VAE-Loco: Versatile Quadruped Locomotion by Learning a Disentangled Gait Representation

Alexander Luis Mitchell, Graduate Student Member, IEEE, Wolfgang Xaver Merkt, Member, IEEE, Mathieu Geisert, Siddhant Gangapurwala, Martin Engelcke, Oiwi Parker Jones, Ioannis Havoutis, Member, IEEE, and Ingmar Posner

Abstract—Quadruped locomotion is rapidly maturing to a degree where robots are able to realize highly dynamic maneuvers. However, current planners are unable to vary key gait parameters of the in-swing feet midair. In this article, we address this limitation and show that it is pivotal in increasing controller robustness by learning a latent space capturing the key stance phases constituting a particular gait. This is achieved via a generative model trained on a single trot style, which encourages disentanglement such that application of a drive signal to a single dimension of the latent state induces holistic plans synthesizing a continuous variety of trot styles. We demonstrate that specific properties of the drive signal map directly to gait parameters, such as cadence, footstep height, and full-stance duration. Due to the nature of our approach, these synthesized gaits are continuously variable online during robot operation. The use of a generative model facilitates the detection and mitigation of disturbances to provide a versatile and robust planning framework. We evaluate our approach on two versions of the real ANYmal quadruped robots and demonstrate that our method achieves a continuous blend of dynamic trot styles while being robust and reactive to external perturbations.

Index Terms—Legged locomotion, representation learning, robot learning.

I. INTRODUCTION

QUADRUPED locomotion has advanced significantly in recent years, extending its capability toward applications of significant value to industry and the public domain. Driven primarily by advances in optimization-based [1], [2], [3], [4], [5] and reinforcement learning (RL) based methods [6], [7], [8], quadrupeds are now able to robustly plan and perform dynamic maneuvers, making them an increasingly popular and reliable choice for tasks, such as inspection, monitoring, search, and rescue or goods delivery in difficult. However, despite recent advances, important limitations remain. Due to the complexity of the system, models used for gait planning and control are often overly simplified and handcrafted for particular and pre-determined contact schedules [1], [9]. In the worst case, this can limit the versatility of the robot as the models deployed are failing to exploit the full capability of the underlying hardware (e.g., [1], [10], and [11]). In particular, fixed gait parameters limit the ability of the robot to react to and reject external disturbances, such as pushes to the robot’s base. For example, of significant value to industry and the public domain. Driven primarily by advances in optimization-based [1], [2], [3], [4], [5] and reinforcement learning (RL) based methods [6], [7], [8], quadrupeds are now able to robustly plan and perform dynamic maneuvers, making them an increasingly popular and reliable choice for tasks, such as inspection, monitoring, search, and rescue or goods delivery in difficult. However, despite recent advances, important limitations remain. Due to the complexity of the system, models used for gait planning and control are often overly simplified and handcrafted for particular and pre-determined contact schedules [1], [9]. In the worst case, this can limit the versatility of the robot as the models deployed are failing to exploit the full capability of the underlying hardware (e.g., [1], [10], and [11]). In particular, fixed gait parameters limit the ability of the robot to react to and reject external disturbances, such as pushes to the robot’s base. For example,
the ability to adjust the robot’s swing trajectories on demand (e.g., the swing height, length, and timing) allows the robot to stabilize itself. The feet can be placed faster and into positions ahead of the center of mass. There are works that are capable of making adjustments to a predetermined contact schedule. The first category of method updates the contact schedule utilizing simplified dynamics’ models [12]. The second type solves the contact timings in advance by adjusting the switching times between optimization segments [13], [14] and uses reduced-order models. The last category optimizes dynamics over gait schedules, footstep lengths, and heights. These are often computationally expensive [3], [4] meaning that varying the gait parameters is not achievable in real time. A limitation of all these methods is that they are unable to adjust key gait parameters, in particular the contact timings, of the feet midair. Furthermore, these adjustments are limited to small perturbations around a default value. The result of which is that these methods are unable to react quickly to external perturbations irrespective of the terrain the robot is traversing. In contrast, the ability to control key gait parameters—such as contact timing, swing height, and full-support duration—on-the-fly would enable a smooth interpolation between dynamic maneuvers, allowing for the swift reaction to external stimuli. This leads to significantly more versatile locomotion.

Inspired by recent work on a quadruped that achieves a crawl gait via the traversal of a learned latent space [15], we approach the challenge of continuous contact schedule variation from the perspective of learning and traversing a structured latent space. This is enabled by learning a generative model of locomotion data which, in addition to capturing relevant structure in the latent space, also enables the detection and mitigation of disturbances to provide a versatile and robust planning framework. In particular, we train a variational autoencoder (VAE) [16], [17] on short sequences of state-space trajectories taken from a single gait type (trot) and predict a set of future states. We show that the resulting latent space has an interpretable structure, lending itself to the generation of a variety of trot styles, depending on how the latent space is traversed. In fact, examining trot trajectories in latent space reveals an oscillatory drive signal, which controls fundamental aspects of the gait. We subsequently find that, by overwriting this trajectory with a synthetic drive signal, we can continuously control the robot’s gait properties while the robot is executing the motion. Parameters of this drive signal can be mapped explicitly to gait parameters, such as cadence, footstep height, and full-support duration. With this, we can continuously vary the contact timings of feet midair. In fact, we can accelerate the cadence of the swing feet from four steps a second at takeoff to eight at touch down within the duration of a single footstep. This constitutes a novel capability in quadruped control. We emphasize that this ability to generalize over gait styles emerges from training on a single gait type: a trot gait with constant parameters.

We illustrate the efficacy of our approach by generating a range of continuously blended trajectories first on the real ANYmal B quadruped robot—a medium-sized platform (35 kg) standing 0.5 m tall. Subsequently, with no retraining, we repeat the experiment on the heavier ANYmal C quadruped, which weighs in at 50 kg and delivers twice the peak torque of the former platform. This demonstrates not only a transfer from simulation to the real robot but also to a dynamically dissimilar platform crucially without retraining. While the latent space is learned using examples only from a specific gait style, our approach is able to synthesize behaviors significantly beyond this training distribution. We choose to limit our analysis to locomotion on flat ground in order to fully explore and understand this novel control paradigm.

In addition, we leverage our generative approach to both characterize and react to external perturbations. A large impulse applied to the robot’s base triggers a spike in the evidence lower bound (ELBO) that clearly identifies the disturbance as out-of-the-distribution seen during training. Inspired by the work of Moyer et al. [18], which states that an increase in cadence is both a response to slip and a form of push recovery in humans, our planner automatically increases the robot’s cadence to aid in counteracting the disturbance. This demonstrates a marked improvement in robustness.

To the best of our knowledge, our method is the first that enables the continuous online adaptation of the robot’s gait characteristics during a swing phase while the robot is walking. It provides a versatile and data-driven approach to quadruped locomotion, which additionally allows for disturbance detection and recovery.

A. Statement of Contributions

The primary contribution of this article and its conference version Next Steps [19] is a novel planning methodology, which realizes locomotion with continuously variable gait parameters. Indeed, the prior art is only able to alter the future timings of quadruped’s contact schedule [12], [13], while the methods presented here facilitate the variation of timings for in-swing feet midair. Varying the foot-step speed of the airborne feet in response to detected disturbance is shown to increase the robot’s ability to reject large push disturbances. In this article, we deploy the VAE without retraining on a brand new platform with significantly different dynamics: the ANYmal C.

This article extends our previous work, Next Steps [19] in order to showcase an in-depth analysis of the method previously proposed. This includes significant extensions to key sections of Next Steps as well as new analysis. In particular, we have extended the description and justification for the oscillatory drive signal in Section III-A. We have significantly expanded the interpretation of the latent-space structure and explain how the trajectory in this space looks (see Section VI-A). Additionally, we compare the nominal trajectory in latent space to that seen during the disturbance and recovery phase, significantly expanding on the analysis of the disturbance rejection experiments (see Section VI-H). We also extend the discussion of the ablation study of the model’s sensitivity to hyperparameters (see Section VI-C).

In this article, we introduce new aspects of analysis. First, we show how backpropagating the binary cross entropy (BCE) through the VAE’s encoder is essential for structuring the latent
space (see Section VI-B). The results of which are crisp decision boundaries and axis alignment of the latent space. This results in an interpretable and disentangled representation. Second, we analyze the receptive field of the VAЕ’s encoder in order to understand how the robot’s gait phase is inferred (see Section VI-D). Inference of the gait phase is crucial for successful closed-loop planning in latent space. Third, we compare the dynamic feasibility of the trajectories from the VAЕ planner to those seen during training, see Section VI-F, and show that the VAЕ-planner’s locomotion trajectories remain dynamically feasible as they are significantly varied. Next, we compare the distribution of realizable gait parameters output from the VAЕ planner to those seen in the dataset (see Section VI-G). We show the range of gaits achievable using our approach and compare this to the distribution seen during training. Finally, after showing that the approach can transfer from simulation to the real ANYmal B in Next Steps, we push this limit and deploy the VAЕ planner on ANYmal C without retraining (see Section VI-I). This experiment demonstrates the extent to which the VAЕ planner is able to generalize to out-of-distribution scenarios. In particular, ANYmal C exhibits significantly different dynamics to ANYmal B, which are detailed in Section VI-I.

II. RELATED WORK

Planning and control for quadruped locomotion have advanced in leaps and bounds in recent years. For example, hierarchical-optimization frameworks split locomotion tasks into a series of smaller problems. Examples in this area include Dynamic Gaits (DGs) [1], [12], [20]. DG and Grandia et al. [20] enable a quadruped robot (such as ANYmal) to execute a wide variety of DGs (e.g., trot, pace, lateral walk, and jump) with real-time motion planning and control. However, to achieve this impressive range of behaviors, all these methods provide each gait type with its own contact schedule and utilize an environment-specific footstep planner, ultimately limiting their capabilities. Bledt and Kim [21] address the shortcomings of hierarchical planning approaches by learning a set of heuristic operating ranges in order to increase the overall dynamic range of quadruped locomotion. This is similar in philosophy to the work we present here. However, we achieve a broad dynamic range by learning a distribution over feasible trot gaits. This distribution is then sampled via our drive signals resulting in flexible and dynamic trot locomotion. This allows us to continually adjust the foot-swing timing of the airborne feet and is achievable much faster and to a broader degree than prior art. We distinguish this ability from methods that perturb the swing duration of feet, which are yet to break contact [12], [13], [14]. In addition, Rathod et al. [12] require a heuristic metric for synchronizing the contact scheduler with the current contact state. Our approach sidesteps this requirement by performing closed-loop feedback directly in the learned latent space.

Latent-space approaches for planning and control learn useful and typically low-dimensional representations that can be used to control complex dynamics, without relying on known system models. Classic examples include deep variational Bayes filters (DVBF) [22] and embed-to-control (E2C) [23]. DVBF produces dynamically consistent trajectories by traversing continuous paths in latent space, while E2C learns a linear system model in which control problems can be solved. Conditional neural movement primitives approach [24] is a more recent latent-space approach for robotic arms that generalizes between a variety of tasks, such as pick-and-place and obstacle avoidance. Other recent works, such as UPN and PlaNet [25], [26], shows impressive capabilities in simulation but are yet to be applied to the real-world systems, including floating-base robots.

In the locomotion domain, Li et al. [27] propose an approach, which utilizes a learned latent-action model to create different locomotion trajectories. However, this latent-action space does not capture robot dynamics and samples trajectories via a random shooting method. In contrast, First Steps [15] learns a structured latent space based on feasible robot configurations to capture the complete robot dynamics. First Steps defines a set of performance predictors that can be used in an optimization framework to control the robot. In practice, these performance predictors can be viewed as symbolic inputs (e.g., “left front leg up”) but drive the robot in continuous space. However, because First Steps is trained on static snapshots of robot configurations, it does not learn from observable dynamics and, thus, requires more explicit structuring of the latent space than is necessary. Our previously published work Next Steps [19] addresses this shortcoming and significantly extends this framework to effective and robust closed-loop planning and control. In this article, we provide a significantly more in-depth study into a continuous variation of the gait parameters via planning in a structured latent space. In doing so, we further justify the utilization of an oscillatory drive signal, analyze the quality of the VAЕs trajectories, and push the limits of domain transfer through deployment on a new robotic platform without retraining.

The Motion VAЕ (MVAЕ) [28] learns to represent dynamic trajectories in a structured latent space for the locomotion of computer-animated humanoids. This is similar to our emphasis here on learning representations for dynamic trajectories in the context of locomotion. However, moving from simulated to real physical systems, as is required for robotic applications, necessitates tackling additional complexities, such as latency, hard real-time requirements, and actuator dynamics. In this work, we tackle these challenges and demonstrate that a single gait style contains sufficient richness to learn a structured latent space that can be exploited to manipulate gait characteristics that generalize beyond the range seen during training. Unlike MVAЕ, our approach does not train on multiple gait styles, despite succeeding in producing them.

Other approaches build a model over multiple gait types and styles without utilizing a structured latent space. An example of which is [29]. This utilizes a vision-based system to sample a footstep schedule and gait for the terrain ahead. These form the input to an model-predictive control (MPC) approach. Alternatively, we choose not to condition the gait parameters on vision but allow the operator to directly choose the gait style. Additionally, our system is able to generate a broad range of trot trajectories while being trained on a single demonstration of a trot gait with fixed parameters. Hence, we propose our method
as an alternative to [29] for creating a queryable model over different types of locomotion styles.

Finally, a study conducted concurrently with our own [30] yields variation between gait types (walk and trot). It utilizes an RL approach, which employs a phase iterator similar to our drive signal. However, this phase iterator is enforced, while our gait dynamics are discovered automatically purely from exposure to trot trajectories with constant parameters.

### III. APPROACH

Our aim is to use unsupervised learning to infer a structured latent space that facilitates real-time and smooth variation of key gait parameters (see Fig. 1). We conjecture that the structure can emerge from the exposure of a suitable generative model to a gait with predetermined and constant characteristics, such as cadence, swing height, and full-support duration. Specifically, we propose that, due to this structure, continuous latent trajectories result in robot locomotion (see Fig. 2). By inspecting this structure, we discover a disentangled latent space where gait parameters are axis aligned within this space (see Section VI-A). Periodic trajectories in latent space can then be decoded back to smooth robot locomotion, as depicted in Fig. 1. Subsequently, the VAE is deployed as a planner in a real-time control loop.

**VAE Architecture:** We train a VAE [16], [17] to create a structured latent space using observed dynamic data. The input to the VAE $X_k$ at time step $k$ consists of $N$ robot states sampled from simulated trot gaits with constant parameters (e.g., cadence and foot-step height). These state-space quantities are values we wish either to control or are required to infer the gait phase. These are joint angles, end-effector positions in the base frame, joint torques, contact forces, the base velocity, and the base pose evolution relative to a control frame, which is updated periodically. These quantities are denoted as $x_k = [\mathbf{q}_k, \mathbf{\dot{q}}_k, \mathbf{\tau}_k, \mathbf{\dot{\tau}}_k, \hat{\mathbf{c}}_k, \Delta \mathbf{c}_k]$, where $k$ is the time step. Note that velocities and accelerations do not form a part of the VAEs input but are inferred from the input history. This has dual benefits: 1) it yields a lower dimensional input space; and 2) it prevents sensitivity to fast-changing quantities, such as the recorded joint accelerations during inference.

To deploy the VAE planner in a closed-loop framework, we encode the input history at the control frequency $f_c$. However, due to restrictions on the VAEs size caused by the tight computation bounds required for real-time control, the encoder input $X_k$ is constructed using states spaced at a frequency of $f_{enc}$

$$X_k = [x_{k-r(N-1)}, \ldots, x_{k-r}, x_k]$$  \hspace{1cm} (1)

where $r = f_c/f_{enc}$ is the ratio between the control and encoder frequencies. The input $X_k$ is created every time step by sampling from an input buffer, which stores every robot state $x$ from time step $k$ to $k-r(N-1)$ at the control frequency. We provide full details of the setting used in our experiments in Section IV-B.

The VAEs decoder output $\hat{X}_k^+$ predicts the current robot state $x_k$ as well as $M$ future ones sampled at a frequency of $f_{dec} = f_c$

$$\hat{X}_k^+ = [\hat{x}_k, \hat{x}_{k+1}, \ldots, \hat{x}_{k+M}]$$  \hspace{1cm} (2)

Here, $+$ denotes the future steps and $\hat{\cdot}$ denotes the predictions.

As the *desired-feet-in-contact* is an input to the tracking controller, we also want to predict which of the four feet are in contact, $s_k$, at the current time step $k$, as well as $J$ steps in the future. Inspired by First Steps [15], we, therefore, utilize a feet-in-contact performance predictor $g_{pp}(\mathbf{z}_k)$. This is attached to the latent space, which estimates the probability of each foot being in contact

$$\hat{S}_k = [\hat{s}_{k-N}, \ldots, \hat{s}_{k-J+1}]^\top.$$ \hspace{1cm} (3)

To command the base twist of the robot, a high-level action $a_k$ is utilized. This represents longitudinal ($x$), lateral ($y$), and yaw ($\theta$) twists in the robot’s base frame. The latent state $x_k$ and the action $a_k$ form the input to the decoder.

**Training the VAE:** We train the VAE and performance predictor together. The VAEs training loss is the modified ELBO formulation found in [31]. This loss consists of a reconstruction loss (mean-squared error) plus the Kullback–Leibler (KL) divergence $D_{KL}$ between the inferred posterior $q(\mathbf{z}|X_k)$ and the prior $p(\mathbf{z})$, multiplied by a hyperparameter $\beta$

$$L_{ELBO} = \text{MSE}(\hat{X}_k^+, X_k^+) + \beta D_{KL}[q(\mathbf{z}|X_k)||p(\mathbf{z})].$$ \hspace{1cm} (4)

These ELBO terms are then summed with the BCE loss between the predicted feet-in-contact and the recorded ones. The latter term is scaled by $\gamma$, resulting in the overall loss

$$L = L_{ELBO} + \gamma \text{BCE}(S_k, \hat{S}_k).$$ \hspace{1cm} (5)

The VAE training loss (4), as seen in prior work [31], is responsible for any subsequent disentanglement found in the latent space. The reconstruction error is weighed against the decomposition of the latent space using the hyperparameter $\beta$. This constraint encourages an efficient latent representation, containing only the required information for reconstruction, hence acting to regularize the latent space. As shown in [31], the $D_{KL}$ term used
with an isotropic unit Gaussian ($p(z) = \mathcal{N}(0, I)$) encourages conditional independence within $z$.

In our approach, as well as that of First Steps [15], a structured latent space is encouraged by backpropagating gradients from the performance predictor’s loss through to the encoder input. However, we hypothesize that useful structuring of this space is inferred from the continuous trajectories used for training, and, in contrast to First Steps, no explicit labeling for each stance is required.

### A. Control Over the Gait Parameters

Once the VAE has been trained as described above, we find that the learned latent space is disentangled. We discover that an oscillation injected into one latent dimension decodes the robot taking steps. Further analysis detailed in Section VI-A finds that adding a second oscillation into the latent space varies the robot’s step length. The former oscillation we denote as the drive signal, and the latter as the trot signal. We find that, in order to control the robot, we only need to inject the drive signal into latent space and can infer the trot signal. By modulating and visualizing the joint-space output from the decoder, we discover that varying the amplitude and phase of the drive signal leads to continuously variable trot locomotion. Therefore, we choose a specific drive signal with features that map directly to the gait parameters we wish to control. These parameters are the robot’s cadence, stance duration, and step height.

While any periodic oscillation (e.g., $\sin$) decodes to robot locomotion, we choose a drive signal featuring specific components allowing for control over the robot’s cadence, swing duration, and step height. The drive signal chosen here is a modified $\sin^3$ oscillation with amplitude $A_k$ and phase $\phi_k$

$$z_{k, d_s} = A_k \sin^3(\phi_k).$$

The amplitude $A_k$ controls the foot-swing height, and the phase $\phi_k$ governs the cadence and support duration.

To control the robot’s swing and stance duration separately, we set the drive-signal’s time period $T_k$ and we employ a stance counter $\epsilon_k$. The time period $T_k$ is equal to the swing duration, while the time that the drive signal is equal to zero is extended by $\epsilon_k$ time steps to introduce a full-support duration of $(\epsilon_k/f_c) \text{ s}$. Hence, once both $T_k$ and $\epsilon_k$ are used together, the phase dynamics are

$$\phi_{k+1} = \begin{cases} 
\phi_k & \text{if } \phi_k \mod \pi = 0 \text{ and } k \epsilon < \epsilon_k \\
\phi_k + 2\pi/T_k & \text{otherwise}
\end{cases}$$

and, in tandem, the counter $\epsilon_k$ is updated as follows:

$$k \epsilon \leftarrow \begin{cases} 
k \epsilon + 1 & \text{if } \phi_k \mod \pi = 0 \text{ and } k \epsilon < \epsilon_k \\
0 & \text{otherwise}.
\end{cases}$$

A $\sin^3(\cdot)$ drive signal is chosen as it is smooth over its domain meaning that the decoded trajectories will also be smooth. Additionally, a $\sin^3(\cdot)$ drive-signal’s gradient is zero at $\phi_k = K\pi$ for $K \in \mathbb{Z}$. Therefore, there is a continuous transition between the $\sin^3(\cdot)$ part of the drive signal and the points where the drive signal is held artificially zero for $\epsilon_k$ control ticks. The resulting signal is shown in Fig. 3 along with the decoded contact schedule and swing trajectory.

### B. Planning for Closed-Loop Control

Once the VAE is trained, it is fast enough to act as a planner in a closed-loop controller. Thus, our approach can react to external disturbances and mitigate against the real-world effects, such as unmodeled dynamics and hardware latency. For closed-loop control, we begin by encoding a history of robot states from the raw sensor measurements to infer the current gait phase. We store a buffer of past robot states and sample from this at $f_{enc}$ to create the encoder’s input.

With an estimate of the current latent variable, we overwrite latent dimension $d_z$ with the drive signal (see Section III-A). Next, we employ a second-order Butterworth filter to smooth the latent trajectory and further smooth the locomotion plan. In essence, the drive signal encourages the decoder to output the next open-loop prediction, while the other latent variables infer the gait phase from the raw sensor input. This process is repeated at the control frequency (400 Hz).

The latent variable $z_k$ and a desired base twist $a_k$ are decoded to produce $X^+_k = g_{dec}(z_k, a_k)$. From this, the joint-space trajectory $Q_k$ and local base velocity $\dot{C}_k$ are extracted and derived or integrated to produce the base and joint positions, velocities, and accelerations. These parameters and the predicted contact schedule are sent to the whole-body controller (WBC) [32]. The WBC solves a hierarchical-optimization problem to calculate the joint torques, which are commands sent to the actuators. The series of constraints enforced by the WBC are contact creation, friction constraints, and torque limits. Next, the WBC applies forward kinematics to the VAEs trajectory to track it in task space. Note that the WBC does not compensate for infeasible plans, i.e., the VAEs’ trajectories, as shown in Section VI-E, are dynamically consistent; otherwise, the robot fails to walk.

### C. Disturbance Detection and Response

Our approach is able to both detect and react to disturbances. The VAE is trained using canonical feasible trajectories. Therefore, any disturbances are characterized as out-of-distribution with respect to the training set. Given the generative nature of our approach, this discrepancy is quantified during operation by...
the trained model via the ELBO [see (4), where $\beta$ is set to 1]. We will show in the evaluation (see Section VI-H) that even a rudimentary response strategy serves to increase the range of disturbance the system can reject.

### IV. IMPLEMENTATION DETAILS

In order to deploy the VAE planner on board the ANYmal quadruped, there are a number of real-world constraints that affect the VAE planner. The first of which is that, in order to deploy the VAE in a real-time control loop, there is a constraint on the VAEs inference time. Second, we discuss how we address the simulation to reality gap. We discuss the VAEs specific architecture, the hyperparameters used to train the model, and finally, specific the specific criteria required for domain transfer.

#### A. Dataset Generation

To train the VAE and create the structured latent space, we require a set of continuous trot trajectories. We restrict these trajectories to quadruped locomotion over flat ground only so that we can study and understand how to generate versatile locomotion utilizing a structured latent space. The training trajectories are generated using DGs [1]. DG is a hierarchical planning and control framework, which is used with a fixed contact schedule and predefined footstep heights. The swing, full-stance durations, and footstep heights set to 0.5 s, 75 ms, and 0.10 m, respectively.

DG is made up of a footstep planner, a base motion planner, and a WBC. The footstep planner computes the next four steps over the gait period using an inverted-pendulum model. The base motion planner solves for the base trajectory over the gait period using a centroidal dynamics model [11]. The latter is constrained with a zero-moment point (ZMP) [10] criterion, the footstep positions, and schedule from the footstep planner. The WBC [32] outlined in Section III-B converts the task-space trajectories to joint feedforward torques, reference positions, and velocities: these are sent to the actuators.

The dataset is generated by uniformly sampling desired base twist and executing DG in the RaSim physics simulator [33]. The fidelity of the simulation is improved by modeling the dynamics of the series-elastic actuators (SEAs) [34] in the ANYmal’s joints using an actuator network [6]. We specifically utilize the network found in [8]. This network takes into account the commanded positions, velocities, feedforward torques, and low-level proportional derivative (PD) gains. The actuator network is essential for good performance as the input response of SEAs depends on a history of states, inputs, and the low-level control law.

#### B. VAE Architecture Details

Fig. 1 outlines the approach’s architecture, and here, we describe the VAEs specific details. Prior to the ablation study in Section VI-C, the VAEs encoder, decoder, and stance performance predictor have two hidden layers and widths of 256 units, using exponential linear unit (ELU) nonlinearities [35]. The encoder input is created using $N = 80$ robot states sampled at 200 Hz—representing a history of 0.4 s—from the encoder input, which is of size 5120 units. The input is compressed via a latent space of 125 units, which is concatenated with an action of 3 units. Next, the decoder outputs the current state and the next $M = 19$ robot states at the control frequency of 400 Hz (preview horizon 47.5 ms and output size: 1216 units). The performance predictor predicts the current foot-in-contact and two future states. Finally, hyperparameters used for training are $\beta = 1.0$ and $\gamma = 0.5$, with a learning rate of $1 \times 10^{-3}$ using the Adam optimizer. Training is terminated after $1 \times 10^6$ gradient steps.

#### C. Domain Transfer

To achieve domain transfer and deployment on the real robot, some modifications are required. The contact forces are artificially set to zero when the robot measures no contact (as determined by a probabilistic contact estimator) [36]. This is necessary since the real-world ANYmal robot measures large contact forces even during swing motion, as these forces are inferred from torque residuals and are affected by model error. In contrast, simulators estimate no contact force during swing. As mentioned in Section III-B, a Butterworth filter with a cutoff frequency of 10 Hz is employed to smooth the latent trajectory. Due to the strict computation budget for real-time control, inference times for the VAE are restricted to at most 1 ms, which imposes constraints on the capacity of the model (see Section VI-C for an ablation of model hyperparameters). Our largest model takes approximately 1 ms for the VAE computation, which is roughly equal to the computation time of the WBC.

#### V. EXPERIMENTAL DESIGN

In this section, we explain how the latent-space properties are discovered and how we assess the performance of the VAE planner deployed on the real robot. In doing so, we motivate the following guiding questions. The aim of these questions is to analyze the latent space once the VAE is trained and to analyze the capabilities of the VAE as a flexible and robust locomotion planner, on a real quadruped (see Fig. 4). We investigate the following:

1) the structure induced in the latent space (see Section VI-A);
2) the effect backpropagating the BCE gradients through the encoder has on the latent-space structure (see Section VI-B);
3) the sensitivity of our approach to variations in key hyperparameters (see Section VI-C);
4) which parts of the encoder’s input are used to infer the robot’s gait phase (see Section VI-D);
5) to what extent the locomotion parameters can be varied online (see Section VI-E);
6) the feasibility of the locomotion plans produced (see Section VI-F);
Fig. 4. Closed-loop control of the real ANYmal quadruped using our V AE planner. This demonstrates a user-controlled variation of gait parameters on-the-fly. Here, colored rectangles represent the full-stance phase, while white space denotes the swing duration. The top row shows a trot gait with an introduced quadrupedal stance phase (gait cycle of 0.75 s – swing 312.5 ms, stance 62.5 ms; a gait cycle consists of a swing phase for each of the leg pairs). Next, the swing duration $T_s$ is reduced using the time-period slider (gait cycle of 0.5 s – swing 188 ms, stance 62.5 ms). The third row illustrates the effect of reducing the stance duration counter $c_s$ to produce trot with reducing full-stance phases (gait cycle of 250 ms – swing 125 ms, stance 0.0 s). Finally, transition into standing occurs when the drive-signal amplitude $A_k$ is reduced to zero.  

7) a comparison between the gait parameters seen during training and those shown in experiments using the V AE planner (see Section VI-G);
8) the degree to which the disturbance detection, coupled with a rudimentary recovery strategy, further increases the robustness of our approach (see Section VI-H);
9) whether the V AE planner can be deployed successfully on the next generation ANYmal C robot without retraining (see Section VI-I).

See the following video for an extended set of experiments along with a brief description of our approach.  

A. Investigating the Latent Space

We wish to examine if there is any structuring in the latent space as well as investigate if any locomotion properties are disentangled within. This knowledge is crucial in understanding how to solve for locomotion trajectories in latent space.

**Latent-Space Structure:** Structure in latent space manifests itself as clustered latent variables. During training, we expect that points in latent space which are of the same gait type will become gathered together. To verify this, we sample a set of random latent variables that we pass through the stance performance predictor. These points are plotted and color coded based upon their predicted stance. The resulting plot can be found in Section VI-A, specifically in Fig. 5.

**Latent-Space Disentanglement:** We wish to see what trajectories in latent space look like and if any of the dimensions within are interpretable. State-space trajectories of the robot trotting from the test set are encoded into latent space, and the subsequent latent-space paths are visualized. These paths in latent space are oscillatory and each oscillation has its own phase.

In order to understand what each oscillation encodes, we artificially inject sine waves into each latent dimension in turn. Decoding these trajectories and visualizing the joint-space paths reveals that the latent space is indeed disentangled, see Section VI-A for full details. This revelation informs how we solve for locomotion paths in latent space.

B. Backpropagating the BCE Through the Encoder

As in First Steps [15], we encourage the latent space to become structured by backpropagating the BCE loss for the contact state of the feet through the encoder. We train an alternative VAE, where we detach the BCE gradients at the latent space. This means that the latent-space structure is not informed by the prediction loss of the feet-in-contact performance predictor. The resulting latent space of the VAE with detached gradients is
compared with the original. First, we look for the crispness of the decision boundaries between stance clusters. Second, we look to see if the decision boundaries are aligned with any dimensions in latent space. This analysis is aimed to investigate the importance of using the contact performance predictor to generate feasible locomotion.

C. Analyzing the Encoder’s Receptive Field

In our experiments on the real robot, the VAE is able to infer the robot’s gait phase from raw sensor inputs. We wish to understand which parts of the encoder’s input are utilized to infer this. We utilize activation maximization [37] to measure the flow of gradients through the encoder. This procedure requires backpropagating from some randomly sampled input encoding until the encoder output matches a predetermined target value. The target is chosen to match key points along the robot’s gait. This analysis is aimed to investigate the importance of using the contact performance predictor to generate feasible locomotion.

D. Studying the VAEs Sensitivity to Key Hyperparameters

The VAEs architecture is ablated and the VAE planner is tested in simulation until the gait phase can no longer be inferred. This leads to noisy and jerky trajectories in simulation. First, the size of the latent dimension is reduced from 125 to a minimum of 6 in strides of 32 units. Second, the number of input states \( N \) is reduced from 80 units to 60, sampled at 200 Hz, meaning that the encoded input history reduces from 0.4 s to 0.3 s. Third, the encoder frequency is halved to 100 Hz with \( N = 80 \). Finally, the widths of the encoder, decoder, and performance predictor are reduced in increments of 32 units from 256 to a minimum of 64.

E. Varying the Gait Parameters Online

We wish to vary the robot’s gait parameters continuously while the robot is walking. This is straightforward for the robot to do. We utilize a simple ROS publisher which sets the drive-signal’s amplitude, time period, and stance duration. The robot’s base twist is commanded independently using the action \( a_k \). In fact, we utilize a navigation waypoint following controller, which produces the action so that the robot walks around a square trajectory in our lab.

F. Analyzing the Dynamic Feasibility of the VAEs Trajectories

The previous guiding question asks how broad the set of maneuvers from the VAE planner are. We now compare the dynamic feasibility of these trajectories to plan from DG. Note that this dynamic feasibility comparison is undertaken as a postexperiment analysis. On the robot, the output of the VAE planner is sent directly to the WBC unaltered.

To evaluate the dynamic feasibility of trajectories synthesized by our VAE, we measure the distance of the ZMP [10] to the support line (i.e., when one pair of legs is swinging) and then compare this distance with that in the synthetic dataset. The ZMP is a commonly used criterion in model-based legged robot control [1], [10]. This analysis is performed for the trajectories in the dataset as a baseline to which we can compare. Dynamically stable trajectories have the ZMP as close as possible to the support line.

G. Measuring the Distribution of Gait Parameters Achievable With the VAE

The gait parameters set during the robot experiments are recorded meaning that we can measure the distribution of visited states. To show the entire range of movements and compare them against those seen during training, we plot a box and whisker plot of the stance and swing durations. This shows the ability of the VAE planner to generalize producing gait parameters not seen in the training distribution.

H. Detecting and Reacting to Disturbances

We wish to analyze the degree to which disturbances can be detected and whether increasing the robot’s cadence helps the robot to recover from them. As mentioned, we monitor the ELBO as this is a measure of the evidence for an encoded input relative to the learned distribution. Hence, a large spike in this value [given the formulation in (4)] is a consequence of a disturbance.

To perturb the robot, we utilize a push broom and disturb the robot’s base. If the ELBO value surpasses a predetermined constant, we characterize the event as a disturbance. The ELBO value is calibrated, as during normal operation, the ELBO remains below this “disturbance” threshold. This is easily found by walking the robot using the VAE planner for 2 min while recording the ELBO.

Following the disturbance detection, the VAE planner automatically reduces the drive-signal’s time period to increase the robot’s cadence. This response is inspired by human locomotion in response to slippage [18]; A group of participants encounters a slippery surface and their reaction is recorded, resulting in an increase in cadence.

We evaluate the effectiveness of increasing the robot’s cadence by repeating the push experiment with and without the cadence increase. The base velocity is utilized to measure the size of the disturbance. Larger velocities arise from bigger pushes. We make a comparison between the reactive VAE planner and a constant cadence version.

In addition, the latent-space trajectory is inspected during and after a disturbance. We are interested in how this is affected by out-of-distribution occurrences, such as a shove from a push broom or a kick to the robot’s base.

I. Deployment on ANYmal C Without Retraining

The VAE, which is trained using simulated ANYmal B locomotion data, is deployed on ANYmal C without retraining. Since our generative model has successfully transferred from simulation to the real robot, a key question is can the VAE

2To view the full range of movements, see the following video https://youtu.be/GT2WLh2Ackc
planner work on a different robot with similar kinematics, but significantly different dynamics? Therefore, we deploy the VAE planner on a different robot, which has a few key differences. First, the ANYmal C has around twice the torque limit of ANYmal B, 80 N·m and 40 N·m, respectively. Second, ANYmal C is significantly heavier than ANYmal B weighing in at 50 kg to B’s 35 kg. Finally, ANYmal C’s actuators have a lower bandwidth due to the additional reduction gearing. The only alterations made to the VAE planner are first the torques are standardized due to the additional reduction gearing. The only alterations made to the VAE planner are first the torques are standardized to reflect the difference in peak torque demand, and second, the WBCs internal dynamics model is updated to ANYmal C.

As previously discussed, the VAE planner can be utilized to measure differences between the learned distribution and the encoded one. Therefore, this approach is able to characterize differences between the two robots. Again, we utilize the ELBO for this, and we record this quantity when the VAE planner is deployed on ANYmal C. The ELBO distribution can be separated in the KL divergence and the reconstruction error. The former provides insight into the distribution of encoded values and the latter acts as a prediction loss.

To verify if the differences in KL divergence collected from both robots are statistically significant, we utilize a Mann–Whitney U-test [38]. Our null hypothesis is that the median of the distribution from ANYmal B is equal to the median of the distribution from ANYmal C. The values of KL divergence are collected from deploying the same VAE planner on both robots. In total, we analyze $2.5 \times 10^4$ values from each robot, which is equivalent to 62.5 s at 400 Hz. The null hypothesis is rejected if the resulting $p$-value is less than our chosen statistical significance value of 0.1%. This procedure is repeated for the two sets of reconstruction errors gathered from each robot.

VI. EXPERIMENTAL RESULTS

Now that we have posed our guiding questions and stated how to answer them, we present the results. These questions can be split up into two categories: introspection of the VAE; and analysis of the VAE as a planner deployed on two real-world platforms. See the following video for a complete set of experiments along with a brief description of our approach: https://youtu.be/GT2WLh2AcKc.

A. Structure Induced in the Latent Space

The latent space is inspected to discover what structure exists and if any locomotion properties are disentangled within.

Latent-Space Structure: Fig. 5 shows samples from the latent space color coded by their predicted stance. For trot, there are four stances: the full-support phase, left front and right hind in contact, another support phase, and finally right front plus left hind in contact. Fig. 5 reveals that the latent space has emerged clustered by stance and that, due to the ordering of these stances, a periodic trajectory decodes to a trot gait. This favorable structure is inferred from the continuous trot trajectory input during training.

The ordering of the clusters matches the sequence of the trot gait. The reason for this is that the encoder’s input contains information about the evolution of the robot’s state over time. This additional information coupled with the training paradigm means that continuous trajectories in latent space match with continuous trajectories in state space. This is in contrast to the work in First Steps [15]. First Steps generates a crawl gait via gradient descent in a structured latent space. However, the input to that model is a static snapshot of the robot’s state with no temporal information (e.g., $x_k$). Since there is no information about the evolution of the robot’s state in the encoder’s input, additional structuring was required to order the clusters in the crawl gait sequence.

Latent-Space Disentanglement: By examining the latent variables, we discover that oscillations injected into just two dimensions in the latent space decode to continuously varying trot trajectories. This result stems from a latent space where variation in footstep length and cadence is aligned along one dimension, while variation in footstep distance lies along another. Specifically, the time period of what we denote the drive-signal oscillation controls the robot’s cadence, while its amplitude is proportionate to the footstep height. In addition, the amplitude of the second signal, which is $\pi/2$ out-of-phase with the drive signal, controls foot-swing length and as such is denoted as the trot signal. Given that other work has tried to explicitly build this structure into locomotion systems [30], it is important to emphasize that this disentanglement emerges in our study as a result of the training paradigm and data. Only the synthetic drive signal needs to be injected into the latent space for closed-loop control; the trot signal is inferred.

Visualizing the Latent-Space Trajectory: We plot the injected drive signal and inferred trot signal in Fig. 5 in red and blue, respectively. When these two signals are plotted against one another, they combine to form a cycle in latent space. This is plotted as the black-vector field in Fig. 5. This cycle is annotated to show how the latent-space trajectory visits
by halving it to 100 Hz, while is reshaped such that each row is the robot depicts a saliency map where a bright color denotes in comparison with the nominal case. S S C 30 P A R R T rotated at roughly way. Note that the decision boundaries in the detached case are prediction, the robot will change contact state in an unpredictable this dimension. In contrast, the detached gradient latent space, meaning that stance classification results mostly from these gradients are detached at the latent space. These boundaries for the stance clustering. These boundaries are aligned perpendicular to the drive-signal dimension in latent space. In the key above the latent-space images, a filled-in circle represents a closed contact and a circular outline denotes an open contact. (a) BCE gradients backpropagated through the encoder. (b) BCE gradients detached at the latent space. each stance cluster in turn. We also show the corresponding contact schedule with images of the robot in their matching configurations.

To begin at point (1), the robot has four feet-in-contact, before moving anticlockwise tracing out a triangular lobe in the red region to point (2). This lobe forms the first foot swing. The trajectory continues anticlockwise through the magenta region (3) before tracing another lobe through the blue area back to point (1). This cycle repeats as the robot takes more steps.

B. Backpropagating the BCE Loss Through the Encoder

We compare the latent space created by backpropagating the BCE gradients through the encoder to the vanilla case, where these gradients are detached before the latent space in Fig. 6. The latent space structured using the BCE gradients exhibits crisp decision boundaries for the stance clustering. These boundaries are aligned perpendicular to the drive-signal dimension in latent space, meaning that stance classification results mostly from this dimension. In contrast, the detached gradient latent space has fuzzy decision boundaries. This is less than ideal as the predicted contact state is an input to the controller and vitally affects the propagation of the dynamics model. If this is a noisy prediction, the robot will change contact state in an unpredictable way. Note that the decision boundaries in the detached case are rotated at roughly 30° in comparison with the nominal case. This results in the latent space being no longer axis aligned with the drive signal. In order to create locomotion trajectories and have independent control over the gait parameters, namely step height and length, the drive signal needs to be rotated to align with the decision boundaries. This is not required for the nominal case, where the BCE gradients are backpropagated through the encoder. Therefore, the crisp decision boundaries and the axis alignment justify our decision—and more crucially highlights the importance—of using the BCE gradients to structure the latent space.

C. Sensitivity to Hyperparameters

As mentioned in Section IV-C, real-time performance is only achievable if VAE inference can be performed under 1 ms. Therefore, we report the results of our ablation study where the VAEs channel capacity is systematically reduced. Table I summarizes the results of this study. First, the latent dimension is reduced from 125 to a minimum of 6. The VAE with latent size of six units is deployed successfully in simulation and on the real robot. Next, the VAEs width is reduced in 32-unit increments and a limit of 128 units is found. This corresponds to a reduction in channel capacity by 52.8%.

Additionally, the window of time used to construct the VAEs input is reduced from 0.4 s to 0.3 s while maintaining an encoder frequency of 200 Hz. This results in poor open-loop performance as the VAE is no longer able to learn the gait phase. This is a result of the swing duration in the dataset, which is 0.45 s. Therefore, an input history over the last 0.3 s is insufficient to capture the gait phase. Next, we investigated reducing the encoder’s sampling frequency f_{enc} by halving it to 100 Hz, while the history remains sampled over 0.4 s. Although this speeds up inference, the resulting trajectories are less smooth than the 200 Hz encoder and the robot is not stable during the closed-loop operation.

As a result of these analyses, the minimum VAE architecture requires a latent space of six units, an input history of 0.4 s sampled at 200 Hz, and a hidden layer size of 128 neurons. We confirm this by deploying this reduced model on the real robot.

D. Analyzing the Encoder’s Receptive Field

It is crucial that the VAE is able to infer the robot’s gait phase from raw sensor input to permit successful latent-space planning. This poses the question “which parts of the encoder input are used to infer this.”

Fig. 7 depicts a saliency map where a bright color denotes the areas of high gradient. These are the areas where the VAEs encoder focuses to infer the gait phase. The left map in Fig. 7 shows the receptive field of the encoder over the entire input X_k. Here, the input X_k is reshaped such that each row is the robot state x_k, see (1). The columns represent specific robot quantities

```
| Latent size | Model width |
|-------------|-------------|
| 6           | F           |
| 29          | -           |
| 61          | -           |
| 125         | -           |
| 64          | F           |
| 96          | F           |
| 128         | T           |
| 192         | T           |
| 256         | T           |
```

Original VAE model denoted as (O) is found in the bottom right-hand corner of this table. The latent-space size is reduced to 6, while the width remains 256. The width is reduced to 64 in units of 32. The smallest model, which transfers to the real robot, has a width of 128 and a latent size of 6.
that we have highlighted in Fig. 7. The areas of high gradient are the contact forces and the most recent joint torques. The left part of Fig. 7 shows the contact forces normal to the ground plane \( \lambda_{z,k} \). The lightest areas in these two subfigures are of the highest gradient meaning that the encoder focuses on these parts in order to infer the gait phase.

Fig. 7. Visualization of the encoder’s receptive field with respect to the gait phase. The encoder input is reshaped such that the robot state quantities, such as joint torques, are in the same column. We also show the contact forces normal to the ground plane \( \lambda_{z,k} \). The lightest areas in these two subfigures are of the highest gradient meaning that the encoder focuses on these parts in order to infer the gait phase.

The results of this are summarized in Fig. 9. The distribution of the ZMP positions is similar for both DG (dataset) and a range of the VAE-planner’s operating modes. Crucially, this remains true despite the maximum swing duration generated by the VAE planner being up to 3.2 times faster than in the training data. We conclude that the representation is good enough to generalize the robot’s dynamics shown here. Empirically, we have further been able to steer the robot with arbitrary and fast-changing input (125 ms) in the third row. Here, the colored contact schedule captures the changes in swing duration as it occurs in real time.

**Stance Duration:** Following a successful reduction in cadence, the stance duration is reduced and the robot transitions into a trot with negligible full-stance phase: \( \epsilon_k = 0 \). Trot gaits with little to no full-support phase are particularly challenging maneuvers for the system in general, as there is reduced control authority to correct for accrued base pose error. During the swing phase of this gait style, only feet across the diagonal are in contact resulting in a line contact limiting the robot’s ability to steady its phase. The transition to a negligible full-stance phase is captured in the third row, where the colored stance duration reduces in length. Note that the full-stance duration does not reduce to 0 s. The mean lowest stance duration is found to be 19.4 ms. We can see this in Fig. 10, which shows the spread of achievable swing and stance durations compared with those in the dataset. The right subplot shows the stance durations, where we see that the bottom whisker of plot does not quite touch zero. The reason for this is that the full-support region in the latent space must be traversed. This part of latent space is highlighted in purple in Fig. 2. It is possible to introduce a nonsmooth drive signal, which jumps this region, but this discontinuous drive signal would introduce a large acceleration into the resulting locomotion trajectory. This might not be trackable by the WBC. Instead, we recommend utilizing a smooth and continuous drive signal.

**Footstep Height:** We vary the amplitude of the drive signal smoothly to zero as is seen in the bottom row (see Fig. 4). The footstep height reduces to zero as the white space in the contact schedule disappears and the robot remains standing. Beyond versatility, e.g., to increase swing heights to overcome irregular ground height, this capability further enables a safe, smooth, and natural transition into and out of the VAE control mode (i.e., to start and come to a halt).

### E. Varying Locomotion Parameters Online

We leverage the disentangled latent space to smoothly transition between gait parameters while the robot is walking. Crucially, cadence, stance duration, and footstep height can be varied during any phase of the gait by modulating the drive-signal’s parameters (see Section III-A). This results in operating modes that vary from those seen during training. Examples of walking motion using the VAE planner on the real robot are shown in Fig. 4 and during disturbance rejection in Fig. 8.

**Swing Duration:** The swing duration is varied over a large operating window on the ANYmal robot. This begins with a swing time period starting at 312.5 ms and is smoothly varied until the swing duration reaches 125 ms, more than doubling the step rate. In parallel, we alter the robot’s heading, demonstrating the independence of the action and the latent-space dynamics. Specifically, the top row of Fig. 4 shows the nominal swing duration of 312.5 ms as the robot turns clockwise, tracking a constant angular velocity command. Following this, we demand a slightly faster swing 188 ms and a constant angular velocity anticlockwise, before transitioning to the fastest swing (237 ms) in the third row.

On the robot, the trajectories from the VAE planner are sent directly to the WBC unaltered. Here, we compare dynamic feasibility of the trajectories, as shown in Section VI-E, after the experiment in order to compare with the training dataset’s distribution generated using DG. The position of the ZMP relative to the support line is utilized as a metric for this comparison. The optimal trajectory will have the ZMP lie on the support line.

The results of this are summarized in Fig. 9. The distribution of the ZMP positions is similar for both DG (dataset) and a range of the VAE-planner’s operating modes. Crucially, this remains true despite the maximum swing duration generated by the VAE planner being up to 3.2 times faster than in the training data. We conclude that the representation is good enough to generalize the robot’s dynamics shown here. Empirically, we have further been able to steer the robot with arbitrary and fast-changing input.
Fig. 8. Push recovery following a shove to the base. A disturbance to the robot causes the ELBO of the form in (4) to rise above a predetermined threshold (red areas). This identifies a disturbance and triggers an increase in cadence from 250 to 125 s as a rudimentary response. This reduction in the swing duration is visible in the contact schedule. To appreciate the results fully, see the following video https://youtu.be/GT2WLh2Ackc.

Fig. 9. Comparison of the distributions of swing and full-stance durations between the dataset used for training and the motions executed on the real robot with the VAE and different drive-signal parameters during the test presented in Section VI-E.

Fig. 10. Distribution of the signed distance between the ZMP and the support line during the two-stance phases (i.e., one pair of legs was in swing). In orange, the distribution with upper and lower bounds (dashed lines) from the dataset, which uses a swing duration of $T_k = 0.5$ s. In blue, the distribution of the VAE trajectories shown over a range of modes deployed on the real robot.

actions for $x$, $y$, and yaw rates, issued from a remote control while being able to interpolate the gait style.

G. Generalization to Different Trot-Gait Styles

The VAE is able to produce a far broader range of different trot styles than those in the dataset. The ranges of swing and stance durations produced by the VAE in Section VI-E and those in the dataset are summarized in Fig. 10. As discussed previously, the dataset is constructed utilizing a predetermined and constant swing duration of 500 ms, with stance duration of 75 ms. The VAE-planner’s swing duration varies between 312 and 125 ms and its stance ranges between 325 ms to negligible duration. This demonstrates that the VAE planner is able to generalize to a variety of trot styles by navigating the structure within the disentangled latent space, even when trained using a very restrictive training set.

H. Disturbance Detection and Recovery

Our approach is able to detect and mitigate disturbances applied to the robot using a heuristic policy that monitors the ELBO loss while the robot is walking. The robot is pushed with a broom resulting in a large spike in the ELBO loss. A spike above a certain threshold is characterized as a disturbance. For our push experiments, we choose the ELBO threshold value to be 11.0. We find this value by walking the robot around the lab using the VAE planner and choosing a value above what is seen during normal operation.

The VAE planner in nominal conditions is able to reject a wide range of impulses applied to the robot’s base. However, this operating window is enlarged by increasing the robot’s cadence as soon as a disturbance is detected. This reactive VAE planner halves the robot’s swing duration from a nominal 250 to 125 ms once the ELBO threshold is surpassed.

We show the VAE planner detecting and reacting to the push broom disturbance in Fig. 8. The top row shows the robot being pushed violently before recovering in three to four steps. At point (b), the ELBO spikes above the threshold indicating the disturbance. Throughout the next 1.5 s, the cadence is halved. This is visible in the contact schedule—the white space reduces in width, see point (c). At points (d) and (e), the robot has fully recovered. The ELBO and contact schedule plots continue on and the robot is pushed twice more in this figure.

A comparison between the reactive VAE planner and a constant cadence version is drawn in Fig. 12. The constant cadence VAE planner uses swing durations of 313 and 250 ms, while the reactive version halves the robot’s cadence to 125 ms as described. Here, we see that the range of rejectable push disturbances increases as the swing duration decreases.

Finally, we illustrate what happens to the latent-space trajectory during and after the disturbance. The nominal latent-space
We successfully deploy the VAE planner trained using simulated ANYmal B data on ANYmal C. Note that the VAE is not retrained. The differences between the two robots are significant as ANYmal C’s joint torque limits are 80.0 N·m instead of 40.0 N·m, and ANYmal C’s total mass is 50 kg instead of 35 kg. Also, note that the WBCs internal model for both kinematics and dynamics is updated when the VAE planner is deployed on ANYmal C. However, the joint trajectory output from the VAE planner is the input to the WBC, meaning that joint trajectories from the VAE planner are suitable for ANYmal C.

Despite not retraining, the VAE planner is able to command ANYmal C’s heading and control the robot’s gait parameters. In essence, the experiment in Section VI-E is repeated with the same VAE planner, but deployed on ANYmal C. Fig. 13 shows ANYmal C with the VAE-planner deployed onboard, along with the contact schedule. The latter shows the variation in both swing and stance duration as the robot walks.

Comparing ANYmal B and C Using ELBO: The ELBO can be used to further compare the differences between encoding the raw ANYmal B and C data. As mentioned in Section V-I, key parameters, such as joint torque limits, are standardized before encoding. Analysis of the ELBO is split into a comparison of the KL divergence and mean-squared reconstruction error between ANYmal B and ANYmal C.

The KL-divergence term is particularly affected by latency in the input. For example, if the states in the encoder input are not sampled exactly at the encoder frequency, the KL-divergence term will increase. The distributions of these values are quite different between each robot. The KL-divergence values for ANYmal B show a steady mean drift upward, which is negligible for ANYmal C. The median value for ANYmal C is also noticeably lower than for ANYmal B.

As described in Section V-I, we compare the KL-divergence values from both robots together using a Mann–Whitney U-test [38]. As a reminder, we record the KL divergence for both robots and analyze the sets of size $2.5 \times 10^5$. The resulting U-statistic is $1.82 \times 10^9$ and the $p$-value is $p < 0.001$. Therefore, the null hypothesis is rejected and the difference in the median values of the KL divergence from both robots is statistically significant.

The reason for ANYmal C’s lower median value is down to the relative performance of the onboard computers on the two ANYmals. ANYmal C has a much faster CPU, meaning that the VAE-planner’s control loop is comfortably under 2.5 ms required for real-time control. The result of which is the encoder’s input is sampled at the desired encoder frequency. In contrast, on average, the control loop on ANYmal B violates the real-time control law roughly 20% of the time during the experimental run in Fig. 10. This introduces latency into the control loop, which causes the increase in KL divergence.

The reconstruction error for ANYmal B is, unsurprisingly, lower than for ANYmal C. The median value for ANYmal
The reconstruction error is $5.74 \times 10^{-3}$, while for ANYmal C, it is $1.58 \times 10^{-2}$. These median values are compared using the Mann–Whitney U-test conducted over two sets containing $2.5 \times 10^4$ values. The U-statistic is $8.26 \times 10^1$ resulting in a $p$-value of $p < 0.001$. This results in the rejection of the null hypothesis, meaning that the two distributions are statistically different. Since the VAE planner is trained using data from the ANYmal B, a lower reconstruction error is expected.

VII. FUTURE WORK

The most significant limitation of the current system is that, despite the VAE planner being able to interpolate between a broad range of dynamic maneuvers, the resulting locomotion gaits are constrained to follow the trot contact sequence and operate on flat ground. This stems from the decision to only train the model using demonstrations of the trot gait with fixed parameters and on flat ground. These decisions were made so that we could inspect and interpret the resulting latent space thoroughly in this restrictive domain.

To generate different contact schedules other than trot, it is necessary to train a new VAE with different gait trajectories. These different contact schedules should similarly become embedded in latent space. This provides an opportunity to train the VAE with even more DGs, such as bound as well as asymmetrical gaits.

Another limitation is that the current VAE planner has not been exposed to the trajectories required in order to traverse uneven terrain. Therefore, we wish to train the model on a variety of different locomotion gaits generated when the quadruped walks in unstructured environments. The resulting latent space will be analyzed and the VAE deployed as a planner so that the quadruped can operate in these environments. This will discover first how well the VAE planner reacts to external disturbances resulting from uneven terrain, and second, the range of motion the VAE planner can produce in response to the unstructured terrain.

VIII. CONCLUSION

In this article, we presented a robust and flexible approach for locomotion planning from the perspective of traversing a structured latent space. This was achieved by utilizing a deep generative model to capture relevant structure from locomotion data and enabled the detection and mitigation of disturbances. The latent space was disentangled to a degree where key salient locomotion features were automatically discovered from a single style of trot gait. An investigation of this latent space revealed a two-dimensional representation that encapsulates the underlying dynamics of the system. This disentanglement was exploited using a drive signal with which dynamically consistent locomotion was generated. Crucially, the amplitude and phase of the drive signal directly control the gait characteristics, namely the cadence, swing height, and full-support duration. Once deployed, the ease with which modulation of the drive signal gives rise to seamless interpolation between gait parameters is demonstrated. Despite generalizing to remarkably distinct trot styles compared with the training distribution, the entire range of VAE trajectories remains dynamically consistent. Additionally, utilizing a generative model affords the ability to characterize disturbances as out-of-the-distribution seen during training. Although the VAE planner is able to reject a broad range of impulses applied to the robot’s base, this window is broadened by increasing the cadence as soon as the disturbance is detected. Finally, we showed that the approach readily transferred to a kinematically similar but dynamically different platform without needing to be retrained. This helps to showcase the VAEs ability transfer to new unseen domains.

ACKNOWLEDGMENT

The authors would like to thank O. Groth for useful discussions and the use of the SCAN facility.

REFERENCES

[1] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, “Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots,” *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2261–2268, Jul. 2018.
[2] C. Mastalli et al., “A feasibility-driven approach to control-limited DDP,” *Auton. Robots*, vol. 46, pp. 1–21, 2022.
[3] O. Melon, R. Orsolino, D. Surovik, M. Geisert, I. Havoutis, and M. Fallon, “Receding-horizon perception trajectory optimization for dynamic legged locomotion with learned initialization,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 9805–9811.
[4] 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 Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1560–1567, Jul. 2018.
[5] C. Mastalli et al., “Agile maneuvers in legged robots: A predictive control approach,” 2022. [Online]. Available: https://arxiv.org/abs/2203.07554
MITCHELL et al.: VAE-LOC0: VERSATILE QUADRUPED LOCOMOTION BY LEARNING A DISENTANGLED GAIT REPRESENTATION 3819

[6] J. Hwangbo et al., “Learning agile and dynamic motor skills for legged robots,” Sci. Robot., vol. 4, no. 26, pp. 5872–5885, 2019.

[7] S. Gangapurwala, A. Mitchell, and I. Havoutis, “Guided constrained policy optimization for dynamic quadrupedal robot locomotion,” IEEE Robot. Autom. Lett., vol. 5, no. 2, pp. 3642–3649, Apr. 2020.

[8] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, “RLOC: Terrain-aware legged locomotion using reinforcement learning and optimal control,” IEEE Trans. Robot., vol. 38, no. 5, pp. 2908–2927, Oct. 2022.

[9] A. W. Winkler, F. Farshidian, D. Pardo, M. Neunert, and J. Buchli, “Fast trajectory optimization for legged robots using vertex-based ZMP constraints,” IEEE Robot. Autom. Lett., vol. 4, no. 4, pp. 2201–2208, Oct. 2019.

[10] M. Vukobratovic and B. A. Borovac, “Zero–moment point—Thirty five years of its life,” Int. J. Humanoid Robot. Res., vol. 1, pp. 157–173, 2004.

[11] D. E. Orin, A. Goswami, and S.-H. Lee, “Centroidal dynamics of a humanoid robot,” Auton. Robots, vol. 35, pp. 161–176, 2013.

[12] N. Rathod et al., “Model predictive control with environment adaptation for legged locomotion,” IEEE Access, vol. 9, pp. 145710–145727, 2021.

[13] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, “An efficient optimal planning and control framework for quadruped locomotion,” in Proc. IEEE Int. Conf. Robot. Autom., 2017, pp. 93–100.

[14] C. Mastalli et al., “Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion,” in Proc. IEEE Int. Conf. Robot. Autom., 2017, pp. 1096–1103.

[15] A. L. Mitchell et al., “First steps: Latent-space control with semantic constraints for quadrupedal locomotion,” in Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., 2020, pp. 5343–5350.

[16] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” in Proc. Int. Conf. Learn. Representations, 2014, pp. 1–13.

[17] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” in Proc. 31st Int. Conf. Mach. Learn., 2014.

[18] B. E. Moyer, A. J. Chambers, M. S. Redfern, and R. Cham, “Gait parameters as predictors of slip severity in younger and older adults,” Ergonomics, vol. 49, pp. 329–343, 2006, pp. 1278–1286.

[19] A. L. Mitchell et al., “Next steps: Learning a disentangled gait representation for versatile quadruped locomotion,” in Proc. IEEE Int. Conf. Robot. Autom., 2022, pp. 10564–10570.

[20] R. Grandia, A. Taylor, A. D. Ames, and M. Hutter, “Multi-layered safety for legged robots via control barrier functions and model predictive control,” in Proc. IEEE Int. Conf. Robot. Autom., 2021, pp. 8352–8358.

[21] G. Bledt and S. Kim, “Extracting legged locomotion heuristics with regularized predictive control,” in Proc. IEEE Int. Conf. Robot. Autom., 2020, pp. 406–412.

[22] A. Karl, M. Soecih, J. Bayer, and P. van der Smagt, “Deep variational Bayes filters: Unsupervised learning of state space models from raw data,” in Proc. Int. Conf. Learn. Representations, 2016, pp. 1–13.

[23] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller, “Embed to control: A locally linear latent dynamics model for control from raw images,” in Proc. 28th Int. Conf. Neural Inf. Process. Syst., 2015, pp. 2746–2754.

[24] M. A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, “Universal planning networks,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 4732–4741.

[25] D. Hafer et al., “Learning latent dynamics for planning from pixels,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 2555–2565.

[26] T. Li, R. Calandra, D. Pathak, Y. Tian, F. Meier, and A. Rai, “Planning in learned latent action spaces for generalizable legged locomotion,” IEEE Robot. Autom. Lett., vol. 6, no. 2, pp. 2682–2689, Apr. 2021.

[27] H. Y. Ling, F. Zinno, G. Cheng, and M. van de Panne, “Character controllers using motion VAEs,” ACM Trans. Graph., vol. 39, 2020, Art. no. 40.

[28] A. Agrawal, S. Chen, A. Rai, and K. Sreenath, “Vision-augmented quadrupedal locomotion on discrete terrain using motion libraries,” in Proc. IEEE Int. Conf. Robot. Autom., 2022, pp. 4708–4714.

[29] Y. Yang, T. Zhang, E. Courmans, J. Tan, and B. Boots, “Fast and efficient locomotion via learned gait transitions,” in Proc. Conf. Robot. Learn., 2021, pp. 773–783.

[30] I. Higgins et al., “β-VAE: Learning basic visual concepts with a constrained variational framework,” in Proc. Int. Conf. Learn. Representations, 2017, pp. 1–12.

[31] C. D. Bellisco, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, “Dynamic locomotion and whole-body control for quadrupedal robots,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2017, pp. 3359–3365.

[32] J. Hwangbo, J. Lee, and M. Hutter, “Per-contact iteration method for solving contact dynamics,” IEEE Robot. Autom. Lett., vol. 3, no. 2, pp. 895–902, Apr. 2018.

[33] G. A. Pratt and M. M. Williamson, “Series elastic actuators,” in Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., 1995, pp. 399–406.

[34] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (ELUs),” in Proc. Int. Conf. Learn. Representations, 2016, pp. 1–22.

[35] M. Hutter et al., “ANYmal—A highly mobile and dynamic quadrupedal robot,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2016, pp. 38–44.

[36] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, “Visualizing higher-layer features of a deep network,” in Proc. Int. Conf. Mach. Learn. Workshop. Feature Hierarchies, Montréal, QC, Canada, 2009, pp. 1–13.

[37] H. B. Mann and D. R. Whitney, “On a test of whether one of two random variables is stochastically larger than the other,” Ann. Math. Statist., vol. 18, no. 1, pp. 50–60, 1947.

[38] M. Mistry, J. Buchli, and S. Schaal, “Inverse dynamics control of floating base systems using orthogonal decomposition,” in Proc. IEEE Int. Conf. Robot. Autom., 2010, pp. 3406–3412.

Alexander Luis Mitchell (Graduate Student Member, IEEE) received the M.Eng. degree in information engineering (control theory) from the University of Oxford, Oxford, U.K., in 2018.

Subsequently, he enrolled with the University of Oxford as a doctoral candidate. He is currently co-supervised by Professor Ingmar Posner and Dr. Ioannis Havoutis with Oxford Robotics Institute, Oxford. His research interests include optimal control and learning-based methods for path planning and control of legged robots.

Wolfgang Xaver Merkt (Member, IEEE) received the B.Eng. (Hons.) degree in mechanical engineering with management and the M.Sc.(R) and Ph.D. degrees in robotics and autonomous systems from the University of Edinburgh, Edinburgh, U.K., in 2014, 2015, and 2019, respectively.

He is currently a Senior Researcher with Oxford Robotics Institute, University of Oxford, Oxford, U.K., and with Ioannis Havoutis. During his Ph.D., he worked on trajectory optimization and warm starting optimal control for high-dimensional systems and humanoid robots under the supervision of S. Vijayakumar. His research interests include fast learning- and optimization-based methods for planning and control, locomotion planning, and legged robots.

Mathieu Geisert received the M.Eng. degree in aerospace from Institut Supérieur de l’Aéronautique et de l’Espace SUPAERO, Toulouse, France, in 2013, and the Ph.D. degree in robotics from LAAS-CNRS, Toulouse, France, in 2018.

He then joined the humanoid robotics team Gepetto with LAAS-CNRS, Toulouse, France, where he worked for five years in different positions. From 2018 to 2021, he worked on quadruped robots as a Postdoctoral Researcher with Dynamic Robot Systems Group, Oxford Robotics Institute, University of Oxford. Since 2021, he has been working as a controls Research Engineer with Arrival Ltd., Howald, Luxembourg. His research focuses on control and learning for legged robots locomotion.
Siddhant Gangapurwala received the B.E. degree in electronics from the University of Mumbai, Mumbai, India, in 2016. In 2017, he joined the AIMS program as a doctoral candidate with the University of Oxford. In 2018, he joined the Dynamic Robot Systems (DRS) Group of Oxford Robotics Institute to pursue his D.Phil. degree with focus on machine learning and optimal control-based approaches for robotic locomotion over uneven terrain. He continues to work with DRS as a Postdoctoral Researcher on locomotion, manipulation, and locomanipulation.

Martin Engelcke received the M.Eng. degree in information engineering and D.Phil. (Ph.D.) degree in generative modelling for scene understanding from the University of Oxford, Oxford, U.K., in 2015 and 2020, respectively. He conducted his doctoral and subsequent postdoctoral research between 2015 and 2021 with Oxford Robotics Institute, under the supervision of Prof. Ingmar Posner. He is currently a Research Scientist with DeepMind. His research interests include generative models, representation learning, and reinforcement learning.

Oiwi Parker Jones received the D.Phil. degree in natural language processing from the University of Oxford, Oxford, U.K., in 2010. He went on to do postdoctoral work in language neuroscience, computational modeling, and the development of neuroimaging methods for large datasets with University College London and University of Oxford. He is currently a Senior Postdoctor with Applied AI Lab, Oxford Robotics Institute, Honorary Fellow of the Nuffield Department of Clinical Neurosciences, and Hugh Price Fellow in computer science with Jesus College, University of Oxford.

Ioannis Havoutis (Member, IEEE) received the M.Sc. degree in artificial intelligence and the Ph.D. degree in informatics from the University of Edinburgh, Edinburgh, U.K., in 2007 and 2011, respectively. He is a Lecturer in robotics with the University of Oxford, Oxford, U.K. He is a part of Oxford Robotics Institute and a co-lead of the Dynamic Robot Systems Group. His focus is on approaches for dynamic whole-body motion planning and control for legged robots in challenging domains. From 2015 to 2017, he was a Postdoctor with Robot Learning and Interaction Group, Idiap Research Institute. From 2011 to 2015, he was a Senior Postdoctor with Dynamic Legged System Lab, Istituto Italiano di Tecnologia.

Ingmar Posner received the M.Eng. degree in electronic systems engineering from Aston University, Birmingham, U.K., in 2001, and the D.Phil. degree in bioacoustics from the University of Oxford, Oxford, U.K., in 2005. He is a Professor of engineering science with Oxford University, Oxford, where he leads Applied Artificial Intelligence Lab. His research aims to enable machines to robustly act and interact in the real world—for, with, and alongside humans. His track record in machine perception and decision making includes seminal work on large-scale learning from demonstration, representation learning for scene understanding and prediction as well as 3-D object detection and machine introspection. His current research interests revolve around the use of structured latent spaces for robot perception, planning, and control. He is the founding Director of Oxford Robotics Institute and, in 2014, cofounded Oxbotica, multiaward winning provider of mobile autonomy software solutions.