Inverse Optimal Control for Multiphase Cost Functions

Wanxin Jin, Dana Kulić, Jonathan Feng-Shun Lin, Shaoshuai Mou, Sandra Hirche

***Abstract*—We consider a dynamical system whose trajectory is a result of minimizing a multiphase cost function. The multiphase cost function is assumed to be a weighted sum of specified features (or basis functions) with phase-dependent weights that switch at some unknown phase transition points. A new inverse optimal control approach for recovering the cost weights of each phase and estimating the phase transition points is proposed. The key idea is to use a length-adapted window moving along the observed trajectory, where the window length is determined by finding the minimal observation length that suffices for a successful cost weight recovery. The effectiveness of the proposed method is first evaluated on a simulated robot arm, and then demonstrated on a dataset of human participants performing a series of squatting tasks. The results demonstrate that the proposed method reliably retrieves the cost function of each phase and segments each phase of motion from the trajectory with a segmentation accuracy above 90%.**

***Index Terms*—Inverse optimal control, multiphase cost functions, human motion segmentation, recovery matrix.**

I. Introduction

INVERSE optimal control (IOC) techniques are able to find a latent cost function that explains an observed trajectory generated by a system under the corresponding optimal control policy. The technique has been applied to many fields from robotics to biomechanics, where knowing the underlying cost function enables applications such as apprenticeship learning [1], human-robot collaboration [2], human motion analysis [3], etc.

Most existing IOC studies assume that the observed system trajectory is generated from a *stationary* cost function, which is constructed as a linear combination of given features (or basis functions) with unknown weights. Though valid in many cases, these methods are restrictive when applied to complex or long-term behaviors where different cost functions may be active in different phases. For example, prior studies [4], [5] show that human motions tend to minimize different costs in different circumstances depending on tasks and environmental conditions.

This has motivated us to investigate the IOC problem for *multiphase* cost functions. We consider the observed trajectory as a concatenation of multiple phases of motion, where each phase is characterized by a distinct cost function parameterized as a linear combination of given features with phase-dependent cost weights. We focus on not only recovering the cost weights of each phase, but also identifying the boundary of each phase in the observed trajectory.

W. Jin and S. Mou are with the School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN 47906 USA (e-mail: wanxinjin@gmail.com; mous@purdue.edu); D. Kulić is with Monash University, Melbourne, Australia (e-mail: dana.kulic@monash.edu); J. F. Lin is with the Department of Electrical Computer Engineering, University of Waterloo, Waterloo, Canada (e-mail: jonathan.lin@uwaterloo.ca); S. Hirche is with the Chair of Information-oriented Control, Technical University of Munich, Munich, Germany (email: hirche@lsr.ei.tum.de); (*Corresponding author: Wanxin Jin*)

A. Related Work

Prior IOC methods can be grouped into two classes. One is a nested structure where the forward optimal control problem is computed repeatedly in an inner loop while the cost function (cost weights) is updated in the outer loop. This formulation was first employed to learn cost functions for Markov decision processes, where IOC is also known as inverse reinforcement learning. Representative works include [6], where the cost weights are computed by maximizing the margin between costs of predicted and observed trajectories, and [7], where the weights are solved to maximize the entropy of the trajectory probability distribution while matching the feature values of observation. In [8], cost weights are computed by minimizing the deviation of predicted trajectories from the observed ones. Despite successful applications such as robot navigation [9] and autonomous driving [10], a shortcoming with these nested methods is the huge computational cost for repeatedly solving the optimal control problem in the inner loop.

The second category of IOC methods directly computes the cost weights using a set of optimality equations. In those methods, the extent to which the optimality conditions are violated by the observed trajectories is evaluated. Representative works include [11], where Karush-Kuhn-Tucker (KKT) conditions are used, [12] using the Pontryagin's maximum principle, and [13] using Euler-Lagrange equations. Recently, [14] developed an inverse KKT approach to address contacts and constraints in robot manipulations. Recoverability for IOC problems has also been considered in those methods; for instance, if a system remains at an equilibrium point, the trajectory, though satisfying optimality conditions, cannot be used for cost function recovery. In [15], the authors propose a sufficient condition to check the recoverability of cost weights.

To our knowledge, theoretical formulations and solutions to the IOC problem for multiple-phase cost functions appear to be very rare, although direct multiphase optimal control is well-established [16]. The most relevant work is [4], where human motion is segmented to multiple phases by discriminating different underlying cost functions. The authors process the trajectory by a sliding window of fixed length (manually set), and the weights are solved by minimizing the KKT conditions [17]. Although successfully segmenting motion trajectories, the KKT method under data of incomplete trajectory cannot ensure correct recovery of cost functions, as demonstrated later in this paper.

B. Contributions

To address the limitations of existing methods in handling multiphase cost function recovery problems, we develop a

new IOC technique. We hypothesize that the observed system trajectory is a result of minimizing a multiphase cost function, which at each phase is constructed as a linear combination of selected features with unknown phase-dependent cost weights. We recover the cost weights of each phase and estimate phase transition points by using an adaptive-length window sliding along the trajectory. At each time step, the cost weights under the window are incrementally computed. The window length is determined based on the *recovery matrix* [18], which seeks to find the minimal observations that suffice for a successful cost weight recovery. The output of the recovery is a sequence of cost weights indexed by time, which describes the changing of cost function over time. Based on the output, each phase can be segmented.

The advantages of the proposed IOC method include: (i) it takes as input a union feature set that contains all relevant phase features, and may also include additional irrelevant phase features, which facilitates applications where the knowledge of exact features is not available; (ii) as phase transition points are estimated, each phase is extracted, thereby providing a solution to motion segmentation or identification problems; and (iii) as the computation of the cost weights is performed incrementally, the method can be used in on-line settings.

We apply the proposed method to a human motion experiment. Using a dataset of participants performing a series of squatting tasks, we applied the proposed method to recover the multiphase cost function underlying the continuous human motion trajectory, and then segmented the motion into different phases. The experimental results illustrate the capability of the proposed method to analyze real motion datasets.

The rest of the paper is constructed as follows. The problem is formulated in Section II. In Section III, we focus on recovering the cost function within a single phase. We then propose the multiphase IOC method in Section IV. The experimental results are presented in Section V. Discussions are presented in Section VI, and conclusions drawn in Section VII.

C. Notation

The column operator $\text{col}\,\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_k\}$ stacks its vector arguments into a column (i.e. concatenation vector). $\boldsymbol{x}_{k:k+n}$ denotes a stack of multiple vectors from time k to $k+n$, that is, $\boldsymbol{x}_{k:k+n} = \text{col}\,\{\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, ...\boldsymbol{x}_{k+n}\}$. $\boldsymbol{A}$ (bold-type) denotes a block matrix. Given a vector function $\boldsymbol{f}(\boldsymbol{x})$ and a constant $\boldsymbol{x}^*$, $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}^*}$ denotes the Jacobian matrix with respect to $\boldsymbol{x}$ evaluated at $\boldsymbol{x}^*$. Zero matrix/vector is denoted as $\boldsymbol{0}$, and identity matrix denoted as $\boldsymbol{I}$, both with appropriate dimensions. $\sigma_i(\boldsymbol{A})$ denotes the ith smallest singular value of matrix $\boldsymbol{A}$. $\boldsymbol{A}'$ denotes the transpose of matrix $\boldsymbol{A}$.

II. Problem Formulation

Consider the discrete-time system[1]

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_{k+1}), \quad \boldsymbol{x}_0 \in \mathbb{R}^n, \tag{1}$$

where $\boldsymbol{f}(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$ is differentiable; $\boldsymbol{x}_k \in \mathbb{R}^n$ denotes the state; $\boldsymbol{u}_k \in \mathbb{R}^m$ is the control input; and k is the time index. Suppose that the system trajectory of states and inputs $(\boldsymbol{x}^*_{1:T}, \boldsymbol{u}^*_{1:T})$ is a sequential concatenation of p phases, and each phase minimizes a *different* cost function. The overall cost of the system trajectory is denoted by

$$J(\boldsymbol{x}_{1:T}, \boldsymbol{u}_{1:T}) = \sum_{j=1}^{p} \sum_{k=T_j+1}^{T_{j+1}} C^{(j)}(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{2}$$

where $C^{(j)}(\boldsymbol{x}, \boldsymbol{u})$ is the running cost of the jth-phase motion, and interval $(T_j, T_{j+1}]$ is the time horizon for the jth-phase motion with $T_1 = 0$ and $T_{p+1} = T$. Here, we call T_j $(1 < j \leq p)$ a *phase transition point*. We construct the jth-phase running cost as

$$C^{(j)}(\boldsymbol{x}, \boldsymbol{u}) = {\boldsymbol{\omega}^{(j)}}' \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{u}), \tag{3}$$

where $\boldsymbol{\phi} = [\phi_1, \phi_2, \cdots, \phi_r]' \in \mathbb{R}^r$ is called the *union feature vector*, and $\boldsymbol{\omega}^{(j)} \in \mathbb{R}^r$ are the *cost weights* of the jth-phase motion. We say that the feature

$$\phi_i \text{ is } \begin{cases} relevant & \text{if } \ \omega_i^{(j)} \neq 0 \\ irrelevant & \text{otherwise} \end{cases} \tag{4}$$

for the jth-phase motion. Here, $\omega_i^{(j)}$, i.e. the ith entry of $\boldsymbol{\omega}^{(j)}$, is the cost weight for ϕ_i with $1 \leq i \leq r$.

Given the observed trajectory $(\boldsymbol{x}^*_{1:T}, \boldsymbol{u}^*_{1:T})$ and the union feature vector $\boldsymbol{\phi}$, we aim to (i) recover the cost weights $\boldsymbol{\omega}^{(j)}$ of each phase and (ii) estimate the location of the phase transition points T_j $(1 < j \leq p)$. Note that the cost weights of each phase can only be recovered up to a non-zero scaling [15]. Thus, for the jth-phase motion $(1 \leq j \leq p)$, we call the recovered cost weights $\hat{\boldsymbol{\omega}}^{(j)}$ a *successful recovery* if $\hat{\boldsymbol{\omega}}^{(j)} = c\boldsymbol{\omega}^{(j)}$ with $c > 0$; specific c can be obtained by normalization [11].

III. Single-Phase Cost Function Recovery Using the Recovery Matrix

In this section, we first focus on recovering the cost function within a single phase. As developed in our previous work [18], we aim to find the minimal number of observations of state-input pairs that suffices for a successful recovery.

We consider a segment of the state and input trajectory of system (1), denoted as $(\boldsymbol{x}^*_{t:t+l-1}, \boldsymbol{u}^*_{t:t+l-1})$, where t represents the *observation starting time* and l is called the *observation length*. Suppose that $(\boldsymbol{x}^*_{t:t+l-1}, \boldsymbol{u}^*_{t:t+l-1})$ are from a single phase, say the jth phase (that is, $T_j < t \leq t+l-1 \leq T_{j+1}$). Using Pontryagin's maximum principle [20], we have

$$\boldsymbol{\lambda}_k - \frac{\partial \boldsymbol{f}'}{\partial x_k^*}\boldsymbol{\lambda}_{k+1} - \frac{\partial \boldsymbol{\phi}'}{\partial x_k^*}\boldsymbol{\omega}^{(j)} = \boldsymbol{0} \tag{5}$$

$$\frac{\partial \boldsymbol{f}'}{\partial u_k^*}\boldsymbol{\lambda}_k + \frac{\partial \boldsymbol{\phi}'}{\partial u_k^*}\boldsymbol{\omega}^{(j)} = \boldsymbol{0} \tag{6}$$

for any $t \leq k \leq t+l-1$, where $\boldsymbol{\lambda}_k$ is the costate for the optimal control system. By writing the above equations in matrix form for all $t \leq k \leq t+l-1$, we have [18]

$$\boldsymbol{F}_x(t,l)\boldsymbol{\lambda}_{t:t+l-1} - \boldsymbol{\Phi}_x(t,l)\boldsymbol{\omega}^{(j)} = \boldsymbol{V}(t,l)\boldsymbol{\lambda}_{t+l} \tag{7}$$

$$\boldsymbol{F}_u(t,l)\boldsymbol{\lambda}_{t:t+l-1} + \boldsymbol{\Phi}_u(t,l)\boldsymbol{\omega}^{(j)} = \boldsymbol{0} \tag{8}$$

[1] In (1), at time k, we denote the control input as $\boldsymbol{u}_{k+1}$ instead of $\boldsymbol{u}_k$ for notation simplicity of following discussions, as adopted in [19].

where $\boldsymbol{\lambda}_{t:t+l-1} := \text{col}\{\boldsymbol{\lambda}_t, \cdots, \boldsymbol{\lambda}_{t+l-1}\}$ and

$$\boldsymbol{F}_x(t,l) = \begin{bmatrix} I & \frac{-\partial \boldsymbol{f}'}{\partial \boldsymbol{x}_t^*} & & \\ 0 & I & \ddots & \\ & & \ddots & \frac{-\partial \boldsymbol{f}'}{\partial \boldsymbol{x}_{t+l-2}^*} \\ 0 & & \cdots & I \end{bmatrix} \quad \boldsymbol{\Phi}_x(t,l) = \begin{bmatrix} \frac{\partial \boldsymbol{\phi}'}{\partial x_t^*} \\ \frac{\partial \boldsymbol{\phi}'}{\partial x_{t+1}^*} \\ \vdots \\ \frac{\partial \boldsymbol{\phi}'}{\partial x_{t+l-1}^*} \end{bmatrix}$$

$$\boldsymbol{F}_u(t,l) = \begin{bmatrix} \frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{u}_t^*} & & & \\ & \frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{u}_{t+1}^*} & & \\ & & \ddots & \\ & & & \frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{u}_{t+l-1}^*} \end{bmatrix} \quad \boldsymbol{\Phi}_u(t,l) = \begin{bmatrix} \frac{\partial \boldsymbol{\phi}'}{\partial u_t^*} \\ \frac{\partial \boldsymbol{\phi}'}{\partial u_{t+1}^*} \\ \vdots \\ \frac{\partial \boldsymbol{\phi}'}{\partial u_{t+l-1}^*} \end{bmatrix}$$

$$\boldsymbol{V}(t,l) = \begin{bmatrix} \boldsymbol{0} & \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_{t+l-1}^*} \end{bmatrix}'_{\in \mathbb{R}^{nl \times n}} \tag{9}$$

with $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_T^*} = \boldsymbol{I}$. The dimensions of the above matrices are: $\boldsymbol{F}_x(t,l) \in \mathbb{R}^{ln \times ln}, \boldsymbol{F}_u(t,l) \in \mathbb{R}^{lm \times ln}, \boldsymbol{\Phi}_x(t,l) \in \mathbb{R}^{ln \times r}$ and $\boldsymbol{\Phi}_u(t,l) \in \mathbb{R}^{lm \times r}$. To facilitate the computation of the cost weights $\boldsymbol{\omega}^{(j)}$ in (7) and (8) (by combining the two equations and eliminating the unknown variables $\boldsymbol{\lambda}_{t:t+l-1}$), we define the following *recovery matrix* [18]:

$$\boldsymbol{H}(t,l) = \begin{bmatrix} \boldsymbol{H}_1(t,l) & \boldsymbol{H}_2(t,l) \end{bmatrix} \in \mathbb{R}^{ml \times (r+n)}, \tag{10}$$

with

$$\boldsymbol{H}_1(t,l) = \boldsymbol{F}_u(t,l)\boldsymbol{F}_x^{-1}(t,l)\boldsymbol{\Phi}_x(t,l) + \boldsymbol{\Phi}_u(t,l) \tag{11}$$

$$\boldsymbol{H}_2(t,l) = \boldsymbol{F}_u(t,l)\boldsymbol{F}_x^{-1}(t,l)\boldsymbol{V}(t,l). \tag{12}$$

Then, the cost weights satisfy the equation

$$\boldsymbol{H}(t,l) \begin{bmatrix} \boldsymbol{\omega}^{(j)} \\ \boldsymbol{\lambda}_{k+l} \end{bmatrix} = \boldsymbol{0}. \tag{13}$$

The recovery matrix $\boldsymbol{H}(t,l)$ has the following properties (the interested reader can refer to [18] for detailed proofs):

(1) iterative property: for any $1 \le t \le t+l-1 < T$,

$$\begin{aligned} \boldsymbol{H}(t,l+1) &= \begin{bmatrix} \boldsymbol{H}_1(t,l+1) & \boldsymbol{H}_2(t,l+1) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{H}_1(t,l) & \boldsymbol{H}_2(t,l) \\ \frac{\partial \boldsymbol{\phi}'}{\partial \boldsymbol{u}_{t+l}^*} & \frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{u}_{t+l}^*} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \frac{\partial \boldsymbol{\phi}'}{\partial \boldsymbol{x}_{t+l}^*} & \frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{x}_{t+l}^*} \end{bmatrix} \end{aligned} \tag{14}$$

with initial $\boldsymbol{H}(t,1)$ for the single observation of $(\boldsymbol{x}_t^*, \boldsymbol{u}_t^*)$

$$\begin{aligned} \boldsymbol{H}(t,1) &= \begin{bmatrix} \boldsymbol{H}_1(t,1) & \boldsymbol{H}_2(t,1) \end{bmatrix} \\ &= \begin{bmatrix} (\frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{u}_t^*} \frac{\partial \boldsymbol{\phi}'}{\partial \boldsymbol{x}_t^*} + \frac{\partial \boldsymbol{\phi}'}{\partial \boldsymbol{u}_t^*}) & \frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{u}_t^*} \frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{x}_t^*} \end{bmatrix}. \end{aligned} \tag{15}$$

(2) rank non-decreasing: for any $1 \le t \le t+l-1 < T$,

$$\text{rank}\, \boldsymbol{H}(t,l) \le \text{rank}\, \boldsymbol{H}(t,l+1). \tag{16}$$

(3) rank upper bound: for any $T_j < t \le t+l-1 \le T_{j+1}$ with $1 \le j \le p$,

$$\text{rank}\, \boldsymbol{H}(t,l) \le r+n-1. \tag{17}$$

Based on the recovery matrix and its properties, the lemma below provides a method for using the minimal observation length to recover the cost weights within a single phase [18].

Lemma 1. *Suppose that the observations of the state and input trajectory of the system (1) start from time t and are within the jth phase, i.e. $T_j < t \le t+l-1 \le T_{j+1}$. The recovery matrix $\boldsymbol{H}(t,l)$ is incrementally updated with new observations via (14)-(15). The minimal observation length that suffices for a successful recovery of the jth phase cost weights $\boldsymbol{\omega}^{(j)}$, defined as $l_{\min}(t)$, is*

$$l_{\min}(t) = \min\left\{l \,|\, \text{rank}\, \boldsymbol{H}(t,l) = r+n-1\right\}. \tag{18}$$

If a vector $\text{col}\{\hat{\boldsymbol{\omega}}, \hat{\boldsymbol{\lambda}}\} \neq \boldsymbol{0}$ *with* $\hat{\boldsymbol{\omega}} \in \mathbb{R}^r$ *is a solution to*

$$\boldsymbol{H}(t, l_{\min}(t)) \begin{bmatrix} \hat{\boldsymbol{\omega}} \\ \hat{\boldsymbol{\lambda}} \end{bmatrix} = \boldsymbol{0}, \tag{19}$$

then $\hat{\boldsymbol{\omega}}$ is a successful recovery for $\boldsymbol{\omega}^{(j)}$.

Proof. Please refer to [18] for the proof. ■

Lemma 1 states that the cost weights of a single phase can be recovered using minimal observations of state-input pairs of that phase: once the nullity of the recovery matrix is one, then any non-zero vector in the kernel corresponds to a successful recovery. Using Lemma 1, we assume that the horizon of the jth phase satisfies $T_{j+1} - T_j \ge l_{\min}(t)$; and due to (16) and (17), $\text{rank}\, \boldsymbol{H}(t, l_{\min}(t)) \le \text{rank}\, \boldsymbol{H}(T_j+1, T_{j+1}-T_j) \le r+n-1$. We thus establish the following assumption for the cost weight 'recoverability' of each phase.

Assumption 1. *Given the union feature vector ϕ in (3) and the recovery matrix defined in (10), the state and input trajectory of the j-th phase $(\boldsymbol{x}_{T_j+1:T_{j+1}}^*, \boldsymbol{u}_{T_j+1:T_{j+1}}^*)$ satisfies*

$$\text{rank}\, \boldsymbol{H}(T_j+1, T_{j+1}-T_j) = r+n-1. \tag{20}$$

for all $j = 1, 2, \ldots, p$.

Assumption 1 provides the condition under which the cost weights of each phase can at least be recovered using the whole phase horizon. The validity of this assumption depends on the informativeness of the data in each phase [18] and also the union feature set used. To understand this, we consider the following contradiction: assume that given a very large union feature set, there exist non-unique feature combinations which can be used to characterize the jth phase, i.e. there exist two independent cost weight vectors, say $\boldsymbol{\omega}^{(j)}$ and $\tilde{\boldsymbol{\omega}}^{(j)}$, for which (7) and (8) hold, and it follows that $\text{rank}\, \boldsymbol{H}(T_j+1, T_{j+1}-T_j) < r+n-1$ [18].

IV. Multiple-phase Cost Function Recovery

Based on the previous discussions for the single-phase case, we now consider multiphase cost function recovery.

A. The Approach

Under Assumption 1, the idea for recovering the multiphase cost function is to compute the cost weights over time using an observation window moving along the trajectory $(\boldsymbol{x}_{1:T}^*, \boldsymbol{u}_{1:T}^*)$. The procedure includes two ingredients: first, a window, with the starting position at time t with the adaptive length denoted as $l(t)$, moves forward along the trajectory while recovering the cost weights, denoted as $\hat{\boldsymbol{\omega}}(t)$, using the trajectory data within that window via (19); second, in an inner loop, the window length $l(t)$ is incrementally determined by finding the

minimal observation length $l_{\min}(t)$ defined in (18). Therefore, the procedure is a recursive application of Lemma 1 along the trajectory, and the output is the recovered weights $\hat{\boldsymbol{\omega}}(t)$. Note that the index t in $\hat{\boldsymbol{\omega}}(t)$ and $l(t)$ is used to indicate where the corresponding observation window starts.

For the above process, we analyze the following two cases: first, the observations and recovery are performed within the same phase, as shown in Fig. 1, and second, the observations are crossing a phase transition point, as shown in Fig. 2.

1) Observations within the Same Phase: without loss of generality, we suppose that a window with starting time t is in the jth phase, as illustrated in Fig.1. When the minimal observation window length $l_{\min}(t)$ (18) is successfully found within the same phase, i.e. $T_j < t \leq t+l_{\min}(t)-1 \leq T_{j+1}$, as shown in Fig. 1, the recovery process is just the single-phase case as discussed in the previous section.

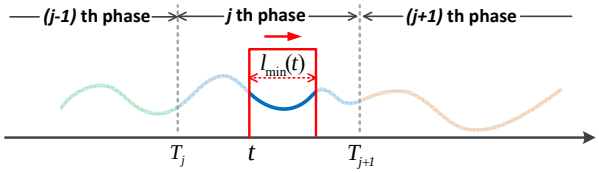


Figure 1: An illustration where the minimal observation length is found within the same phase. Here, the observation window is colored in red.

2) Observations over a Phase Transition Point: we now focus on the case where the window (with the starting time t) is over a phase transition point, say T_{j+1}, as illustrated in Fig.2. This case only happens when the observations from t to T_{j+1} (i.e. of length $T_{j+1}-t+1$ as shown in the upper panel in Fig. 2) cannot reach the minimal observation length (18), that is, $\operatorname{rank} \boldsymbol{H}(t, T_{j+1}-t+1) < n+r-1$. In what follows, based on the bottom panel in Fig. 2, we fix the window starting time t and discuss the rank values of $\boldsymbol{H}(t,l)$ while increasing l from $(T_{j+1}-t+1)$. First, we have the following lemma.

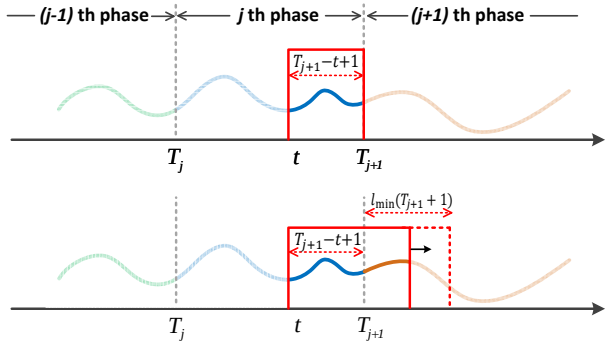


Figure 2: An illustration when the window is over a phase transition point. The upper panel shows the case where the window ends at T_{j+1}; the bottom panel shows that the window length increases to include the data of the $(j+1)$th phase.

Lemma 2. *Suppose that Assumption 1 holds, and the window starts at time $t \in (T_j, T_{j+1}]$ which satisfies $\operatorname{rank} \boldsymbol{H}(t, T_{j+1}-t+1) < n+r-1$. Then there exists an observation length l with*

$$l \in (T_{j+1}-t+1, T_{j+1}-t+1+l_{\min}(T_{j+1}+1)] \tag{21}$$

such that

$$\operatorname{rank} \boldsymbol{H}(t,l) \geq r+n-1 \tag{22}$$

where $l_{\min}(T_{j+1}+1)$ denotes the minimal observation length from the starting time $T_{j+1}+1$.

Proof. Proof by contradiction: assume that $\operatorname{rank} \boldsymbol{H}(t,l) < r+n-1$ holds for all $T_{j+1}-t+1 < l \leq T_{j+1}-t+1+l_{\min}(T_{j+1}+1)$. However, due to the rank non-decreasing in (16), we have

$$\begin{aligned}&\operatorname{rank} \boldsymbol{H}(t, T_{j+1}-t+1+l_{\min}(T_{j+1}+1))\\ &\quad \geq \operatorname{rank} \boldsymbol{H}(T_{j+1}+1, l_{\min}(T_{j+1}+1)) = r+n-1.\end{aligned}$$

contradicting the assumption. This completes the proof. ■

Based on Lemma 2, we consider the following two subcases when increasing l to include the data of the next phase:

Case A: if there exists l such that $\operatorname{rank} \boldsymbol{H}(t,l) = r+n-1$ holds, we still can compute non-trivial weights $\hat{\boldsymbol{\omega}}$ through (19); however, such $\hat{\boldsymbol{\omega}}$ may not be a successful recovery of $\boldsymbol{\omega}^{(j)}$ as it is computed using the data from two phases, in this case we call $\hat{\boldsymbol{\omega}}$ a *"degenerate recovery"*; and

Case B: otherwise, increasing l will result in a *direct* jump to $\operatorname{rank} \boldsymbol{H}(t,l) = r+n$. In this case, just from the rank value we can say that the window includes a phase transition point.

From the above discussions, we note that when the observations from the previous phase are not sufficient to produce a successful recovery (i.e. the window starting time t satisfying $\operatorname{rank} \boldsymbol{H}(t, T_{j+1}-t+1) < n+r-1$), increasing the window length to include the data of the next phase may not lead to $\operatorname{rank} \boldsymbol{H}(t,l) = n+r$ due to degenerate recoveries. This will lead to a possible failure to detect the phase transition points by only observing the rank condition of the recovery matrix (However, as we shall discuss in Section V and VI, in practice we have not encountered this issue).

To circumvent the limitations due to degenerate recovery, we use the cost weights computed at each window to facilitate the estimation of phase transition points. Specifically, as the observation window moves along the trajectory, at any starting time $t \in (T_j, T_{j+1}], (1 \leq j \leq p)$, the cost weights, denoted as $\hat{\boldsymbol{\omega}}(t)$, are computed by:

Case A: if increasing l results in $\operatorname{rank} \boldsymbol{H}(t,l) = n+r-1$, here denoted as $l_{\min}(t)$, compute the weights $\hat{\boldsymbol{\omega}}(t)$ via (19).

Case B: otherwise (increasing l only leads to $\operatorname{rank} \boldsymbol{H}(t,l) = n+r$, indicating that the window includes a phase transition point), set $\hat{\boldsymbol{\omega}}(t) = \hat{\boldsymbol{\omega}}(t-1)$.

Consequently, by checking the changes of $\hat{\boldsymbol{\omega}}(t)$ over time t, the phase transition point T_{j+1} can be estimated. Considering the degenerate recovery that may happen in Case A, the estimated phase transition point, denoted as $\hat{T}_{j+1}$, is always bounded by

$$t_{\max}^{(j)} \leq \hat{T}_{j+1} \leq T_{j+1} \tag{23}$$

where $t^{(j)}_{\max}$ is the last starting time in the jth phase such that $\operatorname{rank} \boldsymbol{H}(t, T_{j+1}-t+1) = n+r-1$ holds, that is,

$$\begin{aligned} t^{(j)}_{\max} = \quad & \arg\max \ t \\ \text{s.t.} \quad & \operatorname{rank} \boldsymbol{H}(t, T_{j+1}-t+1) = n+r-1. \end{aligned}$$

As we will demonstrate later, when the observation window includes a phase transition point, degenerate recoveries happen infrequently. This is because in most cases the trajectory data of the next phase always violates the optimality condition of the previous phase. Thus, usually when the window includes a small number of data points of the next phase, the rank of the recovery matrix immediately jumps to $(n+r)$.

B. Implementation

We first describe our implementation for checking the rank condition (18) and computing the weights in (19) considering data noise, near-optimality of trajectories, and computational error. To verify $\operatorname{rank} \boldsymbol{H}(t,l) = n+r-1$, we check the metric

$$\kappa(t,l) = \frac{\sigma_2(\bar{\boldsymbol{H}}(t,l))}{\sigma_1(\bar{\boldsymbol{H}}(t,l))} \geq \gamma \quad \text{with} \quad l \geq \frac{r+n}{m}. \tag{24}$$

Here, $\bar{\boldsymbol{H}}(t,l) = \frac{\boldsymbol{H}(t,l)}{\|\boldsymbol{H}(t,l)\|_F}$ is the normalized recovery matrix where $\|\cdot\|_F$ is the Frobenius norm; and γ is a rank index threshold. We use the metric $\kappa(t,l)$ to check the rank condition because when (18) holds, $\kappa(t,l)$ will increase to infinity, thus facilitating the rank check process [18]. For the computation of (19), we use the following minimization

$$\begin{bmatrix} \hat{\boldsymbol{\omega}} \\ \hat{\boldsymbol{\lambda}} \end{bmatrix} = \arg \min_{\operatorname{col}\{\hat{\boldsymbol{\omega}}, \hat{\boldsymbol{\lambda}}\}} \left\| \boldsymbol{H}(t, l_{\min}(t)) \begin{bmatrix} \hat{\boldsymbol{\omega}} \\ \hat{\boldsymbol{\lambda}} \end{bmatrix} \right\| \tag{25}$$

where $\|\cdot\|$ denotes the l_2-norm, and to avoid trivial solutions we normalize $\operatorname{col}\{\hat{\boldsymbol{\omega}}, \hat{\boldsymbol{\lambda}}\}$ such that $\sum_{i=1}^{r} \hat{\omega}_i = 1$.

Based on the above rules, we now consider the implementation of the recovery procedure to obtain $\hat{\boldsymbol{\omega}}(t)$. Suppose that the window starts at time t. We increase its length $l(t)$ while examining the validity of (24):

Case A: if (24) is fulfilled for a certain increased $l(t)$, here, denoted as $l_{\min}(t)$, then we compute $\hat{\boldsymbol{\omega}}(t)$ via (25).

Case B: otherwise; if (24) cannot be fulfilled for *any* $l(t)$ from $\lceil \frac{r+n}{m} \rceil$ ($\lceil \cdot \rceil$ is ceiling operator) to $T-t+1$, set $\hat{\boldsymbol{\omega}}(t) = \hat{\boldsymbol{\omega}}(t-1)$.

To reduce computational cost, in Case B we do not necessarily need to verify for all $\lceil \frac{r+n}{m} \rceil \leq l \leq T-t+1$ (as indicated by Lemma 3 in Section VI); instead, we use a maximum window length $l_{\max}$ and only examine (24) for $l(t)$ from $\lceil \frac{r+n}{m} \rceil$ to $l_{\max}$. Here, $l_{\max}$ should be larger than any phase horizon, i.e.

$$l_{\max} > T_{j+1} - T_j, \quad \forall\ 1 \leq j \leq p. \tag{26}$$

Thus, we summarize the computation of $\hat{\boldsymbol{\omega}}(t)$ as

$$\hat{\boldsymbol{\omega}}(t) = \begin{cases} \text{computed via (25)}, & \text{if } l_{\min}(t) < l_{\max} \\ \hat{\boldsymbol{\omega}}(t-1), & \text{otherwise} \end{cases} \tag{27}$$

where $l_{\min}(t) < l_{\max}$ means that the window length satisfying (24) is found within $l_{\max}$. Note that in (27) the cost weights for the data points near the trajectory terminal, which do not suffice for a minimal observation length, are also considered: their values remain the same as the previous recovery results.

The overall implementation for recovering the multiphase cost functions is presented in Algorithm 1. We will show how to select the parameters γ and $l_{\max}$ in the experiment section.

Algorithm 1: IOC for multiphase cost functions

Input: trajectory observations ($\boldsymbol{x}^*_{1:T}$,$\boldsymbol{u}^*_{1:T}$);
a union feature vector $\boldsymbol{\phi}$.
Output: recovered cost weights $\hat{\boldsymbol{\omega}}(t)$ at time $t = 1, 2, \cdots, T$.
Parameter: rank index threshold γ (24);
maximum window length $l_{\max}$ (26).
for $t = 1 : T$ **do**
  initialize observation length $l(t)$=$\lceil \frac{r+n}{m} \rceil$;
  initialize the recovery matrix $\boldsymbol{H}(t, l(t))$ (14-15);
  normalize to obtain $\bar{\boldsymbol{H}}(t, l(t))$ (24);
  while $l(t) < l_{\max}$ and *not satisfying (24)* **do**
    extend the observation size $l(t) = l(t) + 1$;
    take the next observation $(\boldsymbol{x}^*_{t+l(t)}, \boldsymbol{u}^*_{t+l(t)})$;
    update $\boldsymbol{H}(t, l(t))$ (14);
    normalize to obtain $\bar{\boldsymbol{H}}(t, l(t))$ (24);
  end
  compute the cost weights $\hat{\boldsymbol{\omega}}(t)$ via (27).
end

V. Experiments

The proposed method is tested in two sets of experiments: (i) we first evaluate the method on a simulated two-link robot arm where the ground truth cost function is known; then (ii) we apply the method to segment continuous human motion.

A. Simulated Robot Arm

On a two-link robot arm, we evaluate the proposed method in terms of observation noise, parameter settings, and comparison with a state-of-the-art method. We define the recovery error e_ω to quantify the multiphase IOC accuracy:

$$e_\omega = \frac{\sum_{t=1}^{T} \inf_{c \neq 0} \|c\hat{\boldsymbol{\omega}}(t) - \boldsymbol{\omega}(t)\|}{T} \tag{28}$$

where T is the overall horizon, and the ground truth $\boldsymbol{\omega}(t) = \boldsymbol{\omega}^{(j)}$ for $T_j < t \leq T_{j+1}$ with $j = 1, 2, \cdots, p$.

The dynamics of a two-link arm is given by [21, p. 209]

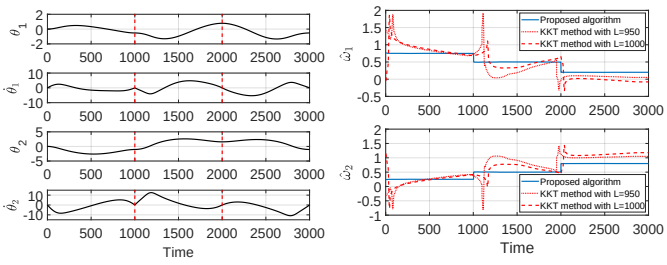
$$M(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + C(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \boldsymbol{g}(\boldsymbol{\theta}) = \boldsymbol{\tau}, \tag{29}$$

where $\boldsymbol{\theta} = [\theta_1, \theta_2]' \in \mathbb{R}^2$ is the joint angle vector; $M(\boldsymbol{\theta}) \in \mathbb{R}^{2\times 2}$ is the inertia matrix; $C(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \in \mathbb{R}^{2\times 2}$ is the Coriolis matrix; $\boldsymbol{g}(\boldsymbol{\theta}) \in \mathbb{R}^2$ is the gravity vector; and $\boldsymbol{\tau} = [\tau_1, \tau_2]' \in \mathbb{R}^2$ are the torques applied to each joint. The parameters used here are as follows [21, p. 209]: the link mass $m_1 = m_2 = 1\text{kg}$, the link length $l_1 = l_2 = 1\text{m}$; the distance from joint to center of mass (COM) $l_{c1} = l_{c2} = 0.5\text{m}$, and the moment of inertia with respect to COM $I_1 = I_2 = 0.5\text{kgm}^2$. By defining

$$\boldsymbol{x} = \begin{bmatrix} \theta_1 & \dot{\theta}_1 & \theta_2 & \dot{\theta}_2 \end{bmatrix}' \quad \text{and} \quad \boldsymbol{u} = \boldsymbol{\tau} = \begin{bmatrix} \tau_1 & \tau_2 \end{bmatrix}', \tag{30}$$

we write (29) in state-space representation and further approximate it to the following discrete-time form [20]

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta \cdot \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_{k+1}) \tag{31}$$



(a) The state trajectory. (b) Recovered cost weights.

Figure 3: Recovery of a three-phase cost function for the robot motion: (a) the state trajectory of the two-link robot arm with the red dotted lines indicating the phase transition points of ground-truth; the ground-truth cost weights for each phase is $\boldsymbol{\omega}^{(1)} = [0.75, 0.25]'$, $\boldsymbol{\omega}^{(2)} = [0.5, 0.5]'$, and $\boldsymbol{\omega}^{(3)} = [0.2, 0.8]'$; and (b) shows the recovered results by the proposed method (blue solid lines) and the KKT method [4] (red dotted/dashed lines).

where $\Delta = 0.001$s is the discretization interval.

The motion of the robot arm contains three phases, and each phase has different extents and cost functions. The union feature vector is $\boldsymbol{\phi} = [\tau_1^2, \tau_2^2]'$, where τ_i^2 $(i = 1, 2)$ denotes a quadratic basis function of torque τ_i. In Phase I, the robot moves from $\boldsymbol{x}_0 = \boldsymbol{x}_{T_1} = [0, 0, 0, 0]'$ at $T_1 = 0$ to $\boldsymbol{x}_{T_2} = [-\frac{\pi}{6}, 0, -\frac{\pi}{3}, 0]'$ at $T_2 = 1000$ (1s) with cost weights $\boldsymbol{\omega}^{(1)} = [0.75, 0.25]'$; in Phase 2, from $\boldsymbol{x}_{T_2}$ to $\boldsymbol{x}_{T_3} = [\frac{\pi}{4}, 0, \frac{\pi}{2}, 0]'$ at $T_3 = 2000$ (2s) with $\boldsymbol{\omega}^{(2)} = [0.5, 0.5]'$; and in Phase 3, from $\boldsymbol{x}_{T_3}$ to $\boldsymbol{x}_{T_4} = [-\frac{\pi}{6}, 0, -\frac{\pi}{3}, 0]'$ at $T_4 = 3000$ (3s) with $\boldsymbol{\omega}^{(3)} = [0.2, 0.8]'$. The multiphase optimal control is solved by GPOPS [16] and the optimal state trajectories are shown in Fig. 3a.

Given the union feature vector $\boldsymbol{\phi}$, we apply Algorithm 1 to recover the multiphase cost weights from the trajectory given in Fig. 3a. We set $\gamma = 100$ and $l_{\max} = 1000$. The results $\hat{\boldsymbol{\omega}}(t)$ are plotted in Fig. 3b in blue solid lines. From Fig. 3b, we can see that the cost weights of each phase and the phase boundaries are successfully recovered. For example, we observe that the first phase is from time 0 to 1001 with the average weights $[0.7501, 0.2499]'$ (by averaging $\hat{\boldsymbol{\omega}}(t)$ for $1 \le t \le 1001$); the second phase from 1002 to 2000 with the average weights $[0.4998, 0.5002]'$; and the third phase from 2001 to 3000 with the average weights $[0.1998, 0.8012]'$.

1) Observation Noise: we evaluate the performance of the proposed method by adding varying levels of white Gaussian noise to the trajectory data, and the results are listed in Table I. From Table I, we can conclude that (i) the average minimum observation length increases if the trajectory noise is high; and (ii) since the increased window length includes more data points to mitigate noise, the recovery accuracy maintains high.

Table I: Recovery performance under different noise levels

Noise level	e_ω	Avg. $l_{\min}$*	$\hat{T}_1$ and $\hat{T}_2$	e_ω for KKT method *
$1e-4$	0.0019	145.3	1004, 2000	0.15
$1e-3$	0.0016	236.0	1004, 2000	0.15
$1e-2$	0.0014	554.5	1004, 2002	0.18

* The averaged $l_{\min}$ is computed by averaging $l_{\min}(t)$ for $1 \le t \le T$; and e_ω for KKT method is evaluated with window length $L = 950$.

2) Comparison with Related Work: we compare the proposed method with the KKT method [4]. In [4], a window of manually-specified length moves along the trajectory, and the weights are computed by minimizing the violation of KKT conditions. Here, we set the window length $L = 950$ and $L = 1000$, and plot the corresponding recoveries in Fig. 3b using red dotted and red dashed lines, respectively. We also test the recovery error for $L = 950$ under different noise levels and summarize the results in Table I.

The results illustrate that although being able to discriminate motion phases, the KKT method does not consistently produce the correct cost weights (the recovery errors for $L = 950$ and $L = 1000$ are 0.18 and 0.15, respectively), and the estimated phase transition points have high errors. We also find that the KKT method is sensitive to the choice of window length: a larger window length will improve recovery accuracy, but may lead to inaccuracy for the phase transition point estimation.

This is because the KKT method only uses current window data and does not consider the influence of future data beyond the window on the recovery; i.e., it establishes the optimality equations by neglecting the right-hand term of (7), inevitably leading to a recovery error. This future information is encoded in the costate $\boldsymbol{\lambda}$ in (19) in our formulation. When the observed data is of 'low richness', e.g. of a small window length, the influence of future information becomes relatively significant, thus leading to a large recovery error. Thus, the KKT method always requires a large window, but this will potentially deteriorate the accuracy of phase boundary detection. In Fig. 3b, we also observe that the KKT method results in a detection delay for the first phase transition point. This may be because when the window is over the transition point, the included data from the first phase is more expressive compared to that of the second phase, thus contributing more to the computed cost weight and making the results look more like the ones of the first phase.

3) Setting Parameters: the value of $l_{\max}$ does not affect the recovery performance as long as it satisfies (26). To facilitate implementation, in practice we can choose $l_{\max} = T/j$, where $j \in \{1, 2, \cdots, p\}$. Here, p is the number of motion phases (or an approximation if not explicitly known) and T is the overall time horizon of the trajectory. Note that a larger $l_{\max}$ will lead to higher computational cost as more trajectory points of the next phase need to be checked when the observation window is over a phase transition point. In the above simulation, as $p = 3$, we choose $j = p$ and thus $l_{\max} = 3000/3 = 1000$ for lower computation cost.

To choose γ, we inspect its influence on recovery performance. We set $l_{\max} = 1000$ and vary γ to observe the changes of average window length and recovery error. Results in Fig. 4 show that larger γ increases the window length and improves the accuracy because more observations can compensate for uncertainties (noise). But above 250, further increasing γ will not significantly improve accuracy, implying the insensitivity of changing γ to the recovery and the flexibility of choosing γ. Therefore, we use the following two rules to find a proper γ. (I) In the case where the union feature set is explicitly known, we may choose γ from the range of $[100, \infty)$, as demonstrated in the above simulation where we chose $\gamma = 100$ (for lower

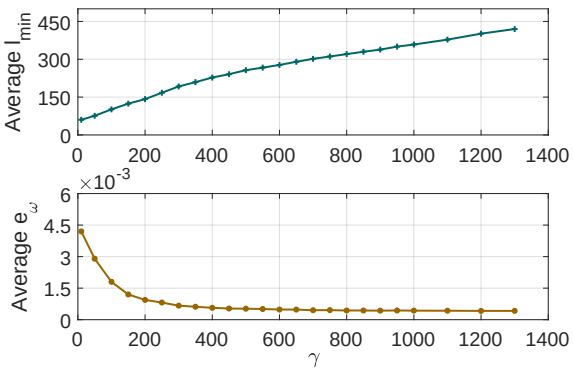
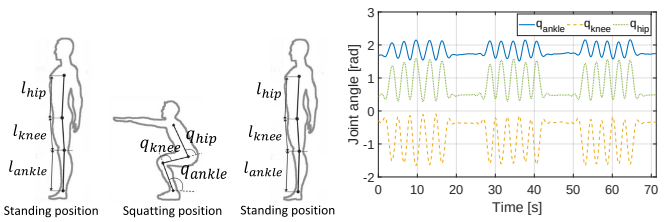


Figure 4: Recovery performance with varying γ

computational cost); for the case where the union features are selected based on empirical knowledge [9], [3], we may choose γ from the range of $(5, \infty)$. This is because if the empirically-selected features are not perfect in the sense of not strictly satisfying the optimality condition (13) for any nonzero cost weights and future costate, then the upper bound property (17) tends to be not valid (increasing l tends to render $\operatorname{rank} \boldsymbol{H}(t,l)$ increase to $n+r$), and thus the metric $\kappa(t,l)$ in (24) becomes smaller. (II) We may first choose a high γ for initial trials, if most of windows reach $l_{\max}$, adjust γ to a smaller one for lower computational cost (because of smaller window length). Note that the above two rules are based on our empiricism.

B. Human Motion Segmentation



(a) A repetition of squat motion[4]. (b) Joint trajectory of 15 squats.

Figure 5: Squat exercise and a sample trajectory for 15 squats.

In the second experiment, the proposed method is tested on a human motion dataset. We choose the human squat motion [22] (Fig. 5a) as it is a common and full-body exercise studied in both athletics and rehabilitation [23].

1) Data Collection: The squat dataset was collected from 6 (5M, 1F, $\mu_{age} = 26.2$) healthy participants. Each participant performed 15 squats (Fig. 5a) in 3 sets, with 5 repetitions in each set. All squats are recorded *in a single recording* via the MotionAnalysis motion capture system, where an 80-marker model was used, providing Cartesian positions. Joint angles were then computed via inverse kinematics [22] and converted to a 3 DOF planar model, as shown in Fig. 5a, corresponding to q_{ankle}, q_{knee}, and q_{hip}.

The motion capture system has a sampling rate $\Delta = 0.01$s. The obtained joint trajectories were smoothed by a moving Savitzky-Golay filter [24] (span 2s and degree 10). This allows to suppress noise and compute smooth trajectory derivatives. The joint velocity and acceleration are then computed by numerical differentiation. Fig. 5b plots the joint trajectories for a sample participant.

2) Body Dynamics Model and Feature Selection: As shown in Fig. 5a, the human body is modeled as a 3DOF (ankle-knee-hip joints) fixed-base articulated system. The dynamics of the modeled human body [4] is given by

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}, \tag{32}$$

where $\boldsymbol{q} = [q_{ankle}, q_{knee}, q_{hip}]'$ is the joint angle vector; $M(\boldsymbol{q}) \in \mathbb{R}^{3\times3}$ is the inertia matrix; $C(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{3\times3}$ is the Coriolis matrix; $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^3$ is the gravity vector; and $\boldsymbol{\tau} \in \mathbb{R}^3$ are the torques generated by each joint. The anthropometrics parameters [25] are used in (32). The torques (trajectories) are computed from (32). We represent (32) in state-space form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ with the state $\boldsymbol{x}$ and input $\boldsymbol{u}$ defined as

$$\boldsymbol{x} = \left[\boldsymbol{q}, \dot{\boldsymbol{q}}\right]' \quad \text{and} \quad \boldsymbol{u} = \boldsymbol{\tau}, \tag{33}$$

respectively, and then discretize it into $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta \cdot f(\boldsymbol{x}_k, \boldsymbol{u}_{k+1})$ with the discretization interval $\Delta = 0.01$s (i.e. sampling rate of the motion capture system).

The unit feature vector $\boldsymbol{\phi}$ in this experiment is chosen based on the previous results obtained in [4], where the following features in Table II were demonstrated to play significant roles in human squat motion [4]. Thus, $\boldsymbol{\phi} = [\phi_1, \phi_2, \phi_3]'$.

Table II: The selected features [4]

Criterion	Feature function (ϕ_i)
Joint acceleration	$\phi_1 = \sum_{i=1}^{3} \ddot{q}_i^2$
Joint jerk	$\phi_2 = \sum_{i=1}^{3} \dddot{q}_i^2$
Joint power	$\phi_3 = \sum_{i=1}^{3} (\tau_i \dot{q}_i)^2$

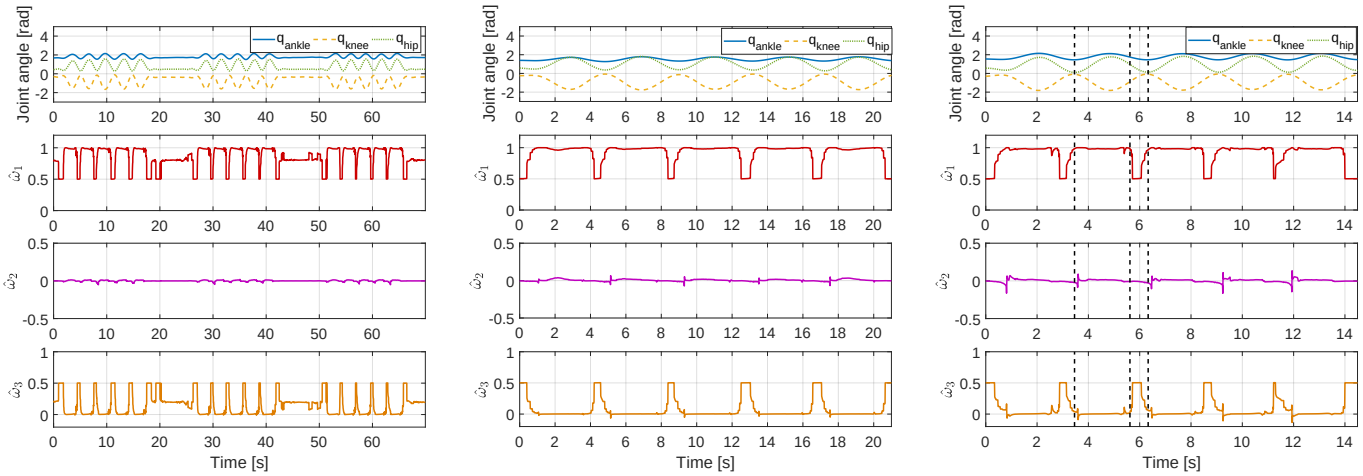
q_1, q_2, and q_3 correspond to q_{ankle}, q_{knee}, and q_{hip}, respectively.

3) Recovery Results: note that for each participant, all 15 squats are in a single recording and we apply the proposed method on the trajectory without manual segmentation.

In Algorithm 1, following the rules given in Section V.A.3, we set $\gamma = 6$ (as described before, the value of γ in practice is always smaller because of the imperfection of union feature set selection); since in each motion set (around 6s) the number of phases is estimated around 10, we set $l_{\max} = 60$ (that is, $(6\text{s})/10/(0.01\text{s})$).

We use the data from Participant 1 (P1), Participant 3 (P3), and Participant 5 (P5) as examples to demonstrate the recovery results. In Fig. 6, the joint trajectories of P1, P3, and P5 are shown in the first row; here, we present the entire motion data (i.e. 3 sets and a total of 15 squats) for P1 and only one set (5 squats) for P3 and P5 to show both overall and local details of the recovery results. Corresponding to the motion data, the recovered cost weights $\hat{\boldsymbol{\omega}}(t)$ are presented in the panels below; here, cost weight $\hat{\omega}_1$, $\hat{\omega}_2$, and $\hat{\omega}_3$ correspond to ϕ_1, ϕ_2, and ϕ_3 in Table II, respectively. We have the following observations:

a) Overall, Fig. 6a shows a reliable multiphase cost function recovery performance. During each squat repetition (i.e.



(a) 15 squats by Participant 1 (data of three sets). (b) 5 squats by Participant 3 (data of one set). (c) 5 squats by Participant 5 (data of one set).

Figure 6: Multiphase cost function recovery for three sample participants. Joint trajectory (filtered) of each participant is plotted in first row: (a) 15 squats in 3 sets by Participant 1; (b) one 5-squat set by Participant 3; and (c) one 5-squat set by Participant 5. The corresponding recovery results are shown below respectively, where $\hat{\omega}_1$, $\hat{\omega}_2$, and $\hat{\omega}_3$ are the cost weights for the acceleration ϕ_1, joint jerk ϕ_2, and power ϕ_3 in Table II, respectively.

Table III: Motion segmentation and multiphase cost function recovery for all participants. Active-squat phases and between-squats phases are segmented by ω_{th}, and the corresponding segmentation accuracy is computed. Using successful segmentations, the average cost weights for both phases are computed. The results by the KKT method [4] are also compared.

Participant	Average $\hat{\boldsymbol{\omega}}$ for active-squat phases		Average $\hat{\boldsymbol{\omega}}$ for between-squats phases		Segmentation accuracy [%]		
	$\omega_{th} = 0.8$	$\omega_{th} = 0.9$	$\omega_{th} = 0.8$	$\omega_{th} = 0.9$	$\omega_{th} = 0.8$	$\omega_{th} = 0.9$	KKT method [4]
1	[0.98, 0.00, 0.02]	[0.98, 0.00, 0.01]	[0.55, −0.01, 0.45]	[0.61, −0.01, 0.40]	96.88%	96.88%	89.54%
2	[0.97, 0.00, 0.03]	[0.98, 0.00, 0.02]	[0.63, −0.01, 0.38]	[0.69, −0.01, 0.32]	100.0%	100.0%	83.51%
3	[0.98, 0.00 , 0.02]	[0.99, 0.01, 0.00]	[0.56, −0.01, 0.44]	[0.63, −0.01, 0.38]	96.67%	96.67%	85.80%
4	[0.96, 0.00, 0.03]	[0.98, 0.01, 0.01]	[0.62, −0.01, 0.38]	[0.70, −0.01, 0.31]	94.44%	100.0%	67.62%
5	[0.98, 0.00, 0.02]	[0.98, 0.00, 0.01]	[0.64, −0.01, 0.37]	[0.65, −0.01, 0.36]	92.89%	93.33%	76.39%
6	[0.98, 0.01, 0.02]	[0.98, 0.01, 0.01]	[0.69, −0.01, 0.31]	[0.73, −0.01, 0.27]	91.15%	90.00%	89.05%

standing-squatting-standing in Fig. 5a), the cost weights $\hat{\boldsymbol{\omega}}(t)$ remain at the value around $[1, 0, 0]$, which indicates that one squat belongs to the same phase in terms of sharing the same cost function. Between two squats where a participant is near (approaching) standing position, the weights change to (around) $[0.6, 0, 0.4]$, indicating that the participants switch to a different control strategy after finishing one squat but before starting the next. Fig. 6a also shows that the cost weights in between two motion sets (where the participants are in the standing position) are around $[0.8, 0, 0.2]$.

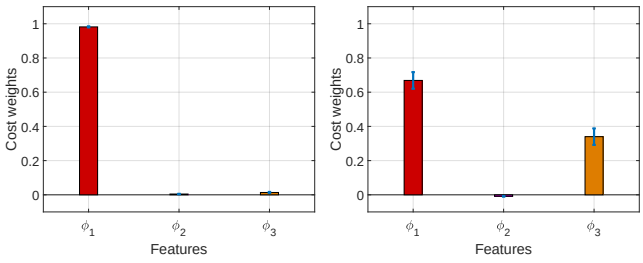
b) Recovery results in Fig. 6b and Fig. 6c show in more detail the changes of the cost weights within a 5-squat set. Below, we use Fig. 6c for analysis. As labeled by the dotted black (vertical) lines, we divide a squat repetition into two motion phases according to different cost functions used:

- *Active-Squat (AS)*: between the first and second dotted lines, during which the participant is flexing hips and knees (to squatting position) and then extending the hips and knees. The recovered results show that the control objective of this phase is to minimize the joint acceleration ϕ_1 (as both $\hat{\omega}_2$ and $\hat{\omega}_3$ are near zeros).
- *Between-Squats (BS)*: between the second and third dotted lines, during which the participant is finishing the hip and knee extension from the previous active squat and then preparing for next one. The cost function to be minimized for this phase is (approximately) $0.6\phi_1 + 0.4\phi_3$.

4) Segmentation Results: in order to automate the segmentation of the active-squat phase and the between-squats phase in each motion set, we define a *segmentation threshold* ω_{th} for $\hat{\omega}_1(t)$ (the most influential weight), and then the segmentation is performed using the following rules: if $\hat{\omega}_1(t) > \omega_{th}$; the current phase is classified as active-squat; otherwise, as between-squats. We evaluate the segmentation accuracy by $\big(0.5 \times (\frac{T_{AS}}{T_{AS}+F_{BS}} + \frac{T_{BS}}{T_{BS}+F_{AS}})\big)$[4], where T_{AS} is the count of the cases where a true active-squat phase is segmented into active-squat (True Positive); T_{BS} when a true between-squats phase is segmented into between-squats (True Negative), F_{AS} when a true active-squat phase is classified as between-squats (False Positive), and F_{BS} when a true between-squats phase is segmented into active-squat (False Negative).

The segmentation results for all participants are summarized in Table III. Here, two thresholds $\omega_{th} = 0.8$ and $\omega_{th} = 0.9$ are used, and the average cost weights for active-squat and between-squats are computed based on all successful segmentations. It can be seen that the proposed method demonstrates a high reliability and accuracy in segmenting different motion phases. The difference in the average cost weights of between-

squats phase for two thresholds is due to the fact that the actual period of a between-squats phase is small (Fig. 6c), thus is more likely to be affected by segmentation threshold values.



(a) Active-squat phase. (b) Between-squats phase.

Figure 7: Recovery results over all participants for active-squat and between-squats phases. Bars denote the mean cost weights, and top line segments denote the standard deviation.

Using $\omega_{th} = 0.9$, we summarize the average cost weights for active-squat and for between-squats over all participants in Fig. 7a and 7b, respectively. It shows that all participants adopt a similar control policy in squat exercise: during active squat, participants focus on minimizing the joint acceleration, while in between squats, they adopt a balanced control policy minimizing both joint acceleration and power. This finding is consistent with previous human motion studies [4], [3], [8].

For comparison, we also perform the motion segmentation using the KKT method [4], as shown in Table II. The proposed method can achieve a segmentation accuracy above 90%, which is higher than that of the KKT method [4] with average accuracy 81.99%.

5) Result Validation: to validate the recovery and segmentation results, we simulate the trajectory of each segmented phase by solving the optimal control problem based on the recovered cost functions. Considering the consistency of the recovery among different squat repetitions (Fig. 6) and different participants (Fig. 7), we just use one squat repetition of a sample participant for illustration. We consider one squat repetition performed by Participant 5 as labeled by the black dotted lines in Fig. 6c. Under segmentation threshold $\omega_{th} = 0.9$, the active-squat phase is from time 3.46s to 5.62s with the average cost weights $[0.99, 0.00, 0.00]$ (by averaging $\hat{\boldsymbol{\omega}}(t)$ within this active-squat phase), and the between-squats is from 5.62s to 6.33s with average cost weights $[0.62, -0.01, 0.39]$. We solve the optimal control problem using these cost functions for both phases [16] and plot the results in Fig. 8. The results show that the simulated trajectory using the recovered cost functions fits well the real data, indicating the validity of the recovered multiphase cost functions in characterizing squat motion.

VI. Discussion

In this section, we provide further insights into the multi-phase cost function recovery and discuss possible extensions.

A. Accurate Estimation of Phase Transition Points

It was noted in Section III and IV that Assumption 1 only guarantees that the proposed method can accurately recover

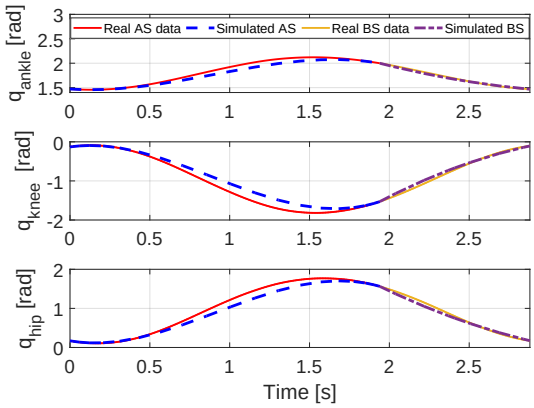


Figure 8: Simulated trajectory using the recovered multiphase cost functions. Solid lines are real motion data (second squat repetition in Fig. 6c): red for the active-squat phase and yellow for the between-squats phase. Dotted lines are the simulated motion: blue for the active-squat phase and brown for the between-squats phase.

the cost weights between T_j and $t^{(j)}_{\max}$, $1 \le j \le p$, and due to degenerate recovery the cost weights between $t^{(j)}_{\max}$ and T_{j+1} may not be recovered correctly. To avoid this limitation, we establish the following assumption.

Assumption 2. *Assumption 1 holds. For the window with the starting time $t \in (T_j, T_{j+1}]$ which satisfies* $\operatorname{rank} \boldsymbol{H}(t, T_{j+1} - t + 1) < n + r - 1$, *there does NOT exist an observation length $l \in (T_{j+1} - t + 1, T_{j+1} - t + 1 + l_{\min}(T_{j+1} + 1)]$ such that* $\operatorname{rank} \boldsymbol{H}(t, l) = r + n - 1$.

Assumption 2 guarantees that the degenerate recovery never occurs for any observation window over a phase transition point. Under this assumption, at the end of each phase where the observations are not sufficient for a successful recovery, the cost weights will be assigned to the ones of the previous successful recovery, and thus the change of $\hat{\boldsymbol{\omega}}(t)$ happens only at T_{j+1}. Thus, the proposed method will produce accurate estimates for all phase transition points.

B. Elimination of Degenerate Recovery

We have noted in Section IV.A that observations over a phase transition point may result in degenerate recovery. The following lemma, however, indicates that degenerate recovery can be eliminated by observing more data.

Lemma 3. *Suppose that Assumption 1 holds, and the window starts at time $t \in (T_j, T_{j+1}]$ that satisfies* $\operatorname{rank} \boldsymbol{H}(t, T_{j+1} - t + 1) = n + r - 1$. *If the window length $l \ge T_{j+1} - t + 1 + l_{\min}(T_{j+1} + 1)$, then*

$$\operatorname{rank} \boldsymbol{H}(t, l) = r + n. \tag{34}$$

Here, $l_{\min}(T_{j+1} + 1)$ denotes the minimal observation length from the starting time $T_{j+1} + 1$.

Proof. Please see the proof in Appendix. ■

Lemma 3 indicates that by observing more data in addition to achieving $\operatorname{rank} \boldsymbol{H}(t, T_{j+1}-t+1) = n+r-1$, we can always obtain a detectable rank condition of $\operatorname{rank} \boldsymbol{H}(t, T_{j+1}-t+1) = n+r$ for checking the presence of the phase transition points. However, one disadvantage of this 'additional observations' strategy is that it will lead to a large error range, e.g., from Lemma 3, we are only able to tell that the phase transition point is located in the interval $[t, T_{j+1} + l_{\min}(T_{j+1}+1)]$ and cannot obtain its more precise location.

C. A Bi-pass Scheme

Another idea to mitigate the influence of degenerate recovery and obtain a more accurate phase transition point estimate is to use a bi-pass framework. Since the proposed method only adopts a forward pass scheme, the detected time where the cost weights changes, denoted as $\tilde{T}^f_{j+1}$, will be no later than the real phase transition point T_{j+1}, i.e. $\tilde{T}^f_{j+1} \leq T_{j+1}$ (as shown in (23)). Similarly, if one also performs a back pass recovery, i.e. moving and recovering in a backward fashion, the detected phase transition point, denoted as $\tilde{T}^b_{j+1}$, will happen no earlier than the real phase transition point: $T_{j+1} \leq \tilde{T}^b_{j+1}$. Combining results from both passes, the estimated phase transition point will fall into the range of $[\tilde{T}^f_{j+1}, \tilde{T}^b_{j+1}]$, which is more precise than the estimate using a single pass.

Although producing a more accurate estimate of the phase transition point, the above bi-pass framework has the following disadvantages compared to single forward pass: 1) significant computational cost is involved, making it difficult for online implementation; and 2) as shown in experiments, degenerate recovery happens infrequently in most cases; thus by applying only forward pass, the estimated point where the cost weights change is already very close to the real one. We may consider the bi-pass based recovery method in our future work.

D. Selection of the Union Feature Set

We now discuss how to select the union feature set when the set of features is not explicitly known. From (4) we note that one advantage of the proposed method is its ability to allow for additional irrelevant features in the union feature vector (the recovered cost weights for those features are equal to zeros). This property implies that one can include as many candidate features as possible into the union feature set. However, too many features will lead to the violation of Assumption 1, as explained in Section III. To further illustrate this, we consider the cost function recovery in Fig. 3a, given different union feature vectors listed in Table IV. We set $\gamma = 100$ and $l_{\max} = 1000$ in Algorithm 1, and the corresponding recovery error for each union feature vector is summarized in Table IV.

As shown in Table IV, more candidate features will significantly increase the minimal observation length required for a successful recovery. Thus, an increasing number of candidate features may lead to the required window length exceeding a phase horizon, causing the violation of Assumption 1 and the failure of the proposed algorithm.

Thus, a principle for union feature set selection is to make the selected set contain as few irrelevant features as possible. One way to form a union feature set is to try an initial feature set which includes many possible candidate features based on prior knowledge (at this point Assumption 1 may be violated), then remove features from the set by trial and error (we may also need to solve the optimal control problems) until a stable recovery performance and a 'compact' union feature set are obtained. A theoretical exploration for the choice of relevant features will be left as our future work.

Table IV: Recovery performance with irrelevant features

Union feature vector *	e_ω	Ave. $l_{\min}$	$\hat{T}_1, \hat{T}_2$
$[\tau_1^2, \tau_2^2]$	0.0018	236.0	$1004, 2000$
$[\tau_1^2, \tau_2^2, \tau_1^3]$	0.0037	333.2	$1004, 2002$
$[\tau_1^2, \tau_2^2, \tau_1^3, \tau_2^3]$	0.0037	386.4	$1004, 2003$
$[\tau_1^2, \tau_2^2, \tau_1^3, \tau_2^3, \tau_1^4]$	0.0052	459.8	$1004, 2003$
$[\tau_1^2, \tau_2^2, \tau_1^3, \tau_2^3, \tau_1^4, \tau_2^4]$	0.0050	505.8	$1004, 2003$

* All candidate features are the polynomial function with the form τ_i^k where the variable τ_i denotes the joint torque applied on the ith joint of the simulated two-arm robot, and k is the exponent.

VII. Conclusions

We consider the inverse optimal control problem where the system trajectory is a concatenation of multiple phases of motion, which are generated by minimizing a phase-dependent cost function. We hypothesize the phase-dependent cost function as a weighted sum of given union features with phase-dependent cost weights. An inverse optimal control method is developed for not only recovering the cost weights of each phase but also estimating the phase transition points.

We solve the multiphase cost function recovery by sliding an adaptive-length window along the observed trajectory while at each position recovering the cost weights under the window. The window length is determined by finding the minimal observation length that suffices for a successful recovery using the recovery matrix. The output is a trace of the recovered cost weights indexed by time, from which the cost function for each phase can be obtained and then each phase can be segmented. We demonstrate the proposed method using real human motion data, and experiments show that the method reliably recovers the cost function of each phase, and segments each motion phase from the trajectory.

Appendix: Proof of Lemma 3

We first prove that (34) holds for $l = T_{j+1} - t + 1 + l_{\min}(T_{j+1}+1)$. We prove this by contradiction: assume $\operatorname{rank} \boldsymbol{H}(t,l) < r+n$ for $l = T_{j+1} - t + 1 + l_{\min}(T_{j+1}+1)$. Due to the rank non-decreasing property (16), $\operatorname{rank} \boldsymbol{H}(t,l) \geq \operatorname{rank} \boldsymbol{H}(t, T_{j+1}-t+1) = n+r-1$; thus $\operatorname{rank} \boldsymbol{H}(t,l) = n+r-1$ has to hold. We thus can find a vector $\operatorname{col}\{\tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{\lambda}}\} \neq \mathbf{0}$ in the kernel of $\boldsymbol{H}(t,l)$. Based on the definition of the recovery matrix in (10-12), we further can find a sequence of costates $\tilde{\boldsymbol{\lambda}}_{t:t+l-1}$ such that

$$\boldsymbol{F}_x(t,l)\tilde{\boldsymbol{\lambda}}_{t:t+l-1} - \boldsymbol{\Phi}_x(t,l)\tilde{\boldsymbol{\omega}} = \boldsymbol{V}(t,l)\tilde{\boldsymbol{\lambda}} \tag{35}$$

$$\boldsymbol{F}_u(t,l)\tilde{\boldsymbol{\lambda}}_{t:t+l-1} + \boldsymbol{\Phi}_u(t,l)\tilde{\boldsymbol{\omega}} = \mathbf{0} \tag{36}$$

where $\boldsymbol{F}_x(t,l)$, $\boldsymbol{\Phi}_x(t,l)$, $\boldsymbol{F}_u(t,l)$, $\boldsymbol{\Phi}_u(t,l)$, and $\boldsymbol{V}(t,l)$ are defined in (9). In the following, we consider two sub-cases:

Case 1: consider $\tilde{\boldsymbol{\omega}} \neq \mathbf{0}$. Splitting both (35) and (36) at time T_{j+1} and recalling the definition of recovery matrix, we have

$$\boldsymbol{H}(t, T_{j+1} - t + 1) \begin{bmatrix} \tilde{\boldsymbol{\omega}} \\ \tilde{\boldsymbol{\lambda}}_{T_{j+1}+1} \end{bmatrix} = \mathbf{0} \tag{37}$$

$$\boldsymbol{H}(T_{j+1} + 1, l_{\min}(T_{j+1} + 1)) \begin{bmatrix} \tilde{\boldsymbol{\omega}} \\ \tilde{\boldsymbol{\lambda}} \end{bmatrix} = \mathbf{0}. \tag{38}$$

As (37) and (38) in fact correspond to the optimality conditions in jth and $(j+1)$th phases, respectively, thus given the actual costates $\boldsymbol{\lambda}^*_{T_{j+1}+1}$ and $\boldsymbol{\lambda}^*_{T_{j+1}+1+l_{\min}(T_{j+1}+1)}$, we have

$$\boldsymbol{H}(t, T_{j+1} - t + 1) \begin{bmatrix} \boldsymbol{\omega}^{(j)} \\ \boldsymbol{\lambda}^*_{T_{j+1}+1} \end{bmatrix} = \mathbf{0} \tag{39}$$

$$\boldsymbol{H}(T_{j+1} + 1, l_{\min}(T_{j+1} + 1)) \begin{bmatrix} \boldsymbol{\omega}^{(j+1)} \\ \boldsymbol{\lambda}^*_{T_{j+1}+1+l_{\min}(T_{j+1}+1)} \end{bmatrix} = \mathbf{0}. \tag{40}$$

As $\operatorname{rank} \boldsymbol{H}(T_{j+1} + 1, l_{\min}(T_{j+1} + 1)) = \operatorname{rank} \boldsymbol{H}(t, T_{j+1} - t + 1) = n + r - 1$, based on (37)-(40) we have $\boldsymbol{\omega}^{(j)} = c_1 \tilde{\boldsymbol{\omega}}$ and $\boldsymbol{\omega}^{(j+1)} = c_2 \tilde{\boldsymbol{\omega}}$ ($c_1 > 0$ and $c_2 > 0$ are two scalars) and thus $\boldsymbol{\omega}^{(j+1)} = \frac{c_2}{c_1} \boldsymbol{\omega}^{(j)}$. This contradicts the fact that jth phase and $j+1$th phase correspond to different cost weights (independent weight vectors).

Case 2: consider $\tilde{\boldsymbol{\omega}} = \mathbf{0}$. Since both (37) and (39) still hold, $\operatorname{rank} \boldsymbol{H}(t, T_{j+1} - t + 1) = n + r - 1$, and $\boldsymbol{\omega}^{(j)} \neq \mathbf{0}$, then $\tilde{\boldsymbol{\lambda}}_{T_{j+1}+1} = \mathbf{0}$ in (37) have to be satisfied. Resorting to (35) and considering the structure of $\boldsymbol{F}_x(t, l)$ in (9), we have the following iteration form for the costates

$$\tilde{\boldsymbol{\lambda}}_k = \frac{\partial \boldsymbol{f}'}{\partial x^*_k} \tilde{\boldsymbol{\lambda}}_{k+1} \tag{41}$$

for $T_{j+1} + 1 \leq k \leq T_{j+1} + l_{\min}(T_{j+1} + 1)$. Given $\tilde{\boldsymbol{\lambda}}_{T_{j+1}+1} = \mathbf{0}$, if $\det\left(\frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{x}^*_k}\right) \neq 0$ ($\det(\cdot)$ is matrix determinant operator), by iteration $\tilde{\boldsymbol{\lambda}}_{T_{j+1}+1+l_{\min}(T_{j+1}+1)} = \tilde{\boldsymbol{\lambda}} = \mathbf{0}$. Thus, $\operatorname{col}\{\tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{\lambda}}\} = \mathbf{0}$, which contradicts the statement that $\operatorname{col}\{\tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{\lambda}}\}$ is non zero vector in the kernel of $\boldsymbol{H}(t, l)$.

Combining the above two cases, we can see that in Lemma 3, $\operatorname{rank} \boldsymbol{H}(t, l) = r + n - 1$ cannot hold for $l = T_{j+1} - t + 1 + l_{\min}(T_{j+1} + 1)$. Thus $\operatorname{rank} \boldsymbol{H}(t, l) = r + n$ holds. Due to the rank nondecreasing property, for any $l \geq T_{j+1} - t + 1 + l_{\min}(T_{j+1} + 1)$, $\operatorname{rank} \boldsymbol{H}(t, l) = r + n$ still holds, which completes the proof.

REFERENCES

[1] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[2] J. Mainprice, R. Hayne, and D. Berenson, "Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces," *IEEE Transactions on Robotics*, vol. 32, no. 4, pp. 897–908, 2016.

[3] B. Berret, E. Chiovetto, F. Nori, and T. Pozzo, "Evidence for composite cost functions in arm movement planning: an inverse optimal control approach," *PLoS computational biology*, vol. 7, no. 10, p. e1002183, 2011.

[4] J. F.-S. Lin, V. Bonnet, A. M. Panchea, N. Ramdani, G. Venture, and D. Kulić, "Human motion segmentation using cost weights recovered from inverse optimal control," in *Proceedings of the 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 1107–1113.

[5] E. Todorov, "Optimality principles in sensorimotor control," *Nature neuroscience*, vol. 7, no. 9, p. 907, 2004.

[6] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 729–736.

[7] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *AAAI*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[8] 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.

[9] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison," in *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1341–1346.

[10] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2641–2646.

[11] A. Keshavarz, Y. Wang, and S. Boyd, "Imputing a convex objective function," in *Proceedings of the 2011 IEEE International Symposium on Intelligent Control (ISIC)*, pp. 613–619.

[12] M. Johnson, N. Aghasadeghi, and T. Bretl, "Inverse optimal control for deterministic continuous-time nonlinear systems," in *Proceedings of the 2013 IEEE 52nd Annual Conference on Decision and Control (CDC)*, pp. 2906–2913.

[13] N. Aghasadeghi and T. Bretl, "Inverse optimal control for differentially flat systems with application to locomotion modeling," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6018–6025.

[14] P. Englert, N. A. Vien, and M. Toussaint, "Inverse kkt: Learning cost functions of manipulation tasks from demonstrations," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1474–1488, 2017.

[15] T. L. Molloy, J. J. Ford, and T. Perez, "Finite-horizon inverse optimal control for discrete-time nonlinear systems," *Automatica*, vol. 87, pp. 442–446, 2018.

[16] M. A. Patterson and A. V. Rao, "Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 1, p. 1, 2014.

[17] A.-S. Puydupin-Jamin, M. Johnson, and T. Bretl, "A convex approach to inverse optimal control and its application to modeling human locomotion," in *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 531–536.

[18] W. Jin, D. Kulić, S. Mou, and S. Hirche, "Inverse optimal control with incomplete observations," *arXiv preprint arXiv:1803.07696*, 2018. [Online]. Available: https://arxiv.org/abs/1803.07696

[19] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," *arXiv preprint arXiv:1206.4617*, 2012.

[20] E. Todorov, "Optimal control theory," *Bayesian brain: probabilistic approaches to neural coding*, pp. 269–298, 2006.

[21] M. W. Spong and M. Vidyasagar, *Robot dynamics and control*. John Wiley & Sons, 2008.

[22] V. Joukov, J. F. S. Lin, K. Westemann, and D. Kulić, "Real-time unlabeled marker pose estimation via constrained extended Kalman filter," in *International Symposium on Experimental Robotics*, 2018.

[23] C. Kisner, L. A. Colby, and J. Borstad, *Therapeutic exercise: foundations and techniques*. Fa Davis, 2017.

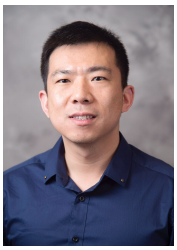
[24] J. Chen, P. Jönsson, M. Tamura, Z. Gu, B. Matsushita, and L. Eklundh, "A simple method for reconstructing a high-quality ndvi time-series data set based on the savitzky–golay filter," *Remote sensing of Environment*, vol. 91, no. 3-4, pp. 332–344, 2004.

[25] R. Dumas, L. Cheze, and J.-P. Verriest, "Adjustments to mcconville et al. and young et al. body segment inertial parameters," *Journal of biomechanics*, vol. 40, no. 3, pp. 543–553, 2007.

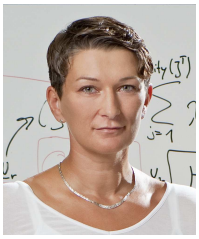
Wanxin Jin received the B.E. degree in automation and M.Sc. degree in control science and engineering from Harbin Institute of Technology, China, in 2014 and 2016, respectively. From 2016 to 2017, Wanxin Jin was a research assistant at the Chair of Information-oriented Control at Technical University Munich, Germany. Since 2017, Wanxin Jin is a research assistant at the School of Aeronautics and Astronautics, Purdue University, West Lafayette, USA. His research interests include inverse learning, learning and control, human-robot interaction, and multi-agent systems.

Dana Kulić received the combined B. A. Sc. and M. Eng. degree in electro-mechanical engineering, and the Ph. D. degree in mechanical engineering from the University of British Columbia, Canada, in 1998 and 2005, respectively. From 2006 to 2009, Dr. Kulić was a JSPS Post-doctoral Fellow and a Project Assistant Professor at the Nakamura-Yamane Laboratory at the University of Tokyo, Japan. In 2009, Dr. Kulić established the Adaptive System Laboratory at the University of Waterloo, Canada, conducting research in human robot interaction, human motion analysis for rehabilitation and humanoid robotics. Since 2019, Dr. Kulić is a professor at Monash University, Australia. Her research interests include robot learning, humanoid robots, human-robot interaction and mechatronics.

Jonathan Lin received the B. A. Sc. (2010), M. A. Sc. (2012), and the Ph. D. (2017) degrees in electrical engineering from the University of Waterloo, Canada. Since 2017, Dr. Lin is a post-doctoral fellow at the University of Waterloo. His research interest includes human motion analysis, motion segmentation and identification, and biomedical engineering in physiotherapy.



Shaoshuai Mou received a Ph. D. degree from Yale University in 2014. After working as a post-doctoral associate at MIT, he joined Purdue University as an assistant professor in the School of Aeronautics and Astronautics in 2015. His research interests include distributed algorithms for control/optimizations/learning, multi-agent networks, UAV collaborations, perception and autonomy, resilience and cyber-security.



Sandra Hirche (M'03-SM'11) received the Diplom-Ingenieur degree in aeronautical engineering from Technical University Berlin, Germany, in 2002 and the Doktor-Ingenieur degree in electrical engineering from Technical University Munich, Germany, in 2005. From 2005 to 2007 she was awarded a Postdoc scholarship from the Japanese Society for the Promotion of Science at the Fujita Laboratory, Tokyo Institute of Technology, Tokyo, Japan. From 2008 to 2012 she has been an associate professor at Technical University Munich. Since 2013 she is TUM Liesel Beckmann Distinguished Professor and has the Chair of Information-oriented Control in the Department of Electrical and Computer Engineering at Technical University Munich. Her main research interests include cooperative, distributed and networked control with applications in human-machine interaction, multi-robot systems, and general robotics. She has published more than 150 papers in international journals, books and refereed conferences. Dr. Hirche has served on the Editorial Boards of the IEEE Transactions on Control of Network Systems, IEEE Transactions on Control Systems Technology, and the IEEE Transactions on Haptics. She has received multiple awards such as the Rohde & Schwarz Award for her PhD thesis, the IFAC World Congress Best Poster Award in 2005 and - together with students - the 2018 Outstanding Student Paper Award of the IEEE Conference on Decision and Control as well as Best Paper Awards of IEEE Worldhaptics and IFAC Conference of Manoeuvring and Control of Marine Craft in 2009.