

# Receding-Horizon Perceptive Trajectory Optimization for Dynamic Legged Locomotion with Learned Initialization

Oliwier Melon , Romeo Orsolino , David Surovik , Mathieu Geisert ,  
Ioannis Havoutis  and Maurice Fallon 

**Abstract**—To dynamically traverse challenging terrain, legged robots need to continually perceive and reason about upcoming features, to adjust the locations and timings of future footfalls and to leverage momentum strategically. We present a pipeline that enables flexibly-parametrized trajectories for perceptive and dynamic quadruped locomotion to be optimized in an online, receding-horizon manner. The initial guess passed to the optimizer affects the compute time to convergence and quality of the solution. We consider two methods for generating these guesses. The first is a *heuristic* initializer which provides a simple guess and requires significant optimization but is nonetheless suitable for adaptation to upcoming terrain. We demonstrate experiments using the ANYmal C quadruped, with fully onboard sensing and computation, to cross obstacles at moderate speeds using this technique. Our second approach uses *latent-mode trajectory regression* to imitate expert data—while avoiding invalid interpolations between distinct behaviors—such that minimal optimization is needed. This enables high-speed motions that make more expansive use of the robot’s capabilities. We demonstrate it on flat ground with the real robot and provide numerical trials that progress toward deployment on terrain. These results illustrate a paradigm for advancing beyond short-horizon dynamic reactions, toward the type of intuitive and adaptive locomotion planning exhibited by animals and humans.

## I. INTRODUCTION

State-of-the-art legged-locomotion controllers can maintain dynamic motion while handling perturbations from external forces on terrain of modest roughness. Alternatively, locomotion planners can reason over longer horizons to strategically plan footholds and paths across more prominent obstacles, such as those seen in industrial settings. However, the benefits of these two paradigms—speed and foresight, respectively—are not trivial to combine. As a result legged robots have limited applicability in scenarios where momentum must be leveraged.

Meanwhile, legged animals can dynamically traverse obstacles with relative ease. They are able to make rapid inferences about their circumstances and subconsciously relate them to relevant prior experiences to effectively plan and execute motions which leverage the properties of their bodies in a coordinated manner. Biological locomotion serves as inspiration to motion generation in robotics.

This work was supported by the UKRI/EPSCRC RAIN Hub [EP/R026084/1] and the EU H2020 Projects MEMMO and THING, the EPSRC grant ‘Robust Legged Locomotion’ [EP/S002383/1] and a Royal Society University Research Fellowship (Fallon). This work was conducted as part of ANYmal Research, a community to advance legged robotics. The authors are with Oxford Robotics Institute, University of Oxford, UK. Email: {omelon, rorsolino, dsurovik, mathieu, ioannis, mfallon}@robots.ox.ac.uk.



Fig. 1. The ANYmal C quadruped using online optimization and real-time perception to plan trajectories to climb a 0.20 m step.

## A. Planning Dynamic Motion Online

Dynamically complex robot motion plans are commonly generated through trajectory optimization—a process which minimizes an objective function while satisfying inequality and equality constraints. The Trajectory Optimization for Walking Robots (TOWR) framework [1] introduced a solution parametrization appropriate for handling the discontinuities inherent to legged locomotion, which ordinarily create problems for gradient descent, without imposing overly rigid assumptions. Adapting this approach for online use carries several additional challenges, however, such as the need to minimize its required computation time and the desire to employ it in a *receding-horizon* manner without compromising its expressiveness.

One increasingly popular strategy for online planning is to produce high-quality initial guesses by leveraging offline experience, so that only minor computation is needed to adapt to a scenario seen at runtime. Many earlier examples of this approach focused on manipulators [2], leading to later studies on more dynamic systems [3], [4], [5]. Recent works have considered more complex variants that may be needed when the relevant solution space is multimodal [6], [7]. For the most part, however, these works impose strong assumptions about the beginning and end of the motion plan being produced. For generalized, sustained behavior by dynamic quadrupeds, a formulation is needed that can handle their multimodality—and to do so without being required to routinely revisit a small and restrictive class of checkpoint states.

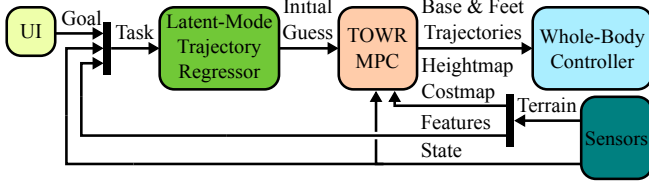


Fig. 2. Overview of the proposed onboard system used for fast replanning and execution of temporally-extended, terrain-aware, dynamic trajectories.

## B. Contributions and Outline

Motivated by current literature discussed in Sec. II, we aim to enable *online replanning* of dynamic motions that can be *sustained indefinitely* while anticipating upcoming terrain features by employing *diverse modes of behavior*. This paper presents a substantial extension and integration of our previous works [5], [7] in pursuit of this goal. Its main contributions are:

- Implementation of an *asynchronous, segmentation-based* trajectory optimization formulation for legged robots based on the single rigid body dynamic (SRBD) model to enable *receding-horizon* planning (Sec. III-A).
- Incorporation of two alternate methods for initializing the nonlinear programming problem (NLP): a heuristic that makes efficient re-use of the previous plan (Sec. III-C), and the data-driven method LMTR that can generalize the locomotion gait in anticipation of terrain (Sec. III-B).
- The evaluation of the proposed system in simulation and on hardware using the real ANYmal C quadruped (Sec. IV).

Figure 2 provides an overview of our replanning framework, which is detailed throughout Sec. III. Additional features include future state prediction based on the actual robot state, update of the predicted state within the solver at each solve iteration, a contact regain behavior and the use of real-time perception from an elevation map to select footholds (Sec. III-D–III-F). Following discussion of the findings in Sec. IV, we summarize our conclusions and identify future directions in Sec. V.

## II. RELATED WORK

### A. Motion Planning via Model Predictive Control

Multiple authors have focused on generation of motions for legged robots. The most successful approaches use trajectory optimization to solve the legged locomotion problem. They may include the kinematics and dynamics of the limbs [8], [9], [10] or they may use simplified models like the centroidal dynamics [11] or the single rigid body dynamics (SRBD) [1]; they can also adopt different strategies for contact scheduling such as completely predefining the timings, only predefining the sequence [12] or fully discovering both [13], [14]. However, only few of these methods are suitable for online motion generation.

To be applicable online, approaches usually rely on simplified dynamics criteria like the Zero-Momentum-Point [15], single rigid body with fixed orientation [16], [17] and

decouple the foothold selection to reduce the computation time [18], [19].

More sophisticated nonlinear MPC approaches use the whole-body model of the robot but have too short optimization horizons to be able to adequately approach large obstacles [20]. Other implementations based on the full robot model employ differential dynamic programming [10] with an efficient feasibility-driven formulation that promises to enable high-frequency receding-horizon control of floating-base platforms. Another family of approaches is mixed-integer programming (MIP) [21], [22] which, however, does not scale well with a large number of decision variables. Other authors successfully use the SRBD which represents a compromise between model accuracy and complexity that allows online replanning [23]. While SRBD assumes the inertia to be invariant, the changing property can be accounted and compensated for without increasing the complexity of the model [19].

### B. Data-Driven Locomotion

Model-based approaches to motion generation may be limited by the computation times or discovery effort needed to construct complex or subtle behavior online. This has lead to data-driven approaches that *amortize* this cost and effort by conducting it offline, so that appropriate actions may be quickly inferred during deployment. Bledt et al. [24] observed correlations between tasks and optimized behaviors to extract heuristics that were formulated as costs to “regularize” the solver used for a predictive controller [17].

Alternately, machine learning approaches such as deep reinforcement learning (RL) reduce the role of humans in curating the amortized knowledge about favorable actions. A hierarchy of two RL policies was used in [25]—one for planning footholds and phase durations, and one for base and end-effector motion. To gradually build up the complexity of the learned behaviors, [26] used curriculum-driven learning for bipedal stepping-stone motions, which were executed in simulation. Unfortunately, these methods become too sample-inefficient when dealing with a large state-space and typically require careful design and tuning of the reward function.

For producing longer-horizon actions, the work [3] “warm-started” a nonlinear model-predictive controller using a learned model derived from a sampling-based approach. Similar use of a function approximator allowed a real quadruped to quickly plan several-steps-long dynamic trajectories which it successfully executed to ascend terrain [5]. Alternately, Guided Trajectory Learning uses consensus optimization to learn a function approximator that only reconstructs feasible motions [27]. In [4], a “memory of motion” of dynamic, collision-free motions has been generated using the *HPP Loco3D* planner [28] for the *Crocodyl* solver [10]. To account for distinct modes of possible behavior, multiple regressors can be trained, as in computer graphics applications capturing the highly dynamic motions of dogs and humans [29], [30]; For robot locomotion, [31] used optimal control demonstrations for imitation learning (IL),

which showed improved sample efficiency over RL. To avoid having to use multiple networks and discrete categories of actions, latent variables can be utilized to represent modal variation in a continuous manner [6], [7].

### III. FRAMEWORK

In this section we present an overview of our online dynamic locomotion replanning and execution approach, shown in Fig. 2. The user, or a high-level planning algorithm, specifies the goal as a SE(2) base state within a couple meters of the robot. The robot's state estimate is provided along with an environment model, typically a dense 2.5D elevation map produced using depth cameras or LIDAR. When the Heuristic is used, the previous solution is employed to initialize the optimization variables of the new NLP. Instead, when the learned initialization is used, the state, the goal and environment features compose the *task* passed into the trajectory regressor (Sec. III-B.3), which quickly returns an initial guess that progresses toward the goal and terminates at a mid-motion state that is suited for approaching the remaining obstacles. Combined with the estimated state and environment heightmap, TOWR-MPC (Sec. III-A) quickly adapts the trajectory segment provided by the regressor. Finally, the whole-body tracking controller follows the resultant base and feet trajectories until the next one becomes available one replanning cycle later.

#### A. Optimization Formulation

To be able to replan dynamically in mid-locomotion, we have chosen to formulate problems where time horizons of foot motions are *asynchronous* to the base horizon as shown in Fig. 3. It enables segmentation of long-horizon motions, training of the regressor and the initialization of the solver for model predictive control. This is accomplished while maintaining the benefits of the phase-based parametrization of the original approach.

The planning problem is defined by nodes whose values  $\mathcal{N}$  and derivatives  $\dot{\mathcal{N}}$  are the optimization variables  $\mathcal{X}$ , the number of which is calculated as:

$$\begin{aligned} \#\mathcal{X} = & (3 + 3 + 3 + 3) \times n_{\text{nodes}}^{\text{base}} && \text{base-motion} \\ & + n_{\text{feet}} \times [3 \times n_{\text{st}} + (3 + 2) \times n_{\text{sw}}] && \text{foot-motion} \\ & + n_{\text{feet}} \times (3 + 3) \times n_{\text{nodes}}^{\text{force/foot/st}} \times n_{\text{st}} && \text{forces} \\ & + n_{\text{feet}} \times (n_{\text{st}} - 1 + n_{\text{sw}}). && \text{durations} \end{aligned} \quad (1)$$

Here, base linear position  $\mathbf{r}(t) \in \mathbb{R}^3$ , linear velocity  $\dot{\mathbf{r}}(t) \in \mathbb{R}^3$ , angular position  $\boldsymbol{\theta}(t) \in \mathbb{R}^3$  and angular velocity  $\boldsymbol{\omega}(t) \in \mathbb{R}^3$  values are multiplied by the number of base nodes  $n_{\text{nodes}}^{\text{base}}$  which is calculated using the durations of the horizon  $\Delta T_{\text{H}}$  and the time step of the base polynomials  $dt_{\text{base-poly}}$  as  $n_{\text{base-nodes}} = 1 + \text{round}(\Delta T_{\text{H}}/dt_{\text{base-poly}})$ .

For a quadruped, the number of legs  $n_{\text{feet}}$  is 4. The chosen number of stance  $n_{\text{st}}$  and swing  $n_{\text{sw}}$  phases for each foot  $i$  is 2 and 1, respectively. The stance phase is defined by the linear position  $\mathbf{p}_i(t) \in \mathbb{R}^3$  values, while the swing phase has both linear position  $\mathbf{p}_i(t) \in \mathbb{R}^3$  and linear velocity

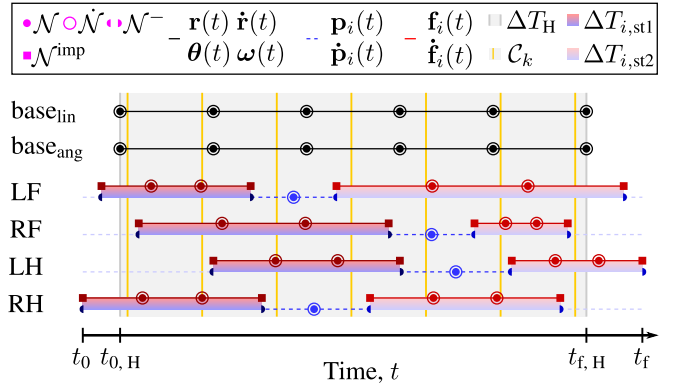


Fig. 3. Our *asynchronous, segmentation-based* problem formulation uses phase-based end-effector parametrization to enable model predictive control.

$\dot{\mathbf{p}}_i(t) \in \mathbb{R}^2$  values; though, the velocity in the z-dimension is an implicit constraint set to 0. The force values and derivatives  $\mathbf{f}(t), \dot{\mathbf{f}}(t) \in \mathbb{R}^3$  are multiplied by the number of force nodes per foot per stance  $n_{\text{nodes}}^{\text{force/foot/st}}$ , which is 3 as forces are constrained to  $\mathbf{0}$  at the start and end of each *swing* phase. For every mid-motion stance phase, this number further decreases to 2 as the forces at the start and end are  $\mathbf{0}$ . These nodes define the connecting cubic Hermite polynomials which can be sampled at any discrete time step for control purposes.

Figure 3 represents the problem formulation for the horizon  $\Delta T_{\text{H}} = 5 dt_{\text{base-poly}}$  which is defined with respect to the first and last base nodes occurring at  $t_{0,H}$  and  $t_{f,H}$ , respectively. The figure shows all the constructing nodes, implicit constraints  $\mathcal{N}^{\text{imp}}$ , polynomials, and phases  $\Delta T$ . In addition, a constraint  $\mathcal{C}_k$ , discretized at a fixed time step  $k$ , is shown; such a type of constraint can be used to enforce the dynamics equality or the range-of-motion inequality constraints. The optimization values that solely represent values for a period of time  $t$  are denoted by  $\mathcal{N}^-$ .

The problem is defined for a time duration  $T$  between time  $t_0$  of the earliest beginning of the *first* set of stances  $\Delta T_{i,\text{st1}}$  and the time  $t_f$  of latest ending of the *second* set of stances  $\Delta T_{i,\text{st2}}$ ; in between, the swing phases  $\Delta T_{i,\text{sw}}$  occur. Swing motions are not re-optimized during their execution, so if a leg is in swing at  $t_{0,H}$ , only the next stance-swing-stance are considered in the optimization.

To determine *realistic* force limits in the normal direction we use the force polytopes [32].

Geometric shapes known as *superquadrics* are used as range-of-motion constraints. They are defined as:

$$|p_{i,x}(t)/A|^a + |p_{i,y}(t)/B|^b + |p_{i,z}(t)/C|^c = 1 \quad (2)$$

They enable longer steps without violating joint limits; and facilitate faster constraint satisfaction due to the nonlinear effect of the exponents  $a, b, c$  that define the curvature of the shapes whose dimensions are specified by the scalings  $A, B, C$ . The shape is expressed in the frame of the base.

#### B. Expert Data and Behavior Cloning

Optimizing complex, long-horizon motion for a nonlinear system in real-time and *from scratch* is usually imprac-



tical because computation time typically greatly exceeds trajectory duration. Furthermore, in order to sustain motion indefinitely, the final boundary condition must account for context *beyond* the current horizon. We attempt to address these issues by using imitation learning of expert trajectories that are generated offline and processed into segments. Learning a model that generalizes across a wide range of experiences could enable online replanning to indefinitely blend between different behaviors. Our approach, which was originally presented in [7] and is summarized in Fig. 4, involves the following:

1) *Generation of Long-Horizon Trajectories*: Offline expert trajectories, which link two stationary robot states using a fixed amount of time and footsteps, are produced using TOWR with analytical costs [5]. For each expert  $e$ , we allow the solver to take some number  $N_e^* \in \mathbb{N}_0$  of iterations to achieve a level of viability specified by a combination of the objective value  $\text{cost}(\mathcal{X})$  and the infeasibility of the primal variables  $\text{inf}_{\text{pr}}(\mathcal{X})$ . “Tasks”, which here consist of initial/final state pairs and any intervening environment information, are drawn from a random distribution so that the expert knowledge accounts for a range of situations. The naive initial guess for each task—a linear interpolation between its two states—is augmented with smooth noise in order to help the optimizer discover diverse strategies and their component behaviors. This attempts to serve a similar role to that of maximum-entropy RL [33], improving the model’s generalization under perturbed states and new scenarios.

2) *Segmentation for Receding-Horizon Use*: Each long-horizon trajectory is processed into segments (as described in Sec. III-A) that may begin and end in mid-motion. In order to account for context beyond the optimization window, these segments are paired with tasks that are defined differently from those of the offline trajectories. The initial state, which is now in mid-motion, is combined with longer-horizon context instead of the final state. This context consists of a goal location that is more than one segment-length away, and any environment features between the initial state and that goal. Thus, while the final state of the segment is part of the *solution* output by the replanner, it becomes a hard constraint in the *problem* the optimizer must solve.

3) *Generalization via Latent-Mode Trajectory Regression*: Due to the expert’s rich behavior inventory, receding-horizon tasks  $\mathbf{x}$  that are similar to each other may be paired with substantially different segments  $\tau$ . This prevents directly fitting a regression model as  $f : \mathbf{x} \rightarrow \tau$ , which can be called “unimodal” since it must smoothly vary based on  $\mathbf{x}$  alone. Instead, a “mode”  $\mathbf{z}$  that represents task-independent variation can be learned in an unsupervised fashion by training a Conditional Variational Autoencoder, where  $\tau$  is the reconstructed variable,  $\mathbf{x}$  is the condition, and  $\mathbf{z}$  is the latent variable.

The decoder module of this model then serves as the regressor deployed online, with the input  $\mathbf{z}$  chosen via nearest-neighbor lookup using a weighted distance metric between a novel task and those encountered in offline experience.

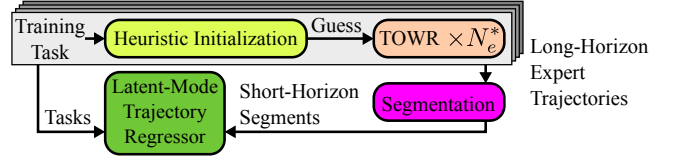


Fig. 4. An overview of the offline system used for generation of expert behaviors and the training of the latent-mode trajectory regressor, a CVAE.

### C. Heuristic Initialization

The nonlinear programming problem can be initialized by the Heuristic or by a segment reconstructed with the LMTR. When the Heuristic is used, assuming that the optimization horizon is at least as long at the time delay between two subsequent replanning cycles, we initialize the variables of the base linear and angular trajectory with the corresponding samples of the solution obtained at the previous replanning cycle. The remaining part of the new base trajectory that does not overlap with the previous solution is linearly interpolated from the last available sample up to the user-defined goal. The footholds corresponding to the first stance phase coincide with the location given by the previous plan whilst the footholds of the second stance phase are located at a nominal distance from target goal. Unlike the LMTR, the Heuristic initializes the NLP with periodic gaits which result in the two stance phases having the same duration ( $\Delta T_{i,\text{st}1} = \Delta T_{i,\text{st}2} = \Delta T_{i,\text{st}}$ ) and in all the feet having the same stance and swing durations  $\Delta T_{\text{st}}$  and  $\Delta T_{\text{sw}}$  (these value can, however, still be varied during the optimization). The duration of the swing phase and of two stance phases of each foot can be set through the duty factor  $d$  and the cycle factor  $c$  such that  $\Delta T_{\text{st}} + \Delta T_{\text{sw}} = (1.0 - c)\Delta T_H$  and  $d = \Delta T_{\text{st}}/T_c$  (where  $T_c$  is the gait cycle time  $T_c = \Delta T_{\text{st}} + \Delta T_{\text{sw}}$ ). The relative time offset  $t_{0,i}$  between the start of the base trajectory and the start of the trajectory of the  $i^{\text{th}}$  foot can be set as an independent fraction  $\alpha_i \in (0, 1)$  such that:  $t_{0,i} = \alpha_i \Delta T_H$ .

### D. State Prediction due to Optimization Delay

Most MPC algorithms target relatively high frequencies of 40 Hz to 200 Hz [17], [24]. Because our framework is concerned with solving longer-horizon dynamic motion, it operates at a much lower frequency of few Hz. As a result, it experiences delays measured in hundreds of milliseconds between measurements of the state and execution of resulting solutions. To manage this issue, the starting point of the computed trajectory corresponds to the time at which it will be executed and occurs after the computation begins. We implement a simple *predictor* to handle this by using the desired position computed by the previous plan and the current tracking error:

$$\hat{\mathbf{x}}(t_{0,k+1}) = \mathbf{x}_k^*(t_{0,k+1}) + \alpha(t)^T \bar{\mathbf{x}}(t) \quad (3)$$

$$\bar{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_k^*(t) \quad (4)$$

Where  $t$  is the current time,  $t_{0,k+1}$  is the time at start of the new plan  $k + 1$ ,  $\hat{\mathbf{x}}(t_{0,k+1})$  is the predicted state used as initial state for the next plan,  $\mathbf{x}_k^*(t)$  is the desired state from the current plan,  $\bar{\mathbf{x}}(t)$  is the current tracking error and  $\alpha$  is a scaling vector.

For the scaling, values of 0 means that the prediction relies only on the desired state while values of 1 fully take into account the tracking error and correspond to the actual state of the robot when prediction is with a short horizon.

For each foot, the scaling vector is set to 1 if it is in stance and this phase corresponds to the same one at the beginning of the next plan, or 0 otherwise. For the base, the scaling factors are tuned between those two values to allow feedback on the actual state while still being close enough to the previous plan for stable numerical behavior. This kind of simple prediction is only valid if the horizon is short with respect to the dynamics of the tracking error. To have a more accurate initial state for the optimized trajectory, the constraint on the initial state is updated with a new prediction between each iteration of the optimizer.

#### E. Terrain Perception

We use the onboard RealSense depth cameras and the GridMap library [34] to perceive the terrain. GridMap constructs a 2.5D heightmap which constrains the footholds and a costmap to aid obstacle avoidance. We use a raw version of the heightmap to match the  $z$  coordinate of the planned footholds to the actual terrain and we use a smooth version of the heightmap to set a cost on each candidate foothold. We implement a cost proportional to the slope of the terrain at the considered foothold. In this way, the gradient-descent based solver prefers to choose footholds away from edges and inclined surfaces. We also process the raw heightmap data to detect edges which become part of the task input provided to the LMTR. Both the heightmap and the edges are updated at a frequency of 6 Hz.

#### F. Whole-Body Controller and Robustness to terrain

To execute the generated plans, the whole-body controller [35] is used to track the trajectory of the base and the end-effectors at 400 Hz. The controller includes different heuristics to improve its behaviour in case of slip or other unexpected events.

To be more robust to errors in the terrain height measured by the depth cameras and to be able to make contact at the right instant, the swing trajectory has been modified. On the planner side, the formulation of splines in TOWR has been amended to allow non-zero vertical velocities for the foot motions at the transition from swing to stance. Additionally, the controller is able to modify the setpoints of the feet computed by the planner by continuing the swing trajectory downward until contact is detected. Those two features allow our controller to quickly react to delayed contacts while having smooth foot trajectories despite the low frequency of the proposed planner.

### IV. RESULTS AND DISCUSSION

The receding-horizon planner was tested with each of the trajectory initialization options, Heuristic and LMTR. The robot was tasked with computing dynamic motions toward a goal. The quality of the initial guesses and the refined trajectories were quantified by analyzing the cost

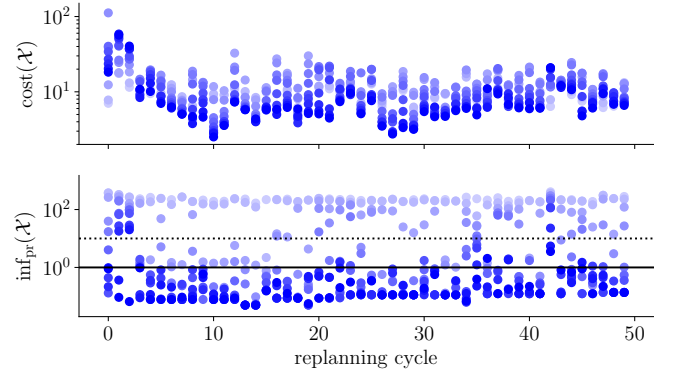


Fig. 5. The values of  $\text{cost}(\mathcal{X})$  and  $\text{inf}_{\text{pr}}(\mathcal{X})$  during refinement of initial guesses provided by LMTR while traversing the terrain shown in Fig. 7 for 50 replanning cycles. The gradient (light to dark) represents subsequent iterations limited to a maximum of 10 per cycle. Horizontal lines show preferred (solid) and acceptable (dotted) thresholds of  $\text{inf}_{\text{pr}}(\mathcal{X})$ .

and constraint violation computed at each iteration of the nonlinear solver. Execution of the replanning pipeline on the ANYmal C quadruped demonstrated its viability on hardware for most scenarios. In the most challenging case—aggressive behaviors on terrain planned by LMTR—hardware viability has not yet been achieved. Instead, we provide numerical trials in which the replanner receives the task that would be seen if the previous plan were followed nominally for the duration of one replanning cycle, and refinement is conducted using the same computation time limit as hardware trials.

#### A. LMTR Setup and Numerical Performance

An LMTR regressor was trained for both environment types. For the *flat-ground* scenario, the offline expert’s initial conditions used random offsets of the lateral position and the yaw relative to a desired line of travel.

The *terrain* scenario included randomized stairstep features spaced apart by 0.25 m to 1 m, with height changes of  $-0.16$  m to  $0.16$  m and edge orientations of  $-15^\circ$  to  $15^\circ$ . During deployment, these terrain features are extracted from the elevation map through post-processing and used as a component of the task input to the LMTR. Promising numerical results were obtained offline and are shown in Fig. 7 demonstrating mathematically-feasible motions computed at 2.5 Hz—this is part of ongoing work that ultimately targets robust deployment on hardware.

When facing an arbitrary scenario within the bounds of the experience set, the network consistently reconstructed a valid initial guess which was refined to a satisfactory level of optimality within the iteration budget available for online use, as shown in Fig. 5. Occasionally, some cycles presented conditions that were challenging to satisfy, potentially due

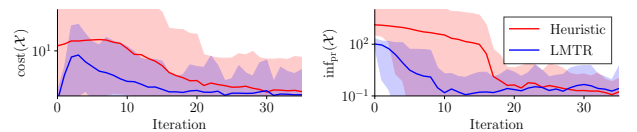


Fig. 6. Mean values of the cost and primal infeasibility at each refinement iteration of 1.2 s long initial guesses provided by the LMTR and Heuristic on flat terrain at 2 Hz for the same set of initial and final conditions.

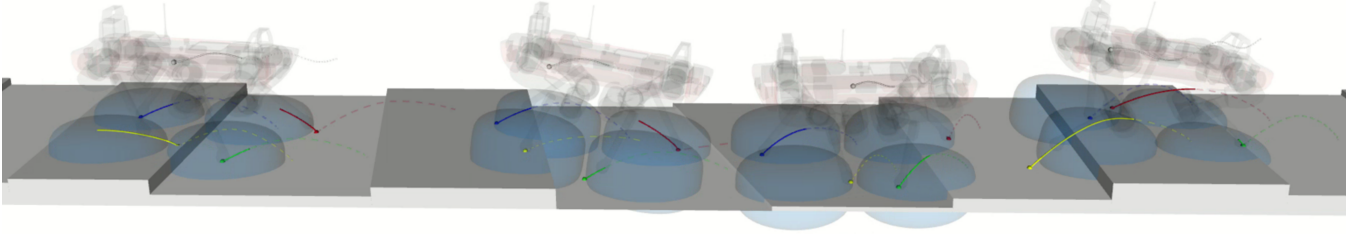


Fig. 7. Snapshots of a typical numerical trial of the LMTR which leverages similar prior experience to quickly plan very dynamic trajectories in receding-horizon fashion while being informed by real-time perception. Steps were of arbitrary depth, height and orientation and the generated plans were refined online by the solver. The dashed lines indicate planned base and feet trajectories while the solid lines emphasize the portion of the plan executed during each MPC control cycle. The blue superquadrics approximate the operational space of the robot's end-effectors by serving as range-of-motion constraints.

to gaps in the regressor's coverage. If refinement failed, the remaining portion of the previous valid trajectory was used for an additional replanning cycle, which was possible due to a greater than 2-to-1 ratio of these two durations.

Figure 6 shows the evolution of the cost and the constraint violation along the iteration of the solver for the Heuristic and the LMTR initialization. While the Heuristic could be used with any predefined gait, to have a fair comparison with respect to the difficulty of the gait, both were evaluated on the gait timing generated by the regressor. The results show that the LMTR initialization was able to converge to a solution around 2 times faster than the Heuristic.

### B. Experimental Evaluation

a) *Heuristic initialization*: The initializations provided by the Heuristic, based on user-commanded target robot base pose, enabled the continuous execution of a walk and a trotting gait during which the robot traversed obstacles of heights up to 0.2m as shown in Fig.1. The base tracking error is shown in Fig.8 for a hardware experiment where the robot walked over obstacles up to 0.13m high for over 5.5 min.

The replanning frequency for the walk gait was 1 Hz with an optimization horizon of 3s. The trot could run at 2 Hz over a replanning window of 1s. For both walk and trot gaits the Heuristic was able to reach speeds up to  $0.1 \text{ m s}^{-1}$  on flat ground and  $0.07 \text{ m s}^{-1}$  while negotiating the obstacles shown in the accompanying video. The average step length on flat ground was 0.25 m for the trot and 0.3 m for the walk.

b) *LMTR initialization*: In the experiments with the *learned* initialization, the LMTR was used to enable a 4m, 12s trot-like motion on flat ground which resulted in the

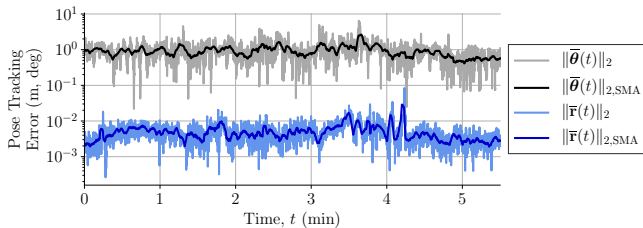


Fig. 8. Base pose tracking errors, and their simple moving averages (SMA), for a 5.5 min execution of the Heuristic on terrain. The root-mean-squared error (RMSE) values of the linear position and orientation were  $6.55 \times 10^{-3} \text{ m}$  and  $1.07^\circ$ , respectively.



Fig. 9. Adaptation by the optimizer to mild height variation of 0.02 m to 0.08 m using initial guesses provided by the LMTR trained for the *flat-ground* scenario.

base velocity in the  $x$ -direction  $\dot{r}_x(t)$  reaching  $0.7 \text{ m s}^{-1}$  and swing lengths of 0.2 m to 0.6 m with durations  $\Delta T_{\text{sw}}$  of 0.3 s to 0.5 s. Thanks to the LMTR initialization, such a plan can be adapted online by the optimizer at frequencies from 2 Hz to 3.3 Hz with planning horizons of 1.2 s to take into account the model of the terrain received from the sensors and walk over obstacles as shown in Fig. 9.

## V. CONCLUSION

This work presents a pipeline for iterative perception-aware planning of dynamic legged locomotion in a receding-horizon fashion via online trajectory optimization with a novel asynchronous, segmentation-based formulation using the single rigid body dynamics model. We present two initialization schemes. The *Heuristic* combines previous solutions with a linearly interpolated guess to accelerate the computation. The *latent-mode trajectory regressor (LMTR)*, which imitates expert data, is used to extend the planning horizon, proposing a behavior dependent on the task at hand, which may include prominent terrain features. We deploy this pipeline on a quadrupedal robot, ANYmal version C, using fully onboard sensing and computation. Hardware trials with the Heuristic demonstrate the ability to traverse terrain obstacles using walking and trotting gaits. The LMTR enabled more dynamic, trot-like locomotion but with irregular phase durations that hint at its capacity to express a diverse inventory of viable behaviors. Numerical results show how its expressiveness is suitable for adapting to challenging terrain while moving at fast pace. Future work will seek to realize these precise and highly dynamic motions via explicitly compensating for the planning model's approximations in lower-level control and by updating the LMTR based on deployed experiences.

## REFERENCES

- [1] 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*, vol. 3, no. 3, pp. 1560–1567, July 2018. 1, 2
- [2] N. Jetchev and M. Toussaint, “Fast motion planning from experience: trajectory prediction for speeding up movement generation,” *Autonomous Robots*, vol. 34, no. 1, pp. 111–127, Jan. 2013. 1
- [3] N. Mansard, A. DelPrete, M. Geisert, S. Tonneau, and O. Stasse, “Using a Memory of Motion to Efficiently Warm-Start a Nonlinear Predictive Controller,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2986–2993. 1, 2
- [4] 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,” *arXiv:2001.11751 [cs, math]*, Feb. 2020. 1, 2
- [5] O. Melon, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, “Reliable Trajectories for Dynamic Quadrupeds using Analytical Costs and Learned Initializations,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 1410–1416. 1, 2, 4
- [6] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, “Learning Latent Plans from Play,” in *Conference on Robot Learning*, May 2020, pp. 1113–1132. 1, 3
- [7] D. Surovik, O. Melon, M. Geisert, M. Fallon, and I. Havoutis, “Learning an Expert Skill-Space for Replanning Dynamic Quadruped Locomotion over Obstacles,” in *2020 Conference on Robot Learning (CoRL)*, Nov. 2020. 1, 2, 3, 4
- [8] H. Dai, A. Valenzuela, and R. Tedrake, “Whole-body motion planning with centroidal dynamics and full kinematics,” in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 295–302. 2
- [9] J. Carpentier and N. Mansard, “Multi-contact Locomotion of Legged Robots,” *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1441–1460, Dec. 2018. 2
- [10] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, “Crocodyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 2536–2542. 2
- [11] D. Orin, A. Goswami, and S. Lee, “Centroidal dynamics of a humanoid robot,” *Autonomous Robots*, vol. 35, pp. 161–176, 2013. 2
- [12] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, “On time optimization of centroidal momentum dynamics,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 5776–5782. 2
- [13] M. Posa, C. Cantu, and R. Tedrake, “A direct method for trajectory optimization of rigid bodies through contact,” *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014. 2
- [14] I. Mordatch, E. Todorov, and Z. Popović, “Discovery of complex behaviors through contact-invariant optimization,” *ACM Trans. Graph.*, vol. 31, no. 4, July 2012. 2
- [15] M. Vukobratovic and B. Borovac, “Zero-moment point thirty five years of its life,” *International Journal of Humanoid Robotics*, vol. 01, no. 01, pp. 157–173, 2004. 2
- [16] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, “Dynamic locomotion in the mit cheetah 3 through convex model-predictive control,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9. 2
- [17] G. Bledt and S. Kim, “Implementing Regularized Predictive Control for Simultaneous Real-Time Footstep and Ground Reaction Force Optimization,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 6316–6323. 2, 4
- [18] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, “Perceptive Locomotion in Rough Terrain – Online Foothold Optimization,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5370–5376, Oct. 2020. 2
- [19] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, “MPC-based Controller with Terrain Insight for Dynamic Legged Locomotion,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 2436–2442. 2
- [20] F. Farshidian, E. Jelavic, A. Satapathy, M. Gifftthaler, and J. Buchli, “Real-time motion planning of legged robots: A model predictive control approach,” in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017, pp. 577–584. 2
- [21] T. Apgar, P. Clary, K. Green, J. C. Grieco, A. Fern, and J. Hurst, “Fast Online Trajectory Optimization for the Bipedal Robot Cassie,” in *Robotics: Science and Systems XIV*, vol. 14, June 2018. 2
- [22] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernandez-Lpez, and C. Semini, “Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, 2018. 2
- [23] O. Cebe, C. Tiseo, G. Xin, H.-c. Lin, J. Smith, and M. Mistry, “Online Dynamic Trajectory Optimization and Control for a Quadruped Robot,” *arXiv:2008.12687 [cs]*, Sept. 2020. 2
- [24] G. Bledt and S. Kim, “Extracting Legged Locomotion Heuristics with Regularized Predictive Control,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 406–412. 2, 4
- [25] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, “DeepGait: Planning and Control of Quadrupedal Gaits Using Deep Reinforcement Learning,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, Apr. 2020. 2
- [26] Z. Xie, H. Y. Ling, N. H. Kim, and M. van de Panne, “ALL-STEPPS: Curriculum-driven Learning of Stepping Stone Skills,” *arXiv:2005.04323 [cs]*, Aug. 2020. 2
- [27] A. Duburcq, Y. Chevalere, N. Bredeche, and G. Boérís, “Online Trajectory Planning Through Combined Trajectory Optimization and Function Approximation: Application to the Exoskeleton Atalante,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 3756–3762. 2
- [28] 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, June 2018. 2
- [29] H. Zhang, S. Starke, T. Komura, and J. Saito, “Mode-adaptive neural networks for quadruped motion control,” *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–11, Aug. 2018. 2
- [30] S. Starke, Y. Zhao, T. Komura, and K. Zaman, “Local motion phases for learning multi-contact character movements,” *ACM Transactions on Graphics*, vol. 39, no. 4, pp. 54:54:1–54:54:13, July 2020. 2
- [31] J. Carius, F. Farshidian, and M. Hutter, “MPC-Net: A First Principles Guided Policy Search,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2897–2904, Apr. 2020. 2
- [32] A. Bratta, R. Orsolino, M. Focchi, V. Barasuol, G. G. Muscolo, and C. Semini, “On the hardware feasibility of nonlinear trajectory optimization for legged locomotion based on a simplified dynamics,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1417–1423. 3
- [33] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmssan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 1861–1870. 4
- [34] P. Fankhauser and M. Hutter, “A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation,” in *Robot Operating System (ROS): The Complete Reference (Volume 1)*, ser. Studies in Computational Intelligence, A. Koubaa, Ed. Cham: Springer International Publishing, 2016, pp. 99–120. 5
- [35] C. Dario Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, “Perception-less terrain adaptation through whole body control and hierarchical optimization,” in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. Cancun, Mexico: IEEE, Nov. 2016, pp. 558–564. 5