Learning an Expert Skill-Space for Replanning Dynamic Quadruped Locomotion over Obstacles

David Surovik, Oliwier Melon, Mathieu Geisert, Maurice Fallon, Ioannis Havoutis
University of Oxford, United Kingdom
{dsurovik,omelon,mathieu,mfallon,ioannis}@robots.ox.ac.uk

Abstract: Function approximators are increasingly being considered as a tool for generating robot motions that are *temporally extended* and express foresight about the scenario at hand. While these longer behaviors are often necessary or beneficial, they also induce multimodality in the decision space, which complicates the training of a regression model on expert data. Motivated by the problem of quadrupedal locomotion over obstacles, we apply an approach that disentangles modal variation from task-to-solution regression by using a conditional variational autoencoder. The resulting decoder is a regression model that outputs trajectories based on the task and a real-valued latent mode vector representing a style of behavior. With the task consisting of robot-relative descriptions of the state, the goal, and nearby obstacles, this model is suitable for receding-horizon generation of structured dynamic motion. We test this approach, along with a trajectory library baseline method, for producing sustained locomotion plans that use a generalized gait. Both options strongly bias planned footholds away from obstacle regions, while the multimodal regressor is far less susceptible to violating kinematic constraints. We conclude by identifying further prospective benefits of the continuous latent mode representation, along with targets for future integration into a hardware-deployable pipeline including perception and control.

Keywords: Legged Locomotion, Trajectory Planning, Unsupervised Learning

1 Introduction

Midsize quadrupedal robots hold promise for quickly transporting substantial sensor payloads across difficult terrain, but they involve greater dynamic complexity than many other types of mobile systems. Recent improvements of both hardware and algorithm capability have led to impressive demonstrations of dynamic quadruped locomotion—usually with a focus upon speed and agility on flat ground as well as robustness to external forces and small-scale irregularities in the terrain [1, 2]. A common aspect of most of these approaches, however, is that they are primarily *reactive* to perturbations at the moment they are felt, rather than *proactive* in terms of anticipating upcoming terrain features. Other methods are tailored to handle large-scale irregularities, such as stairs, ledges, gaps, or rubble, that demand temporally-extended behaviors [3, 4, 5]. However, these approaches typically involve conservative motion that is significantly slower either to execute or to compute.

This work is geared toward providing sustained dynamic locomotion in situations that strongly benefit from foresight in the planning process. We seek to enable quadrupeds to anticipate and flexibly respond to obstacles as they appear at the horizon through significant changes to the near-term plan, including continuous adjustments of the gait and momentum. The targeted capability is suited for handling prominent terrain features such as gaps and ledges.

Background: Trajectory optimization is apt for combining foresight and dynamism, and has been tailored to handle contact discontinuities in the software library Trajectory Optimization for Walking Robots (TOWR) [6]. However, it nominally involves high computational cost and susceptibility poor-quality local optima. Both issues can be greatly reduced by leveraging expert data to provide a high-quality initialization to the optimizer, allowing it to quickly reach a favorable solution. This approach has variously been referred to as "trajectory prediction" [7] or "memory of motion" [8, 9]. In this paper we will use the more explicit term "data-driven trajectory initialization" (DDTI).

DDTI is effectively behavior cloning, a form of imitation learning known to be susceptible to poor generalization and robustness in control applications due to narrow state-space coverage by the expert dataset. These downsides are lessened in the context of trajectory initialization since a degree of refinement can be computed online, and small perturbations can be mitigated by a lower-level tracking controller. Other data-driven approaches, such as reinforcement learning, cannot readily be applied in this scenario due to the difficulty of discovering far-sighted behaviors from scratch.

While the use of temporally-extended actions carries many benefits, it also tends to induce multimodality, which in turn causes issues with the use of function approximators. The classic example is an agent that needs to detour around an obstacle: it can go either left or right, but the average of these examples is an invalid solution. One recourse is to fit a different local expert model for each behavior type, but this can require restrictive assumptions or foreknowledge about possible behaviors. This might not be appropriate for highly versatile systems such as quadrupeds, whose various behaviors may be interrelated in either discrete or continuous ways.

Approach: In light of this, we implement a scheme for DDTI of temporally-extended quadruped behaviors without artificially limiting the expert's expressiveness. We train a conditional variational autoencoder (CVAE) to reproduce segments of long trajectories, conditioned on a task that expresses far-sighted context and relates to the constraints the trajectory must satisfy. Conditional latent distributions then represent the possible characteristic variations between locally-optimal solutions to a given task—in essence, the null-space of the expert's decision. We deploy the CVAE decoder for what we term Latent-Mode Trajectory Regression (LMTR) and demonstrate that it can be used in a receding-horizon manner to avoid a sequence of disallowed foothold regions. Afterward, we identify and discuss additional opportunities improving performance with a latent skill-space.

2 Related Work

Trajectory Libraries: A classic strategy for leveraging precomputed motions is to use a *trajectory library* (TL), where expert solutions are retrieved based upon the similarity of their associated tasks to a novel task encountered online. [10] employed a TL when planning quasi-static walking on rough terrain locomotion for a small quadruped; however, extensive computation was still required for selecting a sequence of behaviors in advance of execution. Other applications of TL have included reaching tasks by either fixed-base arms [7] or humanoids, which must also remain balanced [11].

Trajectory Regression: An increasingly common alternative is to fit a regression model to the relationship between task and solution. Prospective benefits include better adaptation to the task at hand, the lack of need for an explicit lookup metric, and reduced memory requirements. Some works for manipulator path planning regress to the next time step, constructing motion incrementally [12, 13]. Other efforts fit to entire trajectories of fixed scope [14, 9], which may be more appropriate when subject to dynamic constraints as is true for a drone changing position [8, 15], or a humanoid taking a step [16]. Alternately, regressors may be used to apply informed bias during sampling processes [17, 18] or to predict performance attributes of different categories of trajectory [19].

Multimodality: Some DDTI efforts attempt to reconcile regression and multimodality through methods that fit multiple regressors (a Mixture of Experts) and blend or choose between them based on the task [9, 15]. This strategy has also been applied to quadrupeds for animating their limbs in computer graphics [20] and stabilizing their steady-state motion in robotics [1]. By contrast, a *con-*

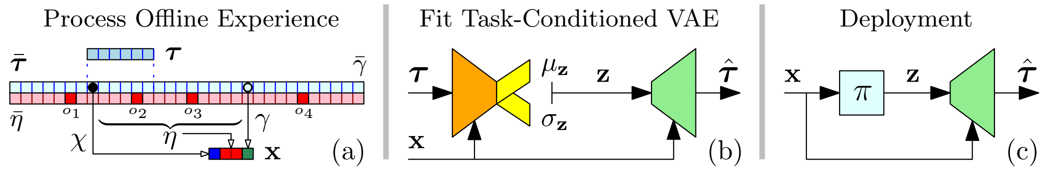

Figure 1: (a) long expert trajectories $\bar{\boldsymbol{\tau}}$ are processed into pairs of tasks $\mathbf{x}$ and trajectory segments $\boldsymbol{\tau}$ of reduced scope. (b) fitting a CVAE generates a smooth space of latent modes $\mathbf{z}$ that capture the variability of expert behavior, enabling good reconstruction quality even if two similar tasks were solved in distinct ways. (c) given a policy π for selecting a mode, the decoder can be deployed for other tasks seen online. This can be done in a receding-horizon fashion.

tinuously varying representation of behavior types (*i.e.*, modes) can be learned using CVAEs [21]. This has been used for predicting [22] or planning [18] wheeled-vehicle motion, as well as in manipulation when demonstration data varies considerably [23, 24]. Our work uses a CVAE in the manner of [23, 24], but for the starkly different application domain of sustained dynamic legged locomotion. We differ by using a behavior-selection model that permits multimodality on a *per-task* basis.

3 Latent-Mode Trajectory Regression

3.1 Expert Solutions

To accommodate problems whose solutions have a structured description, let a trajectory $\boldsymbol{\tau}$ be defined as a collection of different-length sequences of vectors

$$\boldsymbol{\tau} = (\mathbf{a}_{1:N_a}, \mathbf{b}_{1:N_b}, \ldots) \qquad\qquad s = (N_a, N_b, \ldots) \tag{1}$$

where $a, b, \ldots$ represent different categories of state variables or solution parameters, and the set of lengths, s, is termed the "scope". This section will discuss the use of two different scopes used for data generation and behavior cloning, respectively, using bar notation to distinguish variables associated with the former (longer) scope.

The expert data are trajectories $\bar{\boldsymbol{\tau}}$ linking initial states $\bar{\chi}$ and goal states $\bar{\gamma}$, *i.e.*, boundary conditions, across some environment $\bar{\eta}$. These data are generated using two complementary sources of randomization. Tasks $\bar{\mathbf{x}} = (\bar{\chi}, \bar{\gamma}, \bar{\eta})$ are sampled from a parametrized space $\bar{\mathcal{X}}$ defined by the user. Naive initial guesses $\bar{\boldsymbol{\tau}}_0$ are augmented with smooth noise $\bar{\xi}$ before passing them to the optimizer

$$\bar{\boldsymbol{\tau}} = \mathrm{Optimize}\left(\bar{\boldsymbol{\tau}}_0 = h\left(\bar{\mathbf{x}}, \bar{\xi}\right); \bar{\eta}\right) \tag{2}$$

where h is a simple heuristic that performs linear interpolation between the boundary conditions and incorporates the smooth noise in a manner appropriate for the solution structure, *i.e.*, the interdependency of the variable categories $(a, b, \ldots)$. Poor-quality results are filtered out of the long-scope expert dataset $\bar{\mathcal{D}} = \{\bar{\mathbf{x}}, \bar{\boldsymbol{\tau}}\}$ based on cost and constraint violation thresholds.

Segmentation is conducted next, mapping $\bar{\mathcal{D}} \longrightarrow \mathcal{D} = \{\mathbf{x}, \boldsymbol{\tau}\}$ to make the scope of the data appropriate for receding-horizon use and to make regressor fitting more tractable. Segmented trajectories $\boldsymbol{\tau}$ are extracted by moving a sliding window of scope s along the original data $\bar{\boldsymbol{\tau}}$ of scope $\bar{s}$. The solution structure of Eq. 1 may involve asynchronous state variables and require a nontrivial sliding-window rule; see Sec. 4.1 for quadruped-specific details. s should be long enough for online refinement of $\boldsymbol{\tau}$ to meaningfully adapt to errors, but short enough to manage compute expense. Significantly, γ and η can be defined at a further horizon than can be reached with s, allowing an additional margin of foresight.

Fig. 1(a) illustrates the extraction of $\boldsymbol{\tau}$ and the associated task $\mathbf{x} = (\chi, \gamma, \eta)$ from a full-length solution. γ represents a goal set (*e.g.*, the SE(2) component of the state) that the segment is moving toward in a locally optimal manner, and can be extracted from the original trajectory $\bar{\boldsymbol{\tau}}$ beyond the end of the sliding window. In principle, the horizon of γ should be similar to that of η, which expresses the degree of sensory foresight. For floating-base problems, $(\mathbf{x}, \boldsymbol{\tau})$ can be expressed in a coordinate frame defined in relation to the initial state χ. The combination of frame change and scope reduction densify $\mathcal{D}$ relative to $\bar{\mathcal{D}}$. Further data augmentation can be done by extracting several alternate choices of γ from the states the expert reaches beyond s, in a manner similar to [11].

3.2 Behavior Cloning for Multimodal Experts

The aim of behavior cloning is to effectively map from any task in a space $\mathcal{X}$ to an expert-quality solution in $\mathcal{T}$, based upon the discrete set of expert task-solution pairs $\mathcal{D}$. Directly fitting a function approximator to $\mathcal{D}$ would produce the regression model $f : \mathbf{x} \to \boldsymbol{\tau}$. This approach performs poorly in the presence of multimodality, *i.e.*, when multiple distinct solutions exist for a given task

$$(\mathbf{x}_1, \boldsymbol{\tau}_{1,a}) \in \mathcal{D} \qquad\qquad (\mathbf{x}_1, \boldsymbol{\tau}_{1,b}) \in \mathcal{D} \qquad\qquad \boldsymbol{\tau}_{1,a} \neq \boldsymbol{\tau}_{1,b} \tag{3}$$

which indicates multi-modal behavior by the expert. To accommodate multiple modes under a single regressor, its output will be conditioned on a secondary input $\mathbf{z}$, the *mode* or manner of solving the given task. The relationship $f_z : (\mathbf{x}; \mathbf{z}) \to \boldsymbol{\tau}$ can then be expressed smoothly as long as the information within $\mathbf{z}$ resolves the ambiguities and discontinuities present in $\mathcal{D}$.

To avoid restrictive manual assumptions about the form of $\mathbf{z}$, it is desired to learn it in an unsupervised manner. This is done by casting the mode as the latent variable in a conditional variational autoencoder (CVAE) [21] as in Fig. 1(b). With encoder q_θ and decoder p_ϕ, data maps through the CVAE as

$$q_\theta : (\boldsymbol{\tau}; \mathbf{x}) \longrightarrow (\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z) \qquad \mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_z, \mathrm{diag}(\boldsymbol{\sigma}_z)) \qquad p_\phi : (\mathbf{z}; \mathbf{x}) \longrightarrow \hat{\boldsymbol{\tau}} \tag{4}$$

with a diagonal-covariance Gaussian as the conditional latent distribution. Model parameters θ and ϕ are trained with an L1 reconstruction loss and a KL-divergence regularization term weighted by κ

$$\mathcal{L} = \frac{1}{|\mathcal{D}|} \sum_{\mathcal{D}} \Big[\|\boldsymbol{\tau} - \hat{\boldsymbol{\tau}}\|_1 + \kappa D_{KL}\big(\mathcal{N}(\mu_{\mathbf{z}}, \sigma_{\mathbf{z}}) \,||\mathcal{N}(\mathbf{0}, I)\big)\Big] \tag{5}$$

This training process essentially disentangles the degrees of freedom between different *locally optimal* solutions that share the same constraints, *i.e.*, the same task, into the mode space $\mathcal{Z}$. Expert choices of those degrees of freedom are captured by the dataset $\mathcal{D}_z = \{\mathbf{x}, \mathbf{z}\}$. The decoder p_ϕ serves as the regression model f_z needed for behavior cloning via LMTR.

3.3 Mode Selection Policy

To solve novel tasks during deployment, the expert regressor f_z must be paired with a policy for selecting a mode input, $\pi : \mathbf{x} \rightarrow \mathbf{z}$, as in Fig. 1(c).

$$\boldsymbol{\tau} = f_z(\mathbf{x}, \pi(\mathbf{x})) \tag{6}$$

This reveals the LMTR to be the second stage in a hierarchical decision process, with $\mathbf{z}$ as the high-level action and $\boldsymbol{\tau}$ as the low-level one. Accordingly, similar options for representing the $(\mathbf{x}, \mathbf{z})$ relationship can be considered as were for $(\mathbf{x}, \boldsymbol{\tau})$, but now with an output space that is much smoother and low-dimensional.

In [23], an additional function approximator $\pi_{\mathcal{N}} : \mathbf{x} \longrightarrow (\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z)$ is trained to match the output distribution of the encoder q_θ. We emphasize, however, that this forces the learned relationship for $(\mathbf{x}, \mathbf{z})$ to be unimodal despite the capacity for multimodality in the output of f_z, which could be ill-suited to scenarios where the expert uses distinct behaviors for very similar tasks. More expressive output distributions and alternate training methods could be considered for π, but with other complications as noted in Sec. 6.

To combine the simplicity of behavior choice that TL provides and the generalization capability of a regressor, we define π as a nearest-neighbors lookup on the mode experience set $\mathcal{D}_z$.

$$\pi_{nbr}(\mathbf{x}) = \mathbf{z}_i \mid \mathbf{x}_i = \underset{\mathbf{x}_i \in \mathcal{D}_z}{\mathrm{argmin}}\, d(\mathbf{x}_i, \mathbf{x}) \tag{7}$$

As with TL, performance depends strongly upon the distance metric d used in the task space. We use an L1 norm weighted by $W = \mathrm{diag}(\mathbf{w})$ to account for the heterogeneous nature of these variables.

4 LMTR Training

4.1 Expert Quadruped Solutions

The motivation of this work is to learn a *viable* set of diverse, intelligent skills for a sensor-rich quadruped which are subject to the robot's physical limitations and real-world uncertainty. Accordingly, we obtain our expert data using TOWR [6], a trajectory optimization library designed for legged robots in non-uniform environments. This library, when extended with features that promote robustness, has been demonstrated to produce dynamic motion realizable on hardware [25].

TOWR parametrizes legged locomotion trajectories as a collection of splines that describe the time-varying position and velocity of each element—*i.e.*, the floating base and each foot—as well as the ground-reaction forces at the points of contact. The transitions between contact and swing phases can occur at asynchronous epochs chosen by the optimizer, permitting the discovery of arbitrary gaits. Meanwhile, base splines are constructed using polynomials of a fixed duration joined at points called *nodes*. The optimization problem enforces multi-contact dynamics via equality constraints, while inequality constraints limit the forces and kinematics, and integral costs promote smoothness.

As discussed in Sec. 3.1, segmentation of the expert data facilitates online replanning. Fitting a regressor to $\mathcal{D}$ requires all $\boldsymbol{\tau}$ to have consistent structure—the same number, types, and order of optimzation variables. To achieve this, we use a segmented solution scope of (T, N_p), where T is the total duration and N_p is the number of swing phases *per leg*. We furthermore permit generalized gaits by assuming each foot's motion begins and ends in stance—totalling $N_p + 1$ stance phases—but allowing these phases to start and end *asynchronously* relative to the base nodes. For simplicity, segments that stop at the final goal are omitted from $\mathcal{D}$ since most of them are shorter than T.

4.2 Task Space

We define ground-plane obstacles as discrete linear features parametrized by an SE(2) pose $\{o_i = (x_i, y_i, \alpha_i)\}$. These obstacles do not have a definite width; rather, they express a potential field that penalizes the proximity of footholds. Full-length expert solutions begin and end in static stance and must cross two obstacles with randomized spacing and orientation, using scope $\bar{s} = (7.5\text{ s}, 6\text{ swings})$. After interpolating between the boundary conditions, the heuristic h maps the smooth noise ξ to the SE(2) components of the base state. Initial base-relative foot positions are also randomized, and timing variables are initialized to a walking gait that begins by swinging whichever foot is furthest back from its neutral position.

About 25% of the 5,000 solutions are discarded based on cost or constraint values that indicate poor quality and unrealistic behaviors, such as energetic jumping that the hardware system is not capable of executing. From the good solutions, 25,000 segments are sampled with scope $s = (0.8\text{ s}, 1\text{ swing})$. Crucially, the robot state at the segmentation epoch can be mid-motion, *i.e.*, nonzero base velocity and feet at arbitrary stages of swing or stance. Foot parameters describe a stance-swing-stance phase sequence that is asynchronous with the base trajectory and the other feet.

4.3 DDTI Models

Data sizes for this problem setup are $\dim(\mathbf{x}) = 35$ and $\dim(\boldsymbol{\tau}) = 264$. The LMTR decoder and encoder are fully-connected neural networks with hidden layer sizes $[64, 64, 264, 264]$ (reversed for the encoder), and are trained with Adam optimization on Eq. 5 using $\kappa = 0.001$ and with 1% noise added to task inputs. Several choices of latent dimension n are tested, denoted as LMTR-n. Inspection of the distribution of expert modes $\mathbf{z} \in \mathcal{D}_z$, shown in Fig. 2, reveals that $\mathcal{Z}$ is mainly used for representing decisions that *cannot* be unambiguously determined through the task-condition input to the regressor. For example, a foothold could avoid an obstacle through either a short stride or a long one, but not an average one—and thus $\mathcal{Z}$ shows clear organization in terms of stride length. The choice of which foot to move next, which usually has a simple dependence on the initial state, appears nearly uncorrelated and not a good use of the mode-space capacity.

A TL baseline is also considered, *i.e.*,

$$f_{nbr}(\mathbf{x}) = \boldsymbol{\tau}_i \mid \mathbf{x}_i = \underset{\mathbf{x}_i \in \mathcal{D}}{\operatorname{argmin}}\, d(\mathbf{x}_i, \mathbf{x}) \tag{8}$$

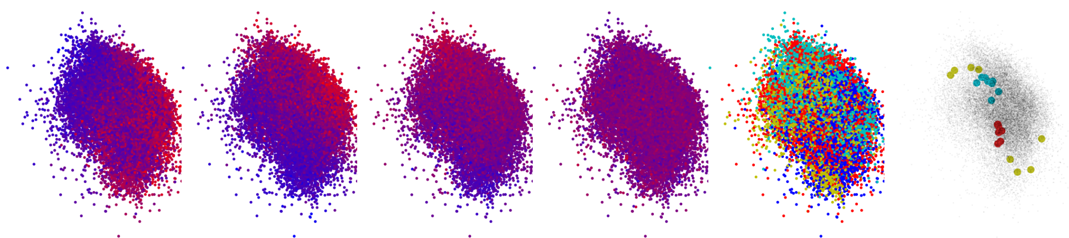
(a) LF stride (b) RF stride (c) Δ yaw (d) Δ height (e) First foot (f) $\mathbf{x}$-neighbors

Figure 2: The 3D distribution of expert mode choices from the training of LMTR-3, colored based on properties of the associated trajectory segments. (a-b) The stride length of each front foot varies steadily across $\mathcal{Z}$—and with independent trends. (c-d) Change in yaw follows a perceptible trend; change in height does not. (e) The latent mode does not appear to store information about which foot to move first. (f) shows the modes associated with several tasks from each of three neighborhoods of $\mathcal{X}$. The yellow distribution demonstrates that $\pi_{nbr} : \mathbf{x} \longrightarrow \mathbf{z}$ can be multimodal.

as well as a unimodal regressor f equivalent to 0-dimensional LMTR. The trajectory lookup time for TL is about 3 ms, as is the combined time for mode lookup $\pi_{nbr}(\mathbf{x})$ and neural network regression f_z in LMTR. The former method has a memory footprint of about 60 MB; the latter, about 10 MB, the majority of which is the mode library $\mathcal{D}_z$. The weight vector $\mathbf{w}$ used in the distance metric was sampled on a coarse 4D grid with separate dimensions for the task variable categories $\chi_{base}, \chi_{feet}, \eta$, and γ.

5 Receding-Horizon Planning

The applicability of DDTI to dynamic quadruped trajectories has been demonstrated in [25], where minimal online refinement was required for successful execution of open-loop plans on hardware. In this work, focus is placed on extension of the DDTI formulation to accommodate sustained use and multimodal behavior; the numerous further challenges of control-integrated deployment in simulation and on hardware are deferred to future work. Experiments thus consist of plan generation via incremental "rollouts" of each DDTI method, *i.e.*, by deriving the task at each replanning cycle from a state directly extracted from the trajectory initialization of the previous cycle.

5.1 Rollout Scheme

DDTI of trajectory *segments* enables behaviors with foresight to be deployed in a *receding-horizon* fashion. This replanning approach is suited for mitigating perturbations as they appear on the perceptive sensory horizon. These perturbations may be registered through η, as new obstacles appear, or through γ, as a mission planner updates the locomotion goal. *Multimodal* DDTI allows the robot to instantaneously switch to an *alternate strategy* for tackling an obstacle, even as it is constrained by its particular contact state and momentum expressed in χ.

At each iteration of a rollout, using replanning period Δt, a new plan is needed beginning from some state $\chi \leftarrow \boldsymbol{\tau}(\Delta t)$ of the previous plan. The environment ahead of the robot is processed to determine η, which we define as the next two discrete obstacle features $\eta \leftarrow (o_i, o_{i+1})$. A mission planner then updates the goal location γ, placing it on the world x-axis a fixed distance ahead of the robot. Given this new task $\mathbf{x} = (\chi, \gamma, \eta)$, trajectory initialization is computed with one of the DDTI methods: Eqs. 6 and 7 for LMTR or Eq. 8 for TL. The rollout testbed randomly generates new obstacles ahead of the robot, varying their world-frame orientation and forward spacing. Though the segment scope extends beyond Δt, only the solution variables with timestamps within the replanning window are accumulated in the rollout.

The parameters and choices in this scheme must balance several considerations. A more distant goal position and environment perception window would enable greater foresight, but only subject to sensing limitations and the modeling demands associated with increased $\dim(\mathbf{x})$. The segment scope s need not reach this horizon (see Fig. 1(a)) as the states within $\boldsymbol{\tau}$ are implicitly appropriate waypoints for $\mathbf{x}$. In an integrated deployment, longer s would allow online refinement of $\boldsymbol{\tau}$ to have

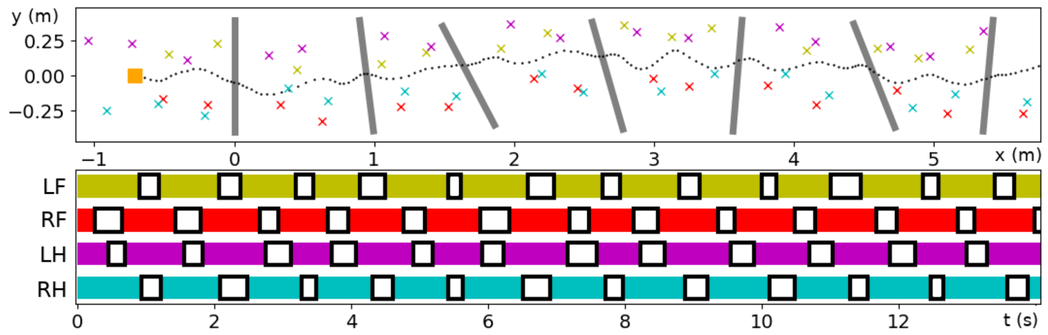

Figure 3: An example rollout of the LMTR replanner over many obstacles, which are potential fields penalizing foothold proximity. The replanner extends the base trajectory (black dots) forward while also placing footholds (x's colored per foot) away from the obstacles (gray lines). The planner's structured outputs remain similar to expert data that meets constraints and minimizes costs. Colored bars indicate the contact phases of each foot; outlined white boxes give the swing phases.

greater influence in correcting approximations, but would also increase its compute cost, along with $\dim(\boldsymbol{\tau})$. The replanning period Δt is constrained by the trajectory compute cost, which is mostly determined by the amount of refinement required. It is most relevant for handling perturbations to χ that would occur during execution; however, we note that small perturbations of this sort could be handled by a tracking controller. Major perturbations away from the experience manifold of $\mathcal{X}$ might be more appropriately handled by a recovery controller than a temporally-extended planner.

An example output obtained with this rollout scheme using $\Delta t = 0.3$ s is shown in Fig. 3. The base path shows that the motion is smooth and sustained; velocity shaping is apparent in the closer spacing of nodes used during careful movement over the obstacles, which is necessary due to the more constrained foot locations. Footstep markers remain distant from the obstacles in most cases. The foot phase plot further shows that the trajectory parametrization allowed expressive variation of gait timings. Concurrent pairs of swing phases, such as in the first half of the rollout, indicate a trot-type gait; a cascading pattern of swing phases (near the end) indicates a walking gait.

5.2 Rollout Performance

We assess aggregate performance in terms of obstacle avoidance and the violation of kinematic limits. Due to the more temporally-extended influence of footholds upon the whole trajectory, we hypothesize that these two constraint types might be less tractable to resolve during online refinement than short-timescale dynamics violations caused by stochasticity of the regressor. Because the obstacles are defined as potential fields, a real-valued metric is used to express obstacle violation—for each obstacle encountered in a rollout, the minimum distance between each foot's closest foothold and the obstacle line is accumulated. Similarly, for each 0.1 s time-step of the base trajectory, the relative forward positions of the contacting feet are computed and any violation of the nominal ± 0.2 m range relative to their neutral locations is logged.

These two metrics are evaluated for a unimodal regressor, for LMTR with varying mode dimensionality, and for TL. Several weight vectors $\mathbf{w}$ are considered for the task-space distance utilized by the latter two methods. 100 rollouts are generated for each replanner, with each rollout lasting 30 s (100 replanning cycles), corresponding to around 11 m traveled and about 13 obstacles crossed.

As shown in Fig. 4(a), the unimodal regressor commonly selects footholds very close to the obstacle, while LMTR and TL both strongly bias their footholds away. Performance of LMTR plateaus at around 6 to 12 latent dimensions, at which point it roughly matches TL in terms of obstacle avoidance. Fig. 4(b), however, reveals that TL has a much higher rate and degree of kinematic violations. Performance trade-offs can be made by prioritizing some categories of task variables over others in the weight vector $\mathbf{w}$ used for nearest neighbors, as shown in Fig. 4(c), where favoring χ or η respectively promotes either the kinematic or the obstacle metric. Altogether, LMTR offers about 5x milder kinematic violation than TL for a given level of obstacle avoidance performance.

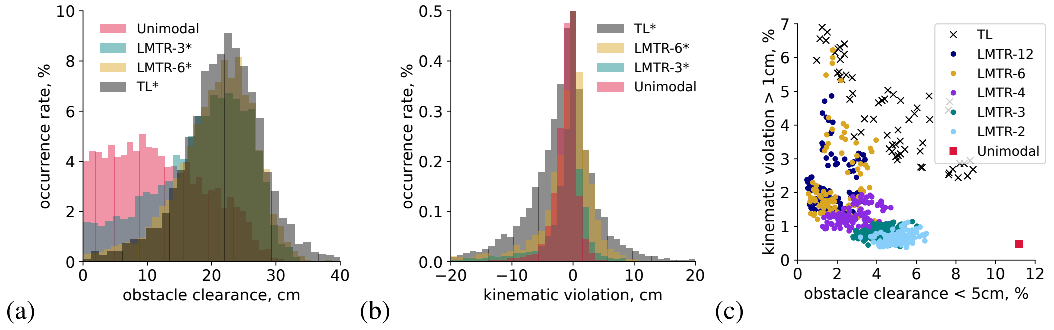

Figure 4: Obstacle avoidance margin and kinematic violation margin are evaluated using several replanner configurations. Star notation indicates selection of the best weight vector for that replanner type and performance metric. (a) probability distribution of the closest foothold to each obstacle being a certain distance away, accumulated across all four feet. (b) probability distribution of a foot violating its nominal range of motion by a given amount; y-axis is zoomed in to expose these uncommon violations. (c) rates of threshold violations for the two performance metrics, where each point per series represents aggregate performance under a different nearest-neighbors weight vector.

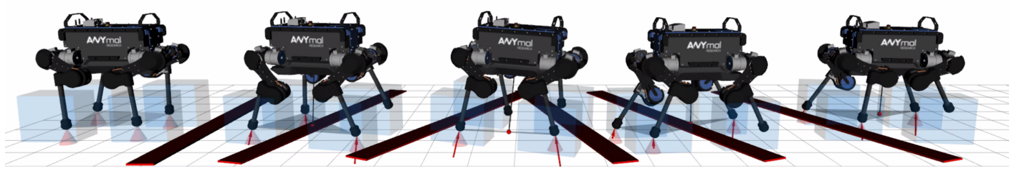

Figure 5: Snapshots from a typical rollout of LMTR-6. The expert's conservative kinematic limits are indicated by blue boxes; friction cones and planned contact forces are shown in red.

Several example rollouts of LMTR can be seen in the attached video supplement, showing further qualitative evidence of approximate feasibility and local optimality of the trajectory initializations it generates. Fig. 5 provides snapshots of one such rollout.

6 Discussion

LMTR matched the simpler TL approach in achieving obstacle avoidance and exceeded it in terms of kinematic feasibility, thanks to its continuous output space. A crucial point for further study is how well each method scales to a wider variety of tasks—*e.g.*, with terrain height or slope included in $\mathbf{x}$—and the new behaviors they warrant. Selective retention or importance-weighting of experiences could help to moderate the growth of TL or to improve the fit of LMTR to less common scenarios.

The fact that neither TL nor LMTR provided completely consistent obstacle avoidance suggests that performance may have been bottlenecked by gaps in the expert knowledge. The ease and feasibility of generating a larger variety of expert data depends on how well the offline optimization process handles increasingly broad task spaces, which cause greater nonconvexity. Key to this is the rate at which the noisy initialization heuristic in Eq. 2 lands in the basins of attraction of good-quality local optima, *i.e.*, the efficiency of trajectory-level exploration. One possibility is to use noise in the latent mode space $\mathcal{Z}$ to push the boundaries of known behaviors and iteratively improve the LMTR in a reinforcement learning scheme related to hierarchical approaches such as [26].

Another clear avenue for improvement is the inference process relating tasks to modes of behavior, as geometric norms are not an ideal similarity measure for large, heterogeneous task descriptions. Fitting a stochastic model of $\pi : \mathbf{x} \longrightarrow \mathbf{z}$ could make the task distance implicit within learning problem. Multimodality on a *per-task* basis would require a more expressive model here than the Gaussian used in [23, 24]. Alternative choices such as mixture models or normalizing flows [27] might meet these needs, but carry with them nontrivial challenges. This is because their increased capacity, when used for *conditional* density estimation, risks overfitting to the variation of the condition—which ultimately relates to the concept of the difference between two tasks.

One more potential benefit of modelling multiple viable behaviors *per task* is the opportunity to make a choice that minimizes the need for online refinement [28] or the cost of execution [19, 9]. Depending on evaluation costs, these critera could be used for actor-critic reinforcement learning of a multimodal policy π [27]. Lastly, a smooth model of this relationship could open the door to behavior selection based upon robustness to noise or approximation error in the task description.

7 Conclusion

We have presented a data-driven approach to quickly generate initializations of legged locomotion trajectories that anticipate prominent obstacles. The approach is tailored for sustaining dynamic motion indefinitely in a receding-horizon fashion. Our results demonstrate that it is vital to account for the high degree of multimodality when applying an expert regressor to the problem of generalized, inherently-discontinuous quadrupedal locomotion, and that this can be done in an unsupervised manner by using a variational autoencoder conditioned on the task. Future work on the DDTI formulation will aim to remove the dependence on nearest-neighbors (and the task-mode library) and to more deliberately leverage the diversity of behaviors available for a given task. Our efforts toward full control pipeline integration and hardware deployment will involve overcoming significant remaining challenges: online refinement of plan segments with asynchronous structure, handling of replanning latency, and robust vision-based extraction of terrain features suited for task description.

Acknowledgments

This work was supported by the EPSRC grant 'Robust Legged Locomotion' [EP/S002383/1], the UKRI/EPSRC RAIN [EP/R026084/1] and ORCA [EP/R026173/1] Hubs, a Royal Society University Research Fellowship (Fallon), and the EU H2020 Projects MEMMO and THING. This work was conducted as part of ANYmal Research, a community to advance legged robotics.

References

[1] J. Carius, F. Farshidian, and M. Hutter. Mpc-net: A first principles guided policy search. *IEEE Robotics and Automation Letters*, 5(2):2897–2904, 2020.

[2] 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)*, pages 6316–6323. IEEE, 2019.

[3] 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*, 5(2):3699–3706, 2020.

[4] M. Geisert, T. Yates, A. Orgen, P. Fernbach, and I. Havoutis. Contact planning for the anymal quadruped robot using an acyclic reachability-based planner. In *Annual Conference Towards Autonomous Robotic Systems*, pages 275–287. Springer, 2019.

[5] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernández-López, and C. Semini. Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization. *IEEE Robotics and Automation Letters*, 3(3):2531–2538, 2017.

[6] 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*, 3(3):1560–1567, 2018.

[7] N. Jetchev and M. Toussaint. Fast motion planning from experience: trajectory prediction for speeding up movement generation. *Autonomous Robots*, 34(1-2):111–127, 2013.

[8] 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)*, pages 2986–2993. IEEE, 2018.

[9] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon. Memory of motion for warm-starting trajectory optimization. *IEEE Robotics and Automation Letters*, 5(2):2594–2601, 2020.

[10] M. Stolle, H. Tappeiner, J. Chestnutt, and C. G. Atkeson. Transfer of policies based on trajectory libraries. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2981–2986. IEEE, 2007.

[11] W. Merkt, V. Ivan, and S. Vijayakumar. Leveraging precomputation with problem encoding for warm-starting trajectory optimization in complex environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5877–5884. IEEE, 2018.

[12] T. Jurgenson and A. Tamar. Harnessing Reinforcement Learning for Neural Motion Planning. In *Robotics: Science and Systems XV*. Robotics: Science and Systems Foundation, June 2019.

[13] M. J. Bency, A. H. Qureshi, and M. C. Yip. Neural path planning: Fixed time, near-optimal path generation via oracle imitation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3965–3972. IEEE, 2019.

[14] D. Forte, A. Gams, J. Morimoto, and A. Ude. On-line motion synthesis and adaptation using a trajectory database. *Robotics and Autonomous Systems*, 60(10):1327–1339, 2012.

[15] G. Tang and K. Hauser. Discontinuity-sensitive optimal control learning by mixture of experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7892–7898. IEEE, 2019.

[16] 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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[17] C. Chamzas, A. Shrivastava, and L. E. Kavraki. Using local experiences for global motion planning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8606–8612. IEEE, 2019.

[18] R. Kusumoto, L. Palmieri, M. Spies, A. Csiszar, and K. O. Arras. Informed information theoretic model predictive control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2047–2053. IEEE, 2019.

[19] A. D. Dragan, G. J. Gordon, and S. S. Srinivasa. Learning from experience in manipulation planning: Setting the right goals. In *Robotics Research*, pages 309–326. Springer, 2017.

[20] H. Zhang, S. Starke, T. Komura, and J. Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)*, 37(4):1–11, 2018.

[21] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491. 2015.

[22] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[23] 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*, pages 1113–1132, 2020.

[24] M. Noseworthy, R. Paul, S. Roy, D. Park, and N. Roy. Task-conditioned variational autoencoders for learning movement primitives. In *Conference on Robot Learning*, pages 933–944, 2020.

[25] 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)*. IEEE, 2020.

[26] J. Co-Reyes, Y. Liu, A. Gupta, B. Eysenbach, P. Abbeel, and S. Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International Conference on Machine Learning*, pages 1009–1018, 2018.

[27] B. Mazoure, T. Doan, A. Durand, J. Pineau, and R. D. Hjelm. Leveraging exploration in off-policy algorithms via normalizing flows. In *Conference on Robot Learning*, pages 430–444, 2020.

[28] D. Berenson, P. Abbeel, and K. Goldberg. A robot path planning framework that learns from experience. In *2012 IEEE International Conference on Robotics and Automation*, pages 3671–3678. IEEE, 2012.