First Steps: Latent-Space Control with Semantic Constraints for Quadruped Locomotion

Alexander L. Mitchell[1,2], Martin Engelcke[1], Oiwi Parker Jones[1], David Surovik[2], Ioannis Havoutis[2], and Ingmar Posner[1]

***Abstract*— Traditional approaches to quadruped control frequently employ simplified, hand-derived models. This significantly reduces the capability of the robot since its effective kinematic range is curtailed. In addition, kinodynamic constraints are often non-differentiable and difficult to implement in an optimisation approach. In this work, these challenges are addressed by framing quadruped control as optimisation in a structured latent space. A deep generative model captures a statistical representation of feasible joint configurations, whilst complex dynamic and terminal constraints are expressed via high-level, semantic indicators and represented by learned classifiers operating upon the latent space. As a consequence, complex constraints are rendered differentiable and evaluated an order of magnitude faster than analytical approaches. We validate the feasibility of locomotion trajectories optimised using our approach both in simulation and on a real-world ANYmal quadruped. Our results demonstrate that this approach is capable of generating smooth and realisable trajectories. To the best of our knowledge, this is the first time latent space control has been successfully applied to a complex, real robot platform.**

I. Introduction

Four-legged robots (quadrupeds) are capable of traversing a broader range of terrains than wheeled robots and can carry larger payloads with longer battery lives than drones ([1], [2], [3], [4]). Unlike wheeled robots, legged robots can choose to place their feet on specific locations within the terrain which can support the mass of the robot. This makes them highly suitable for inspection and monitoring tasks in unstructured domains ([2], [4], [5]). However, this flexibility comes at the cost of platform complexity. Kinematics and dynamics are considerably more complicated for quadrupeds than for wheeled robots or drones. This makes trajectory optimisation, i.e. computing feasible paths for robot end-effectors and for the robot's centre of mass (CoM) given kinematic, dynamic, and environmental constraints, one of the central challenges in their deployment ([6], [7]).

Traditionally, trajectory optimisation for quadrupeds is solved using constrained optimisation. However, the robots feasible joint space and dynamics such as stability, torque limits, and contact forces, require complex often non-differentiable constraints. This makes the optimisation intractable. A typical approach, therefore, is to use approximate, hand-derived dynamic models and arbitrarily reduce the kinematic range of the robot [6]. These linear approximations typically over-simplify the problem and limit platform capabilities, leading to overly narrow convergence basins given feasible initial states (e.g. [6], [8], [9]). As a result, trajectories solved using linear approximations are only suitable for specific use-cases and do not easily generalise to novel situations. In contrast, approaches which account for the full kinodynamic representation of the robot require iterative, sequential optimisations to check for constraint satisfaction [10]. This often makes them computationally infeasible for real-time deployment.

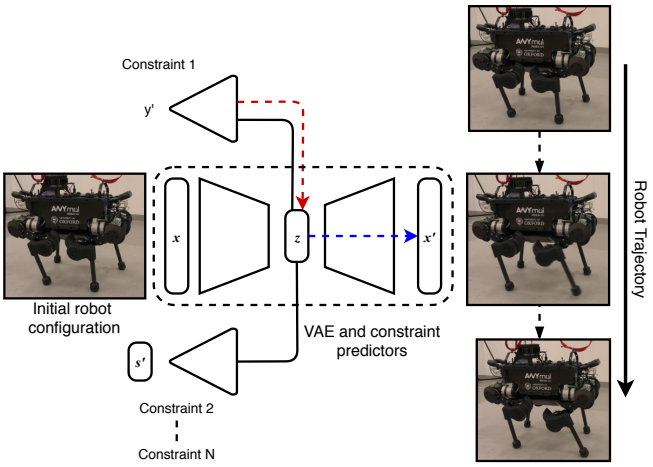

Fig. 1: A VAE encodes the robot state and captures correlations therein in a structured latent space. Once trained, performance predictors (triangles) attached to the latent space predict if constraints are satisfied and apply arbitrarily complex constraints such as robot stability. The flexibility of this approach allows for the application of any number of constraints by performing activation maximisation of a loss summed over all active constraints (see, for example, Eq. 8). Decoding positions in latent space (blue) along the optimisation trajectory translates to movement of the robot.

Here, we propose a radically different approach to quadruped control. Using a generative model of the robot state we perform trajectory optimisation by directly optimising the position in a structured latent space, which captures a statistical model of the robots feasible joint-space (see, Fig. 1). In particular, we employ a *variational autoencoder (VAE)* ([11], [12]) to encode the robot state including joint positions. Inspired by [13], operational constraints are induced via *high-level*, semantic indicators and represented by learned performance predictors operating directly in the latent space. Arbitrarily complex constraints – and goals – can thus be enforced by performing gradient-based optimisation

[1]Applied AI Lab (A2I), [2]Dynamic Robot Systems (DRS)
Oxford Robotics Institute (ORI), University of Oxford
Correspondence to: mitch@robots.ox.ac.uk

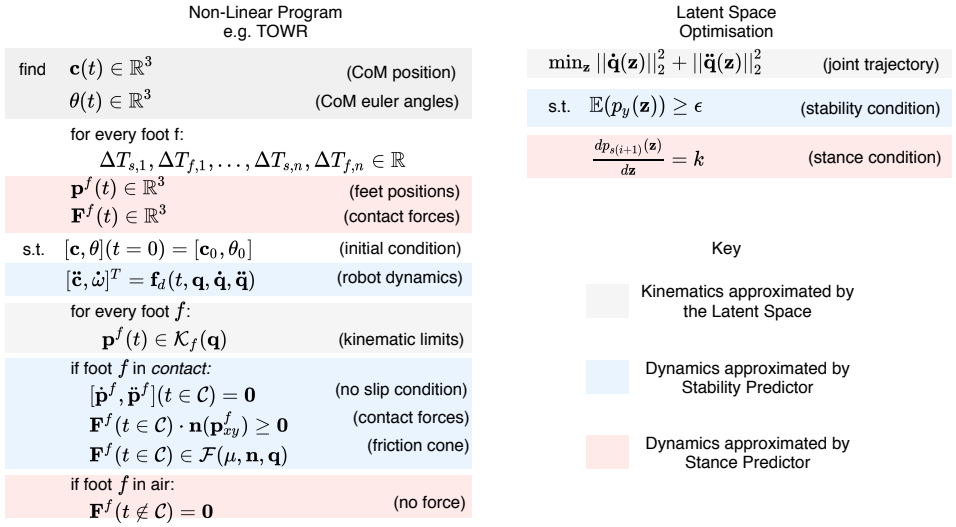

Fig. 2: Conceptual comparison of solving a non-linear program (NLP) such as TOWR with latent space optimisation. The NLP splits locomotion into an optimisation subject to a series of constraints, whilst trajectories in our approach are solved via gradient descent in a structured latent space. Complex constraints such as stability are enforced using learned classifiers (performance predictors) and gradients from differentiating a target loss are used to update a trajectory in latent space. Performance predictors replace multiple and often non-differentiable dynamics exhibited in the NLP.

in the latent space driven by the performance predictors via activation maximisation [14].

By implicitly capturing a full kinodynamic model, our approach enables *efficient* trajectory optimisation without relying on hand-specified dynamics models or linear approximations. In contrast to prior art, it predicts whether constraints are satisfied with a single pass through our neural network. More importantly, it enables direct optimisation of traditionally non-differentiable dynamics (see, Fig. 2) together with the correlations captured by the latent space, and this results in inherently smooth, feasible robot trajectories when decoding poses into state-space along the optimised path in latent space.

We validate our approach to latent-space control in the context of quadruped locomotion and constrain robot stability and end-effector contact dynamics to obtain a walking gait (Fig. 1). Using both simulated and real robot experiments, we demonstrate that our approach to latent-space control is able to find trajectories from unstable to stable robot configurations as well as to execute realisable gait cycles. This is achieved with execution times almost an order of magnitude faster than a comparable analytical approach [15].

To the best of our knowledge this is the first approach to latent-space control for a complex dynamic system which is validated both in simulation and on a real platform.

II. Related Work

As torque controlled robots have become more complex, the weaknesses of traditional trajectory optimisation approaches have become more pronounced. These mostly stem from large configuration-spaces and the complexity of model dynamics. Hand-derived dynamics models such as centroidal dynamics [16] can be used as dynamic constraints ([7], [6]). For example, *TOWR* [6] is a general approach for solving locomotion tasks for any legged robot over known terrain. Specifically, TOWR solves a non-linear program for the centre of mass and feet trajectories whilst optimising the contact forces. This is computationally slow and prohibits online updates to the current plan to react to environmental changes. Furthermore, kinematic constraints restrict the range of the robot's movement. While TOWR is suitable for highly-dynamic locomotion, our approach is conceived to be sufficiently general for low-velocity operation such as locomotion over rough terrain.

[15] estimates robot stability by finding a "margin of stability" while taking into account the robot's centre of mass, friction forces, and torque limits. We use this formulation to create training data for our stability predictor and as a performance baseline. While the original approach requires sampling and evaluating multiple points, our method directly optimises for stable trajectories using a learned stability predictor.

Reinforcement learning (RL) (e.g. [17], [18]) is a promising alternative to optimisation-based approaches. However, RL policies are difficult to train, requiring vast quantities of data, and suffer from unstable gradient estimates. Once trained, specific constraints such as foothold placements are impossible to enforce. In contrast, our approach is flexible: additional constraints are applied by constructing an appropriate loss function summed over all active constraints, which is differentiated to provide a gradient update.

Recent advances in latent-space optimisation also attempt to overcome the limitations of traditional approaches. Universal Planning Networks (UPN) [19] use a gradient-based method in a structured latent space to find trajectories between an initial and final image. Similarly, Embed to Control (E2C) [20] performs system identification and state

estimation from visual inputs for control problems. In addition, a forward dynamics model is learned to traverse the latent space given an input. Both UPN and E2C need to be provided with a goal condition. In contrast, performance predictors, presented here, are capable of predicting if arbitrarily complex constraints, such as robot stability, are satisfied. A target loss is evaluated using these performance predictors and differentiated until the latent variable is guided into the terminal set via gradient descent.

III. Gradient-Based Control in Latent Space with Performance Predictors

A. Learning a Latent Representation of Robot States

VAEs learn compact and smooth representations of high-dimensional data ([11], [12]). We utilise a VAE to encode the robot's joint angles $\mathbf{q}$, feet positions in the base frame $\mathbf{p}^f$ (where $f \in \mathbb{N}$ is the number of feet), joint torque τ, contact forces λ, and the gravity body force vector $\mathbf{g}$, forming the input to the VAE $\mathbf{x} = [\mathbf{q}, \mathbf{p}^f, \tau, \lambda, \mathbf{g}]^T$, $\mathbf{x} \in \mathbb{R}^{51}$. To find these quantities, *static*, snapshots of the robot in random, but feasible joint configurations are sampled under the condition that joint velocities and accelerations are set to zero. Finally, the VAE is trained to reconstruct the input $\mathbf{x}$ via a latent space denoted $\mathbf{z} \in \mathbb{R}^{N_z}$. Training is achieved by minimising the KL-regularised mean-squared error of the reconstruction:

$$\mathscr{L}_{VAE} = MSE(\mathbf{x}, \mathbf{x}') + \beta D_{KL}[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] \tag{1}$$

Given the importance of robot stability in general and information about its current stance for the purpose of locomotion in particular, the VAE is trained jointly with two *performance predictors* [13] which estimate the probability of the robot being stable y' and the probability that it is currently in a specified stance s_i' using a binary cross-entropy (BCE) loss. Gradients from these two classifiers are backpropagated through the VAE encoder and aid in shaping the latent representation. Given two hyperparameters μ_1 and μ_2 for weighting the loss terms, the full training objective can thus be written as:

$$\mathscr{L} = \mathscr{L}_{VAE} + \mu_1 BCE(y, y') + \mu_2 BCE(s_i, s_i') \tag{2}$$

This provides an inductive bias to the model, which structures our latent space so that consecutive stance clusters are adjacent in the latent space (see, Fig. 3). This structure is achieved by the design of the stance labels. Stances in the walk gait where four feet are on the ground are one-hot encoded, whilst intermediary classes with a foot is in the air are two-hot encoded. Therefore, regions exist in latent space where two classifiers are active and, subsequently, consecutive stance clusters are encoded in sequence in the latent space.

B. Constrained Latent-Space Traversal

The performance predictors estimate whether non-linear and often non-differentiable kinodynamic constraints, as well as goal conditions are satisfied. Importantly, formulating these constraints as neural network classifiers renders them

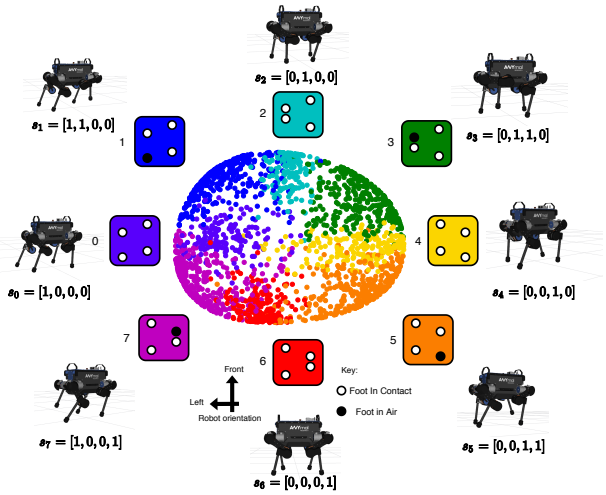

Fig. 3: The latent space (central panel) is structured and clearly shows the stance clusters (coloured). For visualisation, principle component analysis (PCA) [21] with a radial basis function (RBF) kernel reduces the latent space from 64 dimensions to two. Walking is achieved by cycling clockwise through the eight stances, which are represented by coloured diagrams orientated so that forward is up the page and left aligns with reader's left. Open white circles represent the target position of feet in contact with the ground; closed black circles represent feet in the air.

differentiable with respect to the latent variable $\mathbf{z}$. This allows us to update a latent variable sequentially via gradient descent until a constraint is satisfied. This is performed either to steer a trajectory to satisfy a terminal constraint or to enforce a condition upon the entire trajectory. Using gradient-descent to update a latent variable while a performance predictor enforces a particular constraint is a use of *activation maximisation* (AM) [14].

The general case for *activation maximisation* requires one or more differentiable loss functions $\mathscr{L}_k(y, y')$, where y is the target value and y' is the predicted value. Each loss function has an associated step size α_k, which scales the respective gradient step. Finally, an update to the latent variable $\mathbf{z}$ is made via gradient descent in the latent space:

$$\mathbf{z} \leftarrow \mathbf{z} - \nabla \sum_k \alpha_k \mathscr{L}_k(y, y') \tag{3}$$

This approach is inherently flexible as additional constraints are applied simply by adding additional loss functions prior to a gradient update.

C. Activation maximisation for stability

A specific use of constrained latent-space traversal in the context of quadruped control is disturbance rejection. Performance predictors are not only capable of predicting if the robot has become unstable, but also finding smooth and stabilising trajectories by continuously backpropagating the error from a performance predictor back into the model's latent representation $\mathbf{z}$ using activation maximisation (AM). Each latent update $\mathbf{z}$ is decoded to state space $\mathbf{x}'$ to create a trajectory. By choosing to minimise the BCE loss, the

predicted probability that the robot is stable y' is driven close to the desired probability y. Once a step size α_y is selected, AM for robot stability is defined:

$$\mathbf{z} \leftarrow \mathbf{z} - \nabla \alpha_y BCE(y, y') \tag{4}$$

The resulting trajectories are evaluated for their efficacy and feasibility in section IV.

D. Optimisation for Locomotion

Locomotion requires finding a trajectory which allows the quadruped to take a series of steps through a gait sequence while the robot remains stable. The robot is stable when the feet in contact support the CoM without slipping while respecting maximum torque limits. To solve for a walk trajectory in latent space, two performance predictors are required: one for stability and another to move the robot between gait stances (see Fig. 3).

To frame locomotion as an optimisation problem, we minimise the joint velocities $\dot{\mathbf{q}}$ and accelerations $\ddot{\mathbf{q}}$ over an entire trajectory. The joint velocities and accelerations are calculated from the joint angles using a forward-Euler derivative estimation. Intuitively, minimising these values will result in smoother, more readily executable trajectories as the actuators perform best when the input is varied smoothly [22]. Therefore, we solve for a trajectory that minimises the squared L_2-norm of the joint velocity and acceleration of the robot, whilst maximising the robot's stability y and linearly increasing the probability of being in the next stance s_{i+1}. Trajectories are of length N, which is N/f_s seconds long, where f_s is the sampling frequency. The initial robot configuration $\mathbf{x}_0$ is encoded into the latent space to find $\mathbf{z}_0$. N repeats of $\mathbf{z}_0$ are stacked to create $\mathbf{Z} \in \mathbb{R}^{N,N_z}$, and this forms the latent-space trajectory. Concretely, we perform

$$\mathbf{Z}^* = \arg\min_{\mathbf{Z}} \quad \left(||\dot{\mathbf{q}}(\mathbf{Z})||_2^2 + ||\ddot{\mathbf{q}}(\mathbf{Z})||_2^2 \right) \tag{5}$$

$$s.t. \quad \mathbb{E}(p_y(\mathbf{Z})) \geq \varepsilon \tag{6}$$

$$\frac{dp_{s(i+1)}(\mathbf{Z})}{d\mathbf{Z}} = k \tag{7}$$

The constraints in Equations 6 and 7 are converted to costs using Lagrange multipliers λ_i and added to the objective function in Equation 5 to form the locomotion loss

$$\mathscr{L}_{loco} = ||\dot{\mathbf{q}}||_2^2 + ||\ddot{\mathbf{q}}||_2^2 + \lambda_0 BCE(y, y') + \lambda_1 BCE(s, s') \tag{8}$$

$\mathbf{Z}$ is passed through the VAE's decoder to obtain the reconstructed $\mathbf{X}$, and also through the performance predictors to provide stability ($\mathbf{Y}$) and stance ($\mathbf{S}$) estimates. This output is concatenated into $\mathbf{W} = [\mathbf{X}, \mathbf{Y}, \mathbf{S}]$ and multiplied with a selector matrix to compute the quantities needed to evaluate the locomotion loss $\mathscr{L}_{loco}$. $\mathscr{L}_{loco}$ is summed over the entire trajectory and multiplied with a step size α_{loco} before being differentiated with respect to $\mathbf{Z}$ to provide a gradient update such that

$$\mathbf{Z} \leftarrow \mathbf{Z} - \nabla_{\mathbf{Z}} \alpha_{loco} \mathscr{L}_{loco}(\mathbf{Z}) \tag{9}$$

IV. Experiments

We apply constrained latent-space traversal to solve complex planning tasks on a quadruped robot. In particular, we focus on stability and locomotion. Our evaluation first investigates whether our approach to latent-space control is able to provide feasible trajectories when rapid robot stabilisation is required. This is compared to a baseline of performing activation maximisation by backpropagating gradients directly into the input of a feed-forward classifier rather than a structured latent encoding. Secondly, we add a constraint on the robot stance to enable it to take steps while remaining stable throughout the movement. We first describe the dataset of sampled robot configurations and associated ground-truth labels required for training the VAE and the performance predictors. We then provide the architectural details of our networks before presenting results of the individual experiments.

A. Dataset Generation

A dataset is sampled consisting of triplets $\{\mathbf{x}, y, s\}$, where $\mathbf{x}$ is a static robot configuration and $\{y,\ s\}$ are the associated ground truth labels for robot stability (binary) and stance (one out of eight), respectively. The robot's centre of mass location is sampled uniformly within $\pm$120mm in the longitudinal direction and $\pm$100mm in the lateral direction. Feet not in contact with the ground are sampled at heights of $\{0, 40, 80, 120\}$mm. We collect a total of 88,000 triplets whereby 70,400 are used for training (80%) and 17,600 for testing (20%).

Stability: To obtain the ground truth labels for robot stability, we follow the approach in [15], which considers friction limits of the ground and the robot as well as torque limits of the series elastic actuators [22] of the robot. The friction cone model estimates if the robot's feet in contact will slip on the surface. A foot in contact is defined such that the velocity or acceleration of the foot is zero. This means that there are different sets of contact forces and torques for a foot in contact with a surface and a foot which touches the surface without being in contact. The coefficient of friction is assumed to be constant and hence as long as the lateral forces are less than the force normal to the surface multiplied by the coefficient of friction, the robot will not slip.

Stances: Robot stances are defined by which feet are in contact with the ground and the relative ordering of the feet. Specifically, we break up a walking gait into eight phases where each phase is associated with a unique stance (see Fig. 3). The eight stances are made up of four stances with one foot in the air and four with all four feet in contact. Locomotion thus corresponds to cycling through these stances, see (Fig. 3).

B. Architecture Details

The VAE architecture comprises an encoder and decoder, which each have two fully-connected layers with 256 units and ELU non-linearities [23]. The latent space is of width 64 with a zero mean and identity variance Gaussian prior, and a diagonal Gaussian posterior distribution. The stability

classifier consists of three fully-connected layers with a width of 64 units and ELU non-linearities. It has a single output for the stability semantic label. The stance classifier consists of three fully-connected layers with a width of 64 units and ELU non-linearities. It predicts a four element output vector providing the stance encoding (see Section III-A). In the case of the locomotion experiments described below, our model takes the position of the robot's centre of gravity as an additional input. This extra input is reconstructed along the trajectory and was found useful for debugging purposes. We have verified empirically that it does not significantly impact model performance otherwise. As a baseline, a second feed-forward neural network is trained so that activation maximisation is performed directly into input space as opposed to the latent space. This classifier network takes as input the robot data $\mathbf{x}$ and predicts the probability that the robot is stable and which stance the robot is in. The architecture used consists of three fully-connected layers with a width of 64 units and ELU non-linearities.

C. Stability Experiments

The context for this experiment is disturbance rejection. If, for example, the robot is perturbed and no longer stable, we demonstrate how activation maximisation (AM) in a suitable latent space can find a smooth trajectory to a stable pose. At each time step, we backpropagate the stability error from a performance predictor back into the model's latent representation $\mathbf{z}$. This is decoded to $\mathbf{x}'$ and sent as a command to the robot's low-level controller until the robot is stable. We compare AM into a structured latent space (*latent-AM*) with applying AM directly to the input (*input-AM*). The latter scheme bypasses the latent space and, therefore, input-AM does not explicitly account for any correlations between the input state variables.

In each experiment (latent-AM and input-AM), we randomly initialise 1000 unstable configurations. AM is then driven by a binary cross entropy loss with step size $\alpha_y = 1 \times 10^{-3}$; and $\mathbf{z}$ is updated following Eq. 4 for 300 gradient steps.

Our measure of success is the proportion of episodes that feasibly transition from an unstable to a stable configuration. Here we define a stable configuration in terms of the robot's CoM together with the contact points where its feet meet the ground. Hence, it is necessary for both models to also predict which feet are in contact with the ground. Given that these contact points define the vertices of a polygon projected on the ground, the robot is stable when its CoM hovers within the support polygon. Otherwise, if the CoM projection is outside of this polygon, the robot is overextended and considered unstable. Since the robot's CoM can be inferred from its joint configurations and feet positions in the base coordinates, the CoM was not included as an input to the model.

Furthermore, we only consider an experiment a success if the robot configurations which constitute a trajectory are *kinematically* feasible. This means that there are no self-collisions or joint limit violations in all the poses which constitute a trajectory.

We report that the overall proportion of latent-AM experiments that successfully reach a stable pose is 94.6%. However, input-AM success is significantly lower: only 62.6% of input-AM experiments succeeded.

The performance of both latent-AM and input-AM is compared once 95% confidence intervals are inferred using the Wilson score interval [24]. Latent-AM confidence intervals range from 93.1% to 96.2%, whilst input-AM varies from 59.5% to 65.5%. There is no overlap between latent-AM and input-AM confidence bounds, and, hence, there is a significant and substantial difference in performance between the two schemes.

To probe the robustness of the final robot pose, we consider more restrictive definitions of stability which shrink the support polygon, to infer a *stability margin*. Conceptually, CoMs near the centre of the support polygon are more stable than CoMs near the perimeter. The stability margin is defined as the area with which the support polygon can be reduced while the robot remains stable. In practice, trajectory optimisers shrink the support polygon by ∼20% to mitigate any error in estimating the CoM position due to sensor and actuator noise [25]. Table I shows the proportion of experiments which find a stable final pose as the support polygon shrinks in increments ∼6% from its original periphery (no change in the margin) up to 31% smaller polygons. The latent-AM success-rates never drop below 92%, while input-AM performance deteriorates considerably from 63% to 52%. This clearly indicates that latent-AM is significantly more likely to find a robustly stable robot pose.

TABLE I: The percentage of experiments which terminate at a stable robot configuration for both latent-AM and input-AM. 95% confidence intervals are also presented in brackets and calculated using Wilson score interval [24]. To measure the robustness of this stable pose, the support polygon is shrunk by a margin from zero (no margin) to 31% in 6% intervals. With no margin, 95% of latent-AM trajectories reach a stable pose as opposed to only 63% of input-AM plans. The success of both schemes reduces as the margin increases, but input-AM performance degrades severely (∼ 10%). However, latent-AM success remains above 92%.

	Successful Trajectories/ %	
Margin/ %	**Latent AM**	**Input AM**
0	**94.6** (91.9, 95.3)	62.6 (59.6, 65.5)
7	**93.8** (91.2, 94.8)	59.8 (56.7, 62.8)
13	**93.2** (90.9, 94.5)	58.4 (55.3, 61.4)
19	**92.9** (90.6, 94.3)	56.4 (53.3, 59.4)
25	**92.7** (90.6, 94.3)	54.8 (51.7, 57.9)
31	**92.6** (90.5, 94.2)	52.0 (48.9, 55.1)

For a trajectory to be realisable, we emphasise that the robot trajectory must also be *dynamically* feasible. By dynamically feasible we mean that the simulation cannot move in ways that would be physically impossible for a real robot. Such simulated trajectories may end up at a stable configuration, but, because they cannot be implemented, are uninteresting from the perspective of a real robot. We consider an episode infeasible if the simulated trajectory exceeds the maximum joint velocity of 12 rad/s, as reported

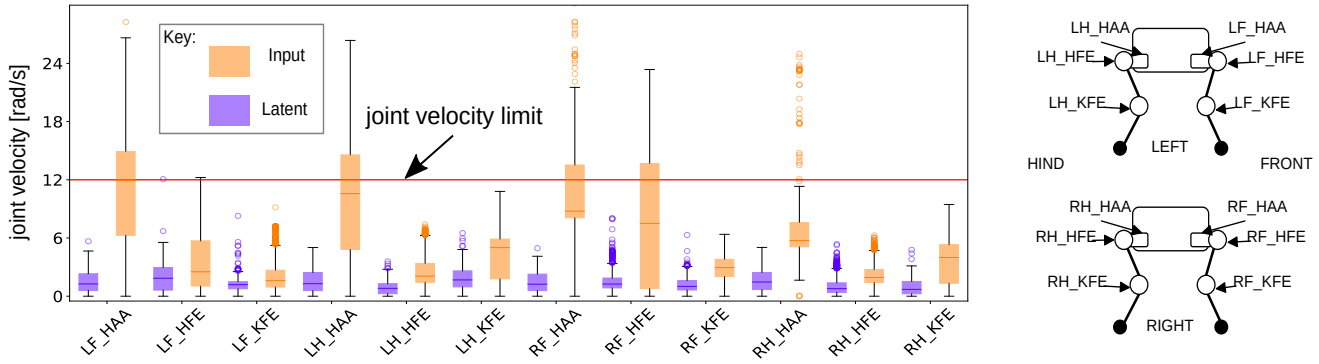

Fig. 4: A joint trajectory is only feasible if maximum permissible joint velocity (12 rad/s) is not exceeded. The proportion of input-AM trajectories which are infeasible is 59.3%, which contrasts latent-AM where no experiments violate the maximum joint velocity. The latent-AM median-maximum velocity is always lower than that of the input-AM results.

in the ANYmal motor specifications [1]. For replicability, we note that our trajectories are sampled at the target control frequency of 200 Hz.

Fig. 4 displays the maximum joint velocities from all the experiments, and input-AM joint velocity violations are visible. Furthermore, no latent-AM experiments exceed the joint velocity limit and therefore all successful trajectories are feasible. In contrast, only 59.3% of all input-AM trajectories were feasible with 95% confidence interval of (56.3%, 62.3%). Crucially, once success and feasibility are considered together, only 25.9% of input trajectories, with a 95% confidence interval of (22.9%, 28.4%) are practically useful for use on the robot. This compares to 94.6% feasibly successful latent trajectories.

Smooth trajectories are essential to avoid unsettling, tipping over, or even damaging the robot. We find that the latent-AM trajectories are significantly smoother than input-AM trajectories, as shown by the derivatives of each joint velocity in Fig. 5.

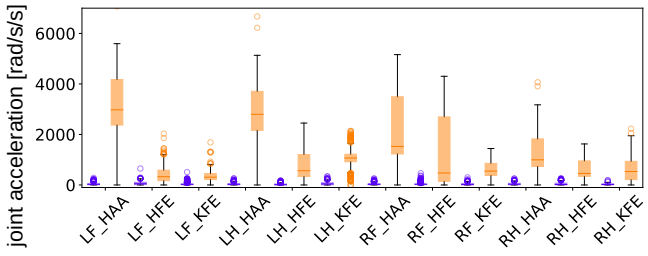

Fig. 5: The maximum joint acceleration serves as a metric of smoothness. Jerky trajectories will unsettle the robot and can cause it to fall over. Here we see that latent-AM trajectories are far smoother whilst exhibiting a narrower spread of absolute accelerations than the input-AM ones.

Analysis of the latent space show that four latent dimensions have a lower variance than the prior distribution. In contrast, the other latent units have a mean close to zero and a near unit variance, which closely map the prior distribution. Interestingly, only the four low-variance latent dimensions are updated by latent-AM, which further indicates that the latent space is exploiting the correlations which exist in the input, and that the effective latent space is four units.

Further investigation shows that the latent space is divided into stable and unstable clusters. Fig. 6 displays stable robot poses (green) surrounded by unstable configurations (red), along with a representative latent trajectory which moves from an unstable robot pose to a stable one. Below this are snapshots from a decoded trajectory sampled from a latent-AM trajectory, showing a smooth movement of the CoM to a stable position. In conclusion, the clustering of stable poses coupled with the correlations captured by the latent space could be the causes for the smoothness exhibited by the latent-AM trajectories.

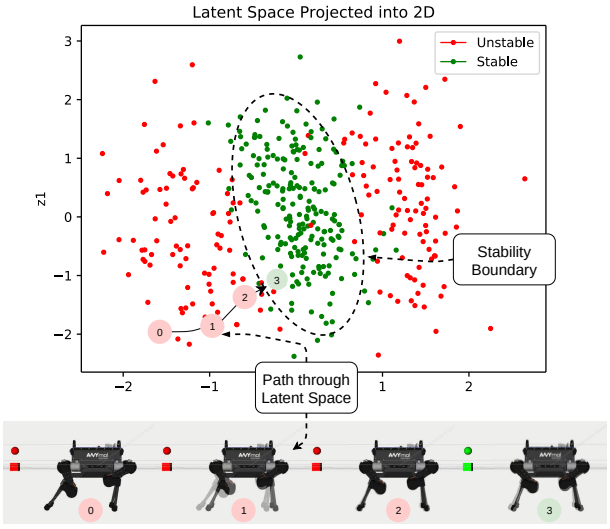

Fig. 6: We project the latent space down to 2D via PCA with an RBF kernel. Here we can see clustering of Stable robot configurations in green surrounded by unstable configurations in red. The robot configuration in pose 0 is initially unstable. Activation maximisation provides a smooth and physically realisable trajectory from this unstable pose to a stable one.

Finally, we note that evaluating a stability criterion takes 0.72ms to compute whilst a comparable analytical solution in

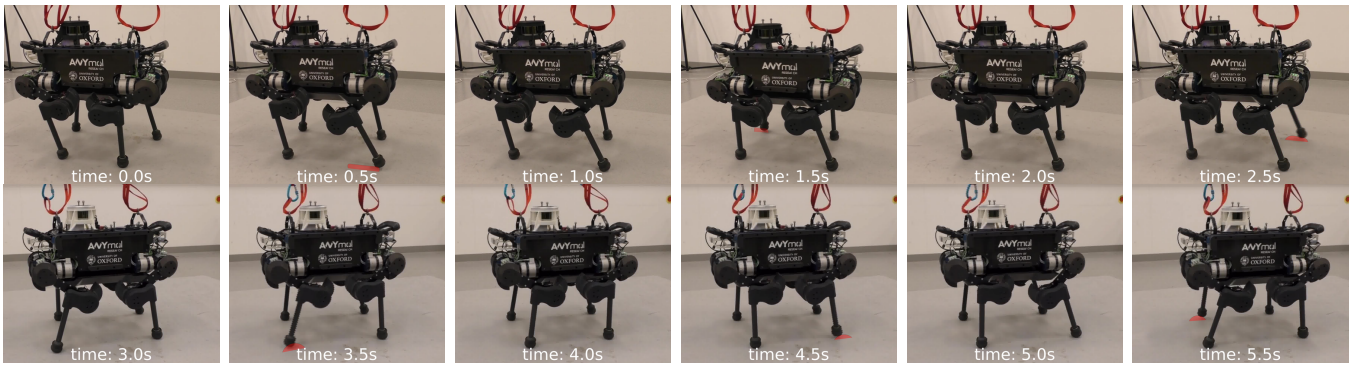

Fig. 7: The first six steps of the locomotion trajectory solved via optimisation in the latent space on the real robot. Here two separate runs with differing camera angles are displayed together. The swept area of each foot swing is displayed in red. A full video of the robot walking is found here.

[15] takes 6.56ms. This comparison uses an Nvidia Quadro P2000 on a laptop with an Intel Xeon(R) 2.9GHz CPU. This improvement in performance is attributed to the analytical solution requiring a series of sequential optimisation, as opposed to a single pass through a neural network on a GPU. Our approach, therefore, is more suitable for applications which require closed loop, online planning.

D. Locomotion

Having shown how AM is used to drive a simulated quadruped to find a smooth path that rejects unwanted perturbations, we show that an extended framework, which optimises over an entire trajectory is used to control other kinds of movement. In particular, we demonstrate that the addition of performance predictors for idealised stance configurations can be used to make a real ANYmal robot walk [3].

In our approach, the stability and stance performance predictors are sufficient to ensure that the locomotion trajectory is feasible and realisable. The overall locomotion cost is described by Eq. 8. It consists of a quadratic cost on the joint velocities and accelerations and of BCE losses. The BCE losses constantly encourage the stability of the trajectory while linearly increasing the probability of being in the next stance. For the walking demonstration, a horizon of $N = 400$ is chosen together with a step size α_{loco} of 2×10^{-4} in Eq. 9, while the Lagrange multipliers in the locomotion loss (Eq. 8) are both set to $\lambda_0 = \lambda_1 = 10^2$.

Snap-shots of a complete walk cycle executed on the real ANYmal robot are presented in Fig. 7. The controller in [7] tracked our locomotion trajectory with a tracking error of $(1.82 \times 10^{-3} rad/s)$, evaluated over a six second trajectory. Hence, this tracking error indicates that the trajectory executed on the real robot closely matches ours optimised via latent-space control. The locomotion results are qualitatively more dynamic than traditional static walk trajectories, as the centre of mass of the robot is at rest for short periods of time and only during phases where all the robot's feet are in contact. This is an encouraging result, which means that a dynamic constraint predictor could produce faster trajectories.

[3]A video of the locomotion trajectory executed on the robot is found at https://youtu.be/z1UopkfIPhI.

V. Conclusions

We propose an effective and novel approach to robot control based on high-level goal optimisation in a structured latent space, culminating in a demonstration of real-world quadruped locomotion. This is radically different from related works in that it directly exploits a statistical model of feasible robot configurations captured in a generative model to achieve smooth and realisable trajectories. These are able to stabilise a perturbed robot as well as optimise for stable and realisable locomotion all while being computable an order of magnitude faster than equivalent analytical approaches. The optimisation is achieved via activation maximisation driven by performance predictors operating at a semantic level. This allows us to render complex, otherwise non-differentiable constraints directly usable in our approach. Our approach to latent-space control is compared with a feed-forward classifier network trained to predict stability directly from the input: the latter classifier also acting as a differentiable stability criterion. Compared to our approach, Input-AM is shown to both be significantly less successful at finding stable robot poses given an initially unstable pose and produces considerably fewer dynamically feasible trajectories than latent-AM. While we have shown the appeal and efficacy of performance predictors for applying complex non-differentiable constraints, we aim to investigate their robustness in future work. Finally, this approach and subsequent results shift the gaze from a traditional control viewpoint to a novel machine learning perspective leveraging deep generative models for complex, real-world robot control.

Acknowledgements

This research was supported by an EPSRC Programme Grant [EP/M019918/1], by the UKRI/EPSRC RAIN Hub [EP/R026084/1] and the EPSRC grant Robust Legged Locomotion [EP/S002383/1]. This work was conducted as part of ANYmal Research, a community to advance legged robotics. The authors would like to thank Mathieu Geisert for his help with running the experiments on the real robot. We would also like to thank Oliwier Melon for a pipe-lining tool and Oliver Groth for his advice throughout the project.

REFERENCES

[1] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "ANYmal - A Highly Mobile and Dynamic Quadrupedal Robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[2] P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter, "Robust Rough-Terrain Locomotion with a Quadrupedal Robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[3] C. Semini, N. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. Caldwell, "Design of HyQ - A Hydraulically and Electrically Actuated Quadruped Robot," *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, 2011.

[4] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[5] C. Gehring, P. Fankhauser, L. Isler, R. Diethelm, S. Bachmann, M. Potz, L. Gerstenberg, and M. Hutter, "ANYmal in the Field : Solving Industrial Inspection of an Offshore HVDC Platform with a Quadrupedal Robot," 2019-08-28, 12th Conference on Field and Service Robotics (FSR 2019); Conference Location: Tokyo, Japan; Conference Date: August 29-31, 2019.

[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*, vol. 3, no. 3, pp. 1560–1567, July 2018.

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

[8] O. Melon, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, "Reliable Trajectories for Dynamic Quadrupeds using Analytical Costs and Learned Initializations," 2020.

[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 Robotics and Automation Letters*, vol. 2, no. 4, pp. 2201–2208, Oct 2017.

[10] T. Bretl and S. Lall, "Testing Static Equilibrium for Legged Robots," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 794–807, Aug 2008.

[21] K. P. F.R.S., "LIII. On Lines and Planes of Closest Fit to Systems of Points in Space."

[11] D. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *International Conference on Learning Representations (ICLR)*, 2014.

[12] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," in *Proceedings of the International Conference on International Conference on Machine Learning (ICML)*, 2014.

[13] Y. Wu, S. Kasewa, O. Groth, S. Salter, L. Sun, O. Parker Jones, and I. Posner, "Imagine That! Leveraging Emergent Affordances for Tool Synthesis in Reaching Tasks," *arXiv preprint arXiv:1909.13561*, 2019.

[14] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing Higher-Layer Features of a Deep Network," University of Montreal, Tech. Rep. 1341, June 2009, also presented at the ICML 2009 Workshop on Learning Feature Hierarchies, Montréal, Canada.

[15] R. Orsolino, M. Focchi, S. Caron, G. Raiola, V. Barasuol, and C. Semini, "Feasible Region: an Actuation-Aware Extension of the Support Region," *arXiv preprint arXiv: 1903.07999*, 2019.

[16] D. Orin, A. Goswami, and S.-H. Lee, "Centroidal Dynamics of a Humanoid Robot," *Autonomous Robots*, vol. 35, 10 2013.

[17] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning Agile and Dynamic Motor Skills for Legged Robots," *CoRR*, vol. abs/1901.08652, 2019.

[18] S. Gangapurwala, A. Mitchell, and I. Havoutis, "Guided Constrained Policy Optimization for Dynamic Quadrupedal Robot Locomotion," 2020.

[19] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, "Universal Planning Networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[20] M. Watter, J. T. Springenberg, J. Boedecker, and M. A. Riedmiller, "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[22] G. A. Pratt and M. M. Williamson, "Series Elastic Actuators," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1995.

[23] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," 01 2016.

[24] E. B. Wilson, "Probable Inference, the Law of Succession, and Statistical Inference," *Journal of the American Statistical Association*, vol. 22, no. 158, pp. 209–212, 1927.

[25] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, Robust Quadruped Locomotion over Challenging Terrain," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2665–2670.