2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Inverse Optimal Control for a Hybrid Dynamical System with Impacts

Navid Aghasadeghi, Andrew Long and Timothy Bretl

Abstract— **In this paper, we develop an approach to inverse optimal control for a class of hybrid dynamical system with impacts. As it is usually posed, the problem of inverse optimal control is to find a cost function that is consistent with an observed sequence of decisions, under the assumption that these decisions are optimal. We assume instead that observed decisions are only approximately optimal and find a cost function that minimizes the extent to which these decisions violate first-order necessary conditions for optimality. For the hybrid dynamical system that we consider with a cost function that is a linear combination of known basis functions, this minimization is a convex program. In fact, it reduces to a simple least-squares computation that—unlike most other forms of inverse optimal control—can be solved very efficiently. We apply our approach to a dynamic bipedal climbing robot in simulation, showing that we can recover cost functions from observed trajectories that are consistent with two different modes of locomotion.**

I. Introduction

Inverse optimal control (IOC) is the problem of recovering a cost function which explains observations of optimal or "expert" trajectories [1]. In some applications, the goal might be to imitate the behavior of an expert. In other cases, such as the study of human motor control, finding the underlying cost function could be precisely what we are interested in. IOC problems are of interest in a wide range of applications, from basic science [2], [3] and locomotion [4], [5], to optimal control of aircraft [6], and more recently aerobatic helicopter flight [7] within the robotics community.

The problem of IOC has been studied in many contexts. In the context of Markov decision processes, IOC has been studied in [8], [9]. IOC algorithms for non-linear continuous systems have been developed in the context of linearly-solvable Markov decision processes [10]. Also, in a previous paper, we have developed a similar approach for IOC in non-linear continuous systems [11]. However, most of these algorithms require solving optimal control problems which introduces a significant computational bottleneck for these algorithms.

Recently, a new formulation of a problem closely related to IOC has been proposed by [12], which completely eliminates the computational bottleneck of previous IOC algorithms. In this formulation, unlike traditional approaches to IOC, where it is assumed that the choice of the decisions is perfect and their observations are noisy, it is assumed that observations of the decisions are perfect, while the choice of the decisions is "approximately optimal". The goal of this approach is to minimize the extent to which first-order necessary conditions of optimality are violated. Minimization and evaluation of these necessary conditions no longer requires solving an optimal control problem, and thus the IOC problem can be solved very efficiently. In fact, the problem reduces to a least-squares minimization.

The contribution of this paper is extending the new formulation of IOC and the notion of "approximate optimality" to hybrid dynamical systems with impacts. Hybrid dynamical systems consist of both continuous and discrete events [13]. For example, bipedal walking is often modeled as a hybrid system consisting of continuous motions of the lower-limbs and discrete impacts of the legs with the ground. Hybrid systems, can be used in many other fields such as robotics [14], [15] and air traffic control [16]. In this paper, we use the first-order necessary conditions for optimality for a class of hybrid dynamical systems introduced in Long et al [17], and we define the notion of approximate optimality for those systems. This notion will then allow us to introduce the IOC problem as a convex least-squares optimization. The performance of our IOC algorithm is tested on a simulated dynamic bipedal climbing robot.

Optimal control and inverse optimal control for hybrid systems are often difficult, due to the nature of these systems having both discrete and continuous events. Examples of works which have incorporated optimal control for walking systems are [18], [19]. The work by Long et al. [17] developed an optimal control approach where the first derivative of the cost function is taken explicitly with respect to the controls. The application of IOC to one class of hybrid systems was explored in [20], where a cost function was estimated to model human yoyo playing. The problem of IOC for hybrid systems is important because such an approach can have implications in lower limb prosthetic design and understanding human locomotion. Moreover, IOC plays an important complementary role to optimal control. While optimal control can produce desired control for a system, the performance of the controller significantly depends on the cost function chosen to be optimized, and IOC approaches can enable the estimation of appropriate cost functions.

The remainder of this paper is divided in the following manner. In Section II we address the problem of optimal control for a class of hybrid dynamical systems discussed in [17]. We then introduce the inverse optimal control problem for hybrid systems in Section III. We discuss our proposed

N. Aghasadeghi is with the Department of Electrical and Computer Engineering and T. Bretl is with the Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA {aghasad1,tbretl}@illinois.edu. A. Long is with the Department of Mechnical Engineering, Northwestern University, Evanston, IL 60208 USA awlong@u.northwestern.edu

solution to this problem in Section IV. The proposed solution is evaluated on a simulated dynamic bipedal climbing robot in Section V. We conclude in Section VI, and we present a discussion about possible future research directions.

II. Optimal Control

In this paper, we focus on the class of hybrid dynamical systems with impacts with following form

$$\begin{aligned} \dot{x} &= f(x,t), \text{ when } (x,\xi) \notin S \\ x^+ &= \Delta(x,\xi), \text{ when } (x,\xi) \in S, \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the state of the system at time t, $\xi \in \Xi$ are impact controls, f is the continuous dynamics, Δ is an instantaneous impact map, S is the impact surface and x^+ is the post-impact state. Once the pair (x,ξ) enters the impact surface S, the state undergoes an instantaneous change to x^+ governed by the impact map Δ. The impact controls ξ can influence both when the impact occurs and the result of the impact. Note that in this class of systems, the controls in the continuous dynamics are either zero or known. Therefore, this hybrid system can be characterized by a finite set of impact controls $\xi = [\xi_1, ..., \xi_N]^T$, which produce post-impact states $x^+ = [x_1^+, ..., x_N^+]^T$, and impact times $\tau = [\tau_1, ..., \tau_N]^T$. In this paper, we use the subscript $k = 1, ..., N$ to denote the k-th impact.

Using the fundamental theorem of calculus along with equation (1), the pre-impact states, represented by x_k^- can be obtained by forward integration of the continuous dynamics with

$$x_{k+1}^- = x_k^+ + \int_{\tau_k^+}^{\tau_{k+1}^-} f_k(x(s),s)ds. \tag{2}$$

The post-impact states are then obtained from

$$x_k^+ = \Delta_k(x_k^-, \xi_k). \tag{3}$$

In this notation, $\tau_k^- = \tau_k^+$, however, we use superscripts $^-$ and $^+$ to denote just before or just after the impact. Furthermore, we assume that the impact time τ_{k+1} is obtained as a function of the last impact time, the impact control input and the previous state as follows

$$\tau_{k+1} = h_{k+1}(\xi_{k+1}, x_k^+, \tau_k). \tag{4}$$

To understand how to perform inverse optimal control, we first need to know how optimal trajectories can be obtained by minimizing a cost function. The impact time denoted by τ_0 and the initial state of the system denoted by $x(\tau_0) = x_0$ are known, and the optimization is performed with respect to the state variables and the impact control inputs as follows

$$\begin{aligned} &\arg\min_{x,\tau,\xi} J(x,\tau,\xi) = \arg\min_{x,\tau,\xi} \sum_{k=1}^{N} L_k(x_k^+, \tau_k), \\ &\text{s.t. } (1)-(4) \ \& \ \xi \in \Xi, \end{aligned} \tag{5}$$

where J is a cost function chosen to encode a desired trajectory for the N impacts, and the function L_k is the cost enforced at time step k. Note that in general, L_k can be any function of the state and impact time of the system. Although we have chosen L_k not to be a function of ξ, this can be easily incorporated as well. Examples of these cost functions can be found in section V-B.

The optimization in (5) can be posed in a different way. As can be seen from equations (1)–(4), a trajectory can be uniquely represented using the initial conditions x_0 and τ_0 and the impact control inputs ξ_k, for $k = 1, ..., N$, i.e. $x_k^+ = x_k^+(x_0, \tau_0, \xi_1, ..., \xi_k)$ and $\tau_k = \tau_k(x_0, \tau_0, \xi_1, ..., \xi_k)$. Therefore, by considering the dependence of state variables on control inputs, optimization (5) can alternatively be written only with respect to the impact controls. In this formulation, the equality constraints are being utilized in the objective, and the new optimization is performed with only constraints on the inputs in the following way

$$\begin{aligned} &\arg\min_{\xi\in\Xi} J(x_0, \tau_0, \xi) = \\ &\arg\min_{\xi\in\Xi} \sum_{k=1}^{N} L_k(x_k^+(x_0,\tau_0,\xi), \tau_k(x_0,\tau_0,\xi)). \end{aligned} \tag{6}$$

In this paper, we assume that the cost function is a weighted combination of some known basis functions. We will denote the basis functions with $\tilde{J} = [\tilde{L}_1, ..., \tilde{L}_M]^T$ and their corresponding weights $\alpha = [\alpha_1, ..., \alpha_M]^T$. Note that the number of basis functions M does not necessarily have to equal the number of time slots N. The produced trajectories are thus the solution to the following optimization problem with known weights α

$$\xi^* = \arg\min_{\xi\in\Xi} \alpha^T \tilde{J}(x_0, \tau_0, \xi) = \arg\min_{\xi\in\Xi} \sum_{m=1}^{M} \alpha_m \tilde{L}_m(x_0, \xi). \tag{7}$$

A technique proposed by [17] for solving this optimization problem is presented later in the paper. Next, we discuss the main focus of this paper, which is the problem of inverse optimal control for the described system.

III. Inverse Optimal Control

Inverse optimal control is the problem of recovering a cost function that would have resulted in an observed trajectory. Traditionally, the formulation of this problem assumes that the trajectories are optimal, but corrupted by noise. We consider a different formulation of the IOC problem, due to [12], in which we assume that observation of the trajectory is perfect, and that this trajectory is only "approximately optimal". The definition of "approximately optimal" is provided later in the paper.

In order for trajectories to be optimal, they have to satisfy the first-order necessary Karush-Kuhn-Tucker (KKT) conditions of optimality [21]. We assume that we can write the control input constraints $\xi \in \Xi$ using a set of Q inequalities $g_q(\xi) \leq 0$ for $q = 1, ..., Q$ and denoted by the function $G = [g_1, ..., g_Q]^T$. For the optimization problem (7), the KKT conditions reduce to the existence of $\lambda \in \mathbb{R}_+^Q$ that satisfy the following conditions

$$\nabla_\xi\left(\alpha^T\tilde{J}(x_0,\tau_0,\xi^*)\right)+\sum_{q=1}^{Q}\lambda_q\nabla_\xi g_q(\xi)=\mathbf{0},$$

$$g_q(\xi)\le 0, q=1,...,Q, \tag{8}$$

$$\lambda_q g_q(\xi)=0, q=1,...,Q, \tag{9}$$

which we can express as residuals

$$\begin{aligned} r_1 =&[\alpha;\lambda]^T[\tilde{\boldsymbol{J}};\boldsymbol{G}]=\mathbf{0}, \\ r_2 =&(g_q(\xi))_+, q=1,...,Q, \\ r_3 =&\lambda_q g_q(\xi)=0, l=1,...,Q, \end{aligned} \tag{10}$$

where $\tilde{\boldsymbol{J}}$ and $\boldsymbol{G}$ denote the Jacobian matrices of $\tilde{J}$ and G with respect to the impact controls $\xi_1,...,\xi_N$, and $\mathbf{0}$ denotes an appropriately-sized vector of all zeros. The notation [;] refers to vertical concatenation of two matrices.

A trajectory is called "approximately optimal" if the condition in (10) is satisfied approximately. More precisely, a trajectory is "approximately optimal" if r_2 is close to zero, and there exists $\lambda\in\mathbb{R}_+^Q$ such that the norm of $[\alpha;\lambda]^T[\tilde{\boldsymbol{J}};\boldsymbol{G}]$ and r_3 is close to zero, for an appropriately chosen norm [12]. In this paper, we will be using the euclidean 2-norm, denoted by $||\cdot||_2$.

Therefore, we pose the IOC problem as the problem of finding a set of weights $\hat{\alpha}$ for an observed trajectory, such that the trajectory is approximately optimal. We write the IOC problem in the following way

$$\{\hat{\alpha},\hat{\lambda}\}=\arg\min_{\alpha,\lambda\in\mathbb{R}_+^Q}||[\alpha;\lambda]^T[\tilde{\boldsymbol{J}};\boldsymbol{G}]||_2^2+||r_3||_2^2, \tag{11}$$

$$\text{s.t. } \alpha\ge 0, \alpha^T\mathbf{1}=1, \tag{12}$$

where $\mathbf{1}$ denotes an appropriately-sized vector of all ones. Note that the addition of constraints (12) to the least square minimization is to avoid getting the trivial solution of $\alpha=\mathbf{0}$. The above optimization is convex in the parameters α and λ, making the proposed IOC algorithm easy to solve. Therefore, unlike traditional IOC algorithms which often require solving multiple optimal control problems, the proposed method only relies on a convex optimization. Also, note that although we are solving this problem with one observed trajectory, the problem can be easily extended to finding a cost function which would explain multiple observed trajectories.

IV. Proposed Solution

The formulation we have chosen for the IOC problem reduces the solution to a simple and computationally efficient least-squares minimization. The solution to (11) relies on two components. The first is finding the gradient of the basis functions with respect to the control inputs. After the derivatives of the basis functions with respect to the impact controls are computed, a least-squares minimization can be solved to recover the estimated weights of each basis function, thus solving the IOC problem. We will now discuss how to compute the derivatives of the basis functions, based on the results in [17].

We first introduce the operator notation $Dg(x)\circ\partial x$ to represent derivatives. This notation reads $Dg(x)$ operates on the perturbation of x, ∂x. Moreover, the notation $D_n g(arg_1, arg_2,\ldots)\circ\partial arg_n$ denotes that $g(\cdot)$ is differentiated with respect to the nth argument, and is known as the slot derivative. For more details on this notation, and the analysis in this section refer to [17].

The optimization (11) requires computing the Jacobian matrix $\tilde{\boldsymbol{J}}$, which in turn relies on computing $D_{\xi_i}\tilde{L}_m(x_0,\xi)\circ\partial\xi_i$ for all i and m. Note that we write $\tilde{L}_m(x_0,\xi)$ since as we discussed earlier, this basis function can be written explicitly as a function of x_0 and ξ. Therefore, while we also use the notation $\tilde{L}_m(x_k^+,\tau_k)$ for simplicity in the derivations below, the basis functions should still be thought of as a function of the initial states and the impact controls. Therefore, the derivatives with respect to the impact controls use the chain rule to take the derivative of the entire expression with respect to the impact control.

As presented in [17], we can derive an explicit expression for the derivative of the cost, even when the differential equations governing the dynamics cannot be integrated analytically. The derivative of the basis function $\tilde{L}_m(x_0,\xi)$ with respect to each set of impact controls, ξ_i, at impact i is given by chain rule

$$\begin{aligned} D_{\xi_i}\tilde{L}_m(x_0,\xi)\circ\partial\xi_i=\Big[&D_1\tilde{L}_m(x_k^+,\tau_k)\circ D_{\xi_i}x_k^+\circ\partial\xi_i \\ &+D_2\tilde{L}_m(x_k^+,\tau_k)\circ D_{\xi_i}\tau_k\circ\partial\xi_i\Big]. \end{aligned} \tag{13}$$

By investigating the structure of (13), the derivative for each impact control can be concisely expressed. It be shown that only a few things need to be computed for each impact in order to compute all the derivatives.

We need a way to compute these right hand side terms. To do so, note that these terms involve the derivative of the impact map and the impact time with respect to the impact control

$$\begin{bmatrix} D_{\xi_i}x_k^+\circ\partial\xi_i \\ D_{\xi_i}\tau_k\circ\partial\xi_i \end{bmatrix}=\begin{bmatrix} D_{\xi_i}\Delta_k(x_k^-,\xi_k)\circ\partial\xi_i \\ D_{\xi_i}\tau_k(\xi_k,x_{k-1}^+,\tau_{k-1})\circ\partial\xi_i \end{bmatrix}.$$

Let

$$\begin{aligned} \Upsilon_k^i&=[D_{\xi_i}\Delta_k(\cdot)\circ\partial\xi_i \quad D_{\xi_i}\tau_k(\cdot)\circ\partial\xi_i]^T \\ C_k^m&=\Big[D_1\tilde{L}_m(x_k^+,\tau_k)\quad D_2\tilde{L}_m(x_k^+,\tau_k)\Big]. \end{aligned}$$

The derivative of the impact map and the impact time with respect to the each impact control is derived in [17]. The results of the derivation are

$$\Upsilon_k^i=\left\{\begin{array}{cc} \left(\prod_{j=i+1}^{k}\Gamma_{k-j+i+1}\right)\Upsilon_i^i & \text{for } k>i \\ \Upsilon_i^i & \text{for } k=i \\ \mathbf{0} & \text{for } k<i \end{array}\right\}$$

where

$$\Gamma_k=\begin{bmatrix} D_1\Delta_k(\cdot)\circ\alpha_k & D_1\Delta_k(\cdot)\circ\beta_k \\ (D_2\tau_k(\cdot))^T & D_3\tau_k(\cdot) \end{bmatrix}$$

$$\alpha_k=\Phi_{k-1}(\tau_k^-,\tau_{k-1}^+)+f_{k-1}(x_k^-,\tau_k)(D_2\tau_k(\cdot))^T$$

$$\beta_k = f_{k-1}(x_k^-, \tau_k) D_3 \tau_k(\cdot) \\ -\Phi_{k-1}(\tau_k^-, \tau_{k-1}^+) f_{k-1}(x_{k-1}^+, \tau_{k-1}),$$

and $\Phi(t,\tau)$ is the state transition matrix [22] calculated from

$$\frac{d}{dt}\Phi(t,\tau) = A(t) \circ \Phi(t,\tau),$$

with $A(t) = D_1 f_{k-1}(x(t), t)$ and $\Phi(\tau,\tau) = I$, where I is the identity matrix.

The derivative of the m-th basis function is then

$$D_{\xi_i} \tilde{L}_m(x_0, \xi) \circ \partial \xi_i = C_k^m \Upsilon_k^i. \quad (14)$$

To compute the derivative of the basis functions with respect to each impact control, we only need to compute $C_k^m, \Upsilon_k^k, \Gamma_k \ \forall\ k = 1, \ldots, N$ and $\forall\ m = 1, ..., M$. The derivatives can now be used in the following least-squares minimization to solve the IOC problem

$$\{\hat{\alpha}, \hat{\lambda}\} = \arg \min_{\alpha, \lambda \in \mathbb{R}_+^Q} ||[\alpha; \lambda]^T [\tilde{\boldsymbol{J}}; \boldsymbol{G}]||_2^2 + ||r_3||_2^2,$$
$$\text{s.t. } \alpha \geq 0, \alpha^T \mathbf{1} = 1.$$

V. Evaluation

In order to evaluate the performance of the proposed IOC approach for hybrid systems, we simulate the ParkourBot, which is a dynamic bipedal climbing robot. We simulate two different locomotion modes for the ParkourBot, namely bounce in place and climbing. Subsequently, we test whether our proposed algorithm could produce cost functions that are consistent with the locomotion mode, and whether the recovered trajectory resembles the observed trajectory.

A. The ParkourBot

The ParkourBot is a biped robot, designed at Carnegie Mellon. The ParkourBot is equipped with two BowLegs and is similar to the one-legged BowLeg hopper in [23]. During flight, the ParkourBot compresses its spring-like legs, storing elastic energy. At impact, this stored energy is then quickly converted to kinetic energy. By controlling the injected energy and the leg angles, the robot is capable of climbing up and down as well as bouncing in place. A cartoon of the robot climbing is seen in Fig 1. (See videos at http://lims.mech.northwestern.edu/RESEARCH/current_projects/Parkourbot/Parkourbot_homepage.html)

A simple model for the ParkourBot assumes a point mass for the body with two massless legs. The configuration of the robot is denoted by $q(t) = [q_1(t), q_2(t)]^T$representing the horizontal and vertical coordinates respectively. The state is defined by $x(t) = [q(t)^T, \dot{q}(t)^T]^T$. The inertial reference frame is in the center of the chute, and the walls are a distance d away from the center. The continuous dynamics of the robot can be written as follows:

$$\dot{x}(t) = f(x(t), t) = [\dot{q}_1(t), \dot{q}_2(t), 0, -g]^T.$$

In our simulation, both legs have a length $l = 0.3$, and $g = 1$.

The impact controls in this simulation are the leg angles, denoted by $\xi = \theta$. The leg angles are measured positive clockwise from the positive horizontal axis on the right side, as shown in Fig. 2.

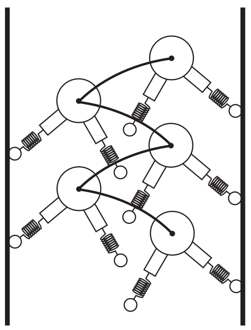

Fig. 1. ParkourBot climbing

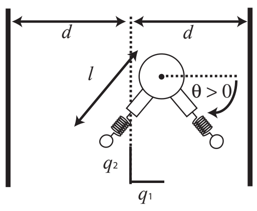

Fig. 2. Schematic of the ParkourBot

The impact time can be calculated from

$$\tau_k = h_k(\theta_k, x_{k-1}^+, \tau_{k-1}) = \frac{\pm d - q_1(\tau_{k-1}) - l cos(\theta_k)}{\dot{q}_1(\tau_{k-1})},$$

where the sign of d depends on the wall that was impacted.

In this setup, the impact impulse only influences the velocity in the direction along the leg. We denote this radial velocity by $\dot{r}(t)$, and we refer to the velocity orthogonal to the leg as the normal velocity $\dot{n}(t)$. These velocities can be calculated using a change of coordinates as follows:

$$\begin{bmatrix} \dot{n}(\tau_k^-) \\ \dot{r}(\tau_k^-) \end{bmatrix} = \begin{bmatrix} \sin(\theta_k) & \cos(\theta_k) \\ -\cos(\theta_k) & \sin(\theta_k) \end{bmatrix} \begin{bmatrix} \dot{q}_1(\tau_k^-) \\ \dot{q}_2(\tau_k^-) \end{bmatrix}$$

Moreover, we can calculate the post-impact velocities at the k-th impact using the following equation:

$$\begin{bmatrix} \dot{n}(\tau_k^+) \\ \dot{r}(\tau_k^+) \end{bmatrix} = \begin{bmatrix} \dot{n}(\tau_k^-) \\ \sqrt{\dot{r}(\tau_k^-)^2 + \frac{2E_k}{m}} \end{bmatrix},$$

where E_k denotes the net energy input to the system. Subsequently, the impact map is given by the following relationship:

$$\Delta_k(x(\tau_k^-), \theta_k) = \begin{bmatrix} q_1(\tau_k^+) \\ q_2(\tau_k^+) \\ \dot{q}_1(\tau_k^+) \\ \dot{q}_2(\tau_k^+) \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sin(\theta_k) & -\cos(\theta_k) \\ 0 & 0 & \cos(\theta_k) & \sin(\theta_k) \end{bmatrix} \begin{bmatrix} q_1(\tau_k^-) \\ q_2(\tau_k^-) \\ \dot{n}(\tau_k^+) \\ \dot{r}(\tau_k^+) \end{bmatrix}.$$

We do not enforce any constraints on the control inputs of the ParkourBot, and thus the IOC equation (11) simplifies.

B. Simulated Basis Functions

In our simulations we choose a set of 11 basis functions. Two different weightings of these basis functions leads to two different cost functions, and two different behaviors. The impact controls of the ParkourBot are optimized for 6 impacts using each cost function. The trajectory obtained following this optimization is then used with the IOC algorithm discussed in this paper, to produce an estimated cost function. The estimated cost function is then used to produce a trajectory, and the resulting trajectory would be compared with the observed optimal trajectory. The basis functions are

$$\tilde{L}_m = \frac{1}{2}(x_m^+ - x_{m,I}^+)^T Q_m (x_m^+ - x_{m,I}^+),$$
$$\text{for } m = 1, ..., 6,$$
$$\tilde{L}_m = \frac{1}{2}(x_{m-5}^+ - x_{m-5,C}^+)^T Q_m (x_{m-5}^+ - x_{m-5,C}^+),$$
$$\text{for } m = 7, ..., 11.$$

All matrices Q_m are chosen as $Q_m = diag(0,1,0,0)$ to only weight the vertical position of the robot. The desired positions are denoted by $x_{k,I}^+ = [\pm 1, -0.5, \mp 1, 1]^T$, for $k = 1, ..., 6$. These basis functions when chosen enforce the ParkourBot to bounce in place. The desired positions for $m = 7, ..., 11$ are denoted by $x_{k,C}^+ = [\pm 1, -0.5 + (k-1)h, \mp 1, 1]^T$, indicating a desired increase of h in the vertical position of the robot after every impact, with a value of $h = 0.15$ in our simulations. The signs change depending on the impacted wall. These basis functions when chosen to enforce the ParkourBot to climb vertically.

C. Bounce In Place Simulation

In this simulation, we obtain an optimal trajectory and control inputs by solving the optimal control problem with initial conditions of $\tau_0 = 0$ and $x(\tau_0) = [-1, -0.51, 1, 1]^T$, and initial controls of $\xi_0 = \{\pi/5, 4\pi/5, \pi/5, 4\pi/5, \pi/5, 4\pi/5\}$, using the cost function defined by

$$\alpha^* = [1,1,1,1,1,1,0,0,0,0,0]^T,$$
$$J(x_0, \tau_0, \xi) = \sum_{m=1}^{11} \alpha_m \tilde{L}_m.$$

The resulting trajectory can be seen in Fig. 3, and the resulting optimal impact controls are

$$\xi^* = [0.764, 2.403, 0.734, 2.404, 0.717, 2.540]^T.$$

We apply the proposed IOC algorithm using a Matlab package for least-squares optimization. Moreover, since the cost function is invariant to a multiplication by a constant, we normalize the recovered weights to match the norm of α^*, resulting in

$$\hat{\alpha} = [1.0463, 1.0289, 1.0109, 0.9867, 0.9928, 0.9304,$$
$$0.0000, 0.0000, 0.0000, 0.0000, 0.0000]^T.$$

Note that the recovered weights are consistent with the optimal weights. We solved the optimal control problem with the recovered weights, and achieved a perfect recovery of the observed trajectory (Fig. 3) and the following controls

$$\hat{\xi} = [0.764, 2.403, 0.734, 2.404, 0.714, 2.540]^T.$$

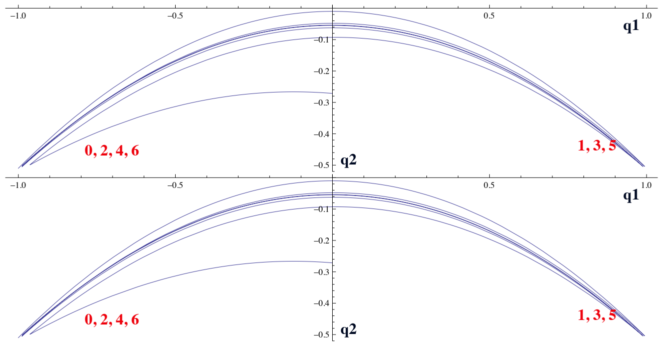

Fig. 3. Bounce in place simulation - observed trajectory (top) and recovered trajectory (bottom), numbered with the impacts

D. Climbing Simulation

In this simulation, we obtain an optimal trajectory and control inputs by solving the optimal control problem with initial conditions of $\tau_0 = 0$ and $x(\tau_0) = [-1, -0.5, 1, 1]^T$, and initial controls of $\xi_0 = \{\pi/5, 4\pi/5, \pi/5, 4\pi/5, \pi/5, 4\pi/5\}$, using the cost function defined by

$$\alpha^* = [1,0,0,0,0,0,1,1,1,1,1]^T,$$
$$J(x_0, \tau_0, \xi) = \sum_{m=1}^{11} \alpha_m \tilde{L}_m.$$

The resulting trajectory can be seen in Fig. 4, and the resulting optimal impact controls are

$$\xi^* = [0.697, 2.572, 0.554, 2.588, 0.556, 2.509]^T.$$

We apply the proposed IOC algorithm using the approach already discussed, and recover the normalized weights of

$$\hat{\alpha} = [0.9944, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,$$
$$0.9944, 0.9944, 0.9944, 0.9956, 1.0264]^T.$$

Note that again the recovered weights are consistent with the optimal weights. We solved the optimal control problem with the recovered weights, and achieved a perfect recovery of the observed trajectory (Fig. 4) and the following controls

$$\hat{\xi} = [0.697, 2.572, 0.554, 2.588, 0.556, 2.509]^T.$$

E. Discussion

In both of these simulations, the observed trajectory was perfectly recovered. The recovered weights for the basis functions are close to the optimal weights, however, the weights are not perfectly recovered. One reason is that the least squares minimization is not unique, since the number of unknowns exceeds the number of equations. This problem may be solved by using additional trajectories from different initial conditions, to ensure a unique solution to the optimization. This topic is very relevant to the problem of IOC, and requires further investigation.

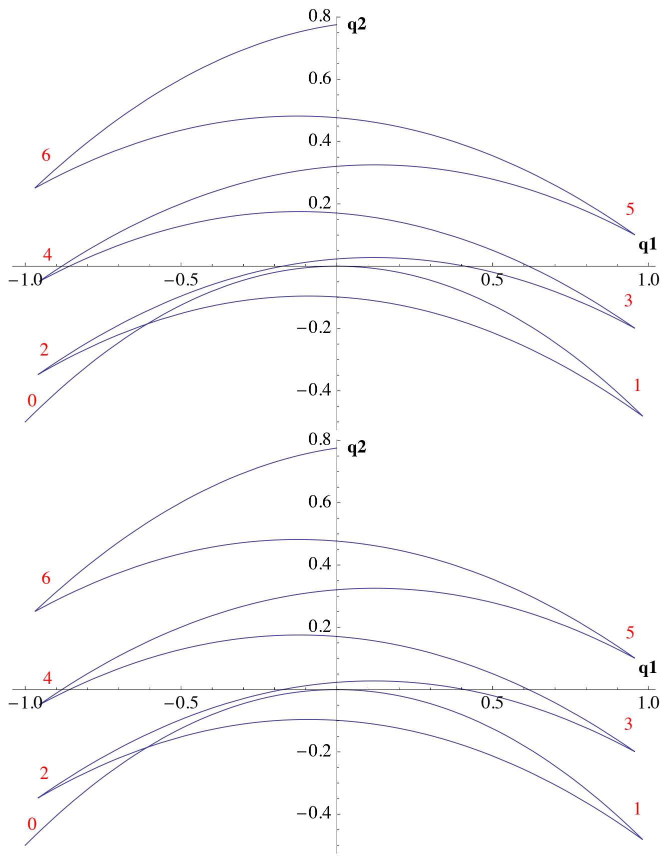

Fig. 4. Climbing simulation - observed trajectory (top) and recovered trajectory (bottom), numbered with the impacts

VI. Conclusions and Future Work

In this paper, we have introduced an IOC algorithm for a class of hybrid dynamical systems. The method of approach is computationally efficient, unlike previous approaches to IOC. We applied our approach to a simulated dynamic bipedal climbing robot, with different simulated locomotion modes. Our algorithm was able to successfully estimate an appropriate cost function based on the mode, and also reproduce a trajectory close to the observed trajectory.

Future extensions of this approach to more complex hybrid systems, can prove to be useful in modeling and understanding human locomotion. This approach can potentially help the development of real-time locomotion mode selection algorithms for prostheses [24]. Moreover, application of this method to observed human locomotion trajectories can help shed light on how humans control locomotion.

VII. Acknowledgments

The authors gratefully acknowledge Miles Johnson, Javad Ghaderi, Figen Oktem and Sophie Puydupin for helpful discussions. This material is based upon work supported by the National Science Foundation under Grant Nos. 0931871 and 0955088.

References

[1] A. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 663–670.

[2] E. Todorov, "Optimality principles in sensorimotor control," *Nature neuroscience*, vol. 7, no. 9, pp. 907–915, 2004.

[3] K. Kording and D. Wolpert, "The loss function of sensorimotor learning," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 26, p. 9839, 2004.

[4] Y. Chitour, F. Chittaro, F. Jean, and P. Mason, "Analysis of optimal control models for the human locomotion," in *IEEE Conference on Decision and Control*, Dec. 2010, pp. 3415–3420.

[5] 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, 04 2010.

[6] M. Krstic and P. Tsiotras, "Inverse optimal stabilization of a rigid spacecraft," *Automatic Control, IEEE Transactions on*, vol. 44, no. 5, pp. 1042–1049, 1999.

[7] P. Abbeel, A. Coates, and A. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

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

[9] B. Ziebart, A. Maas, J. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. AAAI*, 2008, pp. 1433–1438.

[10] K. Dvijotham and E. Todorov, "Inverse Optimal Control with Linearly-Solvable MDPs," in *Proceedings of the Interntional Conference on Machine Learning*. Citeseer, 2010.

[11] N. Aghasadeghi and T. Bretl, "Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 1561–1566.

[12] A. Keshavarz, Y. Wang, , and S. Boyd, "Imputing a convex objective function," 2011.

[13] Y. Hurmuzlu, F. Génot, and B. Brogliato, "Modeling, stability and control of biped robots–a general framework," *Automatica*, vol. 40, no. 10, pp. 1647–1664, 2004.

[14] M. Raibert and E. Tello, "Legged robots that balance," *IEEE Expert*, vol. 1, no. 4, pp. 89–89, 1986.

[15] A. Goswami, B. Thuilot, and B. Espiau, "A study of the passive gait of a compass-like biped robot: symmetry and chaos," *International Journal of Robotics Research*, vol. 17, no. 12, pp. 1282–1301, 1998.

[16] C. Tomlin, G. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multiagent hybrid systems," *Automatic Control, IEEE Transactions on*, vol. 43, no. 4, pp. 509–521, 1998.

[17] A. Long, T. Murphey, and K. Lynch, "Optimal motion planning for a class of hybrid dynamical systems with impacts," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, pp. 4220–4226.

[18] L. Roussel and A. Goswami, "Generation of energy optimal complete gait cycles for biped robots," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 3. IEEE, 1998, pp. 2036–2041.

[19] P. Channon, S. Hopkins, and D. Pham, "Derivation of optimal walking motions for a bipedal walking robot," *Robotica*, vol. 10, no. 02, pp. 165–172, 1992.

[20] K. Mombaur and M. Sreenivasa, "Inverse optimal control as a tool to understand human yoyo playing," *AIP Conference Proceedings*, vol. 1281, no. 1, pp. 394–397, 2010.

[21] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge Univ Press, 2004.

[22] C. Chen, *Linear system theory and design*. Saunders College Publishing Philadelphia, PA, USA, 1984.

[23] B. Brown and G. Zeglin, "The bow leg hopping robot," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 1. IEEE, 1998, pp. 781–786.

[24] H. Varol, F. Sup, and M. Goldfarb, "Real-time gait mode intent recognition of a powered knee and ankle prosthesis for standing and walking," in *Biomedical Robotics and Biomechatronics, 2008. BioRob 2008. 2nd IEEE RAS & EMBS International Conference on*. IEEE, 2008, pp. 66–72.