Reliable Trajectories for Dynamic Quadrupeds using Analytical Costs and Learned Initializations

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

***Abstract*— Dynamic traversal of uneven terrain is a major objective in the field of legged robotics. The most recent model predictive control approaches for these systems can generate robust dynamic motion of short duration; however, planning over a longer time horizon may be necessary when navigating complex terrain. A recently-developed framework, Trajectory Optimization for Walking Robots (TOWR), computes such plans but does not guarantee their reliability on real platforms, under uncertainty and perturbations. We extend TOWR with analytical costs to generate trajectories that a state-of-the-art whole-body tracking controller can successfully execute. To reduce online computation time, we implement a learning-based scheme for initialization of the nonlinear program based on offline experience. The execution of trajectories as long as 16 footsteps and 5.5 s over different terrains by a real quadruped demonstrates the effectiveness of the approach on hardware. This work builds toward an online system which can efficiently and robustly replan dynamic trajectories.**

I. Introduction

Legged robots are soon expected to be used in a range of application domains where advanced mobility is required. The key benefit of such machines is their flexibility in operating on terrain designed for humans: confined spaces and lacking regular structure. This goal implies increased complexity as legged robots are articulated systems with high-dimensional kinematics and dynamics that continually change their contact with the environment. Other hurdles arise due to real-world sensing, actuation limits, state estimation and perturbations. As a result, legged robots need a flexible motion planning approach to efficiently and robustly perform their tasks.

The generation of dynamic motions for these platforms is an open research problem with recent advances focusing on optimization-based approaches. Longer trajectories can produce better system performance, but are more difficult to compute, ruling out the online use of global methods. Gradient descent is less demanding but can get stuck in poor local minima due to the strong non-convexity of the problem. Thus, local optimization is appealing only if the constraints, costs, and the initial guess of the nonlinear program can all be specified effectively.

As shown in Fig. 1 and 2, we combine a learning-based data-driven initialization with an enhanced formulation of the optimization problem of dynamic motion generation, which we demonstrate for a quadrupedal robot walking on uneven terrain. The objective is to produce robust plans over suitably long time horizons with minimal online computation effort.

This work was supported by the UKRI/EPSRC 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, mathieu, dsurovik, ioannis, mfallon}@robots.ox.ac.uk.

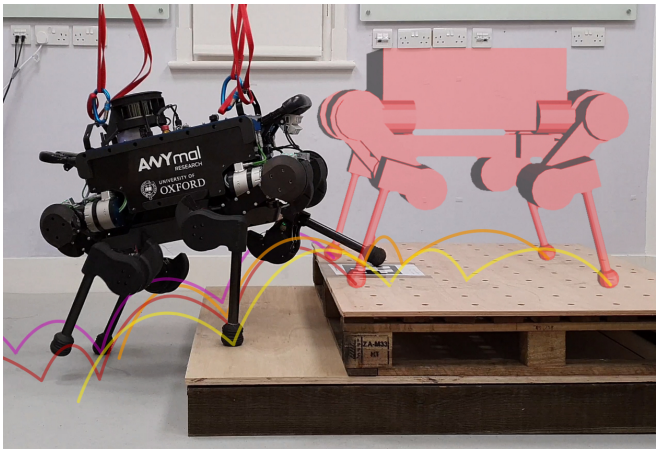

Fig. 1: The ANYbotics ANYmal executing a dynamic stair climb with the proposed approach, as shown in the accompanying video https://youtu.be/LKFDB_BOhl0.

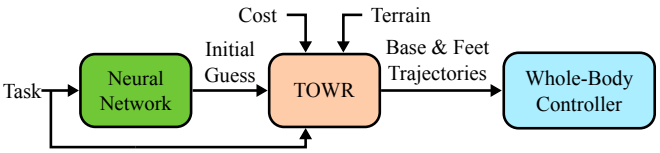

Fig. 2: Overview of the system used for learning, trajectory optimization and execution on the ANYmal quadruped.

A. Contributions

- Extension of the TOWR legged robot motion optimization framework [1] with analytical costs to create dynamic trajectories that can be reliably tracked by a controller.
- A data-driven method to produce good initial guesses which speeds up optimization convergence. The method can compute 16-footstep trajectories in less than 1 second while avoiding poor local minima.
- Significant evaluation in dynamic simulation to demonstrate how the combination of these changes make execution much more reliable than the baseline method.
- Experiments in which the real ANYmal robot trots or walks across flat or uneven terrain while using only onboard sensors for state estimation, verifying the validity of our approach and its suitability for real hardware.

B. Overview

This paper begins with an overview of related state-of-the-art approaches in Sec. II. Descriptions of the baseline trajectory optimizer and tracking controller are then provided in Sec. III, along with new modifications that allow these tools to be effectively combined. Sec. IV describes our method for training effective initialization using offline prior experiences, with some analysis of its benefit. Sec. V describes experimentation on different terrains in both simulation and hardware. The paper concludes with final remarks in Sec. VI.

II. Related Work

1) Trajectory Optimization for Legged Robots: Trajectory optimization (TO) approaches have been used for short to medium scale motion planning in recent research [2], [3]. They usually employ direct methods to transcribe a continuous, infinite-dimensional problem as a discrete, parametrized nonlinear programming problem (NLP) [4].

Legged locomotion optimization problems are subject to discontinuities due to contact transitions of the end-effectors. Two approaches have been used to recast it as a continuous problem. The first reformulates contact as a smooth rather than a discontinuous state [5], [6], [7]. This transforms the discontinuous and minima-prone problem into a continuous problem that can be solved with homotopic methods.

The second approach defines the problem as a succession of phases separated by the contact state transition of each end-effector. In most cases, the timing and position of the contacts are computed externally and the optimization only solves for the centroidal motion of robot [8], [9], [10], [11]. In [1], the contact state of each leg is considered separately, which theoretically allows it to generate different gaits when optimizing the duration of each phase. Moreover, the positions of the footsteps are included in the set of optimized variables. Unlike the previously mentioned approaches, this formulation allows a highly non-convex shape of the problem to be solved by recasting it as a feasibility problem.

2) Data-driven Initialization: Data-driven trajectory initialization schemes have been applied in the domain of manipulation to speed up the computation of smooth paths when reaching past obstacles [12], [13], [14]. Related methods produce multiple initializations in different basins of attraction so as to identify distinct ways of approaching the object to be grasped [15], [16]. Dynamic constraints can additionally be met for tasks such as quickly reaching to catch a thrown object [17] or rejecting large disturbances on underactuated aerial vehicles [18]. For legged locomotion on terrain, a related idea was used to plan individual feasible footsteps, which were then combined by another process into a full motion plan [19].

Most of these efforts map tasks to initial solutions, while others map to segments of solutions [19], or to additional constraints that convexify the problem [15]. Nearly all consider nearest-neighbor lookup and regression on an experience library as a mapping method, while many also consider other function approximators. These include Support Vector Machines [15], [17], Gaussian Process Regression [16], [17], or Artificial Neural Networks (ANNs) [15], [18]. Another research strand is the use of feature-spaces and dimensionality reduction techniques [19], [13], [14], [15], [16]. In each case, the goal is to sufficiently represent past experiences in a manner which can be related to future decisions.

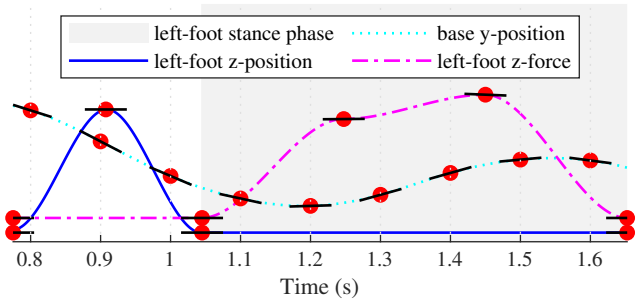

Fig. 3: An illustrative sample of the trajectories computed by the numerical optimizer, for a quadrupedal trot. Red dots and black lines represent values at optimization nodes, interpolated using cubic splines.

III. Motion Generation

In this section we present our motion planning and execution approach. We first review the TOWR trajectory optimization package, which uses a simplified dynamics model of the robot to plan motions for its legs and center of mass (CoM) between initial and final configurations. We present improvements and adaptions that help create trajectories which are more suitable for the real robot. We then overview the tracking whole-body controller that we use to compute the joint torques necessary to execute these generated motions and provide an illustrative evaluation.

A. Trajectory Optimization for Walking Robots (TOWR)

The paper extends the work of Winkler *et al.* and their open-source library TOWR[1], [20]. TOWR is capable of producing highly-dynamic trajectories for a range of walking robots by formulating locomotion as a nonlinear program (NLP). The approach considers single rigid body dynamics (SRBD) of the base, which is assumed to contain all of the system's mass, along with the paths and contact forces of the feet.

The problem is discretized into a numerically-solvable formulation using a collocation-based transcription method. In this case, trajectories are constructed as splines of N cubic Hermite polynomials, where each polynomial is fully defined by the values and the derivative at its start and end nodes. Fig. 3 shows these splines for some of the problem variables. The base trajectory is discretized using a fixed timestep dt (0.1 s), while the feet trajectories and contact forces are discretized with a fixed number of polynomials per phase (2 for the swing trajectory, 3 for the stance forces). Therefore, the number of variables for the feet and contact forces varies with the number of steps, while the number of base-related variables depends on the time horizon. The formulation of

the locomotion problem implicitly constrains some of the variables:

- Forces are null during swing phases.
- Derivatives of the forces and feet positions are zero at the transitions between swing and stance phases.
- A swing node is the highest point of a swing trajectory and its z-dimension derivative is set to 0.
- Feet are fixed in place during a stance phase.

Consequently, the number of optimization variables is then $N = 12T/dt + 20S + 120$, where T is the time horizon, dt the time discretization interval for the base motion and S is the number of steps. Moreover, if the timing of steps is optimized, the duration of the stance and swing phases of each step are included, adding $2S$ variables.

The problem is then transferred to an interior-point method solver (Ipopt [21]) which searches for a solution respecting the following explicit constraints:

- Dynamics of the system (modeled as one rigid body).
- Kinematic limits (the positions of feet, relative to the base, constrained within a box).
- Maximum contact forces and friction pyramids.
- Feet at terrain height during the stance phase.

Since the original TOWR formulation poses only a feasibility problem, the NLP solver generally converges quickly (in less than 40 iterations). While the resulting motion plans are promising, realizing them on hardware is difficult. Oscillation and strong body rolling motions are common while footstep placement and leg terrain clearance are not considered. In this work, we tackle these issues as follows.

1) Smoothing trajectories using costs: To generate more conservative trajectories for motions such as the one shown in Fig. 1, we extended TOWR with analytically-derived costs. A set of i costs $\mathrm{J}_i(t)$ are scaled by weights ω_i, to give a total cost of $\mathrm{J}(t) = \sum \omega_i \mathrm{J}_i(t)$. Integral costs were added to minimize linear velocity in the z-axis and angular velocities of the base, to penalize the magnitude of the ground-reaction forces and their derivatives, and to maintain the desired magnitude of the normal force. The generic cost was

$$\mathrm{J}_i(t) = \int_0^T \left(\left(x_i(t) - x_i^{ref}(t) \right)^2 + \omega_{i,d} \dot{x}_i(t)^2 \right) dt, \quad (1)$$

where $x_i(t)$ is the optimized polynomial and $\omega_{i,d}$ is the weight on its derivative. This penalizes short, but potentially large, deviations from the reference value $x_i^{ref}(t)$. These costs and their Jacobians were computed analytically using the parameters of the polynomials and their relations to the values at each node of the trajectories.

2) Locomotion on uneven terrain: We aim to generate dynamic motions to traverse slopes, steps and stairs. The original TOWR implementation does not take into account swing collisions between spline nodes. To resolve this, we add kinematic constraints to the base of the robot to enforce that it remains a certain distance above the feet; this eliminates the possibility of collision between the base and the ground. To prevent the solver from creating trajectories that pass through or collide with step edges, we added a cost to discourage the robot from selecting footsteps close to these step edges. The selected cost is a differentiable Gaussian function $\sum e^{x_s^2/2\sigma^2}$ where x_s represents the perpendicular distance from each footstep to the edge of the stair. This only affects the footsteps which are close to the edge, while having a negligible effect on the remaining ones. To ensure that collisions with terrain are avoided, both a constraint and a cost are applied to the height of the swinging feet. The constraint ensures that the swing node is a certain distance above each of the adjacent stance nodes while the cost minimizes the swing height to prevent large leg motions that would create angular momentum on the real system.

B. Whole-Body Controller

Once the trajectory has been generated, we use the whole-body controller of Bellicoso *et al.* [22] to track the trajectory of the base and the end-effectors at $400\,\mathrm{Hz}$. The controller contains a state machine which adapts gains in response to slipping and other unexpected contact events.

C. Validation in Simulation

In Fig. 4, we demonstrate the need for more sophisticated planning for a trajectory up and over a step, both with and without the proposed costs. Fig. 4(a) and 4(b) show an overall decrease in peak x- and y-direction forces. Fig. 4(c) shows an increase in the z-direction force during the third second of the trajectory resulting from a corresponding decrease in the x-direction force. Fig. 4(d) shows the inertial stabilization of the base—a significant reduction in the roll and pitch angular velocity of the base can be observed.

Fig. 4(e) and 4(f) show the norm of the tracking errors of the base during the execution of the trajectories by the whole-body controller in the Gazebo physics simulator. Initially, the controller manages to track both trajectories well; after 2 s, the controller can no longer adequately follow the trajectory without the proposed costs, as indicated by the increase in orientation error. At 3.2 s, the robot's front foot collides with the step and the robot falls. Meanwhile, the controller tracking the trajectory generated with the addition of the proposed costs successfully completes the execution.

However, the use of costs and constraints makes the solve time of the optimization problem longer, primarily by increasing the number of iterations; we use machine learning to provide an efficient optimization seed which offsets the extra computation time shown in Table I.

IV. Learning Initializations

Optimization frameworks such as TOWR use solvers which require a guess of the solution $\mathbf{y}$. It can be generated automatically by some map

$$A : \mathbf{x} \rightarrow \mathbf{y}_0 \quad (2)$$

that acts upon the task $\mathbf{x}$, e.g., the pair of initial and desired robot states. Here, primal variables of the interior-point method are initialized with a guess which substantially affects not only the rate of convergence but also the quality of $\mathbf{y}$. As more costs and constraints are used, more local minima arise, and the initial guess becomes even more influential.

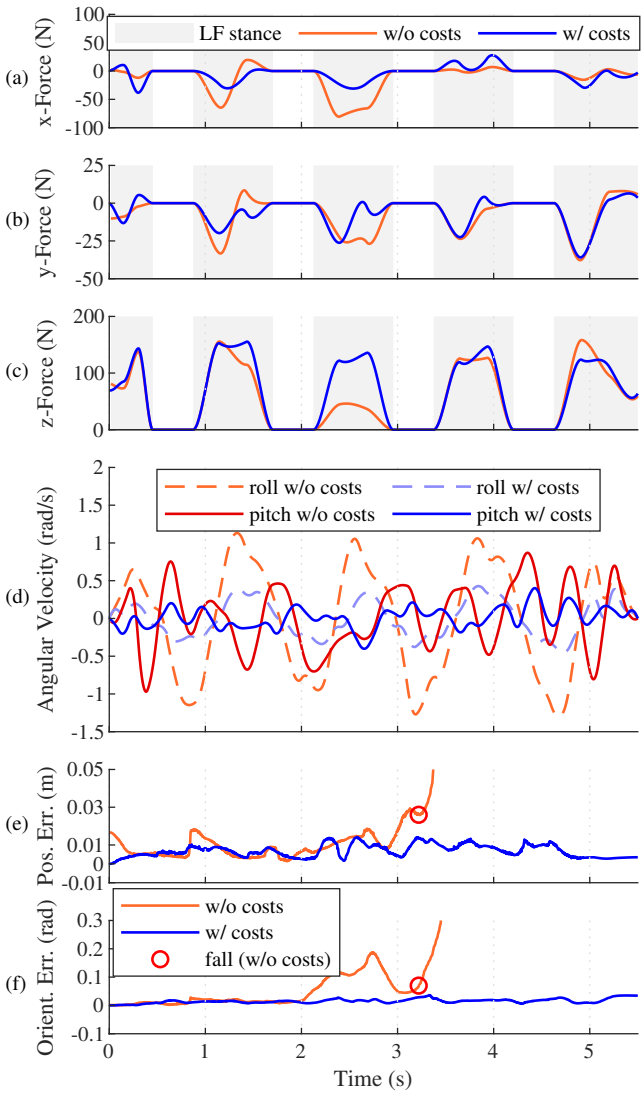

Fig. 4: Simulated climbing a single step, similar to Fig. 7: by adding costs to the optimization problem we can constrain foot forces and base angular velocities to successfully climb a step (Sec. V-.2) in simulation.

The conventional guess generator for TOWR, termed *Heuristic*, linearly interpolates a path for the floating base between the start and goal locations from $\mathbf{x}$. Footsteps are evenly-spaced and transitions between them are evenly-timed according to the selected gait, with contact forces equally distributed to counteract the robot's weight.

The objective of this section is to produce a data-driven initializer, *LearnedInit*, to replace the heuristic such that far less optimization effort (iterations) is required, as sketched in Fig. 5, while furthermore avoiding poor local minima.

A. Methodology

Denoting the heuristic as A_h and an iteration of the optimizer as another map T, optimization can be expressed as $\mathbf{y}_{h,N}(\mathbf{x}) = T^N A_h \mathbf{x}$, with iteration count N selected based upon convergence or available computation time. The

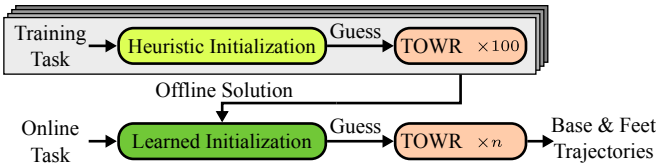

Fig. 5: Learning from previous fully-converged outcomes allows the optimizer to be initialized close to a good solution.

improved initializer is a function approximator A_θ trained to mimic the behavior of the optimization process $T^N A_h$. For a given task space $\mathcal{X}$, its optimal parameters are then defined by

$$\theta = \underset{\theta'}{\operatorname{argmin}} \sum_{\mathbf{x}\in\mathcal{X}} \| A_{\theta'}\mathbf{x} - \mathbf{y}_{h,N}(\mathbf{x}) \|_W \tag{3}$$

with a positive-definite weight matrix W.

In this work A_θ is a fully-connected neural network with 2 to 3 hidden layers, where θ refers to the connection weights. These are determined through supervised learning on a dataset $D = \{\mathbf{x}_i, \mathbf{y}_{h,N}(\mathbf{x}_i)\} = (X, Y)$ with samples $\mathbf{x}_i \in \mathcal{X}$. At present, a given θ is learned for a specific environment; however, this serves as a first step toward contextual planning with $\mathbf{x}$ augmented by local environmental features. The form of $\mathbf{y}$, whose length depends on the total duration and footstep count, is also kept constant.

Given the significant risk of converging to poor local minima, the dataset D is not guaranteed to imply a well-behaved map from $\mathcal{X}$ to the solution space $\mathcal{Y}$. Two additional steps are thus taken to ensure the tractability of the learning problem (3) and the quality of its result.

First, D is filtered based on a threshold of solution cost g_{max} to exclude poor solutions from training:

$$D_{good} = \{\mathbf{x}_i, \mathbf{y}_i \mid g(\mathbf{y}_i) < g_{max}\} \tag{4}$$

Second, since even D_{good} is unlikely to contain only globally optimal solutions, the average performance and uniformity of the learned initialization can potentially be increased by repeating the process of Fig. 5 with θ retrained on optimization outcomes resulting from its previous value. This cycle of moderate exploration and filtering thus lends an aspect of reinforcement learning to the scheme, with particular similarity to the alternation between local optimization and global supervised learning of control laws in Guided Policy Search [23].

The learning method is summarized in Algorithm 1. Notably, $N_\theta < N_h$ can be used due to the faster convergence observed when using learned initialization.

B. Setup

For each of the test environments, which will be fully detailed in Sec. V, a set of about 2000 tasks were sampled from the task space. This space expresses variation of the initial base location and yaw angle. The distribution $g(Y)$ of costs resulting from optimization with *Heuristic* and $N_h = 100$ generally exhibited a long tail of outliers, as seen in Fig. 6, that correspond to poor local minima. The filtering threshold g_{max} was set at the start of this tail, reducing the

Algorithm 1: TRAININITIALIZER

Input: A set of sampled tasks X
Output: Learned initializer parameters θ
$Y \leftarrow \text{TOWR}(A_h(X); N_h)$
for *loop count* **do**
  $D_{good} \leftarrow \text{FILTER}(X, Y; g_{max})$
  $\theta \leftarrow \text{SUPERVISEDLEARNING}(D_{good})$
  $Y \leftarrow \text{TOWR}(A_\theta(X); N_\theta)$
return θ

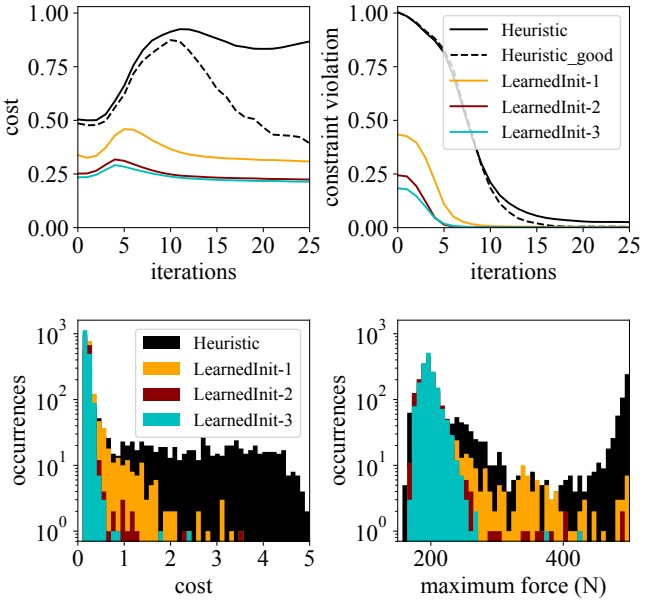

Fig. 6: Optimization performance under different initialization schemes on a large task set. With *Heuristic*, many sampled tasks diverge, producing high costs and forces. Each cycle of training for *LearnedInit* improves the convergence rate, final cost, and maximal force, while eliminating outliers.

initial training set D_{good} to about 300-500 sampled tasks, and a two-layer network was fit to this small dataset. Subsequent learning cycles used $N_\theta = 25$ due to fast convergence, as well as a 3-layer network due to the larger number of samples passing through the filter.

The loss-weighting matrix W consistently weighted all optimization variables within a given category. An initial check was conducted in which some categories were initialized by A_θ and others by A_h. This revealed that the base linear positions, stepping phase durations, and contact forces were the most crucial values to improve through learned initialization. The implication is that it is easier for the optimizer to move along some dimensions of the solution space than others, and that the initializer should prioritize the accuracy of the strongest nonlinear influences. Ultimately, the best performance was obtained using coarsely tuned weights that included all categories.

C. Analysis

This section provides a detailed look at the effect of Algorithm 1 upon the optimization process. As similar trends were observed for all the environments tested, their detailed description is deferred to Sec. V. The results here reflect the *Single Pallet* problem of ascending a ledge. For this problem, the starting base position ranges from between -0.5 m to -1.5 m back from the step and ± 0.75 m laterally from the goal position, with yaw variation in the range $\pm 30°$.

Figure 6 shows the benefits of *LearnedInit* in terms of convergence rate, final cost, and the maximum force experienced by the robot. *Heuristic* initialization often results in poor local minima, and sometimes causes divergence of the optimization process as indicated by high costs and violations of the maximum force constraint (truncated from the plot). Due to the use of filtering, *LearnedInit* exhibits better and more consistent performance, succeeding on most tasks that were failed by *Heuristic*. Retraining *LearnedInit* after optimizing its original output set increases these benefits and eliminates nearly all outliers.

V. EXPERIMENTAL EVALUATION

The proposed approach was evaluated using 3 different terrains of increasing difficulty. As discussed in Sec. III-A, the number of optimization variables depends on the number of steps and the time horizon of the trajectory. While the number of steps was kept constant (16 steps for dynamic walking, 14 for trotting), the time horizon was changed on a per-environment basis to make dynamic trajectories feasible. A different initializer was trained for each gait and environment pair, corresponding to a fixed number of optimization variables.

1) Flat ground: The generated trajectories were tested on flat ground for distances of up to 1.5 m with strides of up to 50 cm. To have highly dynamic motions, the time horizon has been set to 3.5 s which resulted in a velocity of about 0.6 m/s during the middle stage of the trajectory. The first iteration of the learning phase shown in Fig. 5, the heuristic initialization, used to generate the first set of data, based the contact sequence on a trotting gait. For this setup, the number of optimization variables was 848.

2) Single Pallet: The Single Pallet is a standard 1.2×1 m industrial pallet with a sheet of plywood on top (see Fig. 7), whose total height was 16.5 cm. For this experiment, the forward motion was about the same as for the Flat Ground, while the time horizon of the trajectory was increased to 5.5 s. The initial orientation of the base was restricted such that the pallet stays within the field of view of the robot's camera (to detect the pallet position). In this scenario, there were 952 optimization variables.

3) Double Pallet: This two-step environment is shown in Fig. 1. The steps were 14.5 cm and 16.5 cm high. The horizontal distance between steps was 40 cm. The optimization parameters were kept the same as for the Single Pallet; again being 952.

A. Test in dynamic simulation

The tracking errors of the desired base pose for the Single Pallet are shown in Fig. 4(e) and 4(f). To evaluate the performance of the learning-based approach, solutions from three initialization methods were tested on the Double Pallet.

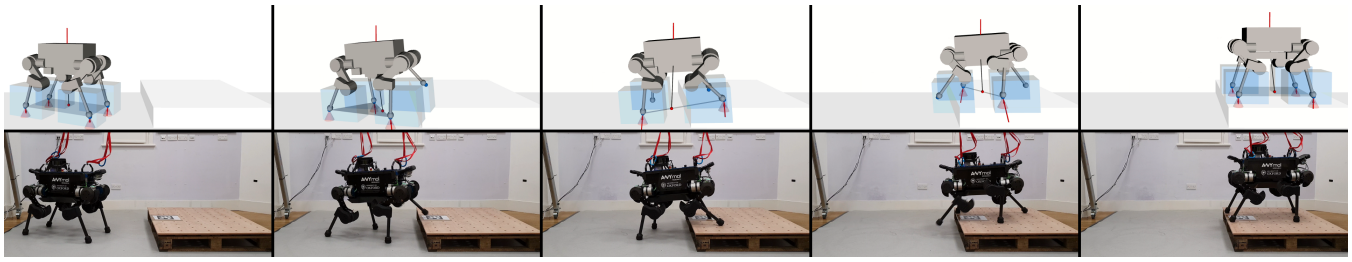

Fig. 7: The comparison of a TOWR-generated, dynamic trajectory (top) with the experimental evaluation (bottom) for a 5.5 s climbing of a pallet with a turn.

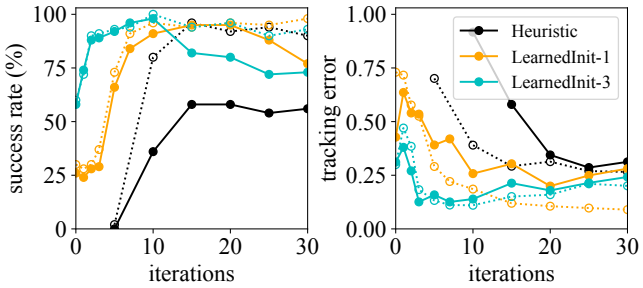

Fig. 8: Performance of trajectories climbing a single pallet under whole-body control in simulation (100 tasks per data point). *LearnedInit* produces solutions that are easier to track, reaching high success rates after minimal online refinement. Fixing phase durations at initialized values (dotted lines) improves optimization stability.

While varying the maximum iteration count, a sample set of 100 tasks was optimized for each. Trials that did not reach the goal state were marked as failures. Tracking accuracy was measured for successful runs to indicate how well-suited the planned trajectories were to the closed-loop system.

Fig. 8 demonstrates that the outputs of *LearnedInit* are executable more than half the time without any further optimization, and the success rate approaches 95% within about 5 iterations of re-optimization. Similar results were achieved for all environments. However, a higher number of iterations caused the optimization to unexpectedly diverge, resulting in a drop in success rates. Shown by the dotted lines in Fig. 8, success rates remained high when a separate set of optimized trajectories, with phase durations fixed at their initialized values, were executed. The observed behavior was caused by the extreme nonlinearity of the problem.

B. Test on the real platform

Table I summarizes the computation times obtained on the onboard locomotion computer (Intel i7-7500U). Three variations of problem formulation have been tested: optimizing all variables, keeping only phase durations fixed, and keeping both phase durations and footstep locations fixed. Not re-optimizing the phase durations gives a great improvement in the computation time per iteration while the success rate is similar, or better. Additionally, the foot trajectories can be taken from the initialization (but eventually adapting the height to correspond to the terrain) and not re-optimized, but this results in a slight increase in the computation time. It appears that it is simpler for the optimization to adapt the foot positions than the base trajectory since the base is coupled to, and constrained by, all the feet.

Optimization Variables	Flat Ground (ms)		Pallets (ms)	
	1st iter.	Mean	1st iter.	Mean
Full	315 [304]	156 [126]	843 [700]	417 [305]
No Phase	210 [202]	69 [60]	385 [371]	117 [108]
No Phase&Feet	282 [282]	84 [74]	534 [524]	177 [145]

TABLE I: Computation time per iteration. The first iter. takes longer due to the solver initialization, subsequent iteration times are approx. constant. The mean was computed for 100 iters. The values w/o costs are shown in the square brackets.

While the robot's sensors could have been used to create a model of the terrain (using its Intel RealSense depth camera), the model of the environment was instead loaded from a virtual model. This ensured repeatability and avoided limits in sensor field of view and the resolution. For the tests using pallets, the robot's front camera was used to read an AprilTag which gave the position and orientation of the obstacle with respect to the robot. The robot's onboard state estimator [24] was used as state input; measureable estimator drift was present. The full system—the generation of the initial guess, the optimization of the trajectory and its execution by the whole-body controller—ran onboard the robot's computer.

Fig. 7 shows the kinematic model and the real robot executing the optimized trajectory on the single pallet while Fig. 1 shows the double pallet. The results show that despite the errors in state and terrain estimation, the robot realized the trajectory to a high degree of accuracy and precision.

VI. Conclusion

This work extended an optimization formulation for walking robot trajectories so that its solutions are not only feasible in theory, but can also be reliably executed on a real quadrupedal robot on a variety of terrains of increasing difficulty. Strong nonconvexity and increased computational expense were offset by generating initial guesses from a neural network trained on filtered experiences gathered offline. We will carry these findings onto our future work which aims to achieve online replanning of terrain-aware dynamic locomotion with a several-step horizon. Future work will integrate environmental perception into the initialization map, more deliberately exploring the nonconvex solution space, and deploying the scheme in a receding-horizon manner for sustained mobility.

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, Jul. 2018. 1, 2

[2] M. Diehl, H. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast Motions in Biomechanics and Robotics*, M. Diehl and K. Mombaur, Eds. Springer, Berlin Heidelberg, 2006, vol. 340, pp. 65–93. 2

[3] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *Intl. J. of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014. 2

[4] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed. Society for Industrial and Applied Mathematics, 2010. 2

[5] E. Todorov, "A convex, smooth and invertible contact model for trajectory optimization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 1071–1076. 2

[6] T. Erez and E. Todorov, "Trajectory optimization for domains with contacts using inverse dynamics," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2012, pp. 4914–4919. 2

[7] I. Mordatch, E. Todorov, and Z. Popovi, "Discovery of complex behaviors through contact-invariant optimization," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 43:1–43:8, Jul. 2012. 2

[8] B. Ponton, A. Herzog, S. Schaal, and L. Righetti, "A convex model of humanoid momentum dynamics for multi-contact motion generation," in *IEEE/RSJ Int. Conf. on Humanoid Robots*, Nov. 2016, pp. 842–849. 2

[9] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 3555–3561. 2

[10] C. Mastalli, M. Focchi, I. Havoutis, A. Radulescu, S. Calinon, J. Buchli, D. G. Caldwell, and C. Semini, "Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2017, pp. 1096–1103. 2

[11] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic Locomotion Through Online Nonlinear Motion Optimization for Quadrupedal Robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2261–2268, Jul. 2018. 2

[12] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, St Paul, MN, USA, May 2012, pp. 3671–3678. 2

[13] 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. 2

[14] W. Merkt, V. Ivan, and S. Vijayakumar, "Leveraging Precomputation with Problem Encoding for Warm-Starting Trajectory Optimization in Complex Environments," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 5877–5884. 2

[15] A. Dragan, G. J. Gordon, and S. Srinivasa, "Learning from experience in manipulation planning: Setting the right goals," in *Proc. of the Intl. Symp. of Robotics Research (ISRR)*. Springer, Jul. 2011. 2

[16] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon, "Memory of motion for warm-starting trajectory optimization," *arXiv:1907.01474 [cs]*, Jul. 2019. 2

[17] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2011, pp. 3719–3726. 2

[18] 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 *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Brisbane, Australia, May 2018, pp. 2986–2993. 2

[19] M. Stolle, H. Tappeiner, J. Chestnutt, and C. G. Atkeson, "Transfer of policies based on trajectory libraries," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, Oct. 2007, pp. 2981–2986. 2

[20] Winkler, A. W. (2018) TOWR - An open-source trajectory optimizer for legged robots in C++. [Online]. Available: http://wiki.ros.org/towr 2

[21] A. Wachter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006. 3

[22] C. D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in *IEEE/RSJ Int. Conf. on Humanoid Robots*, Cancun, Mexico, Nov. 2016, pp. 558–564. 3

[23] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Conf. on Neural Information Processing Systems (NIPS)*, 2014, pp. 1071–1079. 4

[24] M. Bloesch, M. Burri, H. Sommer, R. Siegwart, and M. Hutter, "The two-state implicit filter recursive estimation for mobile robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 573–580, Jan 2018. 6