EKMP: Generalized Imitation Learning with Adaptation, Nonlinear Hard Constraints and Obstacle Avoidance

Yanlong Huang

***Abstract*— As a user-friendly and straightforward solution for robot trajectory generation, imitation learning has been viewed as a vital direction in the context of robot skill learning. In contrast to unconstrained imitation learning which ignores possible internal and external constraints arising from environments and robot kinematics/dynamics, recent works on constrained imitation learning allow for transferring human skills to unstructured scenarios, further enlarging the application domain of imitation learning. While various constraints have been studied, e.g., joint limits, obstacle avoidance and plane constraints, the problem of nonlinear hard constraints has not been well-addressed. In this paper, we propose *extended kernelized movement primitives* (EKMP) to cope with most of the key problems in imitation learning, including nonlinear hard constraints. Specifically, EKMP is capable of learning the probabilistic features of multiple demonstrations, adapting the learned skills towards arbitrary desired points in terms of joint position and velocity, avoiding obstacles at the level of robot links, as well as satisfying arbitrary linear and nonlinear, equality and inequality hard constraints. Besides, the connections between EKMP and state-of-the-art motion planning approaches are discussed. Several evaluations including the planning of joint trajectories for a 7-DoF robotic arm are provided to verify the effectiveness of our framework.**

I. Introduction

In a myriad of applications, e.g., grasping [1], reaching [2], [3] and painting [4] tasks, appropriate planning of robot trajectories in either Cartesian space or joint space is crucial. In this line, imitation learning becomes appealing due to its intuitive and efficient way of transferring human skills to robots. In addition to the reproduction of demonstrated skills, imitation learning aims at adapting the learned skills to new situations without any additional demonstrations, which is highly desirable towards reducing human intervention. Many algorithms have provided this adaptation feature, such as dynamical movement primitives (DMP) [5] and task-parameterized Gaussian mixture model [6] (GMM), whereas it is non-trivial to directly employ them to constrained scenarios, such as obstacle avoidance, equality (e.g., plane constraints in task space) and inequality (e.g., joint limits) constraints, albeit that constraints are almost ubiquitous in various robotics systems.

In order to cope with dynamical environments, many remarkable algorithms have been developed. For example, in [7] potential field and null-space adjustment were introduced into DMP to avoid collisions of robot links with obstacles. Reinforcement learning was combined with imitation learning in [8], [9] so as to find collision-free trajectories. Functional gradient was employed in [10], [11] to optimize an objective function comprising imitation learning and obstacle avoidance costs. However, none of the aforementioned works [7]–[11] takes hard constraints into account. In [12], [13], the problem of imitation learning under linear hard constraints was studied, whereas obstacle avoidance and in particular nonlinear constraints were not addressed, hence handicapping their applications to more complicated tasks.

In this paper, we aim for a novel framework capable of addressing most of the key problems in imitation learning. Specifically, the proposed approach can:

(i) *learn* the probabilistic properties of multiple demonstrations and *adapt* the learned skills to new situations;
(ii) *avoid collisions* between robot links and obstacles;
(iii) satisfy linear and nonlinear *hard constraints* in terms of equality and inequality;
(iv) learn joint position and velocity simultaneously and maintain the corresponding *derivative relationship* while avoiding obstacles and meeting hard constraints;

As kernelized movement primitives (KMP) [14] has exhibited reliable performance in many tasks, e.g., human robot collaboration [4], [14], walking tasks in humanoid [15] and exoskeleton robots [16], we propose to extend KMP towards developing a more generic constrained imitation learning framework, which we will refer to as *extended KMP* (EKMP). More concretely, we explain the rationale of EKMP in Section II. Subsequently, we discuss connections between EKMP and state-of-the-art motion planning algorithms in Section III, including covariant Hamiltonian optimization for motion planning (CHOMP) [17], Gaussian process motion planner (GPMP) [18] and motion planning in reproducing kernel Hilbert space (RKHS) [19]. Finally, we showcase the performance of EKMP through several evaluations in Section IV.

II. Extended Kernelized Movement Primitives

In this section, we first briefly describe the probabilistic modeling of multiple demonstrations in Section II-A. After that, we exploit the extracted probabilistic features and derive EKMP in Section II-B and II-C, as well as illustrate the implementation of EKMP in Section II-D. Some constraints (e.g., linear constraints) are discussed as special cases of EKMP in Section II-E.

A. Learning Probabilistic Features of Demonstrations

Suppose we have access to a set of demonstrations in the form of time t, joint position $\mathbf{q} \in \mathbb{R}^{\mathcal{O}}$ and velocity $\dot{\mathbf{q}} \in \mathbb{R}^{\mathcal{O}}$, denoted by $\mathbf{D} = \{\{t_{n,w}, \mathbf{q}_{n,w}, \dot{\mathbf{q}}_{n,w}\}_{n=1}^{N}\}_{w=1}^{W}$,

Yanlong Huang is with School of Computing, University of Leeds, Leeds LS29JT, UK. y.l.huang@leeds.ac.uk

where N and W represent the trajectory length and number of demonstrations, respectively. Similarly to many previous works, e.g., [6], [14], we use GMM to model the probabilistic distribution of $\mathbf{D}$, leading to

$$\mathcal{P}(t,\boldsymbol{\xi}) \sim \sum_{c=1}^{C} \pi_c \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \tag{1}$$

where $\boldsymbol{\xi} = [\mathbf{q}^\top \ \dot{\mathbf{q}}^\top]^\top \in \mathbb{R}^{2\mathcal{O}}$, C is the number of Gaussian components, π_c, $\boldsymbol{\mu}_c = \begin{bmatrix} \boldsymbol{\mu}_{t,c} \\ \boldsymbol{\mu}_{\xi,c} \end{bmatrix}$ and $\boldsymbol{\Sigma}_c = \begin{bmatrix} \boldsymbol{\Sigma}_{tt,c} & \boldsymbol{\Sigma}_{t\xi,c} \\ \boldsymbol{\Sigma}_{\xi t,c} & \boldsymbol{\Sigma}_{\xi\xi,c} \end{bmatrix}$ respectively denote prior probability, mean and covariance of the *c-th* Gaussain component. Furthermore, with the parameters of GMM in (1), a probabilistic reference trajectory can be retrieved via Gaussian mixture regression (GMR) , giving $= \{t_n, \hat{\boldsymbol{\xi}}_n\}_{n=1}^N$ with $\mathcal{P}(\hat{\boldsymbol{\xi}}_n | t_n) \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n)$. Here, $\mathbf{D}_r = \{t_n, \hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n\}_{n=1}^N$ can be viewed as a probabilistic representation of demonstrations $\mathbf{D}$. Please refer to [6], [14] for more details on the modeling of demonstrations.

B. Problem Description of EKMP

We start with the unconstrained imitation learning method KMP to learn the reference trajectory $\mathbf{D}_r$. Specifically, we parameterize $\boldsymbol{\xi}(t)$ as

$$\boldsymbol{\xi}(t) = \begin{bmatrix} \mathbf{q}(t) \\ \dot{\mathbf{q}}(t) \end{bmatrix} = \boldsymbol{\Theta}^\top(t)\mathbf{w} \tag{2}$$

where the basis function matrix $\boldsymbol{\Theta}(t) \in \mathbb{R}^{B\mathcal{O}\times 2\mathcal{O}}$ is

$$\boldsymbol{\Theta}(t) = [\mathbf{I}_\mathcal{O} \otimes \boldsymbol{\varphi}(t) \quad \mathbf{I}_\mathcal{O} \otimes \dot{\boldsymbol{\varphi}}(t)] \tag{3}$$

with $\boldsymbol{\varphi}(t) \in \mathbb{R}^\mathcal{B}$ being a basis function vector. '$\otimes$' denotes Kronecker product and $\mathbf{w} \in \mathbb{R}^{\mathcal{B}\mathcal{O}}$ denotes the trajectory parameter vector. As suggested in KMP [14], the optimal $\mathbf{w}$ from learning demonstrations can be obtained by maximizing the posterior $\prod_{n=1}^N \mathcal{P}(\boldsymbol{\xi}(t_n)|\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n)$, which is equivalent to the problem of minimizing

$$\mathcal{U}_{IL}(\mathbf{w}) = \sum_{n=1}^{N} \frac{1}{2}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w} - \hat{\boldsymbol{\mu}}_n)^\top \hat{\boldsymbol{\Sigma}}_n^{-1}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w} - \hat{\boldsymbol{\mu}}_n) + \frac{1}{2}\lambda \mathbf{w}^\top \mathbf{w}, \tag{4}$$

where $\frac{1}{2}\lambda\mathbf{w}^\top\mathbf{w}$ with $\lambda > 0$ serves as the regularization term to alleviate the over-fitting issue.

In order to cope with the obstacle avoidance problem, we employ the cost function from [19] and define it using the parametric trajectory in (2), i.e.,

$$\mathcal{U}_{obs}(\mathbf{w}) = \sum_{m=1}^{M} c\big(\mathbf{x}(\mathbf{q}(\tilde{t}_m), u_m)\big), \tag{5}$$

where M denotes the number of points used for calculating the obstacle avoidance cost. $\mathbf{x}(\mathbf{q}(\tilde{t}_m), u_m) \in \mathbb{R}^3$ corresponds to the workspace position of the u_m*-th* body point[1] at joint position $\mathbf{q}(\tilde{t}_m)$. $c(\cdot)$ is the cost function. Here, we choose u_m as the nearest body point to the obstacle at time $\tilde{t}_m$. Similar treatment was also implemented in [7]. Note that other cost functions developed in [10], [17], [20] can be used as well.

Now, we formulate the problem of EKMP as

$$\begin{aligned} \underset{\mathbf{w}}{\textbf{argmin}} \ & \mathcal{U}(\mathbf{w}) = \mathcal{U}_{IL}(\mathbf{w}) + \lambda_{obs}\mathcal{U}_{obs}(\mathbf{w}) \\ \textbf{s.t.,} \ & \begin{matrix} g_{n,1}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w}) \\ g_{n,2}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w}) \\ \vdots \\ g_{n,F}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w}) \end{matrix} \begin{matrix} \geq 0 \\ \geq 0 \\ \\ \geq 0 \end{matrix} , \forall n \in \{1,2,\ldots,N\}, \end{aligned} \tag{6}$$

where $\lambda_{obs} > 0$ is a constant and F denotes the number of nonlinear constraints at each time step[2]. $g_{n,f}$ represents the f*-th* nonlinear constraint acting on $\boldsymbol{\xi}(t_n) = \boldsymbol{\Theta}^\top(t_n)\mathbf{w}$.

The objective function $\mathcal{U}(\mathbf{w})$ consisting of imitation learning and obstacle avoidance costs shares a similar spirit with [10], [11], [21]. However, differing from [10], [21] that optimize discrete waypoints of a trajectory, we propose to learn the trajectory parameter $\mathbf{w}$. A straightforward advantage of trajectory parameterization is that the optimal solution $\mathbf{w}$ of (6) can be used to generate trajectories $\mathbf{q}(t)$ and $\dot{\mathbf{q}}(t)$ simultaneously via (2) which strictly satisfy the derivative relationship—a crucial requirement in many time-contact tasks. To take the striking task as an example, the racket is demanded to strike an incoming ball at a desired position with a desired velocity. If the derivative relationship can not be respected when planning position and velocity trajectories, the racket's actual velocity (i.e., derivative of position) at the striking position will be different from the desired striking velocity, thus failing to strike the ball properly. Note that the same parameterization was also used in [22] for learning unconstrained motions, which was extended in [11] to deal with the obstacle avoidance problem, whereas both [11], [22] relied on explicit design of basis functions, while, as shall be seen later, in EKMP basis functions are alleviated by virtue of kernelization. It is worthwhile to mention that hard constraints are not considered in [10], [11], [21], [22], which however is an essential goal in EKMP.

C. EKMP

Similarly to [17], [19], we approximate $\mathcal{U}(\mathbf{w})$ using its first-order Taylor approximation. Formally, we respectively approximate $\mathcal{U}_{IL}(\mathbf{w})$ in (4) and $\mathcal{U}_{obs}(\mathbf{w})$ in (5) using the current trajectory parameter $\mathbf{w}^c$, giving

$$\begin{aligned} \mathcal{U}_{IL}(\mathbf{w}) &= \mathcal{U}_{IL}(\mathbf{w}^c) + (\bigtriangledown_\mathbf{w}\mathcal{U}_{IL}(\mathbf{w})|_{\mathbf{w}=\mathbf{w}^c})^\top(\mathbf{w}-\mathbf{w}^c) \\ &= \mathcal{U}_{IL}(\mathbf{w}^c) + \big(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^\top\mathbf{w}^c - \boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \lambda\mathbf{w}^c\big)^\top(\mathbf{w}-\mathbf{w}^c) \end{aligned} \tag{7}$$

with

$$\begin{aligned} \boldsymbol{\Phi} &= [\boldsymbol{\Theta}(t_1) \ \boldsymbol{\Theta}(t_2) \ \cdots \ \boldsymbol{\Theta}(t_N)], \\ \boldsymbol{\Sigma} &= \text{blockdiag}(\hat{\boldsymbol{\Sigma}}_1, \ \hat{\boldsymbol{\Sigma}}_2, \ \ldots, \ \hat{\boldsymbol{\Sigma}}_N), \\ \boldsymbol{\mu} &= [\hat{\boldsymbol{\mu}}_1^\top \ \hat{\boldsymbol{\mu}}_2^\top \ \cdots \ \hat{\boldsymbol{\mu}}_N^\top]^\top, \end{aligned} \tag{8}$$

and

$$\begin{aligned} \mathcal{U}_{obs}(\mathbf{w}) &= \mathcal{U}_{obs}(\mathbf{w}^c) + (\bigtriangledown_\mathbf{w}\mathcal{U}_{obs}(\mathbf{w})|_{\mathbf{w}=\mathbf{w}^c})^\top(\mathbf{w}-\mathbf{w}^c) \\ &= \mathcal{U}_{obs}(\mathbf{w}^c) + \big(\widetilde{\boldsymbol{\Phi}}\mathbf{H}^c\big)^\top(\mathbf{w}-\mathbf{w}^c) \end{aligned} \tag{9}$$

[1]The robotic arm is assumed to be covered by a series of body points, please refer to [17], [19] for the details.

[2]The number of trajectory points imposed with constraints is not necessarily the same as that of demonstrations, depending on tasks at hand.

with

$$\begin{aligned}&\widetilde{\mathbf{\Phi}}=[\mathbf{\Theta}(\tilde{t}_1)\ \mathbf{\Theta}(\tilde{t}_2)\ \cdots\ \mathbf{\Theta}(\tilde{t}_M)],\\&\mathbf{H}^c=[\mathbf{H}_1^{c\top}\ \mathbf{H}_2^{c\top}\cdots\mathbf{H}_M^{c\ \top}]^\top,\\&\mathbf{H}_m^c=\begin{bmatrix}\mathbf{J}^\top(\tilde{t}_m,u_m)\bigtriangledown_{\mathbf{x}}\,c(\mathbf{x})|_{\mathbf{x}=\mathbf{x}(\mathbf{q}(\tilde{t}_m),u_m)}\\\mathbf{0}\end{bmatrix},\end{aligned}\tag{10}$$

where $\mathbf{J}(\tilde{t}_m,u_m)=\frac{\partial\mathbf{x}(\mathbf{q},u_m)}{\partial\mathbf{q}}|_{\mathbf{q}=\mathbf{q}(\tilde{t}_m)}$ corresponds to the Jacobian matrix of the u_m-*th* body point when joint configuration is $\mathbf{q}(\tilde{t}_m)$. The zero vector ($\mathcal{O}$-dimensional) in $\mathbf{H}_m^c$ is due to the fact that $\mathcal{U}_{obs}$ in (5) is independent of joint velocity. In the case of only optimizing joint positions, as done in [19], this zero vector in $\mathbf{H}_m^c$ will not exist .

Furthermore, nonlinear constraints $g_{n,f}(\mathbf{\Theta}^\top(t_n)\mathbf{w})$ can be approximated by

$$\begin{aligned}g_{n,f}(\mathbf{\Theta}^\top(t_n)\mathbf{w})&=g_{n,f}(\mathbf{\Theta}^\top(t_n)\mathbf{w}^c)\\&+(\bigtriangledown_{\boldsymbol{\xi}}g_{n,f}(\boldsymbol{\xi})|_{\boldsymbol{\xi}=\mathbf{\Theta}^\top(t_n)\mathbf{w}^c})^\top\mathbf{\Theta}^\top(t_n)(\mathbf{w}-\mathbf{w}^c).\end{aligned}\tag{11}$$

Here, the linearization is a typical way to deal with nonlinear constraints [23]. Let us write

$$\begin{aligned}\boldsymbol{\xi}_n^c&=\mathbf{\Theta}^\top(t_n)\mathbf{w}^c,\\\bigtriangledown g_{n,f}(\boldsymbol{\xi}_n^c)&=\bigtriangledown_{\boldsymbol{\xi}}g_{n,f}(\boldsymbol{\xi})|_{\boldsymbol{\xi}=\mathbf{\Theta}^\top(t_n)\mathbf{w}^c},\end{aligned}\tag{12}$$

where $\boldsymbol{\xi}_n^c$ denotes the predicted trajectory point at time t_n with the current parameter $\mathbf{w}^c$. Using Lagrange multipliers $\boldsymbol{\alpha}=[\alpha_{1,1},\alpha_{1,2},\ldots,\alpha_{1,F},\ldots,\alpha_{N,1},\alpha_{N,2},\ldots,\alpha_{N,F}]^\top$ with $\alpha_{n,f}\geq 0$, we can transform the problem in (6) as

$$\begin{aligned}L(\mathbf{w},\boldsymbol{\alpha})=&\left(\mathbf{\Phi}\mathbf{\Sigma}^{-1}(\mathbf{\Phi}^\top\mathbf{w}^c-\boldsymbol{\mu})+\lambda\mathbf{w}^c+\lambda_{obs}\widetilde{\mathbf{\Phi}}\mathbf{H}^c\right)^\top(\mathbf{w}-\mathbf{w}^c)\\&-\sum_{n=1}^{N}\sum_{f=1}^{F}\alpha_{n,f}\left(g_{n,f}(\boldsymbol{\xi}_n^c)+\left(\bigtriangledown g_{n,f}(\boldsymbol{\xi}_n^c)\right)^\top\mathbf{\Theta}^\top(t_n)(\mathbf{w}-\mathbf{w}^c)\right)\\&+\frac{1}{2}\beta(\mathbf{w}-\mathbf{w}^c)^\top(\mathbf{w}-\mathbf{w}^c),\end{aligned}\tag{13}$$

where $\mathcal{U}_{IL}(\mathbf{w}^c)$ in (7) and $\mathcal{U}_{obs}(\mathbf{w}^c)$ in (9) are ignored since they are constant at the current iteration. The last term $\frac{1}{2}\beta(\mathbf{w}-\mathbf{w}^c)^\top(\mathbf{w}-\mathbf{w}^c)$ with $\beta>0$ is added to mitigate an overlarge update of $\mathbf{w}$. The regularization has also been studied in [10], [17]–[19], but the regularization terms in [10], [17], [18] were defined on trajectory waypoints while in [19] it was defined in RHKS.

By setting the partial derivative of $L(\mathbf{w},\boldsymbol{\alpha})$ w.r.t. $\mathbf{w}$ as zero, we have

$$\begin{aligned}\mathbf{w}=(1-\frac{\lambda}{\beta})\mathbf{w}^c-\frac{1}{\beta}\big(&\mathbf{\Phi}\mathbf{\Sigma}^{-1}(\mathbf{\Phi}^\top\mathbf{w}^c-\boldsymbol{\mu})\\&+\lambda_{obs}\widetilde{\mathbf{\Phi}}\mathbf{H}^c-\mathbf{\Phi}\mathbf{G}^c\boldsymbol{\alpha}\big)\end{aligned}\tag{14}$$

with

$$\begin{aligned}&\mathbf{G}_n^c=[\bigtriangledown g_{n,1}(\boldsymbol{\xi}_n^c)\ \bigtriangledown g_{n,2}(\boldsymbol{\xi}_n^c)\ \ldots\ \bigtriangledown g_{n,F}(\boldsymbol{\xi}_n^c)],\\&\mathbf{G}^c=\mathrm{blockdiag}(\mathbf{G}_1^c,\ \mathbf{G}_2^c,\ \ldots,\ \mathbf{G}_N^c).\end{aligned}\tag{15}$$

Then, we substitute (14) into (13), yielding

$$\begin{aligned}\widetilde{L}(\boldsymbol{\alpha})=&-\frac{1}{2\beta}\boldsymbol{\alpha}^\top\mathbf{G}^{c\top}\mathbf{\Phi}^\top\mathbf{\Phi}\mathbf{G}^c\boldsymbol{\alpha}-\mathbf{Q}^{c\top}\boldsymbol{\alpha}\\&+\frac{1}{\beta}\big((\mathbf{\Phi}^\top\mathbf{w}^c-\boldsymbol{\mu})^\top\mathbf{\Sigma}^{-1}\mathbf{\Phi}^\top+\lambda\mathbf{w}^{c\top}+\lambda_{obs}\mathbf{H}^{c\top}\widetilde{\mathbf{\Phi}}^\top\big)\mathbf{\Phi}\mathbf{G}^c\boldsymbol{\alpha}\end{aligned}\tag{16}$$

with

$$\begin{aligned}&\mathbf{Q}_n^c=[g_{n,1}(\boldsymbol{\xi}_n^c)\ g_{n,2}(\boldsymbol{\xi}_n^c)\ \ldots\ g_{n,F}(\boldsymbol{\xi}_n^c)]^\top,\\&\mathbf{Q}^c=[\mathbf{Q}_1^\top\ \mathbf{Q}_2^\top\ldots\mathbf{Q}_N^\top]^\top.\end{aligned}\tag{17}$$

Furthermore, we can kernelize (16) by using the kernel trick $\boldsymbol{\varphi}(t_i)^\top\boldsymbol{\varphi}(t_j)=k(t_i,t_j)$, where $k(\cdot,\cdot)$ is a kernel function. The kernelized form of (16) is

$$\begin{aligned}\widetilde{L}(\boldsymbol{\alpha})=&-\frac{1}{2\beta}\boldsymbol{\alpha}^\top\mathbf{G}^{c\top}\mathbf{K}\mathbf{G}^c\boldsymbol{\alpha}-\mathbf{Q}^{c\top}\boldsymbol{\alpha}\\&+\frac{1}{\beta}\big((\boldsymbol{\xi}^c-\boldsymbol{\mu})^\top\mathbf{\Sigma}^{-1}\mathbf{K}+\lambda\boldsymbol{\xi}^{c\top}+\lambda_{obs}\mathbf{H}^{c\top}\widetilde{\mathbf{K}}\big)\mathbf{G}^c\boldsymbol{\alpha},\end{aligned}\tag{18}$$

where $\boldsymbol{\xi}^c$ represents the predicted trajectory, i.e.,

$$\boldsymbol{\xi}^c=\mathbf{\Phi}^\top\mathbf{w}^c=[\boldsymbol{\xi}_1^{c\top}\boldsymbol{\xi}_2^{c\top}\ldots\boldsymbol{\xi}_N^{c\ \top}]^\top.\tag{19}$$

$\mathbf{K}$ denotes a $N\times N$ block matrix defined by

$$\mathbf{K}=\mathbf{\Phi}^\top\mathbf{\Phi}=\begin{bmatrix}\mathbf{k}(t_1,t_1)&\mathbf{k}(t_1,t_2)&\cdots&\mathbf{k}(t_1,t_N)\\\mathbf{k}(t_2,t_1)&\mathbf{k}(t_2,t_2)&\cdots&\mathbf{k}(t_2,t_N)\\\vdots&\vdots&\ddots&\vdots\\\mathbf{k}(t_N,t_1)&\mathbf{k}(t_N,t_2)&\cdots&\mathbf{k}(t_N,t_N)\end{bmatrix}\tag{20}$$

with

$$\mathbf{k}(t_i,t_j)=\mathbf{\Theta}(t_i)^\top\mathbf{\Theta}(t_j)=\begin{bmatrix}k_{tt}(i,j)\mathbf{I}_{\mathcal{O}}&k_{td}(i,j)\mathbf{I}_{\mathcal{O}}\\k_{dt}(i,j)\mathbf{I}_{\mathcal{O}}&k_{dd}(i,j)\mathbf{I}_{\mathcal{O}}\end{bmatrix},\tag{21}$$

where $k_{tt}(i,j)=k(t_i,t_j), k_{td}(i,j)=\frac{k(t_i,t_j+\delta)-k(t_i,t_j)}{\delta}, k_{dt}(i,j)=\frac{k(t_i+\delta,t_j)-k(t_i,t_j)}{\delta}, k_{dd}(i,j)=\frac{k(t_i+\delta,t_j+\delta)-k(t_i+\delta,t_j)-k(t_i,t_j+\delta)+k(t_i,t_j)}{\delta^2}$. Here, $\delta>0$ denotes a small constant. Please refer to KMP [14] for more details of the kernelization. Similarly, $\widetilde{\mathbf{K}}=\mathbf{\Phi}^\top\widetilde{\mathbf{\Phi}}$ corresponding to a $N\times M$ block matrix can be obtained.

Since in (18) the quadratic coefficient $-\frac{1}{2\beta}\mathbf{G}^{c\top}\mathbf{K}\mathbf{G}^c$ is a symmetric and negative definite matrix, we can resort to *quadratic programming* (QP) to find the optimal $\boldsymbol{\alpha}^c$ at the current iteration maximizing (18) while satisfying the constraint $\boldsymbol{\alpha}\geq\mathbf{0}$. Therefore, for an arbitrary time t, its corresponding trajectory point can be determined via (14) with $\boldsymbol{\alpha}=\boldsymbol{\alpha}^c$, i.e.,

$$\begin{aligned}\boldsymbol{\xi}(t)&=\begin{bmatrix}\mathbf{q}(t)\\\dot{\mathbf{q}}(t)\end{bmatrix}=\mathbf{\Theta}^\top(t)\mathbf{w}\\&=(1-\frac{\lambda}{\beta})\mathbf{\Theta}^\top(t)\mathbf{w}^c-\frac{1}{\beta}\big(\mathbf{\Theta}^\top(t)\mathbf{\Phi}\mathbf{\Sigma}^{-1}(\mathbf{\Phi}^\top\mathbf{w}^c-\boldsymbol{\mu})\\&+\lambda_{obs}\mathbf{\Theta}^\top(t)\widetilde{\mathbf{\Phi}}\mathbf{H}^c-\mathbf{\Theta}^\top(t)\mathbf{\Phi}\mathbf{G}^c\boldsymbol{\alpha}^c\big).\end{aligned}\tag{22}$$

Again, by employing the kernel trick, we have

$$\begin{aligned}&\mathbf{k}(t)=\mathbf{\Theta}^\top(t)\mathbf{\Phi}=[\mathbf{k}(t,t_1)\ \mathbf{k}(t,t_2)\ \cdots\ \mathbf{k}(t,t_N)],\\&\widetilde{\mathbf{k}}(t)=\mathbf{\Theta}^\top(t)\widetilde{\mathbf{\Phi}}=\left[\mathbf{k}(t,\tilde{t}_1)\ \mathbf{k}(t,\tilde{t}_2)\ \cdots\ \mathbf{k}(t,\tilde{t}_M)\right].\end{aligned}\tag{23}$$

Let us write $\boldsymbol{\xi}^c(t)=\mathbf{\Theta}^\top(t)\mathbf{w}^c$, the updated *trajectory function* corresponding to (22) becomes

$$\begin{aligned}\boldsymbol{\xi}(t)=(1-\frac{\lambda}{\beta})\boldsymbol{\xi}^c(t)-\frac{1}{\beta}\Big(&\mathbf{k}(t)\mathbf{\Sigma}^{-1}(\boldsymbol{\xi}^c-\boldsymbol{\mu})\\&+\lambda_{obs}\widetilde{\mathbf{k}}(t)\mathbf{H}^c-\mathbf{k}(t)\mathbf{G}^c\boldsymbol{\alpha}^c\Big).\end{aligned}\tag{24}$$

Thus, we have obtained a kernelized rule for updating the trajectory function.

D. Implementation of EKMP

To illustrate how the trajectory function can be updated using (24), we here explain the implementation details of the first two iterations. In practice, we can use vanilla KMP to generate an initial trajectory before applying the iteration rule in (24), since KMP provides an analytical solution for imitation learning. Formally, the initial trajectory function from vanilla KMP is

$$\boldsymbol{\xi}^{(0)}(t) = \mathbf{k}(t)\underbrace{(\mathbf{K}+\lambda\boldsymbol{\Sigma})^{-1}\boldsymbol{\mu}}_{\boldsymbol{\gamma}^{(0)}}, \tag{25}$$

which can be used to predict trajectory points at time steps $\{t_n\}_{n=1}^N$, i.e., $\boldsymbol{\xi}^{(0)} = \mathbf{K}\boldsymbol{\gamma}^{(0)}$. Given $\boldsymbol{\xi}^{(0)}$, we can calculate $\mathbf{H}^{(0)}$, $\mathbf{G}^{(0)}$ and $\mathbf{Q}^{(0)}$ via (10), (15) and (17), respectively. Furthermore, the optimal $\boldsymbol{\alpha}^{(0)}$ can be determined by applying the QP optimization to (18). Denote $\mathbf{e}^{(0)} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\xi}^{(0)} - \boldsymbol{\mu}) - \mathbf{G}^{(0)}\boldsymbol{\alpha}^{(0)}$, the updated trajectory function after the first iteration becomes

$$\boldsymbol{\xi}^{(1)}(t) = \mathbf{k}(t)\underbrace{\big((1-\frac{\lambda}{\beta})\boldsymbol{\gamma}^{(0)} - \frac{1}{\beta}\mathbf{e}^{(0)}\big)}_{\boldsymbol{\gamma}^{(1)}} - \widetilde{\mathbf{k}}(t)\underbrace{(\frac{\lambda_{obs}}{\beta}\mathbf{H}^{(0)})}_{\boldsymbol{\rho}^{(1)}}, \tag{26}$$

where the kernel matrices $\mathbf{K}$, $\tilde{\mathbf{K}}$, $\mathbf{k}$ and $\tilde{\mathbf{k}}$ are determined using (20), (21) and (23), $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ are calculated via (8).

In a similar manner, we can predict $\boldsymbol{\xi}^{(1)} = \mathbf{K}\boldsymbol{\gamma}^{(1)} - \widetilde{\mathbf{K}}\boldsymbol{\rho}^{(1)}$, $\mathbf{H}^{(1)}$, $\mathbf{G}^{(1)}$, $\mathbf{Q}^{(1)}$, $\boldsymbol{\alpha}^{(1)}$ and $\mathbf{e}^{(1)} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\xi}^{(1)} - \boldsymbol{\mu}) - \mathbf{G}^{(1)}\boldsymbol{\alpha}^{(1)}$, respectively. Thus, the updated trajectory function after the second iteration is

$$\begin{aligned}\boldsymbol{\xi}^{(2)}(t) = \mathbf{k}(t)&\underbrace{\big((1-\frac{\lambda}{\beta})\boldsymbol{\gamma}^{(1)} - \frac{1}{\beta}\mathbf{e}^{(1)}\big)}_{\boldsymbol{\gamma}^{(2)}} \\ &- \widetilde{\mathbf{k}}(t)\underbrace{\big((1-\frac{\lambda}{\beta})\boldsymbol{\rho}^{(1)} + \frac{\lambda_{obs}}{\beta}\mathbf{H}^{(1)}\big)}_{\boldsymbol{\rho}^{(2)}}.\end{aligned} \tag{27}$$

Observing (26)–(27), we can see that only $\boldsymbol{\gamma}^{(n)}$ and $\boldsymbol{\rho}^{(n)}$ need to be updated iteratively till convergence. Let us denote the number of total iterations as n^*, the *optimal trajectory function* for the problem in (6) will be

$$\boldsymbol{\xi}^*(\cdot) = \mathbf{k}(\cdot)\boldsymbol{\gamma}^{(n^*)} - \widetilde{\mathbf{k}}(\cdot)\boldsymbol{\rho}^{(n^*)}, \tag{28}$$

which is capable of predicting the corresponding trajectory point (including joint position and velocity) at arbitrary time. The entire EKMP algorithm is summarized in Algorithm 1.

E. Special Cases of EKMP

We have derived EKMP under nonlinear hard constraints. Now, we consider the applications of EKMP to deal with some special cases.

Nonlinear equality constraints: we can use inequalities to guarantee equality constraints with a tiny approximation error $\epsilon > 0$, i.e., $g_{n,f}(\boldsymbol{\xi}(t_n)) = 0$ is enforced by $g_{n,f}(\boldsymbol{\xi}(t_n)) + \epsilon \geq 0$ and $-g_{n,f}(\boldsymbol{\xi}(t_n)) + \epsilon \geq 0$.

Adaptations towards desired points: an important feature in imitation learning is the adaptation of learned trajectories towards arbitrary desired points in terms of joint position and velocity, including start-, via- and end-points. Assuming that L desired points, denoted by $\{\bar{t}_l, \bar{\boldsymbol{\xi}}_l\}_{l=1}^L$ with $\bar{\boldsymbol{\xi}}_l = [\bar{\mathbf{q}}_l^\top \ \bar{\dot{\mathbf{q}}}_l^\top]^\top \in \mathcal{R}^{2\mathcal{O}}$, are required in a task. For each desired point $\{\bar{t}_l, \bar{\boldsymbol{\xi}}_l\}$ we can separately define constraints as

$$\begin{aligned} g_{l,f}(\boldsymbol{\xi}(t_l)) &= \mathbf{1}_f^\top(\boldsymbol{\xi}(t_l) - \bar{\boldsymbol{\xi}}_l) + \epsilon \geq 0, \\ g_{l,2\mathcal{O}+f}(\boldsymbol{\xi}(t_l)) &= -\mathbf{1}_f^\top(\boldsymbol{\xi}(t_l) - \bar{\boldsymbol{\xi}}_l) + \epsilon \geq 0 \end{aligned} \tag{29}$$

for $\forall f \in \{1, 2, \ldots, 2\mathcal{O}\}$, where $\epsilon > 0$ is used to control the adaptation precision, i.e., how precisely the adapted trajectory can go through the desired point $\bar{\boldsymbol{\xi}}_l$ at time $\bar{t}_l$. $\mathbf{1}_f \in \mathcal{R}^{2\mathcal{O}}$ is an indicative vector with all elements being zero except for its f-*th* element.

Linear constraints: we can simply write

$$g_{n,f}(\boldsymbol{\xi}(t_n)) = \boldsymbol{\theta}^\top\boldsymbol{\xi}(t_n) + b \geq 0, \tag{30}$$

where $\boldsymbol{\theta}$ and b are constant, representing coefficients of linear constraints. Note that a **plane constraint** can be ensured by $\boldsymbol{\theta}^\top\boldsymbol{\xi}(t_n) + b + \epsilon \geq 0$ and $-\boldsymbol{\theta}^\top\boldsymbol{\xi}(t_n) - b + \epsilon \geq 0$. In the case of **joint position limits**, for each joint i with motion range $[q_i^{min}, q_i^{max}]$, its position limit can be formulated as $\mathbf{1}_i^\top\boldsymbol{\xi}(t_n) - q_i^{min} \geq 0$ and $-\mathbf{1}_i^\top\boldsymbol{\xi}(t_n) + q_i^{max} \geq 0$. Similarly, **joint velocity limits** can also be imposed.

Algorithm 1: EKMP

Initialization
Set λ, β, λ_{obs}, $k(\cdot,\cdot)$ and define $\{\{\mathbf{g}_{n,f}\}_{n=1}^N\}_{f=1}^F$
Extract reference trajectory $\mathbf{D}_r$ from demonstrations
Calculate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ via (8), $\mathbf{K}$ via (20) and $\widetilde{\mathbf{K}}$
Calculate $\boldsymbol{\gamma}^{(0)}$ via (25) and $\boldsymbol{\xi}^{(0)} = \mathbf{K}\boldsymbol{\gamma}^{(0)}$
Calculate $\mathbf{H}^{(0)}, \mathbf{G}^{(0)}, \mathbf{Q}^{(0)}$ via (10), (15) and (17)
Optimize $\boldsymbol{\alpha}^{(0)}$ in (18) using QP and calculate $\mathbf{e}^{(0)}$
Set a maximal iteration number N^*, tolerance error ϵ
Set $\boldsymbol{\rho}^{(0)} = \mathbf{0}$, $iter = 0$
while $iter < N^*$ **do**
  $\boldsymbol{\gamma}^{(iter+1)} \leftarrow (1-\frac{\lambda}{\beta})\boldsymbol{\gamma}^{(iter)} - \frac{1}{\beta}\mathbf{e}^{(iter)}$
  $\boldsymbol{\rho}^{(iter+1)} \leftarrow (1-\frac{\lambda}{\beta})\boldsymbol{\rho}^{(iter)} + \frac{\lambda_{obs}}{\beta}\mathbf{H}^{(iter)}$
  Calculate $\boldsymbol{\xi}^{(iter+1)} = \mathbf{K}\boldsymbol{\gamma}^{(iter+1)} - \widetilde{\mathbf{K}}\boldsymbol{\rho}^{(iter+1)}$
  Calculate $\mathbf{H}^{(iter+1)}$, $\mathbf{G}^{(iter+1)}$, $\mathbf{Q}^{(iter+1)}$
  Optimize $\boldsymbol{\alpha}^{(iter+1)}$ and calculate $\mathbf{e}^{(iter+1)}$
  $\delta_\gamma \leftarrow ||\boldsymbol{\gamma}^{(iter+1)} - \boldsymbol{\gamma}^{(iter)}||$
  $\delta_\rho \leftarrow ||\boldsymbol{\rho}^{(iter+1)} - \boldsymbol{\rho}^{(iter)}||$
  if $||\delta_\gamma|| < \epsilon$ *and* $||\delta_\rho|| < \epsilon$ **then**
    break
  end
  $iter \leftarrow iter + 1$
end
Output: $\boldsymbol{\xi}^*(\cdot) = \mathbf{k}(\cdot)\boldsymbol{\gamma}^{(iter)} - \widetilde{\mathbf{k}}(\cdot)\boldsymbol{\rho}^{(iter)}$

III. Connections with State-of-the-Art Motion Planning Approaches

We now discuss the connections between EKMP and CHOMP [17] (Section III-A), GPMP [18] (Section III-B) and RKHS motion planning [19] (Section III-C).

A. Comparison with CHOMP

Since in practice the implementation of CHOMP updates discrete waypoints of a trajectory [19], we discretize the update rule of EKMP (i.e., (24)) to establish the connection. Supposing that we describe a trajectory by using its waypoints at a series of time steps $\{t_n\}_{n=1}^{N}$. By using (24), we update each waypoint as

$$\underbrace{\begin{bmatrix}\boldsymbol{\xi}(t_1)\\ \boldsymbol{\xi}(t_2)\\ \vdots\\ \boldsymbol{\xi}(t_N)\end{bmatrix}}_{\boldsymbol{\xi}} = (1-\frac{\lambda}{\beta})\underbrace{\begin{bmatrix}\boldsymbol{\xi}^c(t_1)\\ \boldsymbol{\xi}^c(t_2)\\ \vdots\\ \boldsymbol{\xi}^c(t_N)\end{bmatrix}}_{\boldsymbol{\xi}^c} - \frac{1}{\beta}\big(\mathbf{K}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\xi}^c-\boldsymbol{\mu})+\lambda_{obs}\widetilde{\mathbf{K}}\mathbf{H}^c-\mathbf{K}\mathbf{G}^c\boldsymbol{\alpha}^c\big), \tag{31}$$

which can be further simplified by letting $\widetilde{\mathbf{K}}=\mathbf{K}$, yielding

$$\boldsymbol{\xi} = (1-\frac{\lambda}{\beta})\boldsymbol{\xi}^c-\frac{1}{\beta}\mathbf{K}\big(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\xi}^c-\boldsymbol{\mu})+\lambda_{obs}\mathbf{H}^c-\mathbf{G}^c\boldsymbol{\alpha}^c\big). \tag{32}$$

If we let $\lambda=0$, $\boldsymbol{\mu}=\mathbf{0}$ and $\boldsymbol{\alpha}^c=\mathbf{0}$, (32) will become

$$\boldsymbol{\xi} = \boldsymbol{\xi}^c - \frac{1}{\beta}\underbrace{\mathbf{K}\big(\boldsymbol{\Sigma}^{-1}\boldsymbol{\xi}^c+\lambda_{obs}\mathbf{H}^c\big)}_{update\ term}, \tag{33}$$

which shares a similar form with the update rule in CHOMP[3]. However, in CHOMP the coefficient of $\boldsymbol{\xi}^c$ in the update term is $\mathbf{K}\mathbf{K}^{-1}=\mathbf{I}$ rather than $\mathbf{K}\boldsymbol{\Sigma}^{-1}$. Moreover, $\mathbf{K}$ in CHOMP is defined using the finite difference matrix in order to ensure the smoothness of a trajectory, while $\mathbf{K}$ (i.e., (20)) in EKMP is a kernel matrix. Note that hard equality constraints were addressed in [17], but without considering nonlinear inequality constraints.

B. Comparison with GPMP

Let us continue with (32) and if we let $\lambda=0$ and neglect nonlinear hard constraints (i.e., $\boldsymbol{\alpha}^c=\mathbf{0}$), (32) can be rewritten as

$$\boldsymbol{\xi} = \boldsymbol{\xi}^c - \frac{1}{\beta}\underbrace{\mathbf{K}\big(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\xi}^c-\boldsymbol{\mu})+\lambda_{obs}\mathbf{H}^c\big)}_{update\ term}, \tag{34}$$

which resembles the update rule in GPMP[4]. However, in GPMP the coefficient of $\boldsymbol{\xi}^c-\boldsymbol{\mu}$ in the update term is $\mathbf{K}\mathbf{K}^{-1}=\mathbf{I}$ rather than $\mathbf{K}\boldsymbol{\Sigma}^{-1}$. Specifically, $\mathbf{K}$ in GPMP is obtained from a stochastic differential equation and has a specific form related to state transition matrix as well as noise distribution. In contrast, $\mathbf{K}$ in EKMP is a generic kernel function (e.g., Gaussian kernel and periodic kernel), which can be chosen depending on task requirements.

The advantage of using $\boldsymbol{\Sigma}^{-1}$ to weigh $\boldsymbol{\xi}^c-\boldsymbol{\mu}$ in EKMP is that the importance of demonstrations can be incorporated into the process of updating trajectories, i.e., datapoints with small covariance (large consistency implies high importance) will have larger impact on trajectory update than those with large covariance. Similar insights have been discussed in previous works but in the context of unconstrained imitation learning, e.g., [6], [14]. Note that GPMP was extended in [21] to encapsulate the prior distribution of demonstrations, where hard constraints were also neglected.

[3]CHOMP also considered an additional vector dealing with boundary conditions in its update rule.

[4]A projection matrix imposed on the gradient of the obstacle avoidance cost was also considered in GPMP.

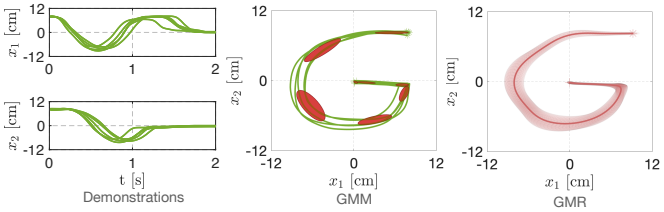

Fig. 1. The modeling of demonstrations (green curves) using GMM and GMR, where red ellipses denote the Gaussian components in GMM, the red curve and the shaded area respectively correspond to the mean and covariance of demonstrations.

In addition, in terms of the trajectory update form, [18], [21] optimizes trajectory waypoints while in EKMP, as shown in (24), the trajectory function is updated. Specifically, unlike [18], [21], in EKMP the derivative relationship between the predicted joint position and velocity can be ensured over the course of optimizing (6), since the update rule (24) is derived from the parametric form (2) and the corresponding derivative relationship is encoded in $\mathbf{k}(\cdot,\cdot)$ (i.e., (21)).

C. Comparison with RKHS Motion Planning

Since RKHS motion planning [19] directly updates trajectory function, we compare it with the update rule in (24). In fact, if we do not consider imitation learning (i.e., setting $\boldsymbol{\Sigma}^{-1}(\boldsymbol{\xi}^c-\boldsymbol{\mu})=\mathbf{0}$) and replace nonlinear constraints with joint position and velocity limits (i.e., letting $F=2\mathcal{O}$ and $\mathbf{G}=-\mathbf{I}$), (24) will become

$$\boldsymbol{\xi}(\cdot) = (1-\frac{\lambda}{\beta})\boldsymbol{\xi}^c(\cdot) - \frac{1}{\beta}\big(\lambda_{obs}\widetilde{\mathbf{k}}(\cdot)\mathbf{H}^c+\mathbf{k}(\cdot)\boldsymbol{\alpha}^c\big), \tag{35}$$

which has the same form as the update rule in [19], except that $\mathbf{k}$ and $\widetilde{\mathbf{k}}$ have different definitions due to the simultaneous learning of $\mathbf{q}$ and $\dot{\mathbf{q}}$ in EKMP, and $\mathbf{H}^c$ includes a zero vector due to the optimization of $\dot{\mathbf{q}}$. Note that the constraints of desired starting and ending joint positions in [19] can be handled by inequality constraints (see Section II-E).

Summary : EKMP can be viewed as a generalization of [17]–[19] towards imitation learning as well as nonlinear hard (in particular inequality) constraints. Moreover, EKMP can learn both joint position and velocity simultaneously while maintaining the corresponding derivative relationship even under nonlinear hard constraints and obstacle avoidance requirement. Specifically, in contrast to [17], [18] that optimize trajectory waypoints, EKMP updates the trajectory function, sharing the same advantages as [19], including fast convergence and smooth trajectories. For the discussion of advantages of optimizing trajectory function over trajectory waypoints, please refer to [19].

IV. Evaluations

We evaluate EKMP through several examples, including 2-D writing tasks (Section IV-A), as well as reaching tasks

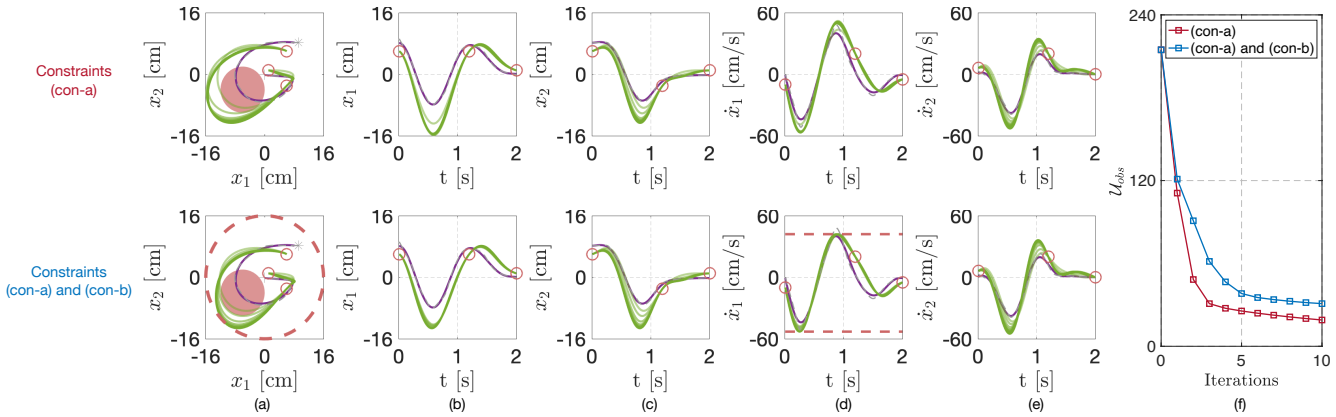

Fig. 2. Evaluations of EKMP under different constraints. (*a*)–(*e*): Dashed gray curves are retrieved by GMR, where '$*$' and '+' denote the start-point and end-point, respectively. Purple curves represent trajectories (obtained by vanilla KMP) used to initialize EKMP. Evolution from light green to dark green corresponds to the updated direction over 10 iterations. Small red circles plot desired points, corresponding to the constraints **(con-a)**. The red solid circle in (*a*) depicts the obstacle. The dashed red circle in (*a*) and dashed lines in (*d*) correspond to the constraints **(con-b)**. (*f*): Obstacle avoidance costs under different constraints.

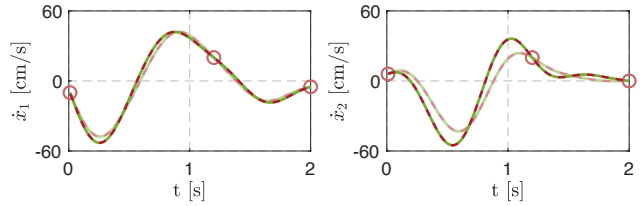

Fig. 3. Derivative of the predicted positions (dashed red curves) and the predicted velocities (green curves) via EKMP after one iteration (light green and red) and ten iterations (dark green and red) when considering both the constraints **(con-a)** and **(con-b)** simultaneously.

in a 7-DoF robotic arm (Section IV-B). The Gaussian kernel $k(t_i, t_j) = \exp(-k_h(t_i - t_j)^2)$ is used in all examples.

A. Writing Task

We here consider the task of writing a letter 'G', where trajectories are planned in 2-D space. Given five demonstrations consisting of time and 2-D position, as shown in Fig. 1 (*left* graph), we use GMM and GMR to model their distribution and subsequently extract a probabilistic reference trajectory, see Fig. 1 (*middle* and *right* graphs). Two groups of constraints are studied in this task:
(con-a): three desired points in terms of 2-D position and 2-D velocity, depicted by the small red circles in Fig. 2;
(con-b): a circle constraint and two velocity constraints

$$\begin{aligned} x_1^2 + x_2^2 &\leq 16^2, \\ -53 \leq \dot{x}_1 &\leq 42. \end{aligned} \tag{36}$$

We first evaluate EKMP under the constraints **(con-a)** and later under both **(con-a)** and **(con-b)**. In both evaluations, an obstacle with a radius $r = 6cm$ located at $[-6.0 \ \ -4.0]^\top cm$ should be avoided. We use the cost function $c(\cdot)$ from [17] for evaluating body points, i.e.,

$$c(d) = \begin{cases} r - d + \dfrac{1}{2}\epsilon_d, & d \leq r, \\ \dfrac{1}{2\epsilon_d}(d - r - \epsilon_d)^2, & 0 < d - r \leq \epsilon_d, \\ 0, & otherwise, \end{cases} \tag{37}$$

where d is the distance from a body point to the obstacle's center, ϵ_d denotes a safety margin and is set as $\epsilon_d = 4cm$. Vanilla KMP and EKMP use the same parameters $\lambda = 0.01$ and $k_h = 4$. Other relevant parameters used by EKMP are $\beta = 340$, $\lambda_{obs} = 110$, $N = M = 200$.

Evaluations are provided in Fig. 2, showing that EKMP is capable of maintaining the shape of demonstrations and avoiding the obstacle while satisfying various constraints after very few iterations. The cost of obstacle avoidance is presented in Fig. 2(*f*), where fast decrease of obstacle avoidance cost is achieved via EKMP. Figure 3 plots derivative of the predicted positions and the predicted velocities. We can see that the derivative relationship is indeed strictly ensured over the course of optimizing trajectories via EKMP.

B. Evaluations on a Robotic Arm

Now, we employ EKMP to plan joint trajectories for a 7-DoF Kinova Gen3 robotic arm so that reaching tasks can be accomplished while avoiding collisions of robot links with the obstacle and fulfilling various hard constraints. We start with five demonstrations (comprising time and joint position) for a reaching task, as shown in Fig. 4, where the GMM modeling of demonstrations is also presented. We define two groups of constraints:
(con-1): two desired points in terms of joint position (7-D) and joint velocity (7-D), as depicted by the red circles in Fig. 5(*a*)–(*c*) and (*e*)–(*g*);
(con-2):

$$\begin{aligned} f(\mathbf{q}_t)|_{t=6} &= [0.60 \ \ -0.15 \ \ 0.60]^\top, \\ f(\mathbf{q}_t)_z &\geq 0.4, \end{aligned} \tag{38}$$

where $f(\cdot)$ represents the forward kinematics of the robot arm and calculates Cartesian position of the robot's end-effector. $f(\mathbf{q}_t)_z \geq 0.4$ demands that the robot's end-effector should always stay above a horizontal plane with a height of $0.4m$. Note that $f(\cdot)$ is a nonlinear function of joint position and we here directly optimize joint trajectories.

The relevant parameters are set as $\lambda = 0.01$, $k_h = 0.2$, $\beta = 700$, $\lambda_{obs} = 120$, the radius of the obstacle is $r = 0.1m$,

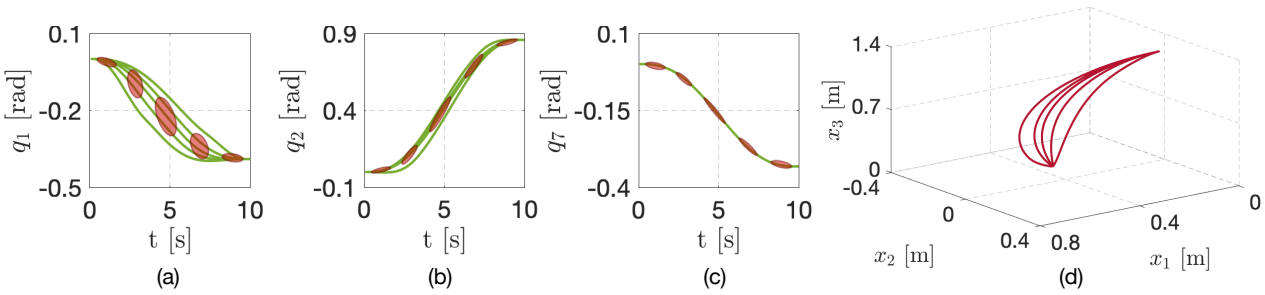

Fig. 4. Demonstrations of the reaching task. (*a*)–(*c*) correspond to the first, second and seventh joints, respectively. (*d*) plots the corresponding Cartesian trajectories of the robot's end-effector.

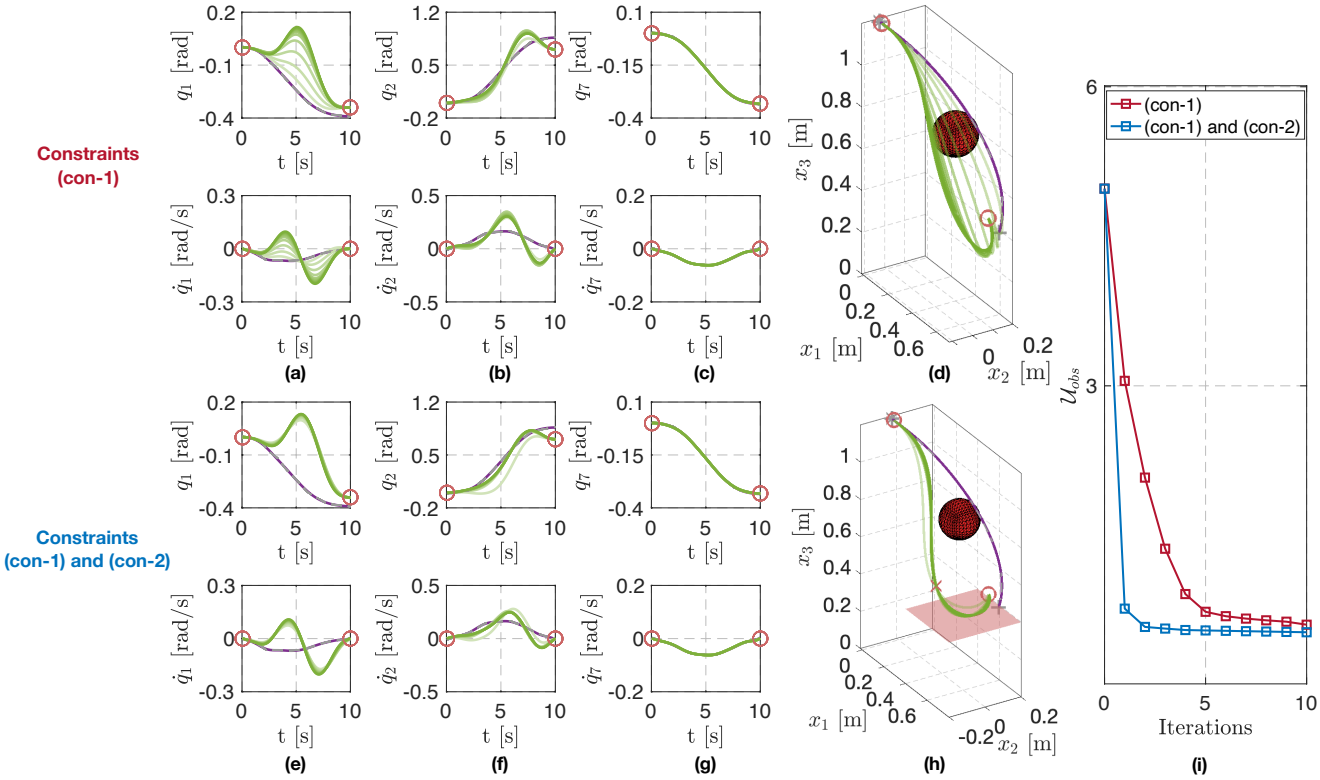

Fig. 5. The planning of joint trajectories via EKMP under different constraints. In (*a*)–(*c*) and (*e*)–(*g*), dashed gray curves are retrieved by GMR; Purple curves depict trajectories generated by vanilla KMP, which are used to initialize EKMP; Evolution from light green to dark green shows the updated trajectories over 10 iterations, where the color changes from light to dark as the iteration number increases; Red circles represent desired point constraints **(con-1)**. In (*d*) and (*h*), gray, purple and green curves plot the corresponding end-effector trajectories of the planned joint trajectories; Red circles depict the end-effector positions corresponding to the desired joint positions; '$*$' and '+' denote the start-point and end-point of the GMR trajectory, respectively; The red solid ball depicts the obstacle. In (h), '$\times$' depicts the desired Cartesian position and the red plane plots the z-direction limit, both corresponding to the constraints **(con-2)** in (38). (*f*): Obstacle avoidance costs under different constraints.

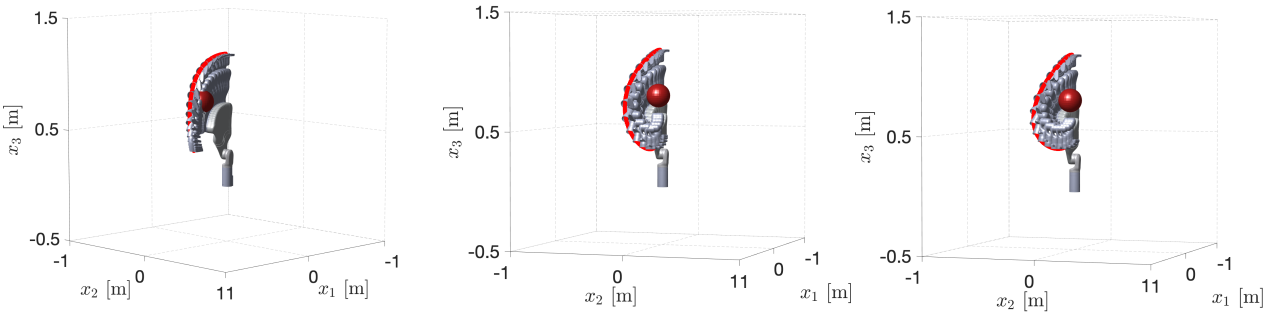

Fig. 6. Different planned trajectories executed on a 7-DoF robotic arm. *Left* graph shows the execution of the joint trajectory generated by vanilla KMP. *Middle* graph corresponds to the optimized joint trajectory after 10 iterations under constraints **(con-1)** while *right* graph is under both **(con-1)** and **(con-2)**.

$\epsilon_d = 0.15m$, $N = M = 21$. Vanilla KMP also uses the same parameters. It is worth mentioning that trajectories with a length of 200 will be generated, which is far longer than N. Since we only impose hard constraints over 21 points in order to reduce the computational cost in each iteration and thus some points from the generated trajectory may not strictly respect the constraints. However, this issue can be trivially solved by strengthening the constraints and in our evaluation the constraint $f(\mathbf{q}_t)_z \geq 0.40 + \epsilon_z$ with $\epsilon_z = 2 \times 10^{-3}$ imposed over 21 points can ensure that $f(\mathbf{q}_t)_z \geq 0.40$ is strictly fulfilled over the generated trajectories (after 9 iterations) consisting of 200 datapoints.

Figure 5 depicts trajectory optimization within 10 iterations using EKMP. From Fig. 5(*a*)–(*c*) and (*e*)–(*g*), we can see that all optimized trajectories (green curves) satisfy the desired point constraints (marked by red circles) defined in **(con-1)**. Specifically, in Fig. 5(*h*), the optimized trajectory at the last iteration goes through the desired Cartesian position (marked by '$\times$') and obeys the z-direction limit (depicted by the red plane), therefore respecting the constraints defined in **(con-2)**. Fast decrease of obstacle avoidance cost is also achieved, as shown in Fig. 5(*i*), where the cost is calculated using the generated joint trajectory (i.e., 200 datapoints).

For the sake of clear observation, Fig. 6 illustrates planned joint trajectories through executing them on the robotic arm. In contrast to vanilla KMP where robot links collide with the obstacle (Fig. 6(*a*)), joint trajectories generated by EKMP are capable of avoiding the obstacle under different hard constraints, see Fig. 6(*middle* and *right* graphs) where the joint trajectories correspond to the ones obtained after 10 iterations. Note that in our evaluations, under **(con-1)** the planned joint trajectories after 3 iterations can avoid the obstacle, while under both **(con-1)** and **(con-2)**, it only needs 1 iteration. This also provides an interesting insight, i.e., proper constraints in Cartesian space could be of help for fast planning of collision-free joint trajectories.

V. Conclusions

We have extended KMP towards a more flexible framework capable of dealing with a bunch of key problems in imitation learning, including learning and adaptation, obstacle avoidance, linear and nonlinear hard constraints. Several evaluations including 2-D writing tasks and joint trajectory planning for reaching tasks verified the performance of our framework. While other constraints such as stiffness and orientation are also important in many scenarios, it would be of interest to address them within the current framework.

Acknowledgement

The author would like to thank Dr. Songyan Xin from the University of Edinburgh for the helpful discussion about evaluations on the robotic arm, and thank Dr. João Silvério from Idiap Research Institute for his comments on this paper.

References

[1] F. Stulp, E. Theodorou, J. Buchli, and S. Schaal, "Learning to grasp under uncertainty," in *IEEE International Conference on Robotics and Automation*, 2011.

[2] M. Howard, S. Klanke, M. Gienger, C. Goerick, and S. Vijayakumar, "A novel method for learning policies from variable constraint data," *Autonomous Robots*, 2009.

[3] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, 2010.

[4] J. Silvério, Y. Huang, F. Abu-Dakka, L. Rozo, and D. Caldwell, "Uncertainty-aware imitation learning using kernelized movement primitives," in *International Conference on Intelligent Robots and Systems*, 2019.

[5] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural Computation*, 2013.

[6] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, 2016.

[7] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *IEEE-RAS International Conference on Humanoid Robots*, 2008.

[8] D. Koert, G. Maeda, R. Lioutikov, G. Neumann, and J. Peters, "Demonstration based trajectory optimization for generalizable robot motions," in *IEEE-RAS 16th International Conference on Humanoid Robots*, 2016.

[9] Y. Huang, J. Silvério, L. Rozo, and D. G. Caldwell, "Generalized task-parameterized skill learning," in *IEEE International Conference on Robotics and Automation*, 2018.

[10] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann, "Guiding trajectory optimization by demonstrated distributions," *IEEE Robotics and Automation Letters*, 2017.

[11] R. A. Shyam, P. Lightbody, G. Das, P. Liu, S. Gomez-Gonzalez, and G. Neumann, "Improving local trajectory optimisation using probabilistic movement primitives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[12] Y. Huang and D. G. Caldwell, "A linearly constrained nonparametric framework for imitation learning," in *IEEE International Conference on Robotics and Automation*, 2020.

[13] M. Saveriano and D. Lee, "Learning barrier functions for constrained motion planning with dynamical systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[14] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *The International Journal of Robotics Research*, 2019.

[15] J. Ding, X. Xiao, N. Tsagaraki, and Y. Huang, "Robust gait synthesis combining constrained optimization and imitation learning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[16] C. Zou, R. Huang, H. Cheng, and J. Qiu, "Learning gait models with varying walking speeds," *IEEE Robotics and Automation Letters*, 2020.

[17] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, 2013.

[18] M. Mukadam, X. Yan, and B. Boots, "Gaussian process motion planning," in *IEEE International Conference on Robotics and Automation*, 2016.

[19] Z. Marinho, B. Boots, A. Dragan, A. Byravan, G. J. Gordon, and S. Srinivasa, "Functional gradient motion planning in reproducing kernel hilbert spaces," in *Robotics: Science and Systems*, 2016.

[20] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *IEEE International Conference on Robotics and Automation*, 2011.

[21] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Towards robust skill generalization: Unifying learning from demonstration and motion planning," in *Conference on Robot Learning*, 2017.

[22] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in neural information processing systems*, 2013.

[23] A. Varol, M. Salzmann, P. Fua, and R. Urtasun, "A constrained latent variable model," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.