Optimal Control Combining Emulation and Imitation to Acquire Physical Assistance Skills

Amirreza Razmjoo, Teguh Santoso Lembono and Sylvain Calinon

***Abstract*— This paper studies exploiting action-level learning (imitation) in the optimal control problem context. Cost functions defined by the optimal control methods are similar to the goal-level learning (emulation) in animals. However, imitating the robot's or others' (e.g. human's) previous experiences (demonstrations) could help the system to improve its performances. We propose to use demonstrations more efficiently by predicting an initialization for the optimal control problems (OCPs) and adding an imitation term to the cost functions. While the predicted initial guess initializes the OCPs close to their local optima, the imitation term guides the optimization, resulting in a faster convergence rate. We test our algorithm in a physical assistive task where a robot should help a human perform a sit-to-stand (STS) task. We define this task as two optimal control problems. The first OCP predicts the human's desired assistance and the other one controls the robot. We have tested our method on different experiments with different conditions for the human in which the robot should quickly solve the two optimization problems exploiting some demonstrations of how it can perform the task. Our proposed method reduced the number of iterations by more than 90% and 70% for the human assistance prediction and the robot controller, respectively, compared to the standard problem which does not take the demonstrations into account.**

I. INTRODUCTION

Skill acquisition encompasses a broad spectrum of learning approaches from action-level *imitation* to goal-level *emulation*. Our motivation to combine both imitation and emulation comes from social learning studies with animals and humans showing the mutual benefits of both learning strategies [1]. While *emulation* requires higher cognitive skills to infer the intended goals behind a set of actions, *imitation* allows to acquire skills in a simple but limited way, by directly mimicking the movements and actions shown to the learner. Because of its higher level nature, *emulation* can easily be misinterpreted as being the only useful mechanism. Ethology studies such as [1] argue that the two are complementary, in the sense that *emulation* allows greater generalization (i.e., the learner understands the purpose of the task and can reproduce it in possibly different ways), while imitation allows to acquire complex task fast, even if all underlying goals are not understood ("copy all, refine later" strategy).

Similarly, in the engineering world, research on optimal control and reinforcement learning (RL) is tightly connected to the *emulation* approach, while research on movement primitives is tightly connected to the *imitation* approach. Emulation makes the system more generalizable, however, we show that imitation/mimicking can improve the learning rate. We test our method with an assistive task in which a robot should provide the required help for different users. We propose an approach to consider human demonstration (i.e., imitiation) in the context of optimal control (i.e., emulation). We compare the convergence rate of the controller with and without the imitation part.

This work was supported by the SWITCH project (https://switch-project.github.io/), funded in part by the Swiss National Science Foundation.

The authors are with the Idiap Research Institute and the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland {amirreza.razmjoo; teguh.lembono; sylvain.calinon}@idiap.ch

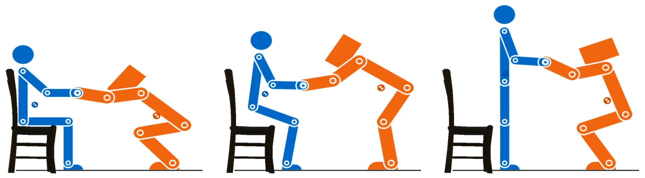
Fig. 1. Models of the human (blue) and the robot (orange). The corresponding CoM positions are presented as semi-filled discs.

The proposed approach also relates to studies in two other fields. The first field is RL in which some works such as [2] show how imitation can accelerate the convergence of RL systems with sparse rewards. While such approaches typically target autonomous skills learning, we study the problem in the context of assistive skills acquisition in which optimal control is more preferable to RL. The second related field is warm starting whose detailed comparison with our approach is described later in this paper.

Optimal control problems typically face two main challenges. The first is nonlinearity, which can be partially solved using iterative approaches to find locally optimal solutions, including differential dynamic programming (DDP) [3] and iterative linear quadratic regulator (iLQR) [4]. When the problem needs to be solved online, approaches relying on trajectory predictions to better initialize these optimization methods have been investigated. This approach is called *warm starting*, which can use methods based on lookup tables [5], [6], Gaussian process regression (GPR) [7], or neural networks (NN) [8]; see [9] for an overview. The other main challenge in optimal control is defining the cost, where some techniques like inverse optimal control (IOC) [10] have been proposed to infer this cost from demonstrations.

Both the warm start and IOC approaches use demonstrations to address the mentioned challenges in optimal control. In this paper, we propose a method to exploit the demonstrations more efficiently by trying to also imitate them rather than merely emulating them. To do this, we modify the cost function (emulating part) with a term that penalizes the system when it deviates far from the demonstrations, and show how this term, in addition to the warm start, can further improve the convergence rate of the optimization problem.

We test our method on a simulated sit-to-stand (STS) task (see Fig. 1) defined as two optimal control problems (OCPs). The first OCP predicts the user's desired help, and the second one predicts the robot torques that are required to provide the predicted assistance. Hereafter, the first and second OCP is called the *human assistance prediction* and the *robot controller*, respectively. In our formulation, we assume that the user can have some disabilities in its ankle, knee, and hip joints. The mass and height of the human can also vary according to each user so that the robot should quickly solve the two OCPs by considering some successful examples of the task. We will show how our approach could be beneficial to achieve this goal by comparing the convergence time of the OCPs with and without the proposed method.

In the remainder of this paper, in Section II, we describe the required background for this study, and we explain the methodology in Section III. The simulation results are presented and discussed in Section IV and V, respectively. Finally, we conclude the paper in Section VI.

II. BACKGROUND

A. STS Task

Physical assistance has been studied in different applications such as dressing [11], [12] and sit-to-stand (STS) [13]. We use the latter to evaluate the benefits of our proposed method. Defining the STS task as an optimization problem is not a new approach. El-Husseiny *et al.* [14] have used the IOC technique to find a quadratic cost function to generate human hip motion during the STS task. Their cost function consisted of two terms, one for states regularization and the other for inputs regularization. In another work, Geravand *et al.* [13] have used a similar approach, but have found a more elaborated cost function that considers different criteria such as the human's final position, Center of Mass (CoM) position at the end of the task, and joint limits. Li *et al.* [15] divided the STS task into two sit-to-perch (STP) and perch-to-stand (PTS) phases. They observed that during the STP phase, the hip joint motions, and during the PTS phase, the motions of the ankle, knee, and hip joints are consistent among the participants. They defined two cost functions for these two phases, which minimize the total torque commands of the human in the STS task. In our work, we define the STS task as in [13], but with a simplified cost function defined manually to generate human-like STS transfer. The cost function is described with more details in Section III-B.

B. Iterative LQR (iLQR)

iLQR is a modified version of the linear quadratic regulator (LQR) for nonlinear systems. In the LQR method, the cost function is defined as a quadratic function, and the dynamic of the system is assumed to be linear. ILQR exploits Taylor expansion to quadratize the cost function c and to linearize the system's dynamics $\boldsymbol{f}$ around their current values w.r.t. the system variables, namely, the states of the system $\boldsymbol{x}$ and the input commands $\boldsymbol{u}$. Then, at each iteration, it solves an LQR problem and updates the current solution. This process continues until it reaches a local optimum. Here, we use the batch form of iLQR, but without any loss of generality, the idea is valid for any other optimization method.

The dynamical system $\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ can be linearly approximated around its current estimations as

$$\begin{aligned} \boldsymbol{x}_{t+1} \approx \hat{\boldsymbol{x}}_{t+1} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t}(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_t}(\boldsymbol{u}_t - \hat{\boldsymbol{u}}_t) \\ \Rightarrow \Delta\boldsymbol{x}_{t+1} \approx \boldsymbol{A}_t \Delta\boldsymbol{x}_t + \boldsymbol{B}_t \Delta\boldsymbol{u}_t. \end{aligned} \tag{1}$$

In (1), $\hat{\boldsymbol{x}}_t$ and $\hat{\boldsymbol{u}}_t$ are the current estimations of state and input variables at t, while $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ are the system and input matrices, respectively. This equation formulates the change of the states at each time as a linear function of the change of the inputs and states at its previous time. With this recursive formula, we can formulate the deviations of all the states as

$$\begin{bmatrix} \Delta\boldsymbol{x}_2 \\ \Delta\boldsymbol{x}_3 \\ \vdots \\ \Delta\boldsymbol{x}_T \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2\boldsymbol{A}_1 \\ \vdots \\ \prod_{t=1}^{T-1} \boldsymbol{A}_{T-t} \end{bmatrix}}_{\boldsymbol{S_x}} \Delta\boldsymbol{x}_1 \quad + \\ \underbrace{\begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{A}_2\boldsymbol{B}_1 & \boldsymbol{B}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{t=1}^{T-2} \boldsymbol{A}_{T-t}\boldsymbol{B}_1 & \prod_{t=1}^{T-3} \boldsymbol{A}_{T-t}\boldsymbol{B}_2 & \cdots & \boldsymbol{B}_{T-1} \end{bmatrix}}_{\boldsymbol{S_u}} \begin{bmatrix} \Delta\boldsymbol{u}_1 \\ \Delta\boldsymbol{u}_2 \\ \vdots \\ \Delta\boldsymbol{u}_{T-1} \end{bmatrix}, \tag{2}$$

where $\boldsymbol{S_u}$ and $\boldsymbol{S_x}$ are transformation matrices that map $\Delta\boldsymbol{u}$ and $\Delta\boldsymbol{x}_1$ to $\Delta\boldsymbol{x}$, respectively. Since the initial states of the system are known ($\Delta\boldsymbol{x}_1 = 0$), we can write

$$\Delta\boldsymbol{x} = \boldsymbol{S_u}\Delta\boldsymbol{u}, \tag{3}$$

where $\Delta\boldsymbol{u}$ and $\Delta\boldsymbol{x}$ are the concatenated vectors of variations of all the input and the state variables, respectively.

The cost function c is a nonlinear function of all the states and inputs, so its quadratized version can be defined as

$$\begin{aligned} c(\boldsymbol{x}, \boldsymbol{u}) \;\approx\; & c(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}) + \Delta\boldsymbol{x}^\top \boldsymbol{g_x} + \Delta\boldsymbol{u}^\top \boldsymbol{g_u} \\ & + \frac{1}{2}\Delta\boldsymbol{x}^\top \boldsymbol{H_x}\Delta\boldsymbol{x} + \frac{1}{2}\Delta\boldsymbol{u}^\top \boldsymbol{H_u}\Delta\boldsymbol{u} \\ & + \Delta\boldsymbol{x}^\top \boldsymbol{H_{xu}}\Delta\boldsymbol{u}, \end{aligned} \tag{4}$$

where

$$\boldsymbol{g_x} = \frac{\partial c}{\partial \boldsymbol{x}} \;,\; \boldsymbol{g_u} = \frac{\partial c}{\partial \boldsymbol{x}}, \tag{5a}$$

$$\boldsymbol{H_x} = \frac{\partial^2 c}{\partial \boldsymbol{x}^2} \;,\; \boldsymbol{H_u} = \frac{\partial^2 c}{\partial \boldsymbol{u}^2} \;,\; \boldsymbol{H_{xu}} = \frac{\partial^2 c}{\partial \boldsymbol{x}\partial \boldsymbol{u}}. \tag{5b}$$

The value of $\Delta\boldsymbol{x}$ in (4) can be replaced from (3), so the quadratized cost would be a function of only $\Delta\boldsymbol{u}$. The optimizer $\Delta\boldsymbol{u}^*$ of (4) can be found by putting its gradient equal to zero, which would result in

$$\Delta\boldsymbol{u}^* = (\boldsymbol{S_u}^\top \boldsymbol{H_u}\boldsymbol{S_u} + 2\boldsymbol{S_u}^\top \boldsymbol{H_{xu}} + \boldsymbol{H_u})^{-1}(-\boldsymbol{S_u}\boldsymbol{g_x} - \boldsymbol{g_u}). \tag{6}$$

TABLE I

Dynamical parameters of a human body segments.

The segment	L_{ratio}	M_{ratio}	$I\ (kg.m^2)$	$\text{CoM}_{\text{ratio}}$
Foot	0.042	0.0145	0.0038	-
Shank	0.250	0.0465	0.0505	0.532
Thigh	0.240	0.0988	0.1502	0.500
Trunk	0.300	0.5080	1.3080	0.530
Upper arm	0.183	0.0270	0.0213	0.430
Forearm	0.160	0.0160	0.0760	0.410

The last step only works when there is no hard constraint, so we define the constraints in this experiment as a part of the cost function, and heavily penalize the system when it violates them.

III. Methodology

We formulate the STS task as two optimal control problems with two separate cost functions to describe the human assistance and the robot controller. These cost functions are minimized by considering the human and robot's dynamical model using iLQR. To improve the convergence rate of the OCPs, we propose to use previous demonstrations to predict the optimal trajectories which are then used to warm start the OCP solver as well as to modify the cost function with an imitation term. In addition, we take advantage of *control primitives* (CPs) to generate smoother behaviors and reduce the dimension of the system states and inputs.

A. Dynamical modeling

We model the human as a planar inverted pendulum with five links, which is inspired from STS literature such as [15], [13]. We assume that the robot assisting the human is also a five-link planner pendulum. Fig. 1 illustrates the two agents during the STS task. In this modeling, the mass and the length of each body segment are calculated according to their proportion to the agent's total mass and height, respectively. Also, the mass of each link is assumed to be located at its CoM position. For each segment, its L_{ratio} (length ratio to the human's height) [16], M_{ratio} (Mass ratio to the human's total mass) [16], I (the moment of inertia) [16], and $\text{CoM}_{\text{ratio}}$ (distance of CoM position of each link from its previous link normalized with the length of each link) [17] are presented in Table I. Note that in the simulation, the values of M_{ratio} and I should be doubled for all of the segments, except for the trunk, as there are two of them in human bodies. While the human can have different total mass and height among different experiments to simulate different subjects, The robot always has 80 kg mass and 1.8 m height.

B. Task definition

1) Human assistance prediction: At this step, the robot tries to find what is the optimal way to help the human by solving an optimization problem whose outputs are the optimal desired external forces and the human reaction during the task. The cost function defined for this step has two main terms. The first one specifies the human to be at the standing position at the end of the task and the other term penalizes high input commands and external forces. The standing position is defined as the human's shanks, thighs and trunk are in the straight-up direction with zero velocity. By changing the cost weight on each of the human's joints, we control its torque value and simulate different disabilities. For example, by giving high cost weight to the human's knee, we emphasize that this joint has some problems and should not apply large torques. In this experiment, we only change the cost weights of the ankle, knee, and hip joints.

There are two other terms in the cost function of the human assistance prediction to make the total CoM position of the human to be at the foot support at the end of the task and to consider the human's joints limits during the task. These two costs penalize the system when it goes out of the feasible region. This step can be defined mathematically as

$$\boldsymbol{u}_H^*, \boldsymbol{F}^* = \underset{\boldsymbol{u}_H, \boldsymbol{F}}{\text{argmin}}\ c_H(m, h, \boldsymbol{R}), \tag{7}$$

where $\boldsymbol{u}_H$ and $\boldsymbol{F}$ are the human torque commands and external forces, respectively, and c_H is the cost function. m and h are the user's total mass and height, respectively, and $\boldsymbol{R}$ is the joints cost weight. The human's optimal joint trajectory $\boldsymbol{x}_H^*$ and its hand trajectory can also be calculated with the outputs of (7) and considering the human's model.

2) The robot controller: The robot should follow the trajectory of the human's hand while it is expected to receive $\boldsymbol{F}^*$ alongside this trajectory. For the first one, we put a term in the robot's cost function to move its end-effector at the planned human's hand trajectory, and for the second one, we regularize the interaction forces around $\boldsymbol{F}^*$. Like the human assistance prediction, the cost function of the robot should be minimized w.r.t. the robot's torques and the interaction forces. Note that interaction forces are functions of the human's and robot's torques and configurations. Thus, by considering them as separate variables, we violate a physical constraint, which is that the end-effector of the robot and the hand of the human should be at the same place every time. We consider this constraint in our cost function. In this way, we do not need to solve the closed-chain kinematic problem of the two systems to find the interaction forces, so we can consider the two agents as two separate systems. The robot's joint limits and the stability of both the human and robot at the end of the task are also considered in the robot controller. That said, the robot controller can be modeled as

$$\boldsymbol{u}_R^*, \boldsymbol{F'}^* = \underset{\boldsymbol{u}_R, \boldsymbol{F'}}{\text{argmin}}\ c_R(\boldsymbol{x}_H^*, \boldsymbol{F}^*, \boldsymbol{u}_H^*), \tag{8}$$

where $\boldsymbol{u}_R$ is the robot's torque commands and $\boldsymbol{F'}$ is the interaction forces. If (8) converges well, we expect $\boldsymbol{F'}^*$ to be equal or at least be close to $\boldsymbol{F}^*$. The two cost functions mentioned above are the emulation part of the STS task. In this paper, we propose to speed up the convergence rate of these two OCPs by exploiting warm starting and imitation.

C. Warm starting

The closer the initial guess is to the system's local optimum, the fewer iteration iLQR needs to find an optimizer. The system can guess a good initial point by exploiting its previous experiences which are, in this paper, some demonstrations gathered by solving the OCPs without considering the imitation term. This could be done either by lookup tables or any regression method that could map the situations to the initial guess. Here, we present the idea with the former one, in the form of k-nearest neighbors (k-NN) with $k = 1$.

D. Imitation cost

The cost function in (4) can be seen as the emulation part of the task. Although it can generalize the task to any situation, it often requires lots of iterations to converge to a solution. To address this problem, we modify c in (4) as

$$c' = c + \lambda(\boldsymbol{p} - \boldsymbol{p}_d)^\top(\boldsymbol{p} - \boldsymbol{p}_d). \tag{9}$$

In (9), $\boldsymbol{p}$ is a feature vector we want to imitate, $\boldsymbol{p}_d$ is its desired value extracted from the demonstrations, and λ is a positive constant. This imitation term has a minimizer that is expected to be close to the minimizer of the main cost function c, so the optimization problem would more likely iterate toward an area that is emphasized by the main cost and the imitation term. We can be more confident about the closeness of the minimizers if the desired value of the chosen feature $\boldsymbol{p}_d$ can be predicted for the new situation with sufficient accuracy. Because of this, this feature should be chosen with some caution. Note that it does not need to be very accurate: just accurate enough to give the system some guidance toward its local optima. In this work, we chose the joint angles as the features, and we predict its values from the demonstrations by using k-NN with $k = 1$, but any applicable method could be used. The value of λ should be small as compared to other values in c in order to not significantly affect the main optimization problem, but it should be large enough to affect the convergence rate. Hence, λ should be adjusted for each task.

E. Control Primitives (CPs)

We assume that the input commands are made up of some basis functions each of which is called a *control primitive (CP)*. Here, we used radial basis functions (RBFs), but other basis functions such as Bernstein polynomials or Fourier series can also be used [18]. The basis functions implicitly enforce the smoothness and the continuity of the control inputs and help the human and the robot behave more naturally. We assume that the input commands can be defined as

$$\boldsymbol{u} = \boldsymbol{\Psi}\boldsymbol{w}_{\boldsymbol{u}} \Rightarrow \Delta\boldsymbol{u} = \boldsymbol{\Psi}\Delta\boldsymbol{w}_{\boldsymbol{u}}, \tag{10}$$

where $\boldsymbol{\Psi}$ is the transformation matrix made up of the basis functions (CPs) and $\boldsymbol{w}_{\boldsymbol{u}}$ is the vector of the CPs' coefficients. With this assumption, (6) should be modified as

$$\Delta\boldsymbol{w}_{\boldsymbol{u}}^* = \Big(\boldsymbol{\Psi}^\top(\boldsymbol{S}_{\boldsymbol{u}}^\top\boldsymbol{H}_{\boldsymbol{u}}\boldsymbol{S}_{\boldsymbol{u}} + 2\boldsymbol{S}_{\boldsymbol{u}}^\top\boldsymbol{H}_{\boldsymbol{x}\boldsymbol{u}} + \boldsymbol{H}_{\boldsymbol{u}})\boldsymbol{\Psi}\Big)^{-1} \boldsymbol{\Psi}^\top(-\boldsymbol{S}_{\boldsymbol{u}}\boldsymbol{g}_{\boldsymbol{x}} - \boldsymbol{g}_{\boldsymbol{u}}). \tag{11}$$

We use a different number of basis functions for the two OCPs, i.e., five basis functions for the human assistance prediction and eight basis functions for the robot controller.

IV. SIMULATION

In this section, we apply our method to a simulated STS task. To gather the demonstrations, we solve the OCPs without the imitation term for N times and record the human's dynamical characteristics (i.e, m, h and $\boldsymbol{R}$) and the coefficients of the CPs for the human assistance prediction and the robot controller. We also record the joint angles of the human and the robot to predict the desired trajectory for the imitation part, however, we map them to a lower dimension as

$$\boldsymbol{x}_\theta = \boldsymbol{\Psi}\boldsymbol{w}_{\boldsymbol{x}_\theta} \quad \Rightarrow \quad \boldsymbol{w}_{\boldsymbol{x}_\theta} = \boldsymbol{\Psi}^\dagger\boldsymbol{x}_\theta, \tag{12}$$

where $\boldsymbol{\Psi}^\dagger$ is the pseudoinverse of $\boldsymbol{\Psi}$. The θ subscript beside $\boldsymbol{x}$ shows that $\boldsymbol{x}_\theta$ only consists of the joint angles and does not include other states. $\boldsymbol{\Psi}$ used for mapping each agent's joint angles is the same as its corresponding CPs.

We then test our method on 100 random situations. The evaluation criteria are the convergence rates and the cost values at different iterations. We compare our method with three other baselines, i.e. the standard problem without any modification (STD), using only the warm start method, and using only the imitation term.

We consider two experiments for the STS task. In Experiment 1, we only change the human's disability type and keep the mass and height constant for all demonstrations and test points. In Experiment 2, we change the mass and the height of the human beside its disability type. The variation of the cost weights is considered only for the ankle, knee, and hip joints. The robot is the same for all of the experiments.

1) Experiment 1: Changing only the human's disability: In this experiment, we assume that we have the same human for the whole experiment whose type of disability is changing. We simulate this by changing the values of $\boldsymbol{R}$, while keeping the values of m and h constant ($m =$ 80 kg and $h = 1.8$ m). For the demonstrations, we choose elements in $\boldsymbol{R}$ corresponding to ankle, knee and hip joints randomly among {1,10,100,1000,10000}. We compute $N =$ 15 demonstrations by solving (7) and (8). During the test phase, we try different situations by initializing the problem with random joint cost weights. Unlike the demonstrations, the system can choose any value between 1 to 10^4 (not limited to the discrete values). We use the logarithmic Euclidean distance for k-NN used for warm starting and extracting desired features $\boldsymbol{p}_d$. Table II shows the results of this experiment for both the human assistance prediction and the robot controller. The proposed method outperforms other approaches; comparing it with the STD method, the combined method reduces the number of iterations by around 90% and 80% for the human assistance prediction and the robot controller, respectively. The data in Table II show that even adding only the imitation term without any warm starting can result in a better convergence rate than the other two methods. The reason why this method affects the human assistance prediction more than the robot controller is that in the former, the cost function is sparse. Namely, it specifies the behavior of the system at a particular time (joint angles and velocities are only relevant at the end of the task) or

TABLE II

Comparing the mean value of number of iterations and convergence time with their standard deviation for different approaches.

		exp1		exp2	
	method	# iterations	conv. time (s)	# iterations	conv. time (s)
Human assistance prediction	STD	80.84 ± 103.77	10.63 ± 18.50	80.24 ± 68.55	10.07 ± 11.15
	Only warm start	53.93 ± 119.67	7.22 ± 19.32	48.62 ± 41.06	5.35 ± 5.77
	Only imitation	14.28 ± 3.28	1.66 ± 0.30	14.31 ± 2.75	1.24 ± 0.65
	Combined	**7.26 ± 3.30**	**1.27 ± 0.58**	**9.33 ± 2.87**	**0.77 ± 0.37**
The robot controller	STD	89.58 ± 71.58	25.23 ± 37.79	101.63 ± 111.81	31.42 ± 53.86
	Only warm start	56.53 ± 74.77	13.72 ± 25.38	57.97 ± 45.00	13.78 ± 16.67
	Only imitation	43.39 ± 27.59	9.30 ± 9.41	46.99 ± 36.97	10.64 ± 13.86
	Combined	**17.90 ± 7.43**	**2.79 ± 1.79**	**28.73 ± 23.78**	**5.32 ± 8.82**

when it goes out of the feasible region (joint limits). Adding the imitation term to this problem provides a trajectory reference to follow and makes the problem less sparse. For the robot controller, however, the robot should already follow a trajectory (the human's hand) and adding another trajectory to follow could be less effective. Nevertheless, the proposed method still has remarkable effects on the number of iterations compared to the other three ones.

We present the value of the cost on a logarithmic at different iterations for all of the methods in Fig. 2. To make a rational comparison between different test points, we normalized all the costs w.r.t. the lowest value achieved from these methods at the 60-*th* iteration. As it is illustrated in this figure, adding the imitation term increases the convergence rate which is more obvious for the points that start far from the local optimum. On the other hand, warm starting helps the system to be initialized at a lower cost. The combined method takes advantages of both methods and outperforms the other ones. The cost value presented in this figure is just for the emulation part c, as the imitation cost is only introduced to help the emulation and is not part of the original task.

2) Experiment 2: Changing mass, height and the human's disability: In this experiment, we repeat Experiment 1, but with varying values for the human's mass and height. We assume that the human's mass varies between 60 to 100 kilograms and the height of the human ranges from 1.6 to 2 meters. To generate $N = 15$ demonstrations, we randomly pick among $\{60, 70, 80, 90, 100\}$ and $\{1.6, 1.7, 1.8, 1.9, 2\}$ to describe the human's mass and height, respectively. However, we do not limit the system to these discrete values for the test points, and the system can choose any value in the corresponding range. Since the data are not in the same range, we define closeness for the k-NN method differently from Experiment 1. We first get the logarithms of the cost weights, then normalize these data as well as the human's mass and height with their mean values and standard deviations. The distances between the demonstrations and the test points are defined with the Euclidean distance of the normalized data. The results of this experiment are presented in Table II and Fig. 2. Comparing with the STD method, the combined method has reduced the number of iterations by around 90% and 70% in the human assistance prediction and the robot controller, respectively. Although using only the imitation term may initialize iLQR with a higher cost value, it can converge even faster than using only the warm starting method, as shown in Fig. 2d.

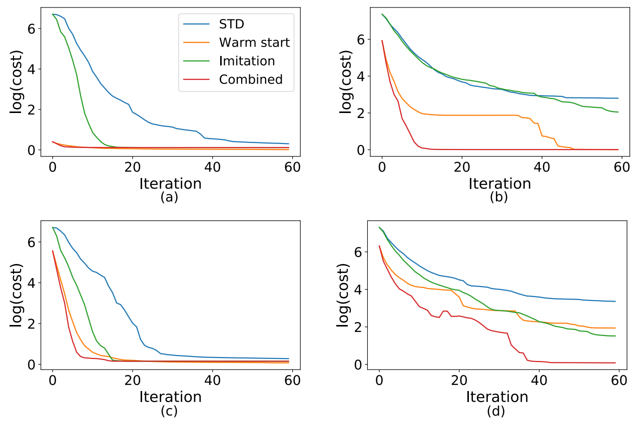

Fig. 2. Comparing the mean values of the normalized costs in logarithmic scale for Experiment 1 (first row) and Experiment 2 (second row). *Left:* The human assistance prediction. *Right:* The robot controller.

V. Discussion

We can evaluate the proposed method with two criteria: 1- how it affects the convergence rate, and 2- how it affects the cost value. Regarding the convergence rate, as it is already discussed, the combined method has notably reduced the number of iterations. As it is also shown in Fig. 2, this method does not have a sensible effect on the cost value, however, it would slightly increase the value of the emulation cost. This behavior is expected, as the system compromises between the emulation part and the imitation part. We can solve this issue by decreasing the value of λ as the number of iterations increases, which is often done in other optimization methods. In this paper, however, we keep the value of λ constant.

In some methods such as feasibility-driven differential dynamic programming (FDDP) [19], it is possible to warm start the controller with the input commands and state variables simultaneously. As studied in [20], initializing with both the control input commands and the state variables is better than predicting just one of them. Our experiments support this observation. Moreover, our method provides a framework to use any feature that can describe the task more accurately. Hence, one can feed the solver with more elaborated features (9) and is not limited only to the optimization variables.

In this experiment, we used k-NN to predict the warm starts or to find the imitation trajectory $\boldsymbol{p}_d$. It worked pretty well in this experiment, but it is not the case for all of the tasks. Using more elaborated techniques like GPR, Gaussian mixture regression (GMR), or neural networks may be more useful for more complicated trajectories. One can also use algorithms proposed in the Learning from Demonstration (LfD) field, such as dynamical movement primitives (DMPs) or task parametrized-GMM (TP-GMM) [21]. We see clear connections between the LfD methods and the optimal control [22]. We hope the proposed idea here could bridge the gap between these two fields.

We presented the idea with the manually-adjusted cost functions, but the idea proposed is independent of a specific cost function and should work for most optimization problems. We will study more accurate cost functions in our future works. Moreover, one of the reasons to work with the manual cost function was that it gave us the freedom to gather random demonstrations. For real applications, it is not possible and we are limited to the available options. Therefore, we may not have access to a diverse and rich demonstration set, for example, we may not be able to gather data from a human whose ankle, knee, and hip joints are severely paralyzed. In such situations, we can program the robot to play the role of the assisted human and ask an expert caregiver to simulate the desired help. This technique which is called switching the role between the robot and the human would be studied more in our future works.

VI. CONCLUSIONS

In this paper, we studied how imitation can be useful in the context of optimal control methods. For this, we modified the cost function by adding an imitation term to follow the desired feature extracted from the demonstrations. Unlike the warm starting methods, the desired feature can be any feature and we are not limited to the input variables of the optimal control method. We tested our algorithm on a simulated assistive task, and we showed that the proposed method had a remarkable effect on the convergence rate of the OCPs, while it does not have any practical effect on the ultimate cost value. In this paper, we studied improvement only in terms of the convergence speed, but we believe that if we extract the features from human-human data, this method can also improve other aspects of the task, such as help the robot to perform tasks more naturally and to be more predictable during the task. These features are important in the tasks where humans have direct interaction with the robot, such as assistive tasks. As a future study, the effectiveness of this method in these areas should be studied as well. Moreover, employing more elaborated learning methods will be considered in future studies. For practical implementation of this method on a real robot, other issues, e.g., unpredictable changes in the human conditions, defining more accurate cost functions, etc. should also be addressed, which are out of the scope of this paper, and would be considered in our future works.

REFERENCES

[1] A. Whiten, N. McGuigan, S. Marshall-Pescini, and L. M. Hopper, "Emulation, imitation, over-imitation and the scope of culture for child and chimpanzee," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 364, no. 1528, pp. 2417–2428, August 2009.

[2] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2018, pp. 6292–6299.

[3] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pp. 4906–4913, 2012.

[4] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *ICINCO*, 2004.

[5] M. Stolle and C. G. Atkeson, "Policies based on trajectory libraries," *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pp. 3344–3349, 2006.

[6] C. Liu and C. G. Atkeson, "Standing balance control using a trajectory library," *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pp. 3031–3036, 2009.

[7] D. Forte, A. Gams, J. Morimoto, and A. Ude, "On-line motion synthesis and adaptation using a trajectory database," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1327–1339, 2012.

[8] N. Mansard, A. Delprete, M. Geisert, S. Tonneau, and O. Stasse, "Using a Memory of Motion to Efficiently Warm-Start a Nonlinear Predictive Controller," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2018, pp. 2986–2993.

[9] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon, "Memory of Motion for Warm-Starting Trajectory Optimization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 2594–2601, 2020.

[10] K. Mombaur, A. Truong, and J. P. Laumond, "From human to humanoid locomotion-an inverse optimal control approach," *Autonomous Robots*, vol. 28, no. 3, pp. 369–383, 2010.

[11] G. Canal, E. Pignat, G. Alenya, S. Calinon, and C. Torras, "Joining high-level symbolic planning with low-level motion primitives in adaptive HRI: application to dressing assistance," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2018, pp. 3273–3278.

[12] T. Tamei, T. Matsubara, A. Rai, and T. Shibata, "Reinforcement learning of clothing assistance with a dual-arm robot," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, 2011, pp. 733–738.

[13] M. Geravand, P. Z. Korondi, C. Werner, K. Hauer, and A. Peer, "Human sit-to-stand transfer modeling towards intuitive and biologically-inspired robot assistance," *Autonomous Robots*, vol. 41, no. 3, pp. 575–592, 2017.

[14] H. El-Hussieny, A. Asker, and O. Salah, "Learning the sit-to-stand human behavior: An inverse optimal control approach," *13th International Computer Engineering Conference (ICENCO)*, pp. 112–117, 2017.

[15] J. Li, L. Lu, L. Zhao, C. Wang, and J. Li, "An integrated approach for robotic Sit-To-Stand assistance: Control framework design and human intention recognition," *Control Engineering Practice*, vol. 107, p. 104680, 2021.

[16] A. Tözeren, *Human body dynamics: classical mechanics and human movement*. Springer Science & Business Media, 1999.

[17] R. Drillis, R. Contini, and M. Bluestein, "Body Segment Parameters; a Survey of Measurement Techniques." *Artificial limbs*, no. 2, pp. 44–66, 1964.

[18] S. Calinon, "Mixture models for the analysis, edition, and synthesis of continuous time series," in *Mixture Models and Applications*, N. Bouguila and W. Fan, Eds. Springer, Cham, 2019, pp. 39–57.

[19] M. Giftthaler, M. Neunert, M. Stäuble, J. Buchli, and M. Diehl, "A Family of Iterative Gauss-Newton Shooting Methods for Nonlinear Optimal Control," pp. 1–9, 2018.

[20] T. S. Lembono, C. Mastalli, P. Fernbach, N. Mansard, and S. Calinon, "Learning how to walk: Warm-starting optimal control solver with memory of motion," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2020, pp. 1357–1363.

[21] S. Calinon and D. Lee, "Learning control," in *Humanoid Robotics: a Reference*, P. Vadakkepat and A. Goswami, Eds. Springer, 2019, pp. 1261–1312.

[22] S. Calinon, "Learning from demonstration (programming by demonstration)," in *Encyclopedia of Robotics*, M. H. Ang, O. Khatib, and B. Siciliano, Eds. Springer, 2019.