Proceedings Paper:
Huang, Y and Caldwell, DG (2020) A Linearly Constrained Nonparametric Framework for Imitation Learning. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). International Conference on Robotics and Automation ICRA 2020, 31 May - 31 Aug 2020, Paris, France. IEEE , pp. 4400-4406. ISBN 978-1-7281-7396-2

https://doi.org/10.1109/ICRA40945.2020.9196821

A Linearly Constrained Nonparametric Framework for Imitation Learning

Yanlong Huang and Darwin G. Caldwell

***Abstract*— In recent years, a myriad of advanced results have been reported in the community of imitation learning, ranging from parametric to non-parametric, probabilistic to non-probabilistic and Bayesian to frequentist approaches. Meanwhile, ample applications (e.g., grasping tasks and human-robot collaborations) further show the applicability of imitation learning in a wide range of domains. While numerous literature is dedicated to the learning of human skills in unconstrained environments, the problem of learning constrained motor skills, however, has not received equal attention. In fact, constrained skills exist widely in robotic systems. For instance, when a robot is demanded to write letters on a board, its end-effector trajectory must comply with the plane constraint from the board. In this paper, we propose *linearly constrained kernelized movement primitives* (LC-KMP) to tackle the problem of imitation learning with linear constraints. Specifically, we propose to exploit the probabilistic properties of multiple demonstrations, and subsequently incorporate them into a linearly constrained optimization problem, which finally leads to a non-parametric solution. In addition, a connection between our framework and the classical model predictive control is provided. Several examples including simulated writing and locomotion tasks are presented to show the effectiveness of our framework.**

I. Introduction

In the community of robot learning, a vital goal is to endow robots with learning capabilities. In this line, *imitation learning* [1], [2], also referred to as *programming by demonstration* or *learning from demonstration* [3], emerges as an important research direction due to its nature and user-friendly features. Remarkably, many imitation learning approaches, such as dynamical movement primitives (DMP) [4], task-parameterized Gaussian mixture model (GMM) [5] and kernelized movement primitives (KMP) [6], have been developed. In addition, imitation learning has achieved great success in ample scenarios, e.g., reaching [7], grasping [8], striking [9] and painting [10] tasks.

However, most of the aforementioned works focus on learning human skills while the possible external or internal constraints are ignored. As we know, robots often encounter various constraints in dynamical environments. To take the wiping task as an example, the robot end-effector trajectory should always lie above the table since trajectories under the table are physically infeasible. Besides, the end-effector trajectory must obey the plane constraint of the table in order to clean the table successfully. It is also worth pointing out that, during the wiping process, robot joint trajectories must comply with the joint limits.

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

Darwin G. Caldwell is with Istituto Italiano di Tecnologia, Via Morego 30, Genoa 16163, Italy. darwin.caldwell@iit.it

In order to cope with the constraints that are imposed to robots, a few approaches have been proposed. For example, the hard (i.e., equality) constraints were studied in [11], [12] and joint limit avoidance[1] was treated in [13], [14]. In contrast to these approaches, we aim at developing a generic framework which can be employed to learn and generalize human demonstrations to new situations, and meanwhile address a variety of constraints (i.e., equality and inequality constraints) that commonly arise in practice. In summary, the new framework should be capable of

(i) inheriting the key features of imitation learning (e.g., learning multiple demonstrations, reproduction as well as adaptation towards via-/end- points in terms of position and velocity);

(ii) taking into account arbitrary linear equality (e.g., plane constraint) and inequality constraints (e.g., a linear combination of action components should be larger or smaller than a predefined value).

To do so, we propose to build the linearly constrained imitation learning framework on the top of our previous work KMP [6], since KMP provides us with most of the crucial features in imitation learning, including the learning and adaptation of human skills associated with time input and high-dimensional inputs. Specifically, we first exploit the probabilistic properties of multiple demonstrations in Section II. Subsequently, we present the constrained framework in Section III. Since the classical model predictive control (MPC) can address the constrained trajectory optimization problem, we provide a connection between our framework and MPC in Section IV. Finally, we evaluate the proposed approach through several simulated examples in Section V.

II. Exploiting Probabilistic Features of Multiple Demonstrations

Following the spirit of probabilistic approaches [6], [15], [16], [17], [18], we model demonstrations from a probabilistic perspective. Let us assume that we have access to M demonstrations $\mathbf{D} = \{\{t_{n,m}, \boldsymbol{\xi}_{n,m}, \dot{\boldsymbol{\xi}}_{n,m}\}_{n=1}^{N}\}_{m=1}^{M}$, where N denotes the trajectory length, $\boldsymbol{\xi} \in \mathbb{R}^{\mathcal{O}}$ and $\dot{\boldsymbol{\xi}}$ respectively correspond to the output and its first-order derivative. For the sake of brevity, we write $\boldsymbol{\eta} = [\boldsymbol{\xi}^\top \dot{\boldsymbol{\xi}}^\top]^\top$. Then, we can employ GMM to model the joint probability distribution $\mathcal{P}(t, \boldsymbol{\eta})$, leading to

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

[1] From an optimization perspective, joint limits can be viewed as a special case of inequality constraints.

with π_c, $\boldsymbol{\mu}_c = \begin{bmatrix} \boldsymbol{\mu}_{t,c} \\ \boldsymbol{\mu}_{\eta,c} \end{bmatrix}$ and $\boldsymbol{\Sigma}_c = \begin{bmatrix} \boldsymbol{\Sigma}_{tt,c} & \boldsymbol{\Sigma}_{t\eta,c} \\ \boldsymbol{\Sigma}_{\eta t,c} & \boldsymbol{\Sigma}_{\eta\eta,c} \end{bmatrix}$ being the prior probability, mean and covariance of the c-th Gaussian component, respectively[2]. Here, C represents the number of Gaussian components. After that, we resort to Gaussian mixture regression[3] (GMR) to retrieve a *probabilistic reference trajectory* $\{t_n, \hat{\boldsymbol{\eta}}_n\}_{n=1}^N$ with $\hat{\boldsymbol{\eta}}_n | t_n \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n)$, where the input sequence $\{t_n\}_{n=1}^N$ (e.g., equal interval sequence) spans the whole input space.

We now denote $\mathbf{D}_r = \{t_n, \hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n\}_{n=1}^N$. In fact, $\mathbf{D}_r$ encapsulates the probabilistic features of demonstrations $\mathbf{D}$, since it estimates the means and covariances of demonstrations over various key input points. In the next section, we will exploit $\mathbf{D}_r$ and propose the linearly constrained imitation learning framework in details.

III. Linearly Constrained Nonparametric Framework for Imitation Learning

Let us first write $\boldsymbol{\eta}(t)$ using a parametric form[4], i.e.,

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

with

$$\boldsymbol{\Theta}(t) = \begin{bmatrix} \boldsymbol{\varphi}(t) & \mathbf{0} & \cdots & \mathbf{0} & \dot{\boldsymbol{\varphi}}(t) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\varphi}(t) & \cdots & \mathbf{0} & \mathbf{0} & \dot{\boldsymbol{\varphi}}(t) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\varphi}(t) & \mathbf{0} & \mathbf{0} & \cdots & \dot{\boldsymbol{\varphi}}(t) \end{bmatrix}, \tag{3}$$

where $\boldsymbol{\varphi}(\cdot) \in \mathbb{R}^B$ represents a basis function vector and $\mathbf{w} \in \mathbb{R}^{B\mathcal{O}}$ denotes the *unknown* parameter vector.

Formally, we formulate the problem of imitation learning with linear constraints as a constrained optimization problem

$$\begin{aligned} \underset{\mathbf{w}}{\textbf{argmax}} \quad & \sum_{n=1}^{N} \mathcal{P}\left(\boldsymbol{\eta}(t_n) | \hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n\right) \\ \textbf{s.t.} \quad & \begin{matrix} \mathbf{g}_{n,1}^\top \boldsymbol{\eta}(t_n) & \geq c_{n,1} \\ \mathbf{g}_{n,2}^\top \boldsymbol{\eta}(t_n) & \geq c_{n,2} \\ \vdots & \\ \mathbf{g}_{n,F}^\top \boldsymbol{\eta}(t_n) & \geq c_{n,F} \end{matrix}, \ \forall n \in \{1, 2, \ldots, N\}, \end{aligned} \tag{4}$$

where we use $\mathbf{g}_{n,f} \in \mathbb{R}^{2\mathcal{O}}$ and $\mathbf{c}_{n,f} \in \mathbb{R}$ to characterize the f-th linear constraint over $\boldsymbol{\xi}(t_n)$ and $\dot{\boldsymbol{\xi}}(t_n)$, and F to denote the number of constraints.

With probability calculations and logarithm transformation, the constrained maximization problem in (4) can be rewritten as

$$\begin{aligned} &\underset{\mathbf{w}}{\textbf{argmin}} \sum_{n=1}^{N} \frac{1}{2}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w} - \hat{\boldsymbol{\mu}}_n)^\top \hat{\boldsymbol{\Sigma}}_n^{-1}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w} - \hat{\boldsymbol{\mu}}_n) + \frac{1}{2}\lambda \mathbf{w}^\top \mathbf{w} \\ &\textbf{s.t.} \ \ \mathbf{g}_{n,f}^\top \boldsymbol{\eta}(t_n) \geq c_{n,f}, \forall f \in \{1, \ldots, F\}, \forall n \in \{1, \ldots, N\}. \end{aligned} \tag{5}$$

[2]Note that vector notations $\mathbf{u}_{t,c}$ and $\boldsymbol{\Sigma}_{tt,c}$ are used to represent scalars.

[3]More details about GMM/GMR can be found in [6], [19], [20].

[4]Similar parametric form was exploited in [4], [21].

Here, the regularized term $\frac{1}{2}\lambda \mathbf{w}^\top \mathbf{w}$ with $\lambda > 0$ is added into (5) so as to alleviate the over-fitting issue[5].

We can solve (5) by introducing Lagrange multipliers $\alpha_{n,f} \geq 0$, leading to the Lagrange function

$$\begin{aligned} L(\mathbf{w}, \boldsymbol{\alpha}) = &\sum_{n=1}^{N} \frac{1}{2}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w} - \hat{\boldsymbol{\mu}}_n)^\top \hat{\boldsymbol{\Sigma}}_n^{-1}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w} - \hat{\boldsymbol{\mu}}_n) \\ &+ \frac{1}{2}\lambda \mathbf{w}^\top \mathbf{w} - \sum_{n=1}^{N}\sum_{f=1}^{F} \alpha_{n,f}(\mathbf{g}_{n,f}^\top \boldsymbol{\Theta}(t_n)^\top \mathbf{w} - c_{n,f}) \end{aligned} \tag{6}$$

with $\boldsymbol{\alpha} = [\alpha_{1,1}, \alpha_{1,2}, \ldots, \alpha_{1,F} \ \ldots\ldots \ \alpha_{N,1}, \alpha_{N,2}, \ldots, \alpha_{N,F}]^\top$. By calculating the derivative $\frac{\partial L(\mathbf{w}, \boldsymbol{\alpha})}{\partial \mathbf{w}}$ and setting as 0, we have[6]

$$\begin{aligned} \mathbf{w}^* = &(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^\top + \lambda\mathbf{I})^{-1}(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \boldsymbol{\Phi}\overline{\mathbf{G}}\boldsymbol{\alpha}) \\ = &\boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{\Sigma})^{-1}(\boldsymbol{\mu} + \boldsymbol{\Sigma}\overline{\mathbf{G}}\boldsymbol{\alpha}), \end{aligned} \tag{7}$$

where

$$\begin{aligned} &\boldsymbol{\Phi} = [\boldsymbol{\Theta}(t_1) \ \boldsymbol{\Theta}(t_2) \ \cdots \ \boldsymbol{\Theta}(t_N)], \\ &\boldsymbol{\Sigma} = \text{blockdiag}(\hat{\boldsymbol{\Sigma}}_1, \ \hat{\boldsymbol{\Sigma}}_2, \ \ldots, \ \hat{\boldsymbol{\Sigma}}_N), \\ &\boldsymbol{\mu} = [\hat{\boldsymbol{\mu}}_1^\top \ \hat{\boldsymbol{\mu}}_2^\top \ \cdots \ \hat{\boldsymbol{\mu}}_N^\top]^\top, \\ &\mathbf{G}_n = [\mathbf{g}_{n,1} \ \mathbf{g}_{n,2} \ \cdots \ \mathbf{g}_{n,F}], \forall n \in \{1, 2, \ldots, N\}, \\ &\overline{\mathbf{G}} = \text{blockdiag}(\mathbf{G}_1, \ \mathbf{G}_2, \ \ldots, \ \mathbf{G}_N). \end{aligned} \tag{8}$$

Furthermore, substituting the optimal $\mathbf{w}^*$ into (6) gives

$$\widetilde{L}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \overline{\mathbf{G}}^\top \boldsymbol{\Sigma}\mathcal{A}\boldsymbol{\Sigma}\overline{\mathbf{G}}\boldsymbol{\alpha} + (2\boldsymbol{\mu}^\top \mathcal{A}\boldsymbol{\Sigma}\overline{\mathbf{G}} + \overline{\mathbf{C}}^\top)\boldsymbol{\alpha} + const \tag{9}$$

with

$$\begin{aligned} &\mathbf{C}_n = [c_{n,1} \ c_{n,2} \ \ldots \ c_{n,F}]^\top, \forall n \in \{1, 2, \ldots, N\}, \\ &\overline{\mathbf{C}} = [\mathbf{C}_1^\top \ \mathbf{C}_2^\top \ldots \mathbf{C}_N^\top]^\top, \\ &\mathcal{A} = -\frac{1}{2}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{\Sigma})^{-1}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{\Phi}^\top\boldsymbol{\Phi}) \\ &\qquad\qquad (\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{\Sigma})^{-1}. \end{aligned} \tag{10}$$

Similarly to KMP, we employ the well-known kernel trick $\boldsymbol{\varphi}(t_i)^\top \boldsymbol{\varphi}(t_j) = k(t_i, t_j)$, with $k(\cdot, \cdot)$ being a kernel function, to (3), giving

$$\mathbf{k}(t_i, t_j) = \boldsymbol{\Theta}(t_i)^\top \boldsymbol{\Theta}(t_j) = \begin{bmatrix} k_{tt}(i,j)\mathbf{I}_\mathcal{O} & k_{td}(i,j)\mathbf{I}_\mathcal{O} \\ k_{dt}(i,j)\mathbf{I}_\mathcal{O} & k_{dd}(i,j)\mathbf{I}_\mathcal{O} \end{bmatrix}, \tag{11}$$

where[7] $k_{tt}(i,j) = k(t_i, t_j), k_{td}(i,j) = \frac{k(t_i, t_j+\delta) - k(t_i, t_j)}{\delta}, k_{dt}(i,j) = \frac{k(t_i+\delta, t_j) - k(t_i, t_j)}{\delta}, k_{dd}(i,j) = \frac{k(t_i+\delta, t_j+\delta) - k(t_i+\delta, t_j) - k(t_i, t_j+\delta) + k(t_i, t_j)}{\delta^2}$. Subsequently, combined with the definition of $\boldsymbol{\Phi}$ in (8), we have the kernel matrix

$$\mathbf{K} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi} = \begin{bmatrix} \mathbf{k}(t_1, t_1) & \mathbf{k}(t_1, t_2) & \cdots & \mathbf{k}(t_1, t_N) \\ \mathbf{k}(t_2, t_1) & \mathbf{k}(t_2, t_2) & \cdots & \mathbf{k}(t_2, t_N) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{k}(t_N, t_1) & \mathbf{k}(t_N, t_2) & \cdots & \mathbf{k}(t_N, t_N) \end{bmatrix}. \tag{12}$$

[5]This treatment has been widely adopted in many regressions, e.g., kernel ridge regression [22], [23].

[6]*Woodbury identity* is used [24]: if $\mathbf{P} \succ 0$ and $\mathbf{R} \succ 0$, $(\mathbf{P}^{-1} + \mathbf{B}^\top\mathbf{R}^{-1}\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{R}^{-1} = \mathbf{P}\mathbf{B}^\top(\mathbf{B}\mathbf{P}\mathbf{B}^\top + \mathbf{R})^{-1}$.

[7]In order to facilitate the kernel application, we approximate $\dot{\boldsymbol{\varphi}}(t) = \frac{\boldsymbol{\varphi}(t+\delta) - \boldsymbol{\varphi}(t)}{\delta}$, with a small constant $\delta > 0$.

Thus, we can rewrite $\mathcal{A}$ in (10) as

$$\mathcal{A} = -\frac{1}{2}(\mathbf{K}+\lambda\boldsymbol{\Sigma})^{-1}(\mathbf{K}\boldsymbol{\Sigma}^{-1}\mathbf{K}+\lambda\mathbf{K})(\mathbf{K}+\lambda\boldsymbol{\Sigma})^{-1}. \quad (13)$$

Let us now revisit the function $\widetilde{L}(\boldsymbol{\alpha})$ of Lagrange multipliers in (9), and denote

$$\begin{aligned}\mathcal{B}_1 &= \overline{\mathbf{G}}^{\top}\boldsymbol{\Sigma}\mathcal{A}\boldsymbol{\Sigma}\overline{\mathbf{G}},\\ \mathcal{B}_2 &= 2\boldsymbol{\mu}^{\top}\mathcal{A}\boldsymbol{\Sigma}\overline{\mathbf{G}}+\overline{\mathbf{C}}^{\top}.\end{aligned} \quad (14)$$

Thus, we can tackle the problem of finding optimal Lagrange multipliers $\boldsymbol{\alpha}$ through maximizing

$$\begin{aligned}\widetilde{L}(\boldsymbol{\alpha}) =& \boldsymbol{\alpha}^{\top}\mathcal{B}_1\boldsymbol{\alpha}+\mathcal{B}_2\boldsymbol{\alpha},\\ s.t.\ & \boldsymbol{\alpha}\geq\mathbf{0}.\end{aligned} \quad (15)$$

It is noted that $\mathcal{A}=\mathcal{A}^{\top}\preceq 0$ and hence $\mathcal{B}_1=\mathcal{B}_1^{\top}\preceq 0$. So, the problem described in (15) is a typical quadratic optimization problem with linear constraints, which can be solved by the classical *quadratic programming*.

Once the optimal $\boldsymbol{\alpha}^*$ is determined, we can apply (7) to the prediction problem. Namely, given a query input t^*, we have its corresponding output as

$$\begin{aligned}\boldsymbol{\eta}(t^*) &= \begin{bmatrix}\boldsymbol{\xi}(t^*)\\ \dot{\boldsymbol{\xi}}(t^*)\end{bmatrix} = \boldsymbol{\Theta}^{\top}(t^*)\mathbf{w}^*\\ &= \boldsymbol{\Theta}^{\top}(t^*)\boldsymbol{\Phi}(\boldsymbol{\Phi}^{\top}\boldsymbol{\Phi}+\lambda\boldsymbol{\Sigma})^{-1}(\boldsymbol{\mu}+\boldsymbol{\Sigma}\overline{\mathbf{G}}\boldsymbol{\alpha}^*)\\ &= \mathbf{k}^*(\mathbf{K}+\lambda\boldsymbol{\Sigma})^{-1}(\boldsymbol{\mu}+\boldsymbol{\Sigma}\overline{\mathbf{G}}\boldsymbol{\alpha}^*),\end{aligned} \quad (16)$$

where

$$\mathbf{k}^* = [\mathbf{k}(t^*,t_1)\ \mathbf{k}(t^*,t_2)\ \cdots\ \mathbf{k}(t^*,t_N)]. \quad (17)$$

Until now, we have explained the linearly constrained imitation learning framework, which we refer to as *linearly constrained KMP* (LC-KMP). Before ending this section, we show a few insights over LC-KMP:

1) **Non-constrained Learning**: if we consider imitation learning without linear constraints, i.e., $\boldsymbol{\alpha}=0$, the prediction of LC-KMP in (16) will become the vanilla KMP.
2) **Partially-constrained Learning:** when only partial linear constraints are required, we can deactivate the remaining constraints by simply setting their corresponding Lagrange multipliers as zero, which can be ensured by adding an additional equality constraint over $\boldsymbol{\alpha}$.
3) **Equality-constrained Learning:** the framework in (4) can be used to tackle equality constraints. For example, the equality constraint $\mathbf{g}_{n,f}^{\top}\boldsymbol{\eta}(t_n)=c_{n,f}$ can be guaranteed by $\mathbf{g}_{n,f}^{\top}\boldsymbol{\eta}(t_n)>c_{n,f}-\epsilon$ and $-\mathbf{g}_{n,f}^{\top}\boldsymbol{\eta}(t_n)>-c_{n,f}-\epsilon$, with an approximation error $\epsilon>0$.
4) **Adaptations with constraints:** we can extend LC-KMP to adapt trajectories towards arbitrary desired points in terms of position and velocity[8]. Given L desired points $\overline{\mathbf{D}}=\{\bar{t}_l,\bar{\boldsymbol{\mu}}_l,\bar{\boldsymbol{\Sigma}}_l\}_{l=1}^{L}$ comprising the desired time $\bar{t}_l$ and its corresponding output distribution (i.e., $\overline{\boldsymbol{\eta}}(\bar{t}_l)\sim\mathcal{N}(\bar{\boldsymbol{\mu}}_l,\bar{\boldsymbol{\Sigma}}_l)$), we can directly concatenate[9] $\overline{\mathbf{D}}$ and $\mathbf{D}$ to obtain an *extended probabilistic reference trajectory* $\mathbf{D}^U$. After that, we exploit $\mathbf{D}^U$ instead of $\mathbf{D}$ in the framework (4), which will generate a trajectory that passes through various desired points while satisfying additional linear constraints.

[8]The adaptation property has been proven essential in many applications, such as grasping an object at difference locations [8] or striking a ping-pong ball at a desired position while having a desired velocity [9].

IV. Connection with Model Predictive Control

In this section, we provide a connection between LC-KMP and linear MPC. For more details about MPC, see, e.g., [25], [26]. Assuming that we have a linear system, described by[10]

$$\boldsymbol{\eta}_{t+1} = \mathbf{A}\boldsymbol{\eta}_t+\mathbf{B}\mathbf{u}_t, \quad (18)$$

with $\mathbf{A}\in\mathbb{R}^{2\mathcal{O}\times 2\mathcal{O}}$ and $\mathbf{B}\in\mathbb{R}^{2\mathcal{O}\times\mathcal{O}}$ being the state and control matrices, and $\mathbf{u}_t\in\mathbb{R}^{\mathcal{O}}$ denoting the control command. The optimization objective in MPC is

$$\begin{aligned}\underset{\mathbf{u}_1,\ldots,\mathbf{u}_{N-1}}{\textbf{argmin}}\ & \sum_{t=1}^{N}(\boldsymbol{\eta}_t-\hat{\boldsymbol{\eta}}_t)^{\top}\mathbf{Q}_t(\boldsymbol{\eta}_t-\hat{\boldsymbol{\eta}}_t)+\sum_{t=1}^{N-1}\mathbf{u}_t^{\top}\mathbf{R}_t\mathbf{u}_t\\ \textbf{s.t.}\quad & \boldsymbol{\eta}_{max}\geq\boldsymbol{\eta}_t\geq\boldsymbol{\eta}_{min},\ \forall t\in\{2,3,\ldots,N\}\\ & \mathbf{u}_{max}\geq\mathbf{u}_t\geq\mathbf{u}_{min},\ \forall t\in\{1,2,\ldots,N-1\}\end{aligned} \quad (19)$$

with $\mathbf{Q}\succeq$ and $\mathbf{R}\succ 0$. $\hat{\boldsymbol{\eta}}_t$ denotes the desired reference trajectory point at time t, while $\boldsymbol{\eta}_{min}$, $\boldsymbol{\eta}_{max}$, $\mathbf{u}_{min}$ and $\mathbf{u}_{max}$ represent the predefined limits.

Following the prediction strategy in MPC (e.g., [20]), we can obtain

$$\underbrace{\begin{bmatrix}\boldsymbol{\eta}_1\\ \boldsymbol{\eta}_2\\ \boldsymbol{\eta}_3\\ \vdots\\ \boldsymbol{\eta}_N\end{bmatrix}}_{\overline{\boldsymbol{\eta}}} = \underbrace{\begin{bmatrix}\mathbf{I}\\ \mathbf{A}\\ \mathbf{A}^2\\ \vdots\\ \mathbf{A}^{N-1}\end{bmatrix}}_{\mathcal{S}_\eta}\boldsymbol{\eta}_1+\underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{B} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{AB} & \mathbf{B} & \cdots & \mathbf{0}\\ \vdots & \vdots & \ddots & \mathbf{0}\\ \mathbf{A}^{N-2}\mathbf{B} & \mathbf{A}^{N-3}\mathbf{B} & \cdots & \mathbf{B}\end{bmatrix}}_{\mathcal{S}_u}\underbrace{\begin{bmatrix}\mathbf{u}_1\\ \mathbf{u}_2\\ \vdots\\ \mathbf{u}_{N-1}\end{bmatrix}}_{\overline{\mathbf{u}}}. \quad (20)$$

Using (20), we can reformulate (19) as

$$\begin{aligned}\underset{\overline{\mathbf{u}}}{\textbf{argmin}}\ & (\overline{\boldsymbol{\eta}}-\widehat{\boldsymbol{\eta}})^{\top}\overline{\mathbf{Q}}(\overline{\boldsymbol{\eta}}-\widehat{\boldsymbol{\eta}})+\overline{\mathbf{u}}^{\top}\overline{\mathbf{R}}\overline{\mathbf{u}}\\ & \textbf{s.t.}\ \mathbf{W}_1\overline{\mathbf{u}}\geq\mathbf{W}_2+\mathbf{V}\boldsymbol{\eta}_1,\end{aligned} \quad (21)$$

where

$$\begin{aligned}\overline{\boldsymbol{\eta}} &= [\boldsymbol{\eta}_1^{\top}\ \boldsymbol{\eta}_2^{\top}\ \ldots\boldsymbol{\eta}_N^{\top}]^{\top},\\ \widehat{\boldsymbol{\eta}} &= [\hat{\boldsymbol{\eta}}_1^{\top}\ \hat{\boldsymbol{\eta}}_2^{\top}\ \ldots\hat{\boldsymbol{\eta}}_N^{\top}]^{\top},\\ \overline{\mathbf{u}} &= [\mathbf{u}_1^{\top}\ \mathbf{u}_2^{\top}\ \ldots\mathbf{u}_{N-1}^{\top}]^{\top},\\ \overline{\mathbf{Q}} &= \text{blockdiag}(\mathbf{Q}_1,\ \mathbf{Q}_2,\ \ldots,\ \mathbf{Q}_N),\\ \overline{\mathbf{R}} &= \text{blockdiag}(\mathbf{R}_1,\ \mathbf{R}_2,\ \ldots,\ \mathbf{R}_{N-1}).\end{aligned} \quad (22)$$

Note that $\mathbf{W}_1$, $\mathbf{W}_2$ and $\mathbf{V}$ denote coefficient matrices, which can be determined by using $\mathcal{S}_\eta$, $\mathcal{S}_u$, $\boldsymbol{\eta}_{min}$, $\boldsymbol{\eta}_{max}$, $\mathbf{u}_{min}$ and $\mathbf{u}_{max}$.

[9]Please refer to [6] for the details of updating the reference trajectory.

[10]Note that $\boldsymbol{\eta}_t$ is interchangeably used with $\boldsymbol{\eta}(t)$.

TABLE I

COMPARISON BETWEEN MPC AND LC-KMP

	MPC	LC-KMP
Model	$\boldsymbol{\eta}(t+1) = \mathbf{A}\boldsymbol{\eta}(t) + \mathbf{B}\mathbf{u}(t)$	$\boldsymbol{\eta}(t) = \boldsymbol{\Theta}^\top(t)\mathbf{w}$
Referecne traj.	$\{\hat{\boldsymbol{\eta}}_t\}_{t=1}^N$	$\{\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n\}_{n=1}^N$
Optimization	$\underset{\mathbf{u}_1,\ldots,\mathbf{u}_{N-1}}{\mathbf{argmin}} \sum_{t=1}^{N}(\boldsymbol{\eta}_t - \hat{\boldsymbol{\eta}}_t)^\top\mathbf{Q}_t(\boldsymbol{\eta}_t - \hat{\boldsymbol{\eta}}_t) + \sum_{t=1}^{N-1}\mathbf{u}_t^\top\mathbf{R}_t\mathbf{u}_t$ **s.t.** $\boldsymbol{\eta}_{max} \geq \boldsymbol{\eta}_t \geq \boldsymbol{\eta}_{min},\ \forall t \in \{2,3,\ldots,N\}$ $\mathbf{u}_{max} \geq \mathbf{u}_t \geq \mathbf{u}_{min},\ \forall t \in \{1,2,\ldots,N-1\}$	$\underset{\mathbf{w}}{\mathbf{argmin}} \sum_{n=1}^{N}\frac{1}{2}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w} - \hat{\boldsymbol{\mu}}_n)^\top\hat{\boldsymbol{\Sigma}}_n^{-1}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w} - \hat{\boldsymbol{\mu}}_n) + \frac{1}{2}\lambda\mathbf{w}^\top\mathbf{w}$ **s.t.** $\mathbf{g}_{n,f}^\top\boldsymbol{\eta}(t_n) \geq c_{n,f}, \forall f \in \{1,2,\ldots,F\}, \forall n \in \{1,2,\ldots,N\}.$
Compact form	$\underset{\overline{\mathbf{u}}}{\mathbf{argmin}}(\mathcal{S}_u\overline{\mathbf{u}} + \mathcal{S}_\eta\boldsymbol{\eta}_1 - \widehat{\boldsymbol{\eta}})^\top\overline{\mathbf{Q}}(\mathcal{S}_u\overline{\mathbf{u}} + \mathcal{S}_\eta\boldsymbol{\eta}_1 - \widehat{\boldsymbol{\eta}}) + \overline{\mathbf{u}}^\top\overline{\mathbf{R}}\overline{\mathbf{u}}$ **s.t.** $\mathbf{W}_1\overline{\mathbf{u}} \geq \mathbf{W}_2 + \mathbf{V}\boldsymbol{\eta}_1,$	$\underset{\mathbf{w}}{\mathbf{argmin}}\frac{1}{2}(\boldsymbol{\Phi}^\top\mathbf{w} - \boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{\Phi}^\top\mathbf{w} - \boldsymbol{\mu}) + \frac{1}{2}\lambda\mathbf{w}^\top\mathbf{w}$ **s.t.** $\overline{\mathbf{G}}^\top\boldsymbol{\Phi}^\top\mathbf{w} \geq \overline{\mathbf{C}}.$
Solution	$\overline{\mathbf{u}}^* = [\mathbf{u}_1^{*\top}\ \mathbf{u}_2^{*\top}\ \ldots\mathbf{u}_{N-1}^{*\top}]^\top$, apply $\mathbf{u}_1^*$ as control command	$\boldsymbol{\alpha}^*$ and $\mathbf{w}^*$, apply $\mathbf{k}^*(\mathbf{K} + \lambda\boldsymbol{\Sigma})^{-1}(\boldsymbol{\mu} + \boldsymbol{\Sigma}\overline{\mathbf{G}}\boldsymbol{\alpha}^*)$ to trajectory prediction.

Furthermore, substituting $\overline{\boldsymbol{\eta}} = \mathcal{S}_u\overline{\mathbf{u}} + \mathcal{S}_\eta\boldsymbol{\eta}_1$ into (21) gives

$$\begin{aligned}\underset{\overline{\mathbf{u}}}{\mathbf{argmin}}&(\mathcal{S}_u\overline{\mathbf{u}} + \mathcal{S}_\eta\boldsymbol{\eta}_1 - \widehat{\boldsymbol{\eta}})^\top\overline{\mathbf{Q}}(\mathcal{S}_u\overline{\mathbf{u}} + \mathcal{S}_\eta\boldsymbol{\eta}_1 - \widehat{\boldsymbol{\eta}}) + \overline{\mathbf{u}}^\top\overline{\mathbf{R}}\overline{\mathbf{u}} \\ &\textbf{s.t. } \mathbf{W}_1\overline{\mathbf{u}} \geq \mathbf{W}_2 + \mathbf{V}\boldsymbol{\eta}_1.\end{aligned} \tag{23}$$

Now, let us recall the optimization problem (5) of LC-KMP, whose impact form is

$$\begin{aligned}\underset{\mathbf{w}}{\mathbf{argmin}}&\frac{1}{2}(\boldsymbol{\Phi}^\top\mathbf{w} - \boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{\Phi}^\top\mathbf{w} - \boldsymbol{\mu}) + \frac{1}{2}\lambda\mathbf{w}^\top\mathbf{w} \\ &\textbf{s.t. } \overline{\mathbf{G}}^\top\boldsymbol{\Phi}^\top\mathbf{w} \geq \overline{\mathbf{C}}.\end{aligned} \tag{24}$$

By comparing (23) and (24), it can be seen that both optimization problems will be equivalent[11] *if* $\mathcal{S}_u = \boldsymbol{\Phi}^\top$, $\widehat{\boldsymbol{\eta}} = \boldsymbol{\mu} + \mathcal{S}_\eta\boldsymbol{\eta}_1$, $\overline{\mathbf{Q}} = \boldsymbol{\Sigma}^{-1}$, $\overline{\mathbf{R}} = \lambda\mathbf{I}$, $\mathbf{W}_1 = \overline{\mathbf{G}}^\top\boldsymbol{\Phi}^\top$ and $\mathbf{W}_2 = \overline{\mathbf{C}} - \mathbf{V}\boldsymbol{\eta}_1$. In fact, the key differences between MPC and LC-KMP lie at two aspects:

1) In contract to MPC that uses the dynamics model (18) for predicting future trajectory, LC-KMP uses the parametric model (2) instead. Note that basis functions in LC-KMP can be ultimately alleviated through the kernel trick, resulting in a non-parametric approach.
2) MPC aims at finding the optimal control command $\overline{\mathbf{u}} \in \mathbb{R}^{(N-1)\mathcal{O}}$, while LC-KMP aims for the optimal trajectory parameter $\mathbf{w} \in \mathbb{R}^{B\mathcal{O}}$.

For the purpose of clear comparison, we summarize the differences between MPC and LC-KMP in Table I.

It is worth pointing out that the unconstrained MPC was studied in [5], [20], where setting $\hat{\boldsymbol{\eta}}_t = \hat{\boldsymbol{\mu}}_t$ and $\mathbf{Q}_t = \hat{\boldsymbol{\Sigma}}_t^{-1}$ in (19) yields the *minimal intervention control* problem

$$\underset{\mathbf{u}_1,\ldots,\mathbf{u}_{N-1}}{\mathbf{argmin}} \sum_{t=1}^{N}(\boldsymbol{\eta}_t - \hat{\boldsymbol{\mu}}_t)^\top\hat{\boldsymbol{\Sigma}}_t^{-1}(\boldsymbol{\eta}_t - \hat{\boldsymbol{\mu}}_t) + \sum_{t=1}^{N-1}\mathbf{u}_t^\top\mathbf{R}_t\mathbf{u}_t, \tag{25}$$

which is equivalent to

$$\underset{\overline{\mathbf{u}}}{\mathbf{argmin}}(\mathcal{S}_u\overline{\mathbf{u}} + \mathcal{S}_\eta\boldsymbol{\eta}_1 - \boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\mathcal{S}_u\overline{\mathbf{u}} + \mathcal{S}_\eta\boldsymbol{\eta}_1 - \boldsymbol{\mu}) + \overline{\mathbf{u}}^\top\overline{\mathbf{R}}\overline{\mathbf{u}}. \tag{26}$$

We can find that minimal intervention control, as a special case of unconstrained MPC, shares similarities with LC-KMP in terms of *imitation*, since minimal intervention control exploits the distribution of demonstrations as well.

[11]The scalar "$\frac{1}{2}$" can be ignored as it does not influence the optimization.

V. EVALUATIONS

In order to verify our framework, several evaluations are provided, including (i) adapting 2D trajectories with/without motion limits, as well as adaptations with full/partial motion limits (Section V-A); (ii) adapting 3D trajectories in various planes (Section V-B); (iii) generating stable waking trajectories for a humanoid robot (Section V-C). The Gaussian kernel $k(t_i, t_j) = \exp(-k_h(t_i - t_j)^2)$ is used in this section.

A. Adaptation with Motion Limits

We first apply LC-KMP to the learning and adaptation of 2D hand-written letter 'G', where five demonstrations comprising input t and 2D output $\boldsymbol{\xi}(t) = [x(t)\ y(t)]^\top$ are used (plotted by solid green curves in Fig. 1 (*top row*)). As a comparison, we separately apply LC-KMP and vanilla KMP to adapt trajectories towards desired points under motion limits. We consider the following constraints:

$$-4 \leq x \leq 10,\ y \geq -4,\ \dot{x} \geq -32,\ \dot{y} \geq -20. \tag{27}$$

Other relevant parameters are $\lambda = 3$ and $k_h = 6$. It can be seen from Fig. 1 (*middle row*) that LC-KMP (solid yellow curves) indeed modulates trajectories towards various desired points (depicted by circles) while respecting the motion limits, where lower and upper limits are shown by blue and red dashed curves, respectively. In contrast, vanilla KMP (dashed green curves) only focuses on trajectory adaptations, ignoring the motion limits.

Furthermore, we test LC-KMP with full constraints and partial constraints. Specifically, in the former case constraints are active over the whole time duration (i.e., $0 < t \leq 2$), while in the latter case constraints are only active when $0.15 < t \leq 2$. We consider the following constraints

$$x \leq 8\ ,\ y \geq -4. \tag{28}$$

As shown in Fig. 1 (*bottom row*), the upper limit of $x(t)$ has a conflict with the x component of the first desired point, thus the adapted trajectory (dashed green curves), that obeys the constraints over the whole time duration, fails to pass through the x component of the first desired point. In contrast, when only partial constraints are active, the adapted trajectory (solid yellow curves) is capable of passing through the first desired point. Note that once the adapted trajectory

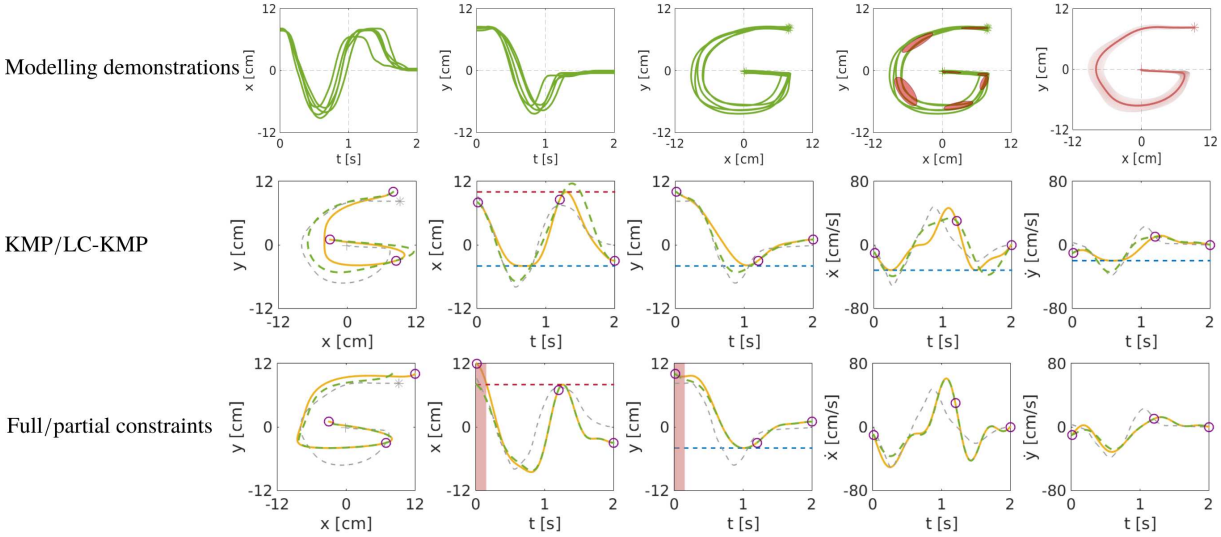

Fig. 1. Learning and adaptation of 2D letter 'G' under motion limits. *Top row* shows demonstrations (green curves) and the corresponding GMM/GMR modeling results. Ellipses represent Gaussian components in GMM, while the pink curve and the shaded pink area respectively depict the mean and the standard deviation of the retrieved trajectory by GMR. *Middle row* shows adaptations by using LC-KMP (solid yellow curves) and vanilla KMP (dashed green curves), where the dashed gray curves denote the mean of the probabilistic reference trajectory. Circles represent desired points. The dashed blue and red curves correspond to the lower and upper limits, respectively. *Bottom row* depicts adaptations with full constraints (solid yellow curves) and partial constraints (dashed green curves) with the shaded red area corresponding to the inactive region.

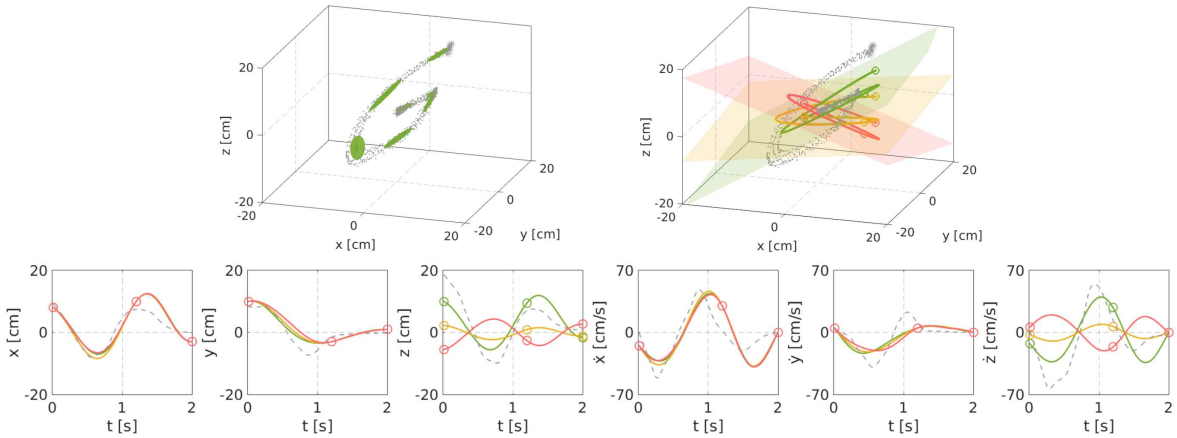

Fig. 2. Adaptation of 3D letter 'G' in different planes. *Top-left* shows GMM modeling of demonstrations, while *top-right* and *bottom* plot trajectory adaptations in different planes (shown by colored planes), where circles denote desired points.

moves out of the deactivate region (depicted by the shaded red area), it will comply with the constraints again.

B. Adaptation with Plane Constraint

We here consider adaptations of 3D letter 'G' in different planes. Five demonstrations (depicted by gray dots in Fig. 2 (*top-left*)) in terms of input t and 3D output $\boldsymbol{\xi}(t) = [x(t)\ y(t)\ z(t)]^\top$ are collected. The relevant parameters in LC-KMP are $\lambda = 5$ and $k_h = 2$. The plane constraint is defined as

$$a_x x + b_y y + c_z z = d. \tag{29}$$

We have three groups of evaluations, whose corresponding parameters are set as: (*i*) $a_x^{(1)} = 1, b_y^{(1)} = 0.2, c_z^{(1)} = -1.1, d^{(1)} = -1$; (*ii*) $a_x^{(2)} = 0.6, b_y^{(2)} = 0.4, c_z^{(2)} = -3, d^{(2)} = 2$; (*iii*) $a_x^{(3)} = 1, b_y^{(3)} = 0.6, c_z^{(3)} = 2, d^{(3)} = 3$. Evaluations are provided in Fig. 2, showing that LC-KMP can learn and adapt demonstrations into different planes and meanwhile considering various start-/via-/end- points in terms of both position and velocity.

C. Stable Walking Trajectories for Humanoid Robot

We now consider a more challenging task, where a stable walking trajectory for a humanoid robot is required. As suggested in [27], [28], the *capture region* which is defined by proper constraints over the position and velocity profiles of the center of mass (CoM) can be used to ensure the stability. In this example, we aim to plan CoM trajectories to accomplish the non-periodic walking (i.e., switch from forward motion to backward motion) over three periods (each period lasts $0.7s$). Specifically, we design the capture regions as[12]

$$\begin{aligned} x_l &\leq a_x x + b_x \dot{x} \leq x_u \\ y_l &\leq a_y y + b_y \dot{y} \leq y_u \end{aligned} \tag{30}$$

[12]The parameters of linear constraints are determined according to the physical features of the simulated humanoid platform [29].

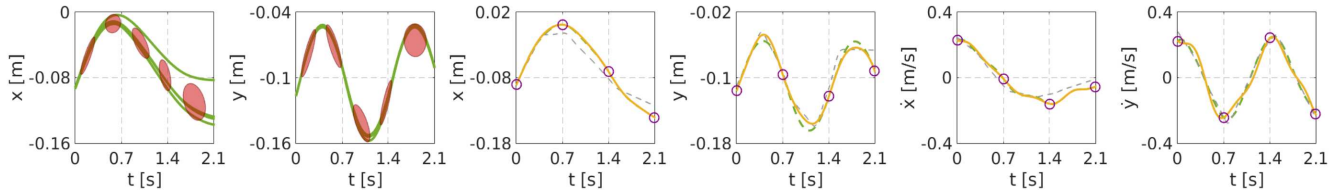

Fig. 3. Learning and adapting walking trajectories. Green solid curves and ellipses denote demonstrations and GMM components, respectively. Yellow solid curves and green dashed curves respectively represent adapted trajectories by using LC-KMP and vanilla KMP. Circles depict desired points.

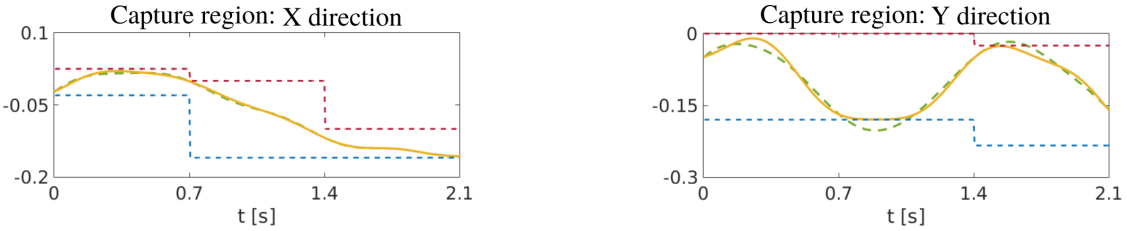

Fig. 4. Evaluations of stability criteria, where yellow solid curves and green dashed curves correspond to LC-KMP and vanilla KMP, respectively. Dashed red and blue curves depict the upper and lower bounds of capture regions, respectively.

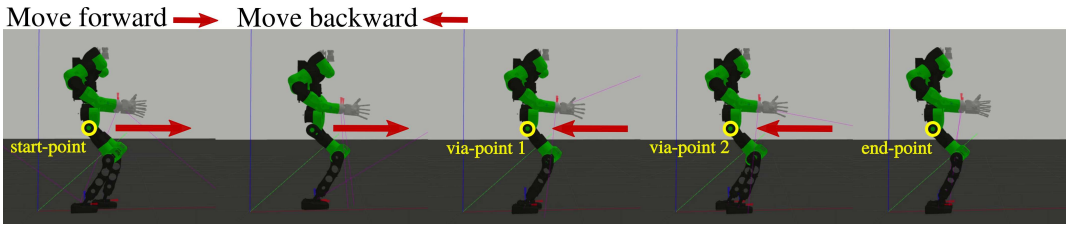

Fig. 5. Snapshots of walking movement of a simulated humanoid robot, where LC-KMP is used to plan the CoM trajectories. Arrows represent the CoM motion direction.

with $a_x = a_y = 1$, $b_x = b_y = (\frac{h_{com}}{g})^{\frac{1}{2}}$, where $h_{com} = 0.8898$ and $g = 9.8$. x_l, x_u, y_l and y_u are set by (*i*) if $0 < t \leq 0.7, x_l = -0.03, x_u = 0.025, y_l = -0.18, y_u = 0$; (*ii*) if $0.7 < t \leq 1.4, x_l = -0.16, x_u = 0, y_l = -0.18, y_u = 0$; (*iii*) if $1.4 < t \leq 2.1, x_l = -0.16, x_u = -0.1, y_l = -0.234, y_u = -0.025$. LC-KMP parameters are $\lambda = 6$ and $k_h = 2$.

We first use optimization solver [30] to generate four training trajectories (solid green curves in Fig. 3), which serve as demonstrations for our framework. For the sake of comparison, both LC-KMP and vanilla KMP are employed to generate the adapted CoM trajectories. As can be seen from Fig. 3 (the third - sixth plots), both LC-KMP (solid yellow curves) and vanilla KMP (dashed green curves) are capable of adapting trajectories towards various desired points (depicted by circles).

In addition, evaluations of $a_x x + b_x \dot{x}$ and $a_y y + b_y \dot{y}$ are provided in Fig. 4, where LC-KMP (solid yellow curves) fulfills the constraints of capture regions in both X and Y directions, while vanilla KMP (dashed green curves) exceeds the capture region in Y direction. Therefore, in contrast to vanilla KMP, LC-KMP can indeed take into account additional linear constraints while performing learning and adaptations of demonstrations. Snapshots of walking motions on the simulation platform [29] are illustrated in Fig. 5.

VI. Conclusions

As an extension of KMP, we have developed an imitation learning framework capable of addressing the learning and adaptation issues while considering additional linear constraints. This framework has been verified through several examples, comprising adapting 2D letters with (partial) motion limits, adapting 3D letter in different planes, as well as planning stable walking trajectories for a humanoid robot.

In this paper, we only focus on learning time-driven trajectories. In fact, due to the kernel treatment, the proposed framework can be extended to the learning of trajectories with high-dimensional inputs. In our previous work [6], KMP has been proven effective in a human-robot collaboration setting, where the user's hand positions (6D input) were used to drive the robot movement straightforwardly. Thus, it would be interesting to exploit LC-KMP in this direction. Besides, incorporating orientation learning [31] into the LC-KMP framework could also be promising.

It is worth mentioning that the constraints throughout this paper are linear. In order to address the non-linear cases, one possible way is to employ the linearization treatment, as done in [32]. In addition, we assume that the constraints are known beforehand, which may prohibit the applications of our framework in highly dynamic environments where constraints are unknown. Thus, further studies on inferring constraints from tasks or demonstrations are needed.

Acknowledgement

We thank Jiatao Ding from Wuhan University and Istituto Italiano di Tecnologia for his help on locomotion evaluations. We also thank João Silvério from Idiap Research Institute and Linyan Han from Southeast University for their comments on this paper.

REFERENCES

[1] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML*, vol. 97, 1997, pp. 12–20.

[2] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

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

[4] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[5] S. Calinon, D. Bruno, and D. G. Caldwell, "A task-parameterized probabilistic model with minimal intervention control," in *2014 IEEE International Conference on Robotics and Automation*. IEEE, 2014, pp. 3339–3344.

[6] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 833–852, 2019.

[7] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[8] F. Stulp, E. Theodorou, J. Buchli, and S. Schaal, "Learning to grasp under uncertainty," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5703–5708.

[9] Y. Huang, D. Büchler, O. Koç, B. Schölkopf, and J. Peters, "Jointly learning trajectory generation and hitting point prediction in robot table tennis," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots*. IEEE, 2016, pp. 650–655.

[10] J. Silvério, Y. Huang, F. Abu-Dakka, L. Rozo, and D. Caldwell, "Uncertainty-aware imitation learning using kernelized movement primitives," in *2019 International Conference on Intelligent Robots and Systems*. IEEE, 2019, pp. 90–97.

[11] M. Howard, S. Klanke, M. Gienger, C. Goerick, and S. Vijayakumar, "A novel method for learning policies from variable constraint data," *Autonomous Robots*, vol. 27, no. 2, pp. 105–121, 2009.

[12] L. Armesto, V. Ivan, J. Moura, A. Sala, and S. Vijayakumar, "Learning constrained generalizable policies by demonstration." in *Robotics: Science and systems*, 2017.

[13] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 489–494.

[14] F. Chaumette and É. Marchand, "A redundancy-based iterative approach for avoiding joint limits: Application to visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 719–730, 2001.

[15] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[16] M. Muhlig, M. Gienger, S. Hellbach, J. J. Steil, and C. Goerick, "Task-level imitation learning using variance-based movement optimization," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 1177–1184.

[17] Y. Huang, J. Silvério, and D. G. Caldwell, "Towards minimal intervention control with competing constraints," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2018, pp. 733–738.

[18] J. Silvério, Y. Huang, L. Rozo, S. Calinon, and D. G. Caldwell, "Probabilistic learning of torque controllers from kinematic and force constraints," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2018, pp. 1–8.

[19] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of artificial intelligence research*, vol. 4, pp. 129–145, 1996.

[20] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, 2016.

[21] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in neural information processing systems*, 2013, pp. 2616–2624.

[22] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," in *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 1998, pp. 515–521.

[23] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[24] K. B. Petersen, M. S. Pedersen *et al.*, "The matrix cookbook," *Technical University of Denmark*, vol. 7, no. 15, p. 510, 2008.

[25] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," in *Nonlinear model predictive control*. Springer, 2009, pp. 345–369.

[26] J. B. Rawlings, "Tutorial overview of model predictive control," *IEEE control systems magazine*, vol. 20, no. 3, pp. 38–52, 2000.

[27] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *2006 6th IEEE-RAS international conference on humanoid robots*. IEEE, 2006, pp. 200–207.

[28] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[29] L. Muratore, A. Laurenzi, E. M. Hoffman, A. Rocchi, D. G. Caldwell, and N. G. Tsagarakis, "Xbotcore: A real-time cross-robot software platform," in *2017 First IEEE International Conference on Robotic Computing*. IEEE, 2017, pp. 77–80.

[30] J. Ding, X. Xiao, and N. G. Tsagarakis, "Nonlinear optimization of step duration and step location," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2019, pp. 2259–2265.

[31] Y. Huang, F. J. Abu-Dakka, J. Silvério, and D. G. Caldwell, "Generalized orientation learning in robot task space," in *2019 IEEE International Conference on Robotics and Automation*. IEEE, 2019, pp. 2531–2537.

[32] A. Varol, M. Salzmann, P. Fua, and R. Urtasun, "A constrained latent variable model," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2248–2255.