}_{\dot{\boldsymbol{x}}\boldsymbol{x}}^k \left(\boldsymbol{\Sigma}_{\boldsymbol{x}}^k\right)^{-1}$, $\boldsymbol{b}_p^k = \boldsymbol{\mu}_{\dot{\boldsymbol{x}}}^k - \boldsymbol{A}_p^k \boldsymbol{\mu}_{\boldsymbol{x}}^k$, $\boldsymbol{\mu}_p^k = \left[\left(\boldsymbol{\mu}_{\boldsymbol{x}}^k\right)^T, \left(\boldsymbol{\mu}_{\dot{\boldsymbol{x}}}^k\right)^T\right]^T$ and $\boldsymbol{\Sigma}_p^k = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{x}}^k & \boldsymbol{\Sigma}_{\boldsymbol{x}\dot{\boldsymbol{x}}}^k \\ \boldsymbol{\Sigma}_{\dot{\boldsymbol{x}}\boldsymbol{x}}^k & \boldsymbol{\Sigma}_{\dot{\boldsymbol{x}}}^k \end{bmatrix}$ are the mean and the covariance of the $k$th Gaussian, respectively. Given a set of $N_p$ demonstrations, this paper addresses the problem of learning the function $f_p(\cdot)$, which is modeled using a GMM, under constraints derived through partial contraction analysis.

**Theorem 3** *If the following constraints are satisfied for the autonomous nonlinear DS in (8),*

$$\left(\boldsymbol{A}_p^k\right)^T \boldsymbol{M}_p + \boldsymbol{M}_p \boldsymbol{A}_p^k \le -\gamma_p \boldsymbol{M}_p, \quad k = 1, 2, \ldots, K_p, \tag{9}$$

$$\boldsymbol{A}_p^k \boldsymbol{x}^* + \boldsymbol{b}_p^k = 0, \quad k = 1, 2, \ldots, K_p \tag{10}$$

*where $\gamma_p$ is a strictly positive scalar constant and $\boldsymbol{M}_p \in \mathbb{R}^{d\times d}$ represents a constant positive definite symmetric matrix, then all trajectories of the system in (8) converge to the goal location $\boldsymbol{x}^*$.*

***Proof*** Consider $K_p$ systems given by

$$\dot{\boldsymbol{y}}_p(t) = A_p^k \boldsymbol{y}_p(t) + \boldsymbol{b}_p^k, \quad k = 1, 2, \ldots, K_p \tag{11}$$

If (10) is satisfied, then $\boldsymbol{y}_p = \boldsymbol{x}^*$ is the common equilibrium of each $k$th system of (11). Furthermore, based on Theorem 1, if (9) is satisfied then every $k$th system of (11) is contracting in a common contraction metric $\boldsymbol{M}_p$ and all the trajectories of each $k$th system of (11) will converge to the goal location $\boldsymbol{x}^*$. Consider a new auxiliary system which is a convex combination of the systems in (11)

$$\begin{aligned}\dot{\boldsymbol{y}}_p(t) &= f_p\left(\boldsymbol{x}(t), \boldsymbol{y}_p(t)\right) \\ &= \sum_{k=1}^{K_p} h_p^k(\boldsymbol{x}(t)) \left(\boldsymbol{A}_p^k \boldsymbol{y}_p(t) + \boldsymbol{b}_p^k\right)\end{aligned} \tag{12}$$

where $\boldsymbol{y}_p(t) \in \mathbb{R}^d$. If (10) is satisfied, the particular solution $\boldsymbol{y}_p = \boldsymbol{x}^*$ of (12) is also its equilibrium point. On defining the virtual dynamics

$$\delta\dot{\boldsymbol{y}}_p \triangleq \frac{\partial f_p}{\partial \boldsymbol{y}_p} \delta \boldsymbol{y}_p \tag{13}$$

and substituting the Jacobian of the auxiliary system (given by $\frac{\partial f_p}{\partial \boldsymbol{y}_p} = \sum_{k=1}^{K_p} h_p^k(\boldsymbol{x}(t)) \boldsymbol{A}_p^k$), for any constant symmetric positive definite $\boldsymbol{M}_p$

$$\frac{\partial f_p}{\partial \boldsymbol{y}_p}^T \boldsymbol{M}_p + \boldsymbol{M}_p \frac{\partial f_p}{\partial \boldsymbol{y}_p} \triangleq \sum_{k=1}^{K_p} h_p^k(\boldsymbol{x}(t)) \left(\boldsymbol{A}_p^k\right)^T \boldsymbol{M}_p + \boldsymbol{M}_p \sum_{k=1}^{K_p} h_p^k(\boldsymbol{x}(t)) \boldsymbol{A}_p^k$$

[2] In $f_p(\boldsymbol{x}(t), \boldsymbol{x}(t))$, the first argument refers to the $\boldsymbol{x}(t)$ in $h_p(\cdot)$ and the second argument refers to the $\boldsymbol{x}(t)$ in the affine part of $f_p(\cdot)$.

$$= \sum_{k=1}^{K_p} h_p^k(\boldsymbol{x}(t)) \left( \left( \boldsymbol{A}_p^k \right)^T \boldsymbol{M}_p + \boldsymbol{M}_p \boldsymbol{A}_p^k \right) \tag{14}$$

Using (9) and (14), we have $\frac{\partial f_p}{\partial \boldsymbol{y}_p}^T \boldsymbol{M}_p + \boldsymbol{M}_p \frac{\partial f_p}{\partial \boldsymbol{y}_p} \leq -\sum_{k=1}^{K_p} h_p^k(\boldsymbol{x}(t)) \gamma_p \boldsymbol{M}_p, \ \forall \boldsymbol{y}_p(t)$. Since $0 \leq h_p^k(\boldsymbol{x}(t)) \leq 1$ and $\sum_{k=1}^{K_p} h_p^k(\boldsymbol{x}(t)) = 1, \ \forall \boldsymbol{x}(t)$, we have

$$\frac{\partial f_p}{\partial \boldsymbol{y}_p}^T \boldsymbol{M}_p + \boldsymbol{M}_p \frac{\partial f_p}{\partial \boldsymbol{y}_p} \leq -\gamma_p \boldsymbol{M}_p, \ \forall \boldsymbol{y}_p(t) \tag{15}$$

Furthermore, taking the time derivative of $V(\delta \boldsymbol{y}_p) = \delta \boldsymbol{y}_p^T \boldsymbol{M}_p \delta \boldsymbol{y}_p$, and using (15) yields $\dot{V}(\delta \boldsymbol{y}_p) \leq -\gamma_p \delta \boldsymbol{y}_p^T \boldsymbol{M}_p \delta \boldsymbol{y}_p$. Hence, the system (12) is contracting with respect to $\boldsymbol{y}_p$ (recall Definition 1). Further, based on Theorem 1, all the trajectories of (12) will globally exponentially converge towards each other.

Now that the convergence of $\boldsymbol{y}_p(t)$ is shown, the convergence of $\boldsymbol{x}(t)$ to $\boldsymbol{x}^*$ remains to be proven. Since the auxiliary system in (12) is contracting with respect to $\boldsymbol{y}_p(t)$, and the trajectory $\boldsymbol{y}_p(t) = \boldsymbol{x}^*$ [a particular solution of (12)] is an equilibrium point, then according to Theorem 2, the trajectories $\boldsymbol{x}(t)$ of (8) will globally exponentially converge to the goal location $\boldsymbol{x}^*$. □

***Remark 3*** The constraints in (9), derived using partial contraction analysis, are different than those presented in Khansari-Zadeh and Billard (2011). Furthermore, note that the constraints in (9) guarantee global exponential stability of the learned model and the rate of convergence, $\gamma_p$, can be tuned for specific applications.

The constrained optimization problem to be solved in order to train the GMM model with the demonstrations can be written as

$$\{\hat{\boldsymbol{\theta}}_p, \hat{\boldsymbol{M}}_p\} = \arg\min_{\boldsymbol{\theta}_p, \boldsymbol{M}_p} \ J_p(\boldsymbol{\theta}_p) \tag{16}$$

$$\text{s.t.} \left( \boldsymbol{A}_p^k \right)^T \boldsymbol{M}_p + \boldsymbol{M}_p \boldsymbol{A}_p^k + \gamma_p \boldsymbol{M}_p \preceq 0, \quad k = 1, \ldots, K_p, \tag{17}$$

$$\boldsymbol{A}_p^k \boldsymbol{x}^* + \boldsymbol{b}_p^k = 0, \quad k = 1, \ldots, K_p, \tag{18}$$

$$\boldsymbol{M}_p \succ 0 \tag{19}$$

$$\boldsymbol{\Sigma}_p^k \succ 0, \quad k = 1, \ldots, K_p, \tag{20}$$

$$0 \leq \pi_p^k \leq 1, \quad k = 1, \ldots, K_p, \tag{21}$$

$$\sum_k \pi_p^k = 1 \tag{22}$$

where $\boldsymbol{\theta}_p = \{\boldsymbol{\mu}_p^1 \ldots \boldsymbol{\mu}_p^{K_p}, \boldsymbol{\Sigma}_p^1 \ldots \boldsymbol{\Sigma}_p^{K_p}, \pi_p^1 \ldots \pi_p^{K_p}\}$ is a vector containing the parameters of the GMM model. Note that the parameters of the matrix $\boldsymbol{M}_p$ are also learned from the demonstrations as opposed to being manually designed. The constraints (17)–(19) ensure the global attraction of the goal location $\boldsymbol{x}^*$ and the constraints in (20)–(22) are a result of using a GMM to model the dynamics. Similar to Khansari-Zadeh and Billard (2011), the cost function $J_p(\boldsymbol{\theta}_p)$ is chosen to be the mean squared error, given by

$$J_p(\boldsymbol{\theta}_p) = \frac{1}{2\mathcal{T}_p} \sum_{n=1}^{N_p} \sum_{t=0}^{T_n} \| \hat{\dot{\boldsymbol{x}}}_n(t) - \dot{\boldsymbol{x}}_n(t) \|^2 \tag{23}$$

where $\mathcal{T}_p = \sum_{n=1}^{N_p} T_n$ is the total number of data points in the demonstrations, $N_p$ is the number of demonstrations, $T_n$ is the number of data points in the $n$th demonstration, and $\hat{\dot{\boldsymbol{x}}}_n(t) = f_p(\boldsymbol{x}_n(t))$ is the predicted state derivative computed based on (8). Note that the learning algorithm, derived in (16)–(22) is a constrained non-convex optimization problem, for which, a local solution can be obtained using standard nonlinear programming algorithms and general purpose solvers, such as sequential quadratic programming (SQP) and active set algorithm. In order to make certain that the solver is provided with a good initialization, the expectation–maximization (E–M) algorithm is used to initialize the parameters of the GMM (Dempster et al. 1977).

## 4 Learning orientation dynamics from demonstrations

Let the angular velocity $\boldsymbol{\omega}(t) \in \mathbb{R}^3$ and the orientation described by the quaternion $\boldsymbol{q}(t) \in \mathbb{S}^3 \subset \mathbb{R}^4$ evolve according to the following differential equations

$$\dot{\boldsymbol{\omega}}(t) = f_o(\boldsymbol{\omega}(t), \boldsymbol{\omega}(t), \boldsymbol{q}(t), \boldsymbol{q}_g) \tag{24}$$

$$\dot{\boldsymbol{q}}(t) = \frac{1}{2} \boldsymbol{\Omega}(\boldsymbol{\omega}(t)) \boldsymbol{q}(t) \tag{25}$$

where $f_o : \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{S}^3 \times \mathbb{S}^3 \rightarrow \mathbb{R}^3$ is a nonlinear continuously differentiable autonomous function that models the orientation dynamics, $\boldsymbol{q}_g \in \mathbb{S}^3$ is the goal orientation, $\boldsymbol{\Omega}(\boldsymbol{\omega}(t)) = \begin{bmatrix} -[\boldsymbol{\omega}(t)]_\times & \boldsymbol{\omega}(t) \\ -\boldsymbol{\omega}(t)^T & 0 \end{bmatrix} \in \mathbb{R}^{4\times4}$, and $[\boldsymbol{\omega}(t)]_\times \in \mathbb{R}^{3\times3}$ is a skew-symmetric matrix formed with the angular velocity vector $\boldsymbol{\omega}(t)$ at time $t$.

### 4.1 Encoding orientation dynamics using contracting GMMs

The nonlinear function $f_o(\cdot)$ defined in (24) is modeled using a GMM, and the resulting autonomous DS is given by[3]

[3] In $f_o(\boldsymbol{\omega}(t), \boldsymbol{\omega}(t), \boldsymbol{q}(t), \boldsymbol{q}_g)$, the first argument refers to the $\boldsymbol{\omega}(t)$ in $h_o(\cdot)$ and the second argument refers to the $\boldsymbol{\omega}(t)$ in the affine part of $f_o(\cdot)$.

$$\dot{\boldsymbol{\omega}}(t) = \sum_{k=1}^{K_o} h_o^k(\boldsymbol{z}(t)) \left(\boldsymbol{A}_o^k \boldsymbol{z}(t) + \boldsymbol{b}_o^k\right) = f_o\left(\boldsymbol{\omega}(t), \boldsymbol{\omega}(t), \boldsymbol{q}(t), \boldsymbol{q}_g\right) \quad (26)$$

where $\boldsymbol{z}(t) = [\boldsymbol{\omega}(t)^T,\ \log\left(\boldsymbol{q}_g * \bar{\boldsymbol{q}}(t)\right)^T]^T \in \mathbb{R}^6$, $\bar{\boldsymbol{q}}(t)$ is the quaternion conjugate of $\boldsymbol{q}(t)$, $h_o^k(\boldsymbol{z}) = \frac{\mathcal{P}(k)\mathcal{P}(\boldsymbol{z}|k)}{\sum_{i=1}^{K_o}\mathcal{P}(i)\mathcal{P}(\boldsymbol{z}|i)}$ is the scalar weight associated with the $k$th Gaussian such that $0 \leq h_o^k(\boldsymbol{z}) \leq 1$ and $\sum_{k=1}^{K_o} h_o^k(\boldsymbol{z}) = 1$, $\mathcal{P}(k) = \pi_o^k$ is the prior probability, $\boldsymbol{A}_o^k = \boldsymbol{\Sigma}_{\dot{\boldsymbol{\omega}}\boldsymbol{z}}^k \left(\boldsymbol{\Sigma}_{\boldsymbol{z}}^k\right)^{-1}$, $\boldsymbol{b}_o^k = \boldsymbol{\mu}_{\dot{\boldsymbol{\omega}}}^k - \boldsymbol{A}_o^k \boldsymbol{\mu}_{\boldsymbol{z}}^k$, $\boldsymbol{\mu}_o^k = [\boldsymbol{\mu}_{\boldsymbol{z}}^k,\ \boldsymbol{\mu}_{\dot{\boldsymbol{\omega}}}^k]^T \in \mathbb{R}^9$ is the mean of the $k$th Gaussian, $\boldsymbol{\mu}_{\boldsymbol{z}}^k \in \mathbb{R}^6$ and $\boldsymbol{\mu}_{\dot{\boldsymbol{\omega}}}^k \in \mathbb{R}^3$ are the mean vectors associated with $\boldsymbol{z}$ and $\dot{\boldsymbol{\omega}}$, respectively, $\boldsymbol{\Sigma}_o^k = \begin{pmatrix} \boldsymbol{\Sigma}_{\boldsymbol{z}}^k & \boldsymbol{\Sigma}_{\boldsymbol{z}\dot{\boldsymbol{\omega}}}^k \\ \boldsymbol{\Sigma}_{\dot{\boldsymbol{\omega}}\boldsymbol{z}}^k & \boldsymbol{\Sigma}_{\dot{\boldsymbol{\omega}}}^k \end{pmatrix} \in \mathbb{R}^{9\times 9}$ is the covariance of the $k$th Gaussian, $\boldsymbol{\Sigma}_{\boldsymbol{z}}^k \in \mathbb{R}^{6\times 6}$ and $\boldsymbol{\Sigma}_{\dot{\boldsymbol{\omega}}}^k \in \mathbb{R}^{3\times 3}$ are the covariance matrices associated with $\boldsymbol{z}$ and $\dot{\boldsymbol{\omega}}$, respectively, and $\boldsymbol{\Sigma}_{\boldsymbol{z}\dot{\boldsymbol{\omega}}}^k \in \mathbb{R}^{6\times 3}$ is the cross-covariance matrix between $\boldsymbol{z}$ and $\dot{\boldsymbol{\omega}}$. The quaternion logarithm $\log(\boldsymbol{q}) : \mathbb{S}^3 \to \mathbb{R}^3$ is defined as

$$\log(\boldsymbol{q}) = \begin{cases} \arccos(v)\frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}, & \boldsymbol{u} \neq 0 \\ [0,0,0]^T, & \text{otherwise} \end{cases} \quad (27)$$

The expression $\log\left(\boldsymbol{q}_g * \bar{\boldsymbol{q}}(t)\right)$ can be viewed as the error or the distance between the quaternion at time $t$ and the goal quaternion in $\mathbb{R}^3$ (Ude 1999). Note that the dynamics in (24) and (25), by design, generate quaternion trajectories that stay in the $\mathbb{S}^3$ manifold. Hence, this design circumvents the need for post-processing of the quaternion trajectory to ensure $\mathbb{S}^3$ membership. Given a set of $N_o$ demonstrations, this paper addresses the problem of learning the function $f_o(\cdot)$, which is modeled using a GMM. The $n$th demonstration consists of the trajectories of angular velocity, angular acceleration, and the orientation represented using quaternions: $\{\boldsymbol{\omega}_n(t), \dot{\boldsymbol{\omega}}_n(t), \boldsymbol{q}_n(t)\}_{t=0}^{t=T_n}$.

**Theorem 4** *If the following constraints are satisfied for the autonomous nonlinear DS in (26),*

$$\left(\boldsymbol{A}_{\boldsymbol{\omega}}^k\right)^T \boldsymbol{M}_o + \boldsymbol{M}_o \boldsymbol{A}_{\boldsymbol{\omega}}^k \leq -\gamma_o \boldsymbol{M}_o, \quad k = 1, 2, \ldots, K_o, \quad (28)$$

$$\boldsymbol{b}_o^k = 0, \quad k = 1, 2, \ldots, K_o \quad (29)$$

$$|\boldsymbol{A}_{\boldsymbol{q}}^k| \neq 0, \quad k = 1, 2, \ldots, K_o, \quad (30)$$

*where $\boldsymbol{A}_{\boldsymbol{\omega}}^k$ and $\boldsymbol{A}_{\boldsymbol{q}}^k$ are sub-matrices of $\boldsymbol{A}_o^k$ obtained by selecting the first, and last three columns of $\boldsymbol{A}_o^k$, respectively, $\gamma_o$ is a strictly positive scalar constant, $|\cdot|$ denotes the determinant of a matrix, and $\boldsymbol{M}_o \in \mathbb{R}^{3\times 3}$ represents a constant positive definite symmetric matrix, then all trajectories of the system in (26) converge to $\boldsymbol{\omega}_g = [0\,0\,0]^T$ while $\boldsymbol{q}(t) = \boldsymbol{q}_g$.*

***Proof*** Consider $K_o$ systems given by

$$\dot{\boldsymbol{y}}_o(t) = \boldsymbol{A}_o^k \boldsymbol{\mathcal{Z}}(t) + \boldsymbol{b}_o^k, \quad k = 1, 2, \ldots, K_o \quad (31)$$

where $\boldsymbol{y}_o(t) \in \mathbb{R}^3$, $\boldsymbol{\mathcal{Z}}(t) = [\boldsymbol{y}_o(t)^T,\ \log\left(\boldsymbol{q}_g * \bar{\boldsymbol{q}}(t)\right)^T]^T$. If (28)–(30) are satisfied, the matrices $\boldsymbol{A}_{\boldsymbol{\omega}}^k$ and are $\boldsymbol{A}_{\boldsymbol{q}}^k$ are guaranteed to be full rank and, as a result, have trivial null spaces. This implies that $\boldsymbol{A}_{\boldsymbol{\omega}}^k \boldsymbol{\omega} = [0\,0\,0]^T$ only when $\boldsymbol{\omega} = [0\,0\,0]^T$ and similarly $\boldsymbol{A}_{\boldsymbol{q}}^k \boldsymbol{e} = [0\,0\,0]^T$ only when $\boldsymbol{e} = [0\,0\,0]^T$. Note that $\log\left(\boldsymbol{q}_g * \bar{\boldsymbol{q}}(t)\right)^T = [0\,0\,0]^T$ only when $\boldsymbol{q}(t) = \boldsymbol{q}_g$. Thus, $\dot{\boldsymbol{y}}_o(t) = [0\,0\,0]^T$ only when $\boldsymbol{y}_o(t) = \boldsymbol{\omega}_g = [0\,0\,0]^T$ and $\boldsymbol{q}(t) = \boldsymbol{q}_g$. Hence, $\boldsymbol{y}_o = \boldsymbol{\omega}_g$ while $\boldsymbol{q}(t) = \boldsymbol{q}_g$ is the common equilibrium for each of the $K_o$ systems in (31). Further, based on Theorem 1, if the constraints in (28) are satisfied, then each $k$th system of (31) is said to be contracting with respect to $\boldsymbol{y}_o$ in a common contraction metric $\boldsymbol{M}_o$ and all their trajectories will converge to the goal location $\boldsymbol{\omega}_g$ while $\boldsymbol{q}(t) = \boldsymbol{q}_g$. Now consider a new auxiliary system, which is a convex combination of the systems in (31), given by

$$\dot{\boldsymbol{y}}_o(t) = f_o\left(\boldsymbol{y}_o(t), \boldsymbol{\omega}(t), \boldsymbol{q}(t), \boldsymbol{q}_g\right) = \sum_{k=1}^{K_o} h_o^k(\boldsymbol{z}(t)) \left(\boldsymbol{A}_o^k \boldsymbol{\mathcal{Z}}(t) + \boldsymbol{b}_o^k\right). \quad (32)$$

If (29) and (30) are satisfied, the particular solution $\boldsymbol{y_o} = \boldsymbol{\omega}_g$ of (32) is also its equilibrium point. On defining the virtual dynamics

$$\delta\dot{\boldsymbol{y}}_o \triangleq \frac{\partial f_o}{\partial \boldsymbol{y}_o} \delta \boldsymbol{y}_o \quad (33)$$

and substituting the Jacobian of the auxiliary system (given by $\frac{\partial f_o}{\partial \boldsymbol{y}_o} = \sum_{k=1}^{K_o} h_o^k(\boldsymbol{z}(t)) \boldsymbol{A}_{\boldsymbol{\omega}}^k$), for any constant symmetric positive definite $\boldsymbol{M}_o$,

$$\begin{aligned} \frac{\partial f_o}{\partial \boldsymbol{y}_o}^T \boldsymbol{M}_o + \boldsymbol{M}_o \frac{\partial f_o}{\partial \boldsymbol{y}_o} &\triangleq \sum_{k=1}^{K_o} h_o^k(\boldsymbol{z}(t)) \left(\boldsymbol{A}_{\boldsymbol{\omega}}^k\right)^T \boldsymbol{M}_o \\ &\quad + \boldsymbol{M}_o \sum_{k=1}^{K_o} h_o^k(\boldsymbol{z}(t)) \boldsymbol{A}_{\boldsymbol{\omega}}^k \\ &= \sum_{k=1}^{K_o} h_o^k(\boldsymbol{z}(t)) \left(\left(\boldsymbol{A}_{\boldsymbol{\omega}}^k\right)^T \boldsymbol{M}_o + \boldsymbol{M}_o \boldsymbol{A}_{\boldsymbol{\omega}}^k\right) \end{aligned} \quad (34)$$

Using (28) and (34), we have $\frac{\partial f_o}{\partial \boldsymbol{y}_o}^T \boldsymbol{M}_o + \boldsymbol{M}_o \frac{\partial f_o}{\partial \boldsymbol{y}_o} \leq -\sum_{k=1}^{K_o} h_o^k(\boldsymbol{z}(t)) \gamma_o \boldsymbol{M}_o,\ \forall \boldsymbol{y}_o(t)$. Since $0 \leq h_o^k(\boldsymbol{z}(t)) \leq 1$ and $\sum_{k=1}^{K_o} h_o^k(\boldsymbol{z}(t)) = 1,\ \forall \boldsymbol{z}(t)$, we have

$$\frac{\partial f_o}{\partial \boldsymbol{y}_o}^T \boldsymbol{M}_o + \boldsymbol{M}_o \frac{\partial f_o}{\partial \boldsymbol{y}_o} \leq -\gamma_o \boldsymbol{M}_o,\ \forall \boldsymbol{y}_o(t) \quad (35)$$

Taking the time derivative of $V(\delta \boldsymbol{y}_o) = \delta \boldsymbol{y}_o^T \boldsymbol{M}_o \delta \boldsymbol{y}_o$, and using (35) yields $\dot{V}\left(\delta \boldsymbol{y}_o\right) \leq -\gamma_o \delta \boldsymbol{y}_o^T \boldsymbol{M}_o \delta \boldsymbol{y}_o$. Hence, the system (32) is contracting with respect to $\boldsymbol{y}_o$ (recall Definition 1). Further, based on Theorem 1, all the trajectories of (32) globally exponentially converge towards each other.

Now that the convergence of $\boldsymbol{y}_o(t)$ is shown, the convergence of $\boldsymbol{\omega}(t)$ to $\boldsymbol{\omega}_g$ remains to be proven. Since the auxiliary system in (32) is contracting with respect to $\boldsymbol{y}_o$, and the trajectory $\boldsymbol{y}_o(t) = \boldsymbol{\omega}_g$ (a particular solution of (32)) is an equilibrium point, then according to Theorem 2, the trajectories $\boldsymbol{\omega}(t)$ of (26) will globally exponentially converge to $\boldsymbol{\omega}_g$. □

The constrained optimization problem to be solved in order to train the GMM model can be written as

$$\hat{\boldsymbol{\theta}}_o, \hat{\boldsymbol{M}}_o = \arg\min_{\boldsymbol{\theta}_o, \boldsymbol{M}_o} \; J_o(\boldsymbol{\theta}_o) \tag{36}$$

$$\text{s.t.} \left(\boldsymbol{A}_{\boldsymbol{\omega}}^k\right)^T \boldsymbol{M}_o + \boldsymbol{M}_o \boldsymbol{A}_{\boldsymbol{\omega}}^k + \gamma_o \boldsymbol{M}_o \preceq 0, \quad k = 1, \ldots, K_o, \tag{37}$$

$$\boldsymbol{b}_o^k = 0, \quad k = 1, \ldots, K_o, \tag{38}$$

$$|\boldsymbol{A}_{\boldsymbol{q}}^k| \neq 0, \quad k = 1, \ldots, K_o, \tag{39}$$

$$\boldsymbol{M}_o \succ 0 \tag{40}$$

$$\boldsymbol{\Sigma}_o^k \succ 0, \quad k = 1, \ldots, K_o, \tag{41}$$

$$0 \leq \pi_o^k \leq 1, \quad k = 1, \ldots, K_o, \tag{42}$$

$$\sum_k \pi_o^k = 1, \tag{43}$$

where $\boldsymbol{\theta}_o = \{\boldsymbol{\mu}_o^1 \ldots \boldsymbol{\mu}_o^{K_o}, \boldsymbol{\Sigma}_o^1 \ldots \boldsymbol{\Sigma}_o^{K_o}, \pi_o^1 \ldots \pi_o^{K_o}\}$ is a vector containing the parameters of the GMM model. The constraints (37)–(40) ensure the global attraction of the goal location $\boldsymbol{\omega}_g$ and the constraints in (41)–(43) are a result of using a GMM to model the dynamics. Similar to (Khansari-Zadeh and Billard 2011), the cost function $J_o(\boldsymbol{\theta}_o)$ is chosen to be the mean squared error and is given by

$$J_o(\boldsymbol{\theta}_o) = \frac{1}{2\mathcal{T}_o} \sum_{n=1}^{N_o} \sum_{t=0}^{T_n} \| \hat{\dot{\boldsymbol{\omega}}}_n(t) - \dot{\boldsymbol{\omega}}_n(t) \|^2 \tag{44}$$

where $\mathcal{T}_o = \sum_{n=1}^{N_o} T_n$ is the total number of data points in the demonstrations and $\hat{\dot{\boldsymbol{\omega}}}_n(t) = f_o\left(\boldsymbol{\omega}_n(t), \boldsymbol{q}_n(t), \boldsymbol{q}_g\right)$ is the predicted state derivative computed based on (26). Similar to the solution approach for the optimization problem in (16)–(22), the expectation–maximization (E–M) algorithm (Dempster et al. 1977) is used to initialize the parameters for (36)–(43). A consolidated step-by-step description of the CDSP algorithm is given in Algorithm 1.

**Algorithm 1:** CDSP: Learning Pose Dynamics from Demonstrations

**Collect Pose Demonstrations**:
1. Record the robot's end-effector pose trajectories as a user guides the robot to perform a desired task;
2. If necessary, using the recorded pose trajectories and the finite differences method, obtain linear velocity, angular velocity, and angular acceleration estimates for the demonstrations;

**Learn Position Dynamics**:
3. Define the design parameters of the position GMM, such as the number of Gaussians;
4. Obtain initial estimates of the GMM parameters using the E–M algorithm;
5. Based on the initialization from the last step, obtain the parameters of the position GMM by solving the optimization problem defined in (16)–(22);

**Learn Orientation Dynamics**:
6. Define the design parameters of the orientation GMM, such as the number of Gaussians;
7. Obtain initial estimates of the GMM parameters using the E–M algorithm;
8. Based on the initialization from the last step, obtain the parameters of the orientation GMM by solving the optimization problem defined in (36)–(43);

**Robot Implementation**:
9. Obtain the initial pose of the end-effector, and define the desired goal pose;
10. Translate the origin of the position system to the desired goal location;
11. Generate the end-effector position trajectory by integrating the learned position dynamics per (8);
12. Translate the end-effector position trajectories back to the robot's coordinate frame;
13. Generate the end-effector orientation trajectory by integrating the learned orientation dynamics per (25) and (26);
14. Convert the generated end-effector pose trajectories from Cartesian space into the joint space;
15. Implement the joint space trajectory using a low-level robot controller.

## 5 Experimental evaluation

Three sets of experiments are conducted to evaluate the CDSP algorithm. The algorithm is run on a desktop computer running Intel i3 processor and 8 GB of memory. The algorithm is coded using MATLAB 2016a and the *fmincon* function is used to solve the constrained optimization problem. Initial estimates of the parameters of the GMM are obtained using the expectation–maximization (E–M) algorithm (Dempster et al. 1977). The first set of experiments uses the Learning Algorithms and Systems Laboratory (LASA) human handwriting library introduced in Khansari-Zadeh and Billard (2011) to learn point-to-point position dynamics. The second set of experiments showcase the ability of CDSP to learn orientation dynamics from synthetic demonstrations generated using a minimum jerk polynomial. In the both sets

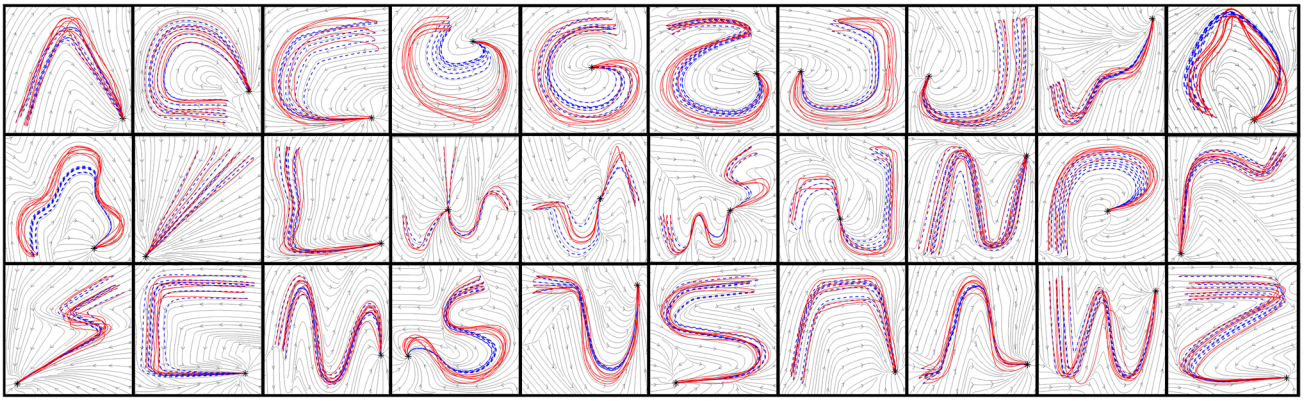

**Fig. 3** Qualitative performance of the CDSP algorithm on the LASA dataset (Color figure online)

of experiments, the CDSP algorithm is compared with state-of-the-art algorithms for learning from demonstrations. In the final set of experiments, the learned models are used in path generation for various tasks performed by a Baxter robot arm. In all the following experiments, the frame of reference for the position trajectories is attached to the desired equilibrium point. Furthermore, similar to Khansari-Zadeh and Billard (2011), Khansari-Zadeh and Billard (2014) and Neumann and Steil (2015), it is assumed that all the demonstrations of a particular shape or motion are intended to converge to the same equilibrium point. In practice, noisy demonstrations are appropriately translated during pre-processing, such that they end at the same point.

### 5.1 Learning position dynamics

The CDSP algorithm is tested on the LASA human handwriting library, introduced in Khansari-Zadeh and Billard (2011), that consists of handwriting motions collected from pen input using a Tablet PC. The library contains a total of 26 handwriting motion sets and four additional sets with more than one movement shape (Multi Models). The qualitative performance of the CDSP algorithm on the benchmark dataset over 30 handwriting motions is shown in Fig. 3. The demonstrations (solid red) and the reproductions (dashed blue) are overlaid on the streamlines (light gray) of each of the learned DSs.

To evaluate the performance of the CDSP algorithm against state-of-the-art algorithms, the CDSP algorithm is compared with 1) SEDS (Khansari-Zadeh and Billard 2011), 2) CLF-DM (NILC and WSAQF) (Khansari-Zadeh and Billard 2014), 3) NiVF (NILC and WSAQF) (Lemme et al. 2013), and 4) $\tau$-SEDS (NILC and WSAQF) (Neumann and Steil 2015) algorithms. It must be noted that, while the algorithms using the WSAQF parametrization result in Lyapunov candidates that are globally valid, those using the NILC parametrization result in Lyapunov candidates that are valid only in predefined local regions. Furthermore, the WSAQF parametrization is known to perform slightly better than the NILC counterpart since the NILC parametrization incorporates simple alignment of the Lyapunov candidates' gradient and the velocity of the demonstrations, which may not be adequate to guarantee a violation-free Lyapunov candidate (Neumann and Steil 2015).

The comparisons are carried out in terms of the reproduction accuracy as measured by swept error area (SEA) (Khansari-Zadeh and Billard 2014). The SEA for a method is given by

$$SEA = \frac{1}{N_d} \sum_{n=1}^{N_d} \sum_{t=0}^{T_n - 1} \mathcal{A}(\hat{\boldsymbol{x}}_n\,(t)\,, \hat{\boldsymbol{x}}_n\,(t+1)\,, \boldsymbol{x}_n\,(t)\,, \boldsymbol{x}_n\,(t+1)) \tag{45}$$

where $\hat{\boldsymbol{x}}_n\,(t)\,,\ \forall t = 0, \ldots, T_n$ is the equidistantly resampled reproduction of the $n$th demonstration with $T_n$ samples, $N_d$ is the number of demonstrations, and $\mathcal{A}\,(\cdot)$ denotes the area of enclosed tetrahedron formed with the points $\hat{\boldsymbol{x}}_n\,(t)\,, \hat{\boldsymbol{x}}_n\,(t+1)\,, \boldsymbol{x}_n\,(t)$, and $\boldsymbol{x}_n\,(t+1)$ as corners [see Khansari-Zadeh and Billard (2014) for details]. The means and standard deviations of SEA for all the state-of-the art algorithms used in this comparison are obtained from the authors of Neumann and Steil (2015). The results of the comparisons of means and standard deviations of SEA are summarized in Fig. 4.

In order to validate the comparisons against the state-of-the-art algorithms, an one-way analysis of variance (ANOVA) is conducted on the obtained results of all the algorithms. It is observed that the SEA means of all the algorithms are statistically significantly different ($p < 0.001$). Further, to analyze how each of the eight algorithms performed against each other, a Tukey's honestly significant differences (HSD) test is conducted. The statistical significance (or insignificance) of the differences among the algorithms' average SEAs are indicated in Fig. 4.

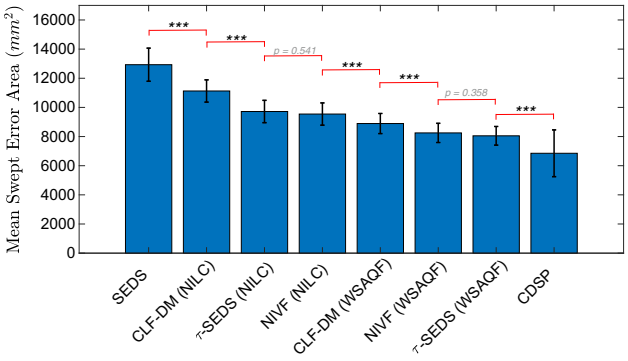


**Fig. 4** The mean swept error area of different algorithms computed over 30 motions (each with seven sample trajectories) of the LASA library. Statistically significant differences ($p < 0.001$) are denoted by *** and the $p$ values of non-significant differences are shown in Gray (Color figure online)

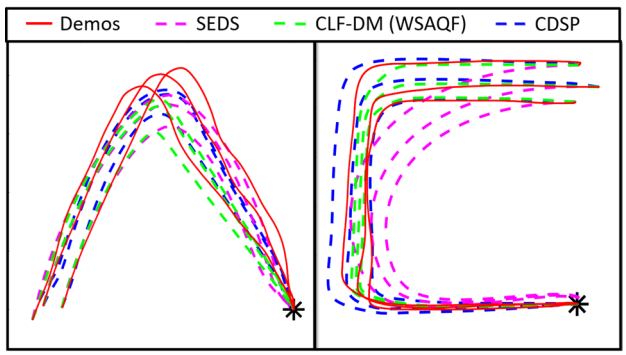


**Fig. 5** Qualitative comparison of the reproductions generated by SEDS, CLF-DM, and CDSP algorithms for a simple shape (*Angle*) on the left and a more complex shape (*SharpC*) on the right (Color figure online)

Further, in Fig. 5, the qualitative performance of SEDS, CLF-DM (WSAQF), and CDSP algorithms on a simple shape (*Angle*) and a more complex shape (*SharpC*)) are shown. The SEDS algorithm does not use a notion of generalized squared distance. Thus, it fails to accurately reproduce complex shapes, such as *SharpC*. The CLF-DM and CDSP algorithms, which use the notion of generalized squared distance, are able to reproduce the *SharpC* shape more accurately.

## 5.2 Learning orientation dynamics

The ability of the CDSP algorithm to learn orientation dynamics is evaluated on a set of 200 synthetic orientation trajectories. A minimum jerk polynomial is sampled between randomly chosen initial and goal quaternions to generate the demonstrations. Each element of both initial and goal quaternions is sampled from a uniformly distributed interval $[-1, 1]$. The sampled trajectories are normalized in order to obtain the reference quaternion trajectories, $\{\boldsymbol{q}_n(t)\}_{t=0}^{t=T_n}$. Based on the obtained quaternion trajectories, the reference angular velocity trajectories, $\{\boldsymbol{\omega}_n(t)\}_{t=0}^{t=T_n}$, are generated. The reference angular acceleration trajectories, $\{\dot{\boldsymbol{\omega}}_n(t)\}_{t=0}^{t=T_n}$, are obtained using the first-order forward finite difference method along with a third-order median filter.

The CDSP algorithm is compared with orientation DMP (ODMP) (Ude et al. 2014), generalized DMP (gen-DMP) (Pastor et al. 2009), TP-GMM (Silvério et al. 2015), CLF-DM (WSAQF) (Khansari-Zadeh and Billard 2014), and SEDS (Khansari-Zadeh and Billard 2011) algorithms. The comparisons are carried out in terms of the orientation reproduction errors, as measured by the log quaternion error. The log quaternion error at time $t$ is given by $\log\left(\boldsymbol{q}_n(t) * \bar{\hat{\boldsymbol{q}}}_n(t)\right)^T$, where $\{\boldsymbol{q}_n(t)\}_{t=0}^{t=T_n}$ is the $n$th reference quaternion trajectory (demonstration), and $\{\hat{\boldsymbol{q}}_n(t)\}_{t=0}^{t=T_n}$ is the corresponding reproduction. To carry out a fair comparison, the number of Gaussians for the CDSP, CLF-DM, and SEDS algorithms is chosen equally to be nine. The design parameters for ODMP, are chosen to be the same as given in Ude et al. (2014) for approximating a desired orientation trajectory. The parameters of the gen-DMP and TP-GMM algorithms are chosen empirically to minimize reproduction errors. An example quaternion reference trajectory and the corresponding reproductions generated by the models trained using different algorithms are shown in Fig. 6. In Fig. 7, the corresponding quaternion reproduction error trajectories of all the methods (in terms of the log quaternion error between the reference and the reproduction trajectories) are shown. In Fig. 8, a reference angular velocity trajectory and its corresponding reproductions by CDSP and ODMP algorithms are shown. The results are computed over 200 runs. In each run, each algorithm is trained on one of the 200 trajectories and tested on the same trajectory. The overall statistics of the comparisons are summarized in Fig. 9.

To validate the comparisons, an one-way ANOVA is conducted on the obtained means and standard deviations of all the algorithms. It is observed that the mean log quaternion errors are statistically significantly different ($p < 0.001$). Further, to analyze how each of the six algorithms performed against each other, a Tukey's HSD test is conducted. The

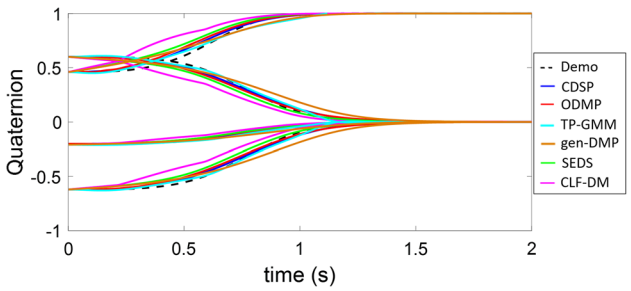


**Fig. 6** An example reference quaternion trajectory overlaid with the corresponding reproductions generated using different algorithms (Color figure online)

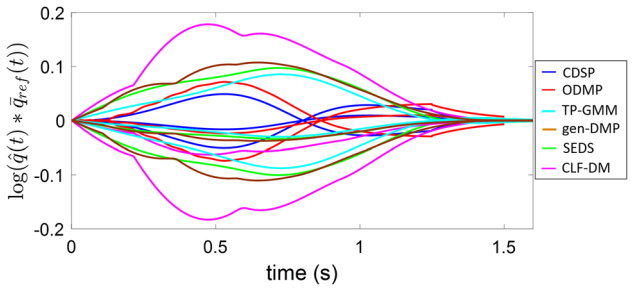


**Fig. 7** The log errors between an example reference quaternion trajectory and the corresponding reproductions generated using different algorithms (Color figure online)

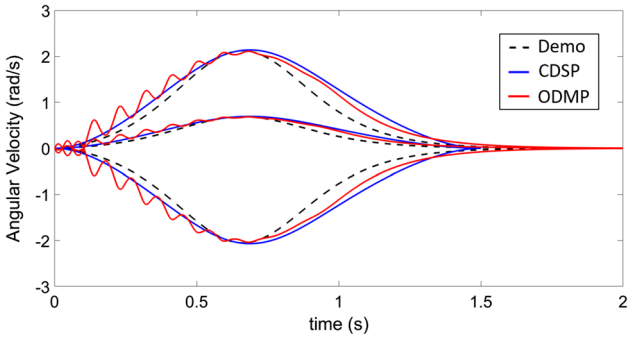


**Fig. 8** Angular velocity trajectories reproduced by the models trained using the CDSP algorithm (solid blue) and the ODMP algorithm (solid red). The reference angular velocity trajectories used for training (dashed black) are also overlaid (Color figure online)

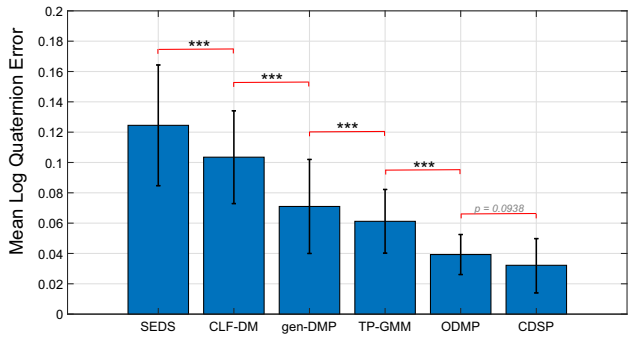


**Fig. 9** The average log quaternion error of different algorithms computed over 50 synthetic orientation trajectories. Statistically significant differences ($p < 0.001$) are denoted by *** and the $p$ values of non-significant differences are shown in Gray

statistical significance (or insignificance) of the differences among the algorithms' average log quaternion errors are indicated in Fig. 9.

## 5.3 Implementation on a robot

For the final experiment, the utility of the CDSP algorithm to generate reference pose trajectories for three different tasks is demonstrated. The tasks are: (1) wire insertion, (2) parts assembly, and (3) water pouring. The wire insertion involved insertion of a wire into a breadboard, the parts assembly task involved attaching a toy construction set block to another, and the water pouring task involved pouring water from a bottle into one of the four mugs placed on the table. Each of these tasks require accurate reproduction of both position and orientation trajectories, and accurate convergence to the goal location in order for the robot to successfully complete them. The tasks are carried out using a seven degree-of-freedom Baxter research robot. For each task, six kinesthetic demonstrations are recorded by manually moving the robot's arm in gravity-compensated mode. The obtained demonstrations contain joint trajectories which are converted to position and orientation trajectories of the end-effector using forward kinematics. First-order forward finite difference method along with a third-order median filter is used to obtain the necessary velocity and acceleration trajectories. These demonstrations are then used to train two GMMs, one for position dynamics and the other for orientation dynamics.

As an example, the demonstrations and the reproductions generated using the trained models for the wire insertion task are shown in Figs. 10 and 11. Further, the reference trajectories generated by the learned models are fed to the robot to perform the tasks autonomously. To execute the trajectories on the Baxter robot, the in-built low-level controller and the inverse kinematics engine, IKFast (Diankov 2010), are used. A sequence of images showing an example of Baxter inserting a wire into a breadboard is provided in Fig. 12. Similar sequences of images showing Baxter performing the construction set block assembly and water pouring tasks are shown in Figs. 13 and 14, respectively.

Furthermore, the execution of each task is carried out a total of 50 times—each time starting from a randomly chosen initial pose and providing a randomly chosen target pose. The CDSP algorithm is able to generate both position and orientation trajectories that converge to the target position and orientation, respectively, on all attempts of all the tasks. Correspondingly, the robot implementation is also carried out 50 times for each task. It is found that the robot successfully (1) inserted the wire into the breadboard, with a tolerance of

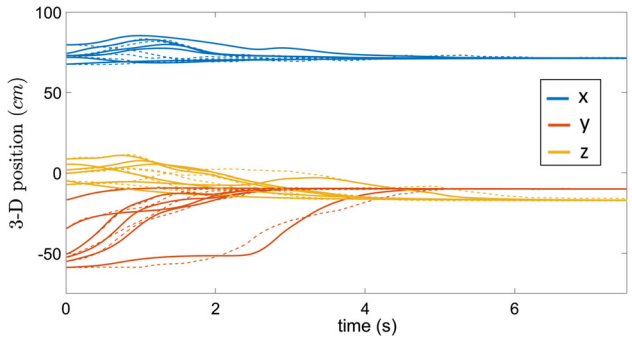


**Fig. 10** 3-D position trajectories reproduced by the trained GMM (solid lines) and the corresponding demonstrations (dashed lines) for the wire insertion task (Color figure online)

$\pm$ 3.5 mm, on 42 instances, (2) assembled two construction set blocks together on 44 instances, (3) poured water in to the target mug on 47 instances. A pouring attempt is considered successful if the stream of water poured by the robot falls entirely inside the target mug.

## 6 Discussion

The learning process is carried out by numerically solving a constrained optimization problem. Due to the non-convex nature of the problem, the learned model is not guaranteed to be the global solution. However, in practice, the CDSP algorithm is shown to be capable of successfully learning a variety of motions as evidenced by the experimental evaluations. Note that the parameters of the contraction matrices, $M_p$ and $M_o$, are learned directly from the demonstrations for each of the motions, as opposed to being manually tuned.

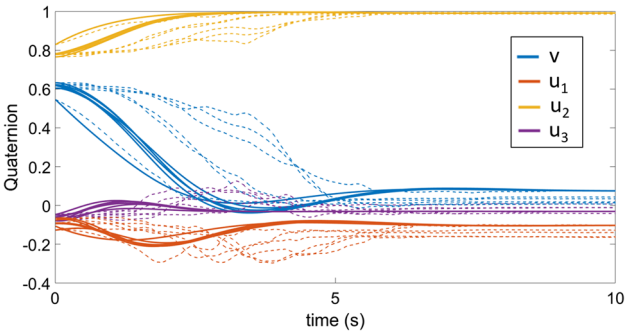


**Fig. 11** Quaternion trajectories reproduced by the trained GMM (solid lines) and the corresponding demonstrations (dashed lines) for the wire insertion task (Color figure online)

In the experiments presented in Sect. 5.1, it is shown that the CDSP algorithm is capable of embedding different shapes in the different parts of the state space of a single DS in the position space (see fourth, fifth, and six shape from the left on the second row of Fig. 3). As shown in Fig. 5, while the SEDS algorithm is capable of accurately reproducing simpler shapes (such as *Angle*), it is not capable of accurately encoding the dynamics of complex shapes (such as *SharpC*). The CLF-DM and CDSP algorithms, on the other hand, are shown to be capable of accurately encoding the dynamics of both shapes. This is due to the fact that, as pointed out Sect. 1, the use of the generalized squared length in the development of constraints allows the CDSP algorithm to accurately model a wider class of motions when compared to the SEDS algorithm. Furthermore, according to the thorough quantitative analyses (including a Tukey's HSD test), the mean SEA of the CDSP algorithm is shown to be statistically significantly lower than that of all the other seven state-of-the-art algorithms used in the comparison ($p < 0.001$). It is also observed that the CDSP algorithm results in one of the highest SEA variances (see Fig. 4). One possible explanation to this observation is that the constant contraction metric $M_p$, used in the development of the constraints, while suitable to accurately model most of the shapes in the LASA library, is too restrictive to accurately model a few highly complex shapes, such as the *DoubleBendedLine* (fourth shape on the

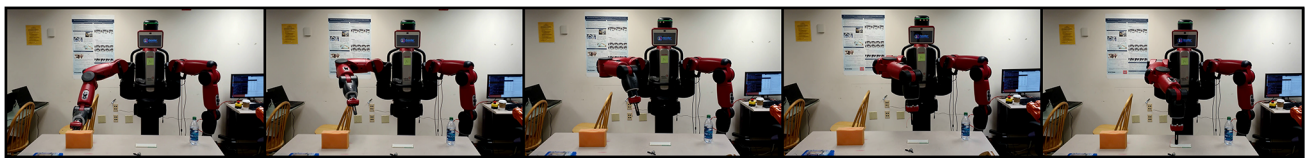

**Fig. 12** A sequence of images showing Baxter autonomously inserting a wire into a bread board

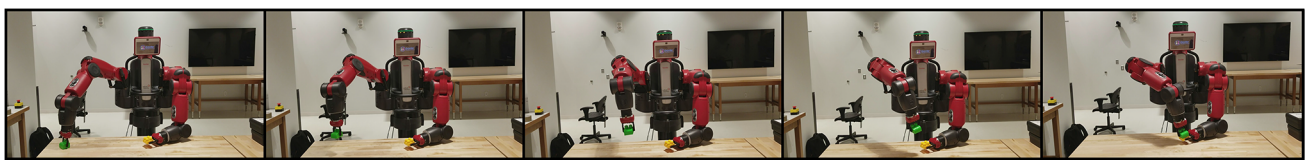

**Fig. 13** A sequence of images showing Baxter autonomously attaching one construction set block to another

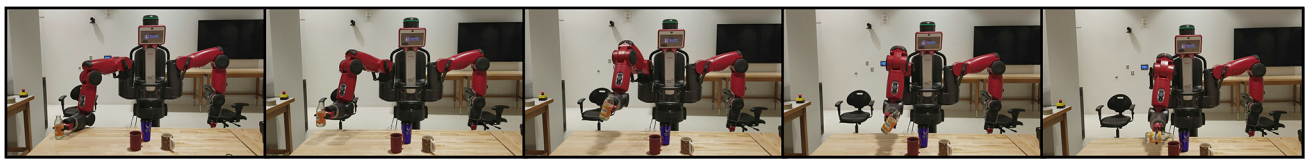

**Fig. 14** A sequence of images showing Baxter autonomously pouring water from a bottle into the correct target mug

first row of Fig. 3) and *JShape2* (seventh shape on the first row of Fig. 3).

In the experiment presented in Sect. 5.2, orientation dynamics are learned from synthetic reference trajectories. Comparative analysis, shown in Fig. 9, reveals that the CDSP algorithm resulted in the lowest average log error when compared to all the other five algorithms evaluated in this experiment used for comparison. Further, according to the Tukey's HSD test, the average log error of the CDSP algorithm's reproductions is statistically significantly ($p < 0.001$) lower than that of the SEDS, CLF-DM, gen-DMP, and TP-GMM algorithms. It is also observed that there is no statistically significant difference between the average log quaternion errors of the CDSP and the ODMP algorithm. However, as shown in Fig. 8, the angular velocity trajectories reproduced by the CDSP algorithm are smoother than those reproduced by ODMP algorithm for the same sample trajectory. It must be noted that, while the quaternion trajectories generated by the CDSP and ODMP algorithms naturally retained $\mathbb{S}^3$ membership, the other four algorithms required an ad-hoc post-normalization step to guarantee membership. Further, the hyper-parameters of the algorithms used in this comparison have to be manually tuned for each reference trajectory. On the other hand, the CDSP algorithm automatically learns its parameters (except for the number of Gaussians) from the demonstrations.

In the experiment presented in Sect. 5.3, the practical utility of the CDSP algorithm to successfully learn pose dynamics of Baxter's end-effector is demonstrated for three different tasks. The learned models are used to generate end-effector reference trajectories for Baxter's arm to perform the desired task. The CDSP algorithm is shown to be capable of modeling a wide variety of tasks, such as wire insertion, parts assembly, and water pouring. Inverse Kinematics is used along with Baxter's low-level controllers to follow the end-effector pose generated by the CDSP algorithm. The robot's low-level controller and low accuracy in tracking the commanded joint angle positions are likely factors that affect the success rate of the wire insertion, block assembly and water pouring tasks.

## 7 Conclusion and future work

The CDSP algorithm for learning arbitrary point-to-point motions is presented. GMMs are used to learn both the position and orientation dynamics from demonstrations. The learned models are then used to generate trajectories for the end-effector motion of a robot. In order to ensure that the trajectories generated by the learned models converge to the goal location, partial contraction analysis-based constraints are developed and enforced in the learning process. Additionally, the system used to model the orientation dynamics is designed such that the special constraints of $\mathbb{S}^3$ are maintained. The experimental evaluations of the CDSP algorithm on the LASA human handwriting library and the synthetically generated orientation data suggest that the CDSP algorithm is able to successfully learn a variety of point-to-point motions. Based on the comparison results against seven state-of-the-art motion generation algorithms on the LASA handwriting library, the CDSP algorithm has the lowest mean swept error area (SEA). The proposed framework of CDSP uses a constant contraction metric in the development of the partial contraction analysis-based constraints. As part of future work, the use of a state-dependent contraction metric will be explored in order to further widen the class of motions that can be learned using the proposed framework.

**Acknowledgements** The authors would like to acknowledge Klaus Neumann and Jochen Steil for providing the SEA means and standard deviations of the seven state-of-the-art algorithms used for comparison presented in Sect. 5.1. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

**Funding** Funding was provided by the UTC Institute for Advanced Systems Engineering (UTC-IASE) of the University of Connecticut and the United Technologies Corporation.

## References

Abbeel, P., & Ng, A. Y., (2004). Apprenticeship learning via inverse reinforcement learning. In *International conference on machine learning* (pp. 1–8). ACM.

Ahmadzadeh, R., Rana, M. A., & Chernova, S. (2017). Generalized cylinders for learning, reproduction, generalization, and refinement of robot skills. In *Robotics: Science and systems*.

Akgun, B., Cakmak, M., Jiang, K., & Thomaz, A. L. (2012). Keyframe-based learning from demonstration. *International Journal of Social Robotics*, *4*(4), 343–355.

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, *57*(5), 469–483.

Billard, A., & Matarić, M. J. (2001). Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, *37*(2), 145–160.

Bowen, C., & Alterovitz, R. (2014). Closed-loop global motion planning for reactive execution of learned tasks. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 1754–1760).

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

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, *4*, 129–145.

Corke, P. I. (2011). *Robotics, vision and control: Fundamental algorithms in Matlab*. Berlin: Springer.

Dani, A. P., Chung, S. J., & Hutchinson, S. (2015). Observer design for stochastic nonlinear systems via contraction-based incremental stability. *IEEE Transactions on Automatic Control*, *60*(3), 700–714.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the E–M algorithm. *Journal of the Royal Statistical Society Series B (Methodological)*, *39*(1), 1–38.

Diankov, R. (2010). Automated construction of robotic manipulation programs. Ph.D. thesis, Carnegie Mellon University.

Dragan, A. D., Muelling, K., Bagnell, J. A., & Srinivasa, S. S. (2015). Movement primitives via optimization. In *IEEE international conference on robotics and automation (ICRA)* (pp. 2339–2346).

Gribovskaya, E., Khansari-Zadeh, S. M., & Billard, A. (2010). Learning non-linear multivariate dynamics of motion in robotic manipulators. *The International Journal of Robotics Research*, *30*(1), 80–117.

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2002). Learning rhythmic movements by demonstration using nonlinear oscillators. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 958–963).

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., & Schaal, S. (2013). Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, *25*(2), 328–373.

Jenkins, O. C., Mataric, M. J., & Weber, S., et al. (2000). Primitive-based movement classification for humanoid imitation. In *IEEE-RAS international conference on humanoid robotics*.

Kalakrishnan, M., Righetti, L., Pastor, P., & Schaal, S. (2012). Learning force control policies for compliant robotic manipulation. In *International conference on machine learning* (pp. 49–50).

Khansari-Zadeh, S. M., & Billard, A. (2010). Imitation learning of globally stable non-linear point-to-point robot motions using nonlinear programming. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2676–2683).

Khansari-Zadeh, S. M., & Billard, A. (2011). Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, *27*(5), 943–957.

Khansari-Zadeh, S. M., & Billard, A. (2014). Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, *62*(6), 752–765.

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*(11), 1238–1274.

Koenig, N., & Matarić, M. J. (2016). Robot life-long task learning from human demonstrations: A bayesian approach. *Autonomous Robots*, *40*(6), 1–16.

Langsfeld, J. D., Kaipa, K. N., Gentili, R. J., Reggia, J. A., & Gupta, S. K. (2014). Incorporating failure-to-success transitions in imitation learning for a dynamic pouring task. In *IEEE/RSJ international conference on intelligent robots and systems*.

Laumond, J. P., Mansard, N., & Lasserre, J. B. (2014). Optimality in robot motion: Optimal versus optimized motion. *Communications of the ACM*, *57*(9), 82–89.

Laumond, J. P., Mansard, N., & Lasserre, J. B. (2015). Optimization as motion selection principle in robot action. *Communications of the ACM*, *58*(5), 64–74.

Lemme, A., Neumann, K., Reinhart, R. F., & Steil, J. J. (2013). Neurally imprinted stable vector fields. In *European symposium on artificial neural networks* (pp. 327–332).

Lohmiller, W., & Slotine, J. J. E. (1998). On contraction analysis for nonlinear systems. *Automatica*, *34*(6), 683–696.

Nemec, B., Tamo, M., Worgotter, F., & Ude, A. (2009). Task adaptation through exploration and action sequencing. In *IEEE-RAS international conference on humanoid robots* (pp. 610–616).

Neumann, K., & Steil, J. J. (2015). Learning robot motions with stable dynamical systems under diffeomorphic transformations. *Robotics and Autonomous Systems*, *70*, 1–15.

Paraschos, A., Daniel, C., Peters, J. R., & Neumann, G. (2013). Probabilistic movement primitives. In *Advances in neural information processing systems* (pp. 2616–2624).

Pastor, P., Hoffmann, H., Asfour, T., & Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. In *IEEE international conference on robotics and automation* (pp. 763–768).

Peters, J., & Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, *71*(7), 1180–1190.

Priess, M. C., Choi, J., & Radcliffe, C. (2014). The inverse problem of continuous-time linear quadratic gaussian control with application to biological systems analysis. In *ASME 2014 dynamic systems and control conference*.

Rai, A., Meier, F., Ijspeert, A., & Schaal, S. (2014). Learning coupling terms for obstacle avoidance. In *International conference on humanoid robotics* (pp. 512–518).

Ravichandar, H., & Dani, A. P. (2015). Learning contracting nonlinear dynamics from human demonstration for robot motion planning. In *ASME dynamic systems and control conference (DSCC)*.

Ravichandar, H., & Dani, A. P. (2016). Human modeling for bio-inspired robotics. In *Intention Inference for human–robot collaboration in assistive robotics* (pp. 217–249). Elsevier.

Ravichandar, H., Salehi, I., & Dani, A. (2017). Learning partially contracting dynamical systems from demonstrations. In *Proceedings of the 1st annual conference on robot learning* (Vol. 78, pp. 369–378). PMLR.

Ravichandar, H., Thota, P. K., & Dani, A. P. (2016). Learning periodic motions from human demonstrations using transverse contraction analysis. In *American control conference (ACC)* (pp. 4853–4858). IEEE.

Rossano, G. F., Martinez, C., Hedelind, M., Murphy, S., & Fuhlbrigge, T. A. (2013). Easy robot programming concepts: An industrial perspective. In *IEEE international conference on automation science and engineering (CASE)* (pp. 1119–1126).

Saunders, J., Nehaniv, C. L., & Dautenhahn, K. (2006). Teaching robots by moulding behavior and scaffolding the environment. In *ACM SIGCHI/SIGART conference on human–robot interaction* (pp. 118–125).

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, *3*(6), 233–242.

Schaal, S., Mohajerian, P., & Ijspeert, A. (2007). Dynamics systems vs. optimal control-a unifying view. *Progress in Brain Research*, *165*, 425–445.

Silvério, J., Rozo, L., Calinon, S., & Caldwell, D. G. (2015). Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 464–470).

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (Vol. 1). Cambridge: MIT Press.

Todorov, E., & Jordan, M. I. (2002). Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, *5*(11), 1226–1235.

Ude, A., Nemec, B., Petrić, T., & Morimoto, J. (2014). Orientation in cartesian space dynamic movement primitives. In *IEEE international conference on robotics and automation (ICRA)* (pp. 2997–3004).

Ude, A. (1999). Filtering in a unit quaternion space for model-based object tracking. *Robotics and Autonomous Systems*, *28*(2), 163–172.

Ude, A., Gams, A., Asfour, T., & Morimoto, J. (2010). Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, *26*(5), 800–815.

Wang, C., Zhao, Y., Lin, C. Y., & Tomizuka, M. (2014). Fast planning of well conditioned trajectories for model learning. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 1460–1465).

Wang, W., & Slotine, J. J. E. (2005). On partial contraction analysis for coupled nonlinear oscillators. *Biological Cybernetics*, *92*(1), 38–53.

Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., et al. (2013). CHOMP: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, *32*(9–10), 1164–1193.

**Harish chaandar Ravichandar** received his M.S. degree in Electrical and Computer Engineering from the University of Florida, Gainesville, FL, USA, in 2014. He is currently working toward his Ph.D. in the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, USA. His current research interests include machine learning, nonlinear dynamical systems and estimation, and human–robot interaction. His work was recognized by the ASEM Dynamics Systems and Controls Conference Best Student Robotics Paper Award in 2015.

**Ashwin Dani** (M'11) received the M.S. and Ph.D. degrees from the University of Florida (UF), Gainesville, FL, USA, in 2008 and 2011, respectively. He was a Post-Doctoral Research Associate at the University of Illinois, Urbana-Champaign, IL, USA. In 2013, he joined the faculty of Electrical and Computer Engineering (ECE) as Assistant Professor at the University of Connecticut, Storrs, CT, USA. He has coauthored over fifty refereed papers, three book chapters, and holds two patents in the area of robotics and vision-based estimation. His current research interests include nonlinear estimation and control, machine learning, human–robot collaboration, autonomous navigation. Dr. Dani serves as a member of the conference editorial board of IEEE Control Systems Society (CSS). His work was recognized by the ASEM Dynamics Systems and Controls Conference Best Student Robotics Paper Award in 2015, ISIF International Conference on Information Fusion Best Student Paper Award—2nd runner up in 2016, IEEE CSS Video Contest Award in 2015, UConn's Outstanding Teaching Award from ECE in 2015 and AAUP-UConn Chapter's Teaching Innovation Award.