Learning to Guide Online Multi-Contact Receding Horizon Planning

Jiayi Wang [1] Teguh Santoso Lembono [2] Sanghyun Kim [3] Sylvain Calinon [2]
Sethu Vijayakumar [1,4] Steve Tonneau [1]

***Abstract*— In Receding Horizon Planning (RHP), it is critical that the motion being executed facilitates the completion of the task, e.g. building momentum to overcome large obstacles. This requires a value function to inform the desirability of robot states. However, given the complex dynamics, value functions are often approximated by expensive computation of trajectories in an extended planning horizon. In this work, to achieve online multi-contact Receding Horizon Planning (RHP), we propose to learn an oracle that can predict local objectives (intermediate goals) for a given task based on the current robot state and the environment. Then, we use these local objectives to construct local value functions to guide a short-horizon RHP. To obtain the oracle, we take a supervised learning approach, and we present an incremental training scheme that can improve the prediction accuracy by adding demonstrations on how to recover from failures. We compare our approach against the baseline (long-horizon RHP) for planning centroidal trajectories of humanoid walking on moderate slopes as well as large slopes where static stability cannot be achieved. We validate these trajectories by tracking them via a whole-body inverse dynamics controller in simulation. We show that our approach can achieve online RHP for 95%-98.6% cycles, outperforming the baseline (8%-51.2%).**

Fig. 1: Snapshots of simulations. Video is available at https://youtu.be/oCsOBHHc9XM

I. Introduction

In this work, we consider the problem of planning multi-contact motions for legged robots to traverse uneven terrain (Fig. 1). This problem is high-dimensional, nonlinear, and subject to discrete changes of dynamics arising from contact switches [1], [2], [3], [4]. Traditionally, multi-contact motions are pre-planned offline and then tracked by a controller [5], [2]. However, in the real world, the pre-planned motion can become invalid due to perturbations, e.g. environment changes and state drifts [5], [6]. To adapt to these changes, online motion (re)-planning is required.

To this end, Receding Horizon Planning (RHP) [6], [7] is a promising solution. Similar to Model Predictive Control (MPC) [8], RHP aims at constantly updating the motion plan for immediate execution based on the state of the robot and the environment. This is usually achieved by solving a finite-horizon Trajectory Optimization (TO) problem [9], which comprises an execution horizon to plan the optimal actions for execution and a prediction horizon to foresee the future. Although the prediction horizon is never executed, it is critical to the success of RHP. Drawing a parallel with Bellman's equation [10], the prediction horizon can be seen as an approximation of the value function, which directs the decision of the execution horizon towards optimal actions that are favorable for the future.

However, approximating the value function with the prediction horizon can significantly increase the computational complexity. To reduce the computation burden, [11] and [12] propose a multi-fidelity TO that relaxes the model accuracy in the prediction horizon. Nevertheless, such a model-based formalism can still struggle to compute online when a long lookahead is needed, e.g. when traversing large slopes.

To further accelerate the computation speed of multi-contact RHP, we explore the possibility of approximating the value function with a learned model. To achieve this, one of the options is to learn a global value function [13] that can reflect the value of the states for a given environment. However, this requires a flexible representation that can parameterize the value function in the coupled state-environment space [14]. In this work, instead of learning the value function directly, we propose to learn an oracle that can predict a local objective—an intermediate goal state that is favourable towards the completion of a given task—based on the current robot state, goal position, and environment model. We then employ a short-horizon TO to compute the optimal actions for reaching the local objective. This is achieved by constructing a local value function based on the local objective and placing it as the terminal cost of the short-horizon TO. We refer to this novel RHP framework as

1 The authors are with the School of Informatics, The University of Edinburgh, United Kingdom.
2 The authors are with the Idiap Research Institute, Switzerland and with EPFL, Switzerland.
3 The author is with the Dept. of AI Machinery, System Engineering Research Division, Korea Institute of Machinery & Materials, South Korea.
4 The author is with The Alan Turing Institute, United Kingdom.
e-mail: jiayi.wang@ed.ac.uk

Locally-Guided Receding Horizon Planning (LG-RHP). The oracle is obtained through supervised learning based on the expert knowledge computed by the long-horizon RHP.

To demonstrate the computational advantage of our approach, we use LG-RHP to plan the centroidal trajectories of the humanoid robot Talos [15] in an online RHP setting, and compare it against the long-horizon RHP as the baseline. This requires the TO to converge within a time budget—the duration of the motion to be executed.

We show that LG-RHP can achieve online computation for 95%-98.6% cycles in the scenarios considered, outperforming the baseline (8%-51.2% cycles computed online). However, due to the prediction error, LG-RHP can reach ill-posed states, from which the TO fails to converge. We show that this issue can be mitigated by a data augmentation technique, in which we add data points to demonstrate how to recover from the states that cause convergence failures.

Our main contributions are:

- We propose to achieve online RHP by approximating the value function with a learned oracle that can predict local objectives as intermediate goals for a given task. These local objectives are then used to construct local value functions to guide a short-horizon TO.
- We realize this approach for planning multi-contact locomotion in an online RHP fashion, and we validate the dynamic feasibility of the planned trajectories by tracking them with a whole-body inverse dynamics controller in simulation.
- We present an incremental training scheme that can iteratively improve the prediction accuracy of the oracle by demonstrating recovery actions from the states that cause convergence failures.

II. Related Work

A. Multi-contact Motion Planning via TO

RHP frameworks usually employ TO to compute the motion plan, which can be time-consuming due to complex dynamics. For instance, although we can generate impressive motions with the whole-body dynamics model [1], [16], [17], the computation time is far from online usage. This is because the whole-body model considers the inertia of every link, which results in high dimensionality and non-convexity.

Alternatively, the centroidal dynamics model [18] has become popular for multi-contact planning [19], [20], [21]. This model has lower dimension, as it only considers the dynamics of the total linear and angular momentum expressed at the Center of Mass (CoM) [18]. Moreover, approximations are introduced to the robot kinematics and the momentum variation induced by the movements of individual robot links. Unfortunately, this model is still non-convex, which can make online computation challenging, except when limiting assumptions (e.g. fixed gait, flat surface) are made [22], [23].

To improve the computation speed, convex inner/outer approximations [21], [24], [25], [26] of the centroidal dynamics are proposed. Nevertheless, convex inner approximations can fail due to a restricted search space [21], while convex outer approximation can violate the dynamics unless it is gradually tightened [26]. To combine the benefit of using both the accurate model and the approximated model, [11] and [12] present a TO formulation that relaxes the model accuracy along the lookahead horizon. However, online computation can still be challenging when a long lookahead is needed.

B. Learning-enhanced Locomotion Planning

Recently, researchers have explored using machine learning to bootstrap locomotion planning. For instance, [27] proposes to learn the evolution of centroidal momenta, which guides an A* planner to generate contact plans. Another line of research tries to accelerate the computation of TO. For example, [28], [29], [7] propose to learn (near)-optimal solutions to warm-start TO. Alternatively, [13], [14], [30], [31] shorten the planning horizon with a learned global value function placed as the terminal cost. However, learning a value function for the multi-contact problem is challenging. The main difficulty is that the value function is defined in a coupled state-environment space, which requires a flexible parameterization that can capture the landscape changes of the value function with respect to environment variations [14]. In contrast, we avoid this issue by learning an oracle to predict intermediate goal states for completing a given task based on the current state, the final goal, and the environment, and then we construct local value functions based on these intermediate goals.

When predicting sequential actions with a learned model, the prediction accuracy can drop dramatically once the robot reaches a state that is un-explored in the training data-set. This problem is known as *distribution shift* [32], which can be mitigated by data augmentation, i.e. adding demonstrations from the states that either appeared from the roll-out of the learned policy [32], [33], or sampled from the expert policy with injected noise [34]. In this work, we present a similar but more targeted data augmentation strategy that only focuses on demonstrating corrective actions from the states that cause convergence failures.

III. Problem Description

Let us denote by $\boldsymbol{x}$ the robot state and $\boldsymbol{u}$ the control input. In traditional RHP frameworks, each cycle aims to compute a motion plan that consists of a state trajectory $\{\boldsymbol{x}_0, ..., \boldsymbol{x}_T\}$, and a control trajectory $\{\boldsymbol{u}_0, ..., \boldsymbol{u}_T\}$ over a finite horizon $[0, T]$. This is achieved by solving a TO problem:

$$\min_{\substack{\boldsymbol{x}_0,\dots,\boldsymbol{x}_T,\\ \boldsymbol{u}_0,\dots,\boldsymbol{u}_T}} \quad \sum_{t=0}^{T} l(\boldsymbol{x}_t, \boldsymbol{u}_t) \tag{1a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{t+1} = \mathcal{F}(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{1b}$$

where l is the running cost[1] and (1b) is the dynamics constraint. Each cycle only executes part of the plan until $t^* < T$. We refer to the horizon being executed as Execution Horizon (EH), and the rest as Prediction Horizon (PH).

[1]We assume that l also attracts the trajectory towards the final goal $\boldsymbol{x}_g$.

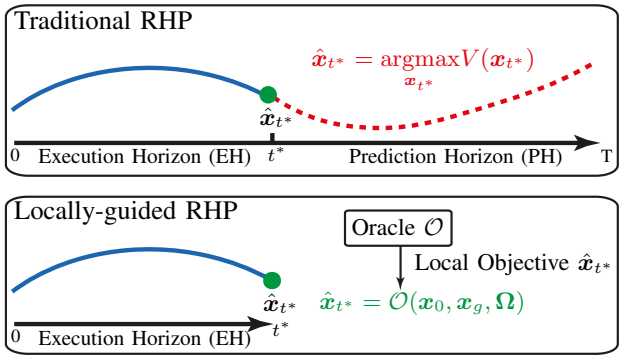

Fig. 2: As Bellman equation (2) suggests, the EH should not only optimize its running cost, but also reach a state $\hat{\boldsymbol{x}}_{t^*}$ that optimizes the value function $V(\boldsymbol{x}_{t^*})$. Since the value function is often hard to obtain analytically, traditional RHP approximates it by expensive computation of trajectories in the PH. In contrast, based on the initial state $\boldsymbol{x}_0$, the final goal $\boldsymbol{x}_g$, and the environment $\boldsymbol{\Omega}$, our LG-RHP employs a learned oracle $\mathcal{O}$ to predict the optimal $\hat{\boldsymbol{x}}_{t^*}$ as a local objective for guiding a short-horizon TO to plan the EH.

Although the PH never gets executed, computing it is critical to RHP. To illustrate this, let us consider the TO formulation from Bellman's principle of optimality,

$$V(\boldsymbol{x}_t) = \min_{\boldsymbol{u}_t} [l(\boldsymbol{x}_t, \boldsymbol{u}_t) + V(\boldsymbol{x}_{t+1})], \tag{2}$$

where $\boldsymbol{u}_t$ is the optimal policy, $V(\boldsymbol{x})$ is the value function, $\boldsymbol{x}_{t+1}$ is the state evolved from the dynamics (1b). Drawing a parallel to the Bellman equation, we note that the EH—which approximates the optimal policy—should optimize its running cost l while leading to a state $\hat{\boldsymbol{x}}_{t^*}$ that optimizes the value function $V(\boldsymbol{x}_{t^*})$. The value function $V(\boldsymbol{x}_{t^*})$ reflects the desirability for the EH to arrive at a certain state $\boldsymbol{x}_{t^*}$. Unfortunately, for complex systems, an analytical expression of the value function is often unavailable. Hence, TO approximates the value function by computing trajectories in the PH (Fig. 2). Nevertheless, this can increase the complexity of TO, which hinders online computation.

To speed up the computation, a promising direction is to replace the PH with an approximate value function model $\tilde{V}(\boldsymbol{x})$. This leads to a short-horizon TO where the planning horizon terminates at t^* (the end of EH), and (1) becomes

$$\min_{\substack{\boldsymbol{x}_0,\dots,\boldsymbol{x}_{t^*},\\ \boldsymbol{u}_0,\dots,\boldsymbol{u}_{t^*}}} \underbrace{\sum_{t=0}^{t^*-1} l(\boldsymbol{x}_t, \boldsymbol{u}_t)}_{\text{EH}} + \tilde{V}(\boldsymbol{x}_{t^*}), \tag{3}$$

where $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$ subject to system dynamics (1b). Based on this idea, [13] proposes to learn a global value function $\tilde{V}(\boldsymbol{x}|\theta)$ parameterized by θ. However, multi-contact problem requires the consideration of environment geometry. This gives rise to the challenge of finding a flexible representation of the coupled state-environment space [14]. In this work, instead of learning a value function over the entire state-environment space, we propose to learn an oracle $\mathcal{O}$ to predict intermediate goal states $\hat{\boldsymbol{x}}_{t^*}$ that are favorable for completing a given task, based on the current state $\boldsymbol{x}_0$, final goal state $\boldsymbol{x}_g$, and environment model $\boldsymbol{\Omega}$:

$$\hat{\boldsymbol{x}}_{t^*} = \mathcal{O}(\boldsymbol{x}_0, \boldsymbol{x}_g, \boldsymbol{\Omega}). \tag{4}$$

We refer to these intermediate goal states as local objectives, and use them to construct local value functions $\tilde{V}(\boldsymbol{x}_{t^*}|\hat{\boldsymbol{x}}_{t^*})$ to attract the short-horizon TO towards the local objective $\hat{\boldsymbol{x}}_{t^*}$. We call this approach Locally-Guided Receding Horizon Planning (LG-RHP) and illustrate the idea in Fig. 2.

Although we can learn to predict the optimal policy of the EH directly from past experiences, the learning error may lead to trajectories that violate system dynamics and cause tracking failures. Hence, we decide to compute the EH using TO, which guarantees the dynamic feasibility. Next, we present how to implement LG-RHP in the context of multi-contact planning and the approach for learning the oracle.

IV. Methods

This section presents the technical details of LG-RHP. In Section IV-A, we briefly describe the RHP formulation for planning multi-contact motions, and we refer the reader to [11] for more details. The long-horizon version of this formulation is used to compute the training data offline, while the short-horizon version is employed in LG-RHP for planning the EH. In Section IV-B, we elaborate the oracle formulation for multi-contact planning, and we present the incremental training scheme in Section IV-C.

Furthermore, the following assumptions are made:

- We plan bipedal walking motion, in which a step consists of 3 phases: pre-swing (double support), swing (single support), and landing (double support).
- EH always plans a single-step motion, and thus the oracle predicts the local objective for making one step.
- We model the robot feet as rectangular patches, and we assign a contact point to each vertex of the rectangle.
- We assume that the environment is made of rectangular surfaces. We pre-define the sequence of these surfaces upon which the swing foot lands, while we optimize the contact timings and the locations (within the surfaces).
- We approximate the kinematic constraints of the CoM and the relative positions of contacts as convex polytopes attached to the feet in contact, with the orientations aligned to the contact surfaces [35].

A. Receding Horizon Planning for Multi-contact Motions

This section briefly describes the RHP formulation for planning multi-contact motions (see [11] for more details). In each planning cycle, given a finite lookahead of n steps, the current robot state $\boldsymbol{x}_{cur}$, the goal state $\boldsymbol{x}_g$, and the sequence of contact surfaces on which the robot will step upon, the RHP framework computes a motion plan of N_{ph} phases by solving a TO problem. The motion plan is composed of:

- $\boldsymbol{\mathcal{X}} = \{\mathcal{X}^1, ..., \mathcal{X}^{N_{ph}}\}$: state trajectory $\mathcal{X}^q$ of all phases $q \in N_{ph}$. Each phase is discretized to N_k knots: $\mathcal{X}^q = \{\boldsymbol{x}_1^q, ..., \boldsymbol{x}_{N_k}^q\}$. We define the state $\boldsymbol{x} = [\boldsymbol{c}^\top, \dot{\boldsymbol{c}}^\top, \boldsymbol{L}^\top]^\top$,

where $\boldsymbol{c}, \dot{\boldsymbol{c}} \in \mathbb{R}^3$ are the CoM position and velocity, and $\boldsymbol{L} \in \mathbb{R}^3$ is the total angular momentum.

- $\boldsymbol{\mathcal{U}} = \{\mathcal{U}^1, ..., \mathcal{U}^{N_{ph}}\}$: discretized control trajectory $\mathcal{U}^q = \{\boldsymbol{u}_1^q, ..., \boldsymbol{u}_{N_k}^q\}$ of all phases. We denote $\boldsymbol{u} = [\boldsymbol{f}_1^\top, ..., \boldsymbol{f}_{N_c}^\top]^\top$, which collects the contact force $\boldsymbol{f}_c \in \mathbb{R}^3$ of all contact points $c \in \{1, ..., N_c\}$.
- $\boldsymbol{\mathcal{P}} = \{\boldsymbol{p}^1, ..., \boldsymbol{p}^n\}$: a list of contact locations, where $\boldsymbol{p}^i \in \mathbb{R}^3$ is the contact location of the i-th step.
- $\boldsymbol{\mathcal{T}} = \{t^1, ..., t^{N_{ph}}\}$: a list of phase switching timings. The time step of each phase is $\tau^q = (t^q - t^{q-1})/N_k$.

The discretized TO is given by:

$$\min_{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{P}}} \quad \sum_{q=1}^{N_{ph}} J^q(\mathcal{X}^q, \mathcal{U}^q) + \phi(\boldsymbol{x}_T) \tag{5a}$$

$$\text{s.t.} \quad h(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{P}}) \leq 0, \tag{5b}$$

where $J^q = \sum_{k=1}^{N_k} \tau^q (\ddot{\boldsymbol{c}}_k^{q\top} \ddot{\boldsymbol{c}}_k^q + \boldsymbol{L}_k^{q\top} \boldsymbol{L}_k^q)$ is the running cost of each phase that encourages smooth trajectory, $\phi(\boldsymbol{x}_T) = (\boldsymbol{x}_T - \boldsymbol{x}_g)^\top (\boldsymbol{x}_T - \boldsymbol{x}_g)$ is the terminal cost that attracts the end state $\boldsymbol{x}_T$ towards the final goal $\boldsymbol{x}_g$, and (5b) collects the constraints described as:

- $\boldsymbol{x}_0 = \boldsymbol{x}_{cur}$, the initial condition constraint.
- $0 \leq t^1 \leq \cdots \leq t^{N_{ph}} \leq T_{max}$ restricts the phase switching timings to increase monotonically.
- $\boldsymbol{p}^i \in \mathcal{S}_i$ constrains each contact $\boldsymbol{p}^i$ to stay on the pre-assigned surface $\mathcal{S}_i = \{\boldsymbol{p} \in \mathbb{R}^3, \boldsymbol{d}_i^T \boldsymbol{p} = e_i, S_i \boldsymbol{p} \leq s_i\}$, where the equality defines the plane containing the surface and the inequalities bound the surface.
- $\boldsymbol{p}^i \in \mathcal{R}_{i-1}$ constrains each contact $\boldsymbol{p}^i$ to be within the reachable space $\mathcal{R}_{i-1}$ of the previous footstep $\boldsymbol{p}^{i-1}$.
- $\boldsymbol{c}_k^q \in \mathcal{K}_l, \forall l \in \mathcal{C}^q$ restricts the CoM position at the k-th knot to stay in the reachable space $\mathcal{K}_l$ established by each active contact $l \in \mathcal{C}^q$ at each phase q.
- $\boldsymbol{x}_{k+1}^q = \boldsymbol{x}^q + \tau^q \mathcal{F}^q(\boldsymbol{x}_k^q, \boldsymbol{u}_k^q)$, the dynamics constraint approximated by forward Euler scheme. We consider the centroidal dynamic model as detailed in [11].

In this work, we use (5) with a long lookahead ($n \geq 2$) to compute the data-set and also employ it as the baseline. For LG-RHP, we plan optimal actions by (5) with 1-step lookahead (only covers EH) along with the adapted terminal cost and the constraints as described in Section IV-B.

B. Oracle Formulation for Multi-Contact Planning

This section presents the oracle formulation for multi-contact planning and the associated variable definitions (Fig. 3-a). Following the idea in Section III, we define the oracle as:

$$\boldsymbol{x}^*, \boldsymbol{p}^*, \boldsymbol{\mathcal{T}}^* = \mathcal{O}(\delta_{l/r}, \boldsymbol{x}_0, \boldsymbol{p}_0, \boldsymbol{\Omega}, \boldsymbol{x}_g). \tag{6}$$

The oracle's output is an intermediate goal configuration after making a step, which includes: i) target CoM state $\boldsymbol{x}^*$; ii) target contact location $\boldsymbol{p}^*$; iii) target phase switching timings $\boldsymbol{\mathcal{T}}^* = \{\tilde{t}^1, \tilde{t}^2, \tilde{t}^3\}$ for the 3 phases that complete the step. To predict this goal configuration, we require:

- $\delta_{l/r} \in \{L, R\}$: swing foot indicator that informs which foot (left/right) will be re-positioned.

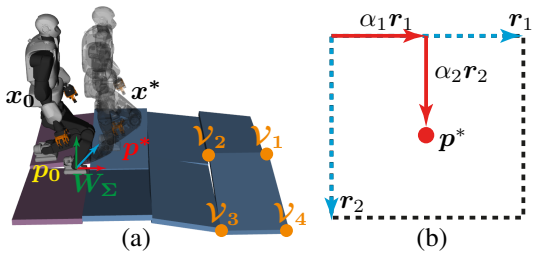

Fig. 3: (a) Variables definition of the oracle. $\boldsymbol{x}_0$ and $\boldsymbol{p}_0$ are the initial CoM state and initial swing-foot position. $\boldsymbol{x}^*$ and $\boldsymbol{p}^*$ are the target CoM state and the contact location respectively. Purple patches are initial contact surfaces, while blue patches are for future steps. Contact surfaces are modeled by their vertices $\mathcal{V}_i$. Spatial terms are defined in the contact frame $\boldsymbol{W}_\Sigma$ established at the non-swing foot. (b) The target contact location $\boldsymbol{p}^*$ is represented as the vector sum of $\alpha_1 \boldsymbol{r}_1$ and $\alpha_2 \boldsymbol{r}_2$, which scale along the contact surface borders $\boldsymbol{r}_1, \boldsymbol{r}_2 \in \mathbb{R}^3$ with the proportion defined by $\alpha_1, \alpha_2 \in [0, 1]$.

- $\boldsymbol{x}_0$: initial CoM state.
- $\boldsymbol{p}_0$: initial contact location of the swing foot.
- $\boldsymbol{\Omega} = \{\mathcal{S}_{l0}, \mathcal{S}_{r0}, \mathcal{S}_1, ..., \mathcal{S}_n\}$: a finite-preview of the environment, where $\mathcal{S}_{l0}, \mathcal{S}_{r0}$ are the surfaces that the left and right feet initially stand upon, $\mathcal{S}_1, ..., \mathcal{S}_n$ are the surfaces that future n steps will land on; We represent each surface with their vertices $\mathcal{S} = \{\mathcal{V}_1, ..., \mathcal{V}_4\}$, $\mathcal{V} \in \mathbb{R}^3$.
- $\boldsymbol{x}_g$: final goal position (fixed in our case).

We define all spatial terms in the contact-foot frame $\boldsymbol{W}_\Sigma$, whose origin locates at the position of the non-swing foot, and the orientation aligns with the surface in contact.

Furthermore, we introduce an on-surface parameterization for the target contact location $\boldsymbol{p}^* = \alpha_1 \boldsymbol{r_1} + \alpha_2 \boldsymbol{r_2}$, which transforms the 3D position as the sum of two vectors, scaling along the surface borders $\boldsymbol{r}_1, \boldsymbol{r}_2 \in \mathbb{R}^3$, and we predict the scaling factors $\alpha_1, \alpha_2 \in [0, 1]$ (Fig. 3-b).

To guide the LG-RHP, we adapt the short-horizon version of (5) that only covers the EH with the following changes. First, we replace the terminal cost with $(\boldsymbol{x}_T - \boldsymbol{x}^*)^\top (\boldsymbol{x}_T - \boldsymbol{x}^*) + (\boldsymbol{p}^1 - \boldsymbol{p}^*)^\top (\boldsymbol{p}^1 - \boldsymbol{p}^*)$ that encourages the end state $\boldsymbol{x}_T$ and the contact location $\boldsymbol{p}^1$ to approach the predicted targets. Second, we narrow down[2] the search space of phase switching timings t^i centered at their predicted values $(1 - \epsilon)\tilde{t}^i \leq t^i \leq (1 + \epsilon)\tilde{t}^i$, where ϵ is a user-defined slack.

C. Incremental Training Scheme

This section describes the incremental training scheme for learning the oracle. The key idea is to improve the prediction accuracy by adding data points to demonstrate corrective actions from the states that cause convergence failures. As depicted in Fig. 4, in each training iteration i, we train an oracle $\mathcal{O}_i$ based on the current data-set $\mathcal{D} = \mathcal{D}_0 \cup \mathcal{D}_1^* \cup ... \cup \mathcal{D}_{i-1}^*$, where $\mathcal{D}_0$ is the initial data-set, and

[2]We empirically find that limiting the search space of the phase switching timings can result in more efficient computation than using costs terms to bias their decisions.

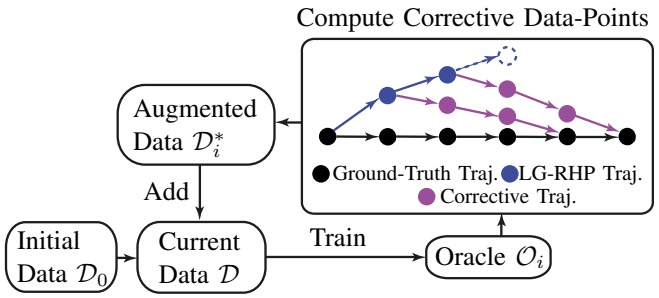

Fig. 4: The incremental training scheme concept. Each iteration i trains an oracle $\mathcal{O}_i$ on the data-set $\mathcal{D}$ that aggregates the initial data-set $\mathcal{D}_0$ and the augmented data-sets $\mathcal{D}_i^*$. The augmented data-set contains recovery actions (purple), starting from 1-3 cycles before LG-RHP fails (dashed blue circle) until the cycle converges back to ground-truth trajectory.

$\mathcal{D}_1^* \cup ... \cup \mathcal{D}_{i-1}^*$ are the augmented data-sets from previous training iterations. The initial data-set $\mathcal{D}_0$ is achieved by rolling out the long-horizon RHP with a 3-step PH[3] over a set of randomly sampled environments, and then extracting the data points from the EH of each cycle. For generating the augmented dataset $\mathcal{D}_i^*$, we firstly use LG-RHP with the currently trained oracle $\mathcal{O}_i$ to plan trajectories in a RHP fashion on the previously sampled environments. Then, we use the long-horizon RHP to compute recovery actions—starting from 1-3 cycles before LG-RHP fails—until the cycle converges back to the ground-truth trajectory (roll-out of long-horizon RHP on the same terrain). We repeat the process until there is no further improvement.

V. Results

To highlight the computational advantage of our approach, we compare Locally-Guided Receding Horizon Planning (LG-RHP) against the baseline (long-horizon RHP) in an online setting, in which the TO should converge within a time budget, i.e. the duration of the motion under execution[4].

A. Experiment Setup

We carry out two case studies: walking on moderate slope (Fig. 5) and large slope (Fig. 6). Planning motions on these terrains is challenging because the admissible contact force is limited by the orientation of the surface in contact, requiring careful selections of contact locations and the timings [26].

We use LG-RHP and the baseline to compute centroidal trajectories of the humanoid robot Talos [15] in RHP fashion. Both LG-RHP and the baseline are tested on the terrains that are unseen during training[5]. We refer to the trial on each terrain as an episode. In each cycle, we warm-start the TO with the solution of two cycles before. A cycle converges online if the TO is solved within the time budget. If it fails to converge online, we still let the TO compute until convergence and continue the episode, unless no solution is found (fail to converge). In LG-RHP, we employ the oracle trained with the best prediction accuracy. Lastly, we validate the planned trajectories by tracking them via a whole-body inverse dynamics controller [36] in simulation (Video: https://youtu.be/oCsOBHHc9XM).

[3]We choose 3-step PH, as we find a longer lookahead does not improve the quality of the motion (cost), while increasing the computation time.

[4]We find the controller can track the planned trajectories without having large deviations in simulation. Thus, we assume the next-cycle motion starts from the terminal state planned for the current cycle, and use the motion duration of the current cycle as the time budget.

[5]We generate the terrains by uniformly sampling the surface orientations, and we split them into training (80%) and testing (20%).

TABLE I: Cycle-wise success rate for CS1.

Method	Offline	Online	Time Out	Fail (Conv.)
LG-RHP	99.05%	**98.63%**	**0.42%**	0.95%
Baseline (1-Step PH)	99.9%	51.21%	48.69%	0.1%

B. Implementation Details

We model TO problems in Python and solve them by the interior-point method of KNITRO [37]. We use the automatic differentiation framework CasADi [38] to provide the gradients and the Hessians. Computations are achieved on a desktop with an Intel i9-CPU (3.6GHz) and 64GB memory.

The oracle is modelled as a Neural Network (NN) of 4 hidden layers, each layer has 256 nodes with ReLu activation functions. The NN is implemented and trained with Tensorflow [39]. Furthermore, we find that the data distributions of the two types of terrains exhibit different modalities, i.e. when traversing the large slope, the robot has larger momentum variations than walking on the moderate slopes. Mixing these data points can result in a discontinuous and unbalanced data-set, on which a single NN can struggle to interpolate. As a result, we decide to train separate oracles for the two types of terrains. The possibility of generalizing across these two modalities is discussed in Section VI.

C. Case Study 1 (CS1): Moderate Slope

This section presents the experiment result on the terrain with moderate slopes (Fig. 5). Although quasi-static motions can be quickly found for this terrain[6] [40], we are interested in planning dynamic walking motions using TO. This provides a unified approach to handle non-quasi-static cases (e.g. large slope) and also allows more efficient task completion.

In this case study, we choose the long-horizon RHP with 1-step PH as the (fastest) baseline. For LG-RHP, we set the slack of the phase switching timing constraints as $\epsilon = 0.6$. We find this can increase the chance of finding a solution (enlarged search space) without sacrificing much computation speed. Each episode plans maximum 28 cycles.

To show the computational benefit of our approach, we compare LG-RHP and the baseline based on their cycle-wise success rate (the percentage of cycles that converged) for online computation. As Table I lists, in offline mode (unlimited time budget), both methods can achieve 99% cycle-wise success rate, while having up to 1% of the cycles fail to converge. However, when online computation is required (consider time out), only LG-RHP maintains a success rate of

[6]This is because the friction cone associated with each contact surface contains force vectors to cancel the gravity.

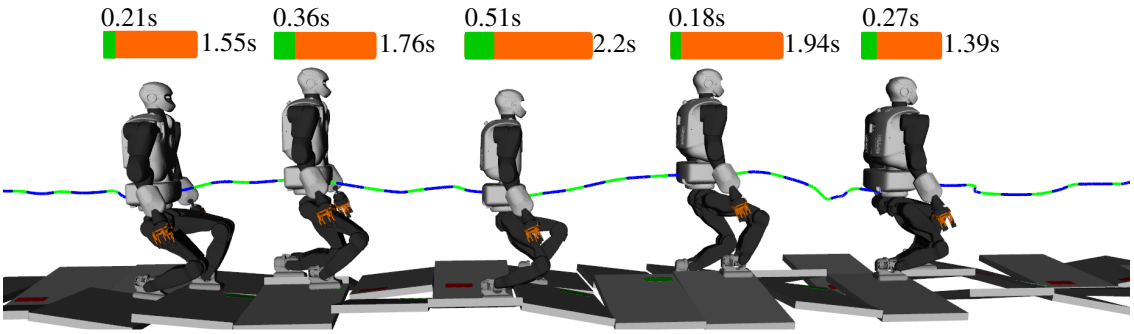

Fig. 5: Snapshots of simulations on moderate slopes (5-12 degrees). We use orange bar to represent the motion duration of current cycle, and green bar as the computation time for the next cycle.

TABLE II: Average computation time of LG-RHP v.s. average time budget for the cycles that converged online.

Terrain	Avg. Comput. Time	Avg. Time Budget
Moderate Slope (CS1)	**0.37 +/- 0.19**s	1.97 +/- 0.23s
Large Slope (CS2)	**0.36 +/- 0.23**s	2.01 +/- 0.48s

TABLE III: 95% CI of the average number of cycles that consecutively converge online (max. 28).

Method	95% CI (CS1)
LG-RHP	**23.5 +/- 2.0** cycles
Baseline (1-Step PH)	0.6 +/- 0.3 cycles

98.63%, whereas the baseline drops to 51.21%. Additionally, as Table II shows, LG-RHP only consumes on average 19% of the time budget. This suggests the potential of using LG-RHP in real robot control, as the remaining time budget can be allocated to the overheads, e.g. data transmission.

Moreover, to demonstrate that LG-RHP can consecutively achieve online RHP in an episode, we check its 95% Confidence Interval (CI) on the number of cycles that continually converged online. As Table III shows, the CI of LG-RHP is 23.5 +/- 2.0 cycles, indicating that LG-RHP can perform online computation for 76.8% to 91.1% cycles of an episode. In contrast, the baseline CI is 0.6 +/- 0.3 cycles, implying its incapability of achieving online RHP. A sequence of snapshots of our simulation can be seen in Fig. 5.

D. Case Study 2 (CS2): Large Slope

This section considers the large slope terrain (Fig. 6), on which the robot cannot maintain static balance. For this terrain, the baseline requires 2-step PH to foresee the large slope early enough and start building its momentum.

Each episode starts from the cycle when the large slope is inside the lookahead horizon and ends at the cycle when the robot gets off the large slope (6 cycles). For LG-RHP, we define the slack of the phase switching timing constraints $\epsilon = 0.15$, as empirically determined to give a good balance between the success rate and the computation speed.

In Table IV, we compare the cycle-wise success rate of LG-RHP and the baseline. In offline mode, the success rate of the baseline and LG-RHP are 99.5% and 95.99% respectively, while the rest of the cycles fail to converge.

TABLE IV: Number of cycles converged online for CS2.

Method	Offline	Online	Time Out	Fail (Conv.)
LG-RHP	95.99%	**95.0%**	**0.99%**	4.01%
Baseline (2-Step PH)	99.5%	7.99%	91.51%	0.5%

TABLE V: Episodic success rate for CS2.

Method	Success	Time Out	Fail (Conv.)
LG-RHP	**76.1%**	**3.8%**	20.1%
Baseline (2-Step PH)	0.0%	98.4%	1.6%

However, when it comes to online mode, the cycle-wise success rate of LG-RHP stays at 95.0%, whereas the baseline decreases to 7.99%. Furthermore, as indicated in Table II, LG-RHP only consumes on average 18% of the time budget.

To examine whether the online RHP can be consecutively achieved in an episode, we compare the episodic success rate of each method. We define an episode as successful if all the cycles have their TO converged online. As Table V lists, the baseline never completes an episode in an online fashion (0.0%), and 98.4% episodes failed due to time out. In contrast, LG-RHP can achieve online RHP over 76.1% episodes, while only 3.8% of the failures are due to time out, and the rest (20.1%) are caused by convergence failures induced by the prediction error of the oracle. A sequence of simulation snapshots are presented in Fig. 6.

E. Improving Prediction with Incremental Training Scheme

This section demonstrates the effectiveness of our incremental training scheme (Section IV-C). In Table VI, we list the episodic success rate of LG-RHP achieved on the training environments with oracles trained from different iterations of the data augmentation process. Our result shows that adding corrective data points of interest can increase the prediction accuracy, which improves the episodic success rate of LG-RHP. The success rate saturates after 5 training iterations.

VI. Discussion

We empirically show that LG-RHP can achieve online RHP of multi-contact motions while consuming on average 19% of the time budget. This suggests that LG-RHP can be used for real-world robot control, as it leaves sufficient time for overheads, e.g. data transmission.

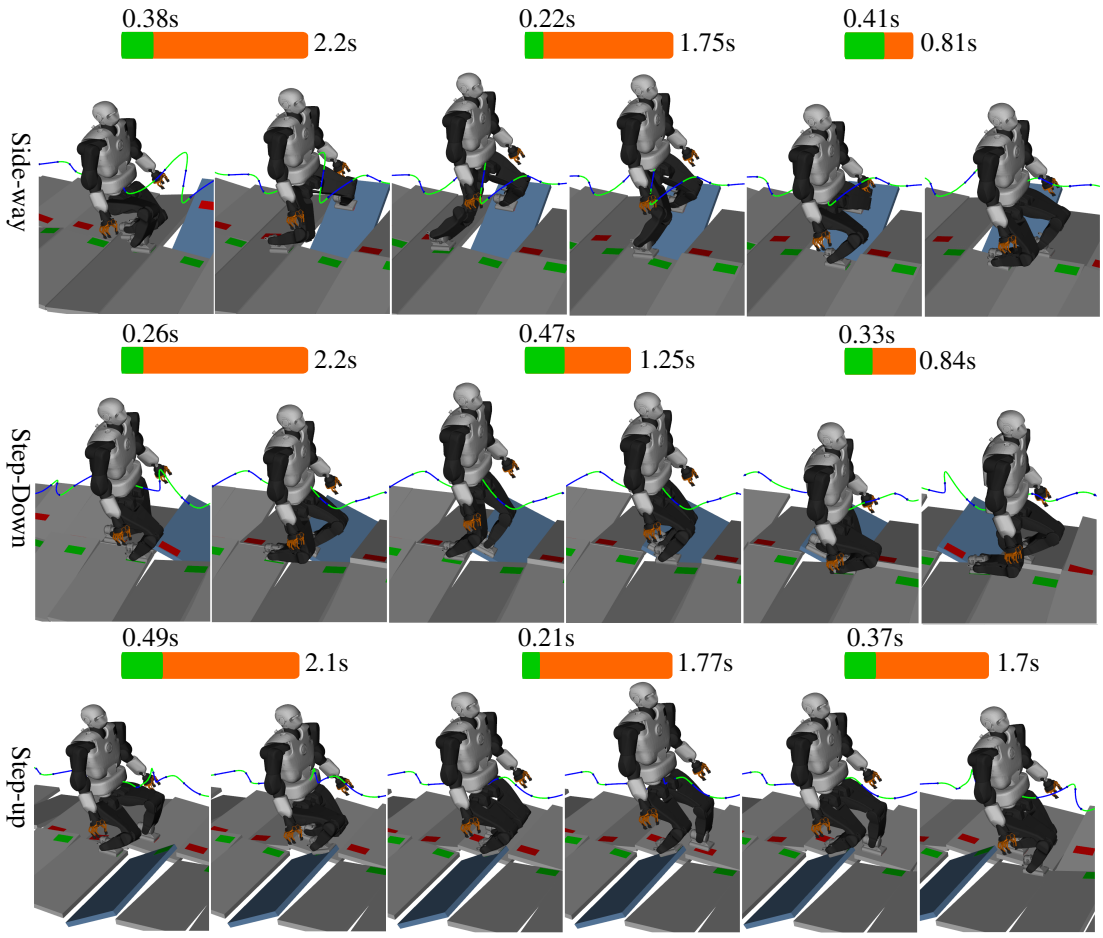

Fig. 6: Snapshots of simulations on the large slope (17-25 degrees). The blue block is the large slope (25 degrees), while the rest are moderate slopes (5-12 degrees). The robot tends to build momentum to achieve dynamic balancing on the large slope. We show that LG-RHP can be used online, as the computation time of the next cycle (green bar) is smaller than the motion duration of the current cycle (orange bar).

TABLE VI: Episodic success rate of different iterations of the incremental training scheme on the training environments.

Terrain	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5
Moderate (CS1)	67.2%	76.2%	80.3%	81.8%	82.1%
Large (CS2)	71.5%	75.3%	79.5%	80.4%	81.0%

However, as mentioned in Section V-B, we find that the data points for the two types of terrains exhibit different modalities. This can impose challenges when training a single NN on the combined data-set. Although we capture the two modalities by using separate NNs, it is worthwhile to explore a more unified approach that can handle multi-modal data, e.g. using mixture density networks [41].

Furthermore, LG-RHP struggles in the following two cases. First, the oracle has prediction errors due to imperfect fitting and insufficient data coverage. This can lead the robot to ill-posed states, and cause failures to converge. Although our incremental training scheme can mitigate this issue, it is hard to reach a 100% success rate. To improve the performance, we can use a Recurrent Neural Network (RNN) or impose safety constraints. Second, although LG-RHP features a short-horizon TO, it is still a nonlinear problem that may time out. To alleviate this issue, we can reduce the number of decision variables by representing trajectories with parameterized curvatures, e.g. Bezier Curves [21].

Lastly, multi-contact problems necessarily require discrete decisions, i.e. the sequence of contact surfaces [40] and the gait pattern [42]. It would be beneficial to extend the oracle to predict these discrete choices.

VII. Conclusion

We present Locally-Guided Receding Horizon Planning (LG-RHP)—a novel RHP framework for planning multi-contact motions. The proposed approach features a learned oracle that can predict local objectives, which are used for building local value functions for guiding a short-horizon TO. Our experiment shows that LG-RHP can achieve online computation for 95%-98.6% cycles, which outperforms the baseline (8%-51.2%). Moreover, LG-RHP only requires around 19% of the time budget, suggesting the potential for real-robot control. However, due to the prediction error, LG-

RHP does not achieve 100% success rates. In future work, we plan to improve the success rate by using RNN or adding safety constraints. We also plan to test LG-RHP on real robot.

ACKNOWLEDGEMENTS

This research is supported by the EU H2020 project Memory of Motion (MEMMO, 780684), EPSRC UK RAI Hub for Offshore Robotics for Certification of Assets (ORCA, EP/R026173/1) and The Alan Turing Institute.

REFERENCES

[1] M. Posa and R. Tedrake, "Direct trajectory optimization of rigid body dynamical systems through contact," in *Algorithmic foundations of robotics X*. Springer Berlin Heidelberg, 2013, pp. 527–542.

[2] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 1560–1567, 2018.

[3] M. Toussaint, K. Allen, K. Smith, and J. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Proc. Robotics: Science and Systems (R:SS)*, 2018.

[4] T. Stouraitis, I. Chatzinikolaidis, M. Gienger, and S. Vijayakumar, "Online hybrid motion planning for dyadic collaborative manipulation via bilevel optimization," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1452–1471, 2020.

[5] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, 2018.

[6] H.-W. Park, P. M. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," in *Proc. Robotics: Science and Systems*, 2015.

[7] O. Melon, R. Orsolino, D. Surovik, M. Geisert, I. Havoutis, and M. Fallon, "Receding-horizon perceptive trajectory optimization for dynamic legged locomotion with learned initialization," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2021, pp. 9805–9811.

[8] C. Mastalli *et al.*, "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2020.

[9] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, 2010.

[10] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[11] J. Wang, S. Kim, S. Vijayakumar, and S. Tonneau, "Multi-fidelity receding horizon planning for multi-contact locomotion," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, 2021, pp. 53–60.

[12] H. Li, R. J. Frei, and P. M. Wensing, "Model hierarchy predictive control of robotic systems," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 3373–3380, 2021.

[13] M. Zhong, M. Johnson, Y. Tassa, T. Erez, and E. Todorov, "Value function approximation and model predictive control," in *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*. IEEE, 2013, pp. 100–107.

[14] R. Deits, T. Koolen, and R. Tedrake, "Lvis: Learning from value function intervals for contact-aware robot controllers," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2019, pp. 7762–7768.

[15] O. Stasse *et al.*, "Talos: A new humanoid research platform targeted for industrial applications," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, 2017, pp. 689–695.

[16] K. H. Koch, K. Mombaur, and P. Soueres, "Optimization-based walking generation for humanoid robot," *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 498–504, 2012.

[17] T. Erez and E. Todorov, "Trajectory optimization for domains with contacts using inverse dynamics," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 4914–4919.

[18] D. E. Orin, A. Goswami, and S. H. Lee, "Centroidal dynamics of a humanoid robot," *Auton. Robots*, vol. 35, no. 2-3, pp. 161–176, 2013.

[19] A. Herzog, S. Schaal, and L. Righetti, "Structured contact force optimization for kino-dynamic motion generation," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 2703–2710.

[20] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Transactions on Robotics*, 2018.

[21] P. Fernbach, S. Tonneau, O. Stasse, J. Carpentier, and M. Taïx, "C-croc: Continuous and convex resolution of centroidal dynamic trajectories for legged robots in multicontact scenarios," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 676–691, 2020.

[22] S. Kajita *et al.*, "Biped walking pattern generation by using preview control of zero-moment point," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, vol. 2, 2003, pp. 1620–1626.

[23] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-dimensional bipedal walking control based on divergent component of motion," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 355–368, 2015.

[24] S. Caron and Q.-C. Pham, "When to make a step? tackling the timing problem in multi-contact locomotion by topp-mpc," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, 2017, pp. 522–528.

[25] H. Dai and R. Tedrake, "Planning robust walking motion on uneven terrain via convex optimization," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, 2016, pp. 579–586.

[26] B. Ponton, M. Khadiv, A. Meduri, and L. Righetti, "Efficient multi-contact pattern generation with sequential convex approximations of the centroidal dynamics," *IEEE Transactions on Robotics*, 2021.

[27] Y.-C. Lin, B. Ponton, L. Righetti, and D. Berenson, "Efficient humanoid contact planning using learned centroidal dynamics prediction," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2019, pp. 5280–5286.

[28] E. Dantec *et al.*, "Whole body model predictive control with a memory of motion: Experiments on a torque-controlled talos," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2021.

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

[30] A. Parag, S. Kleff, L. Saci, N. Mansard, and O. Stasse, "Value learning from trajectory optimization and sobolev descent: A step toward reinforcement learning with superlinear convergence properties," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2022.

[31] J. Viereck, A. Meduri, and L. Righetti, "Valuenetqp: Learned one-step optimal control for legged locomotion," in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 931–942.

[32] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. 14-th Int. Conf. on Artif. Intell. and Stats.* JMLR, 2011, pp. 627–635.

[33] A. Venkatraman, B. Boots, M. Hebert, and J. A. Bagnell, "Data as demonstrator with applications to system identification," in *ALR Workshop, NIPS*, 2014.

[34] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "Dart: Noise injection for robust imitation learning," in *Conference on robot learning*. PMLR, 2017, pp. 143–156.

[35] S. Tonneau, P. Fernbach, A. D. Prete, J. Pettré, and N. Mansard, "2pac: Two-point attractors for center of mass trajectories in multi-contact scenarios," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 5, pp. 1–14, 2018.

[36] A. Del Prete and N. Mansard, "Robustness to joint-torque-tracking errors in task-space inverse dynamics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1091–1105, 2016.

[37] R. H. Byrd, J. Nocedal, and R. A. Waltz, "Knitro: An integrated package for nonlinear optimization," in *Large-Scale Nonlinear Optimization*. Springer US, 2006, pp. 35–59.

[38] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – a software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, Mar. 2018.

[39] M. Abadi *et al.*, "Tensorflow: a system for large-scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[40] D. Song *et al.*, "Solving footstep planning as a feasibility problem using l1-norm minimization," *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[41] C. M. Bishop, "Mixture Density Networks," Aston University, Tech. Rep., 1994.

[42] J. Wang *et al.*, "Automatic gait pattern selection for legged robots," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2020, pp. 3990–3997.