Learning dynamically stable Ultra-Slow Gait in Quadruped Robots

Alex Li Noce[1], Poramate Manoonpong[2,3], Luca Patanè[4], Paolo Arena[1]

[1]DIEEI, University of Catania, Catania, Italy
(e-mail: alex.linoce@unict.it; paolo.arena@unict.it)
[2]Vidyasirimedhi Institute of Science and Technology (VISTEC), Rayong, Thailand
[3]University of Southern Denmark (SDU), Odense, Denmark
(e-mail: poramate.m@vistec.ac.th)
[4]Department of Engineering, University of Messina, Messina, Italy
(e-mail: lpatane@unime.it)

***Abstract*— This research proposes a novel control architecture for dynamically stable ultra-slow quadruped locomotion on unstructured and slippery terrains. The approach is based on Imitation Learning (IL), where two neural controllers are trained to imitate an optimal Quadratic Programming (QP) controller derived from a Linear Time Invariant (LTI) Variation-Based Linearized (VBL) model of the robot. Stability and safety are incorporated through Lyapunov-inspired and Linear Matrix Inequality (LMI) constraints integrated into the training process. A switching strategy between leg pairs enables continuous balance during slow gait execution. Preliminary simulation results demonstrate accurate center-of-mass tracking and stable behavior across gait phases, supporting the feasibility of the proposed method for real-world deployment.**

I. Introduction

Legged robots are a natural choice over wheeled or tracked machines when navigating unstructured or uneven terrain: their superior adaptability and minimal ground contact allow for efficient navigation with less impact on the environment. Robust legged locomotion over rough surfaces remains a major challenge in robotics, with critical applications in such areas as planetary exploration, search and rescue and disaster management. Among all the legged configurations currently available, quadrupeds have become increasingly popular recently, as they combine mechanical robustness with control efficiency and improved battery life. Modern quadrupeds can achieve high speeds on flat terrain and exhibit robust locomotion over a wide range of surfaces thanks to advances in dynamic stability. At velocities up to 3.7 m/s [1], trotting gaits with high step frequencies are typically adopted to reduce body oscillations, which, conversely represent a primary cause of falls when slow motion are adopted. However, in highly unstructured environments, characterized by abrupt surface changes or variable terrain granularity, such as volcanic, planetary, or landslide-prone terrains, ultra-slow gaits become essential to ensure foothold reliability and safe locomotion. In these contexts, legged robots are increasingly favored for traversing steep and irregular planetary surfaces, where traditional wheeled systems often prove ineffective [2]–[8]. In such environments, slip detection and careful locomotion strategies are critical for maintaining stability and safe navigation. To prioritize safety and robustness over speed, adopting an ultra-slow gait is commonly recommended, with locomotion strategies designed to maintain balance even under extreme scenarios, such as when two diagonal legs simultaneously lose contact or grip.

Recently, the authors addressed the problem of balancing on two legs in a four-legged Mini-Cheetah robot simulated in an accurate dynamic environment [9]. Here, the robot was able to achieve and maintain a dynamic equilibrium state with only two diagonal legs fixed to the ground. The approach was based on IL [10]: a supervised neural network was trained by imitation based on data obtained from the QP formulation of an optimal controller, designed on the VBL model of the four-legged robot [11]. In the remainder of the paper, this controller will be referred to as the VBL-QP controller. The neural network used to approximate the optimal controller was formulated as a bias-less three-layer network. An iterative procedure was implemented to distill a neural controller capable of approximating the optimal controller while satisfying the Lyapunov stability and safety conditions formulated under LMI constraints. This strategy enabled the development of a neural controller capable of maintaining stability, with an attraction range comparable to (or slightly larger than) the optimal controller, while being two orders of magnitude faster than the optimal VBL control bandwidth. The robot controlled in this way maintained its stability while lifting two legs and standing on the spot in equilibrium. Based on this first result, a first extension of the strategy is proposed below. Since the robot is able to lift two diagonal legs and maintain dynamic equilibrium, the next ground positions of the lifted leg can be controlled at any point within a certain neighborhood of the previous holding position. This strategy enables an ultra-slow gait in which the robot moves cyclically with two diagonal legs alternately holding the robot body. It should be emphasized that, to date, no existing method allows such an ultra-slow gait with only two supporting legs rigidly fixed to the ground. In the literature, recent advances in reinforcement learning techniques have enabled two-leg balancing on the Go2 robot [12]; however, this is achieved only by allowing the supporting legs to perform small hops on the ground to

compensate for inertial effects that would otherwise lead to instability. In our case, any motion of the supporting legs is strictly avoided, in accordance with the intended applications. The proposed method is applied to the dynamic simulation of the Unitree Go2 quadruped in Gazebo, with details on network training and constraint satisfaction provided in the subsequent sections. The remainder of the paper is organized as follows. Section II presents the proposed methodology, introducing the VBL-QP formulation and the neural controllers. Section III describes the training procedure of the neural architecture, including the Lyapunov-based stability and safety constraints. Section IV outlines the simulation framework and the collected dataset. Section V reports the obtained results, and finally, Section VI concludes the paper.

II. Problem Description

The proposed ultra-slow gait generates forward motion through equilibrium-preserving poses, without relying on predefined locomotion patterns. The neural controller is trained within a shared simulation framework and evaluated on unseen contact configurations, demonstrating robustness under distribution shifts. To account for the inertial asymmetries introduced by the high-fidelity simulation environment and to reflect the inevitable imbalances of the future implementation of the real robot, two different neural networks are used, each responsible for controlling one of the two diagonal pairs of legs. This architectural choice improve the robustness and effectiveness of the learned control policy. Each neural controller is trained via imitation learning with stability constraints in the augmented loss function, inspired by LTI-based methods using Lyapunov theory to design stable controllers and characterize their region of attraction [13]. Although the system is nonlinear, these results provide a conceptual foundation: interpreting the ultra-slow gait as a sequence of quasi-static equilibrium points allows LTI stability principles to be applied locally, stabilizing each configuration before transitioning to the next and extending previous balancing results to full locomotion.

A. The VBL-QP controller

The quadruped robot is described by a reduced $n_\Sigma{}^{th}$-order (with $n_\Sigma = 12$) linearised variational dynamics of the state error on SO(3):

$$\dot{e} = A(x^d)e + B(x^d)\delta u \tag{1}$$

where A and B represent the state matrices, which depend on the reference trajectory; x^d is the desired reference trajectory and e approximates the error between the actual and reference states of the system:

$$e := [e_p \; e_v \; e_R \; e_\omega]^T \tag{2}$$

where the vector elements represent the position, linear velocity, rotation and angular velocity errors, respectively, as in [14]. The overall error e lies in a Euclidean space, while its rotational component e_R is locally parametrized on $\mathrm{SO}(3)$ via the logarithmic map to $\mathrm{SO}(3)$, yielding a minimal three-dimensional representation. The constraint $e \in \mathcal{X}$ defines the admissible error set in $\mathbb{R}^{n_\Sigma}$, ensuring compatibility with the linearized variational framework. δu, the output of the variational controller, is that incremental force to be added algebraically to the reference control input such that: $u = u^d + \delta u$. Assume that e is constrained to lie within a safety condition set:

$$X = \{x \in \mathbb{R}^{n_\Sigma} : -h \leq Hx \leq h, h \geq 0\} \tag{3}$$

where $H \in \mathbb{R}^{n_\Sigma \times n_\Sigma}$ and $h \in \mathbb{R}^{n_\Sigma}$, so it holds: $x(t) \in X, \; \forall t \geq 0$.

Here the reference input is given by non-zero entries only on the vertical components of the forces, f_{iz}, on two holding legs at a time, which equally share the whole body weight:

$$u^d = \{f^d_{1x}, f^d_{1y}, f^d_{1z}, \cdots f^d_{4z}\} = \{0, 0, f^d_{1z}, \cdots, 0, 0, f^d_{4z}\} \tag{4}$$

Based on this linearised model, the optimal controller can be obtained solving a quadratic optimization problem which requires the solution of the associated Riccati equation.

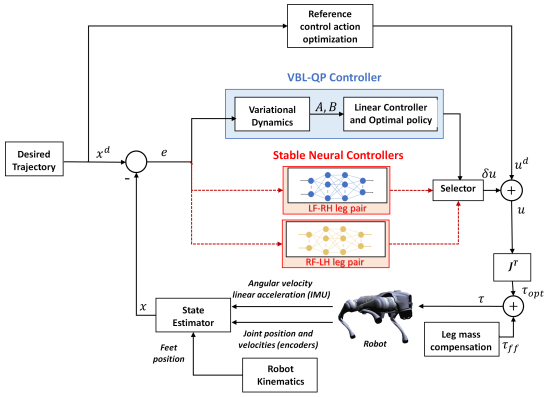


Fig. 1: Block diagram of the entire control system. The robot controller can switch between the VBL-QP and the stable neural controllers trained with IL to balance the robot on either the left front (LF) and right hind (RH) pair of legs or the right front (RF) and left hind (LH) pair of legs.

B. The neural controller

Each of the two neural network (NN) controllers consists of a 3-layer feedforward perceptron without bias with hyperbolic tangent activation function (tanh) in the hidden neurons. This ensures that $NN(0) = 0$. The entire NN representation can be formulated under the standard nonlinear operator form (SNOF) [15] by isolating the nonlinear static activation functions from the linear operators within the NN. This now consists of a feedback loop with an LTI system in the forward path and a static nonlinearity vector of size n_ϕ in the feedback path. The bias-free tanh activation functions are suitable for a sector formulation. In addition, in [16] a loop transformation, $\mathcal{G}(NN) = \widetilde{NN}$ can be applied, which unifies the sectors in which the activation functions are bounded.

Theorem 1. *Consider the feedback loop consisting of the linear system in Eq. 1 and the NN controller in the SNOF formulation, with safety conditions on X defined in Eq. 3. If there exist a positive definite matrix $P \in \mathbb{R}^{n_\Sigma}$ and a vector*

$\lambda \in \mathbb{R}^{n_\phi}$ *with* $\lambda \geq 0$*, such that inequalities expressed by in Eq.5 and in Eq. 6 hold:*

$$\begin{bmatrix} Q_1 & 0 & Q_1 A_\Sigma^\top + L_1^\top B_\Sigma^\top & L_3^\top \\ 0 & Q_2 & L_2^\top B_\Sigma^\top & L_4^\top \\ A_\Sigma Q_1 + B_\Sigma L_1 & B_\Sigma L_2 & Q_1 & 0 \\ L_3 & L_4 & 0 & Q_2 \end{bmatrix} > 0, \tag{5}$$

$$H_i^T Q_1 H_i \leq h_i^2, \quad i = 1 \cdots n_X \tag{6}$$

then the feedback system is locally asymptotically stable in $x = 0$*, with Lyapunov function* $V(x) = x^T P x$*. Moreover, the ellipsoid* $\mathcal{E}(P) := \{x \in \mathbb{R}^{n_\Sigma} : x^\top P x \leq 1\}$*, is an inner approximation of the Region of Attraction within the constraint polytope in Eq.3.*

Here $\Lambda = diag(\lambda), Q_1 = P^{-1} > 0, Q_2 = \Lambda^{-1} > 0$, and the matrices L_i are related to the $\widetilde{NN}$, H_i^T in Eq. 6 denotes the i^{th} row of matrix H in Eq. 3 and h_i is the i^{th} scalar component of the vector bound. Variables $(P, \Lambda, \widetilde{NN})$ that satisfy the Lyapunov condition in inequality 5, after loop transformation, can be recovered by calculating $(Q_1, Q_2, L_1, ..., L_4)$ using the following equations: $P = Q_1^{-1}$, $\Lambda = Q_2^{-1}$ and

$$\mathcal{G}(NN) = \widetilde{NN} = LQ^{-1} \tag{7}$$

where

$$Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}; \; L = \begin{bmatrix} L_1 & L_2 \\ L_3 & L_4 \end{bmatrix} \tag{8}$$

More details about the derivation of this result are presented in [9]. This constrained minimisation problem with different parameters (NN convergence, stability, and safety) can be solved through the Alternating Direction Method of Multipliers (ADMM) algorithm introduced in [17], and employing the following augmented loss:

$$\begin{aligned} \mathcal{L}_a(\widetilde{NN}, Q, L, Y) = \zeta_1 \mathcal{L}_{\mathcal{NN}} - \zeta_2 \log\det(Q_1) + \\ + tr(Y^T(\widetilde{NN}Q - L)) + \frac{\rho}{2}||\widetilde{NN}Q - L)||_F^2 \end{aligned} \tag{9}$$

where $\|\cdot\|_F$ is the Frobenius norm, $\mathrm{Y} \in \mathbb{R}^{(n_u+n_\phi)\times(n_\Sigma+n_\phi)}$ is the Lagrange multiplier and $\rho > 0$ is a regularisation variable which affects the convergence rate of the algorithm [13].

The augmented loss minimization is carried out in two successive steps. In the first step, the Adaptive Moment Estimation (Adam) algorithm adjusts the weights of the neural network to minimize the estimation error along with additional contributions associated with the LMI-based stability constraints. Once the NN weights are updated, the second step is to compute the matrices Q and L by solving the LMI optimization problem with the ADMM while simultaneously updating the matrix Y.

III. Neural Control Architecture

The VBL-QP balancing approach, previously developed for the Mini-Cheetah robot [18], is here adapted to the commercially available Unitree Go2 platform. Using IL, two neural networks are trained to mimic the teacher controller while preserving stability and safety during four-legged contact phases. By alternating the networks and the VBL-QP controller through a selector, the system achieves a slow but stable gait, as shown in Fig.1.

The robot state is estimated using a linear position-velocity Kalman filter, providing the twelve state variables x_i, which are compared to the desired references. The resulting state error e serves as input to the VBL-QP controller, along with the foot-to-CoM distances r and reference forces u^d, as in Eq. 4. The controller outputs an incremental force $\delta u \in \mathbb{R}^{n_u}$, which is combined with the reference forces and converted to joint torques via the transposed Jacobian. During data acquisition, r and u^d are kept constant to satisfy the assumptions of the LTI framework; consequently, the training set includes only e and δu.

This training phase consists of a series of iterative processes, as shown in Fig.2. The training dataset D =(e,δu_t) is generated by recording the error signals and the corresponding corrective actions calculated by the VBL-QP controller during the execution of the balancing task. The collected input (e) and output pairs (δu_t) form the basis for training the network.

The neural controller uses a feedforward architecture consisting of two identical networks. Each network has 12 input nodes connected to the error state vector, two hidden layers with 10 neurons each, and six output nodes that generate the incremental force components δu=[δf_x,δf_y,δf_z] for the two support legs. The hyperbolic tangent is chosen as the activation function for the hidden layers because it is suitable for the specific QC control optimization problem and has advantageous local sector properties within bounded sets centered around the origin. No bias terms are included, which simplifies the function approximation process and maintains compatibility with the optimization framework. The output layer is configured with a linear activation function.

The safety limits h_i (Eq. 3) are given in table I. The algorithm used to train the network was Adam, a stochastic optimization method with an adaptive moment estimation [19]. The training phase of the network was structured into two main steps:

TABLE I: Safety limits of the state variables.

State variables	Safety bounds
Position error $(e_{p_x}, e_{p_y}, e_{p_z})$	$\pm 0.3\ m$
Linear velocity error $(e_{v_x}, e_{v_y}, e_{v_z})$	$\pm 4.0\ m/s$
Rotation matrix error $(e_{R_x}, e_{R_y}, e_{R_z})$	$\pm 1.0\ rad$
Angular velocity error $(e_{w_x}, e_{w_y}, e_{w_z})$	$\pm 5.0\ rad/s$

- **Pre-Training phase**: the first step was a classical accuracy-based training, where each network was trained with the mean squared error (MSE) loss function for 100000 epochs. After the training was completed, the trained networks were directly validated in the dynamic simulator to evaluate their performance. After the successful validation, the process proceeded to the second step where stability constraints were introduced through an augmented loss formulation.

- **Lyapunov Conditions phase**: the augmented loss includes the optimization of the decision variables Q and L, fulfilling the safety and stability requirements defined by the LMI conditions. In this second step, the networks were subjected to a further training phase consisting of a maximum of 100 macro-iterations with 1500 epochs each, using the Adam optimization algorithm. An early termination criterion was introduced that allowed the training process to be terminated when the relative variation of the Frobenius norm of the decision variables fell below a threshold of 0.08. The augmented loss function enforced the LMI-based stability constraints presented in Section II using the linearized robot model matrices A and B as well as the constraints on the state variables h_i and the NN weights obtained from the previous learning iteration (as shown in the green box in Fig. 2).

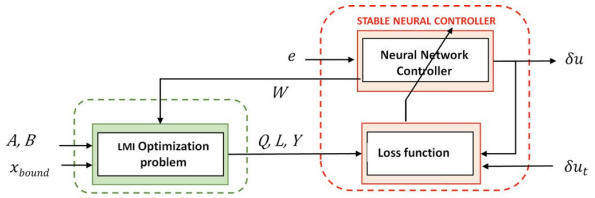


Fig. 2: Scheme of the overall learning strategy.

IV. Simulation Set-up

A. Simulation framework

The dynamic simulator used in this study is based on Gazebo and uses the Open Dynamics Engine (ODE) for a realistic modeling of the dynamics of the Go2 robot. The simulation uses a real-time factor of 1 to ensure that the robot's behavior is modeled in real time. The real-time update rate is set to 5000 Hz, which allows for smooth and fast updates during the simulation. The engine uses a maximum step size of 0.2 ms, with gravity set to $9.81 m/s^2$ to replicate real-world conditions. The solver performs 50 iterations with a Successive Overrelaxation (SOR) parameter of 1.3, while constraints such as Constraint Force Mixing (CFM) and Error Reduction Parameter (ERP) are tuned for accurate and stable contact modeling. These settings enable precise simulations of the Go2 robot's dynamics and interactions with its environment, making it a valuable tool for testing and validating the proposed control methods. Tables II and III report all relevant parameters of the dynamic simulation.

TABLE II: Go2 parameters in the simulation environment.

Parameter	Value	Units	Description
m	15.925	kg	Robot mass
I_{xx}	0.0792	kg $\times m^2$	Inertia on x-axis
I_{yy}	0.2085	kg $\times m^2$	Inertia on y-axis
I_{zz}	0.2265	kg $\times m^2$	Inertia on z-axis
μ	0.8		Friction coefficient
f_{min}	5	N	Minimum reaction force
f_{max}	300	N	Maximum reaction force

TABLE III: Dynamic simulator parameters used for the Go2 simulation.

Parameter	Value (Units)
Real-time factor	1 (-)
Real-time update rate	5000 (Hz)
Max step size	0.0002 (s)
Gravity	(0, 0, -9.81) (m/s²)
Solver type	Quick (-)
Solver iterations	50 (-)
SOR parameter	1.3 (-)
CFM	1×10^{-9} (-)
ERP	0.8 (-)
Max correcting velocity	100.0 (m/s)
Contact surface layer	0.001 (m)

B. Dataset

A complete 37^{th}-order floating-base nonlinear model, structurally equivalent to the model used in [9] for the Mini-Cheetah simulation, was used to represent the robot dynamics and allowed accurate estimation of the balancing moments required for the lifted legs during the balancing maneuvers. Data collection was performed using the VBL-QP controller as an expert rule to ensure stable balancing on two legs. Two separate datasets were collected, each corresponding to a different diagonal leg pair. Each dataset contained sequences of 90,000 elements, corresponding to a total recording time of three minutes at a sampling frequency of 500 Hz. During the acquisition process, the robot was initially balanced on two legs while the other two legs were raised to the equilibrium configuration and then lowered to new target positions. These target positions were randomly generated by introducing a displacement of the center of mass (CoM) with a maximum deviation of 20 cm from the target position. Among the parameters in Table IV, a key change to shorten the training phase concerns the reuse of the previously optimized weight matrices across iterations (Fig. 2).This approach, which was first presented and validated in [9], consists of initializing each new iteration with the final weights from the previous iteration instead of starting the training from scratch. In this work, a ±10% perturbation was applied to the reused weights to promote exploration and avoid local minima, reducing the training to five iterations of 1,500–3,000 epochs each, compared to 90,000 in previous implementations. Optimization focused only on the loss matrices Y, Q, and L, while the network was initialized from the previous optimal weights with small perturbations, reducing computational load and accelerating convergence.

V. Results

In this section, a case study demonstrating the robot performing an ultra-slow gait is presented. To emulate unstructured environments with instability and uncertainty, dynamically spawning pegs were introduced as discrete footholds. While their positions are predefined, the non-uniform and sudden appearance of pegs creates abrupt changes in support geometry, effectively acting as controlled disturbances that require continuous online balance adaptation. Each peg

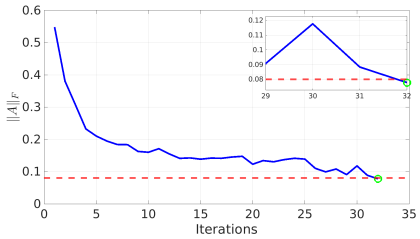
TABLE IV: Neural network training parameters.

Parameter	Value
Input neurons	12
Output neurons	6
Hidden neurons per layer	10
Number of hidden layers	2
Learning algorithm	Adam
Batch size	512
Macro-iterations max	100
n^{acc}_{epoch}	100×10^3
$n1^{stab}_{epoch}$	1.5×10^3
$n2^{stab}_{epoch}$	3×10^3
ρ	4000
η_{NN}	2000
η_{ROA}	100

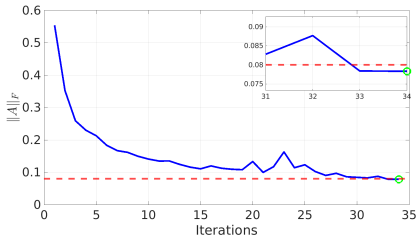
represents a safe contact point that must be reached with high precision. To maintain stability, the displacement of the desired CoM between consecutive steps is limited to approximately 10 cm.

Locomotion follows a sequential diagonal gait in which one diagonal leg pair swings while the others maintain support. Each diagonal phase lasts about 10 s and is separated by a 4 s quadrupedal stabilization phase enforced by the VBL controller. A full locomotion cycle, composed of two diagonal phases, restores the initial contact configuration in about 20 s. The control system operates at a fixed timestep of dt=0.002 s. Figures 3(a) and 3(b) illustrate the evolution of the key term in the augmented loss function, specifically the Frobenius norm, during the second training phase. This phase is devoted to satisfy Lyapunov-based constraints, and the results indicate that both networks successfully meet the early stopping criterion in approximately 30 iterations, demonstrating rapid convergence. Fig. 4 shows the robot's CoM and foot trajectories during the ultra-slow gait. The CoM tracks the reference with small oscillations after leg lifts, while foot trajectories confirm stable coordination during alternating support phases. The controller achieves a CoM RMSE of 0.024 m and MAE of 0.021 m, with a CoM success rate of 97.49%. Although switching events increase the tracking error to about 0.065 m, stability is preserved throughout the motion. In order to evaluate the performance of the NN controller, it is necessary to monitor its ability to enforce Lyapunov stability constraints during the training process. Despite the robot's minimal velocity, oscillations appear shortly after one pair of legs enters the swing phase, highlighting the system's dynamic sensitivity and the neural controller's effectiveness in stabilizing the root body in real time. This setup also allows comparison with the baseline VBL method, which handles balance adequately but lacks the computational efficiency of the NN controller. The network's fast feedforward pass enables real-time control, while training is performed offline. The proposed NN controller improves both speed and efficiency over conventional VBL-QP control, reproduces its performance, and extends the Region of Attraction. Its generalization is confirmed by perturbation tests and by path-following along dynamically spawned pegs, demonstrating walking behaviors beyond those seen during training.

Regarding the temporal aspects, it should be emphasised that the methodology with the NN controller allows real-time performance during balancing. As for the training phase, the algorithm was run on a laptop with an Intel i9 with 32 GB of RAM and an Nvidia RTX A2000 GPU, even if the procedure is performed offline, in order to have an idea of the time required to achieve the desired performance. While the NN training (phase 1) was run on the GPU using Pytorch, the optimization for the Lyapunov condition (phase 2) was run on Matlab using the laptop CPU. The total time required for one macro iteration in phase 2 was about $20s$, while phase 1 took about 10 minutes. Figure 5 reports four consecutive poses of the robot while performing the ultra-slow gait, alternating a two-leg balancing pose to a four leg static pose, in which the VBL takes control and imposes the forward motion. The robot is able to target the lifted legs towards specific ground positions, simulating safe holding points.



(a)



(b)

Fig. 3: Frobenious norm in training of both couple of legs: (a) FL-RH leg pair, (b) RF-LH leg pair.

VI. Conclusions

In this work, an ultra-slow gait was implemented for a highly agile four-legged robot. This gait is intended for use in extremely complex situations where the terrain characteristics are inconsistent and unreliable and up to two legs lose their footing. In this situation, the robot is able to lift the two most unreliable diagonal legs while maintaining dynamic balance. The robot has the ability to find the safest position for the lifted legs before steering them to the ground. The overall control strategy consists of two NN controllers trained by mimicking linear optimal controllers with stability and safety constraints. The robot after the training phase, succeeds in moving ultra-slowly following an imposed direction by alternating a two-leg balancing pose to

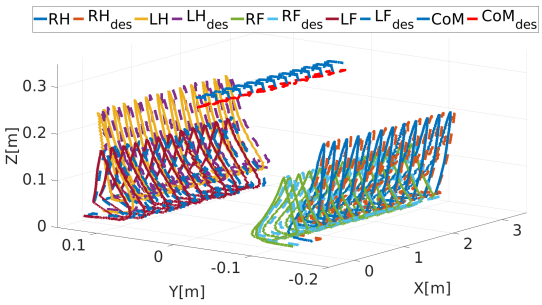


Fig. 4: Trajectory of the CoM and feet in 3D space. The motion is generated by alternating the neural network activation for RF–LH and LF–RH leg pairs, with step progression guided by peg displacement, ensuring balance along the planned path.

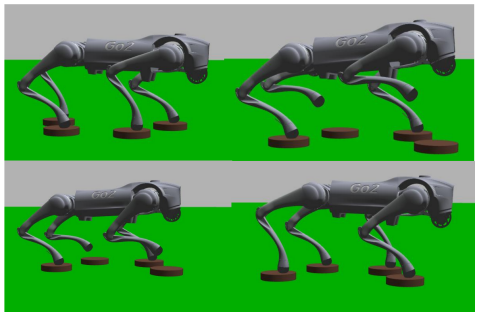


Fig. 5: Snapshots of the ultra-slow gait phases highlighting balance and precise foot placement on pegs. The extended stance enhances stability during transitions, emphasizing dynamic balance and foot planning in constrained environments.

a four leg static configuration, driven by a VBL controller. In fact, whereas the NN controllers allow a very high control band, particularly useful in the two-leg balancing pose, the four standing leg configuration does not need particular performance and can be, in this initial stage, controlled by a traditional VBL. Next stage will be to distill a unique NN controller, only for the sake of uniformity. The initial simulation results reported here are currently being further optimized so that the robot is able to move on a terrain characterized by different slopes and a series of limited safe positions with different heights. This will open up the possibility of using these robots for climbing in very challenging, sloping and wild environments. The main novelty lies in generating a stable slow gait from a purely equilibrium-based controller, without predefined locomotion patterns. The neural controller is trained and tested on significantly different contact configurations, highlighting the generality of the stability-driven framework across multiple behaviors. Sim-to-real transfer remains ongoing due to challenges such as model mismatch, contact uncertainty, delays, and unmodeled dynamics; therefore, this work focuses on simulation validation as a key step toward real-world deployment.

ACKNOWLEDGMENT

PM acknowledges the support of the Reinventing University project, funded by the Ministry of Higher Education, Science, Research and Innovation of Thailand.

REFERENCES

[1] B. Katz, J. D. Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6295–6301.

[2] S. Vyas, F. Stark, R. Kumar, H. Isermann, J. Haack, M. Popescu, J. Middelberg, D. Mronga, and F. Kirchner, "Quadrupeds for planetary exploration: Field testing control algorithms on an active volcano," *arXiv:2510.18600*, 2025. [Online]. Available: https://arxiv.org/abs/2510.18600

[3] A. Sanchez-Delgado, J. C. V. Soares, V. Barasuol, and C. Semini, "Towards proprioceptive terrain mapping with quadruped robots for exploration in planetary permanently shadowed regions," *arXiv:2510.18986*, 2025. [Online]. Available: https://arxiv.org/abs/2510.18986

[4] Z. Peijin, L. Qikai, Q. Ripeng *et al.*, "A review of legged robots for planetary exploration in complex extraterrestrial terrain," *J. Robot*, vol. 47, no. 6, pp. 817–843, 2025. [Online]. Available: https://robot.sia.cn/en/article/cstr/32165.14.robot.250060

[5] Y. Nisticò, S. Fahmi, L. Pallottino, C. Semini, and G. Fink, "On slip detection for quadruped robots," *Sensors*, vol. 22, no. 8, p. 2967, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/8/2967

[6] Y. Ambe and F. Matsuno, ""leg-grope walk": strategy for walking on fragile irregular slopes as a quadruped robot by force distribution," *Robomech Journal*, vol. 3, p. 7, 2016. [Online]. Available: https://link.springer.com/article/10.1186/s40648-016-0046-2

[7] M. Focchi, R. Orsolino, M. Camurri, V. Barasuol, C. Mastalli, D. G. Caldwell, and C. Semini, "Heuristic planning for rough terrain locomotion in presence of external disturbances and variable perception quality," *arXiv:1805.10238*, 2018. [Online]. Available: https://arxiv.org/abs/1805.10238

[8] H. Kolvenbach, P. Arm, E. Hampp, A. Dietsche, V. Bickel, B. Sun, C. Meyer, and M. Hutter, "Traversing steep and granular martian analog slopes with a dynamic quadrupedal robot," *arXiv:2106.01974*, 2021. [Online]. Available: https://arxiv.org/abs/2106.01974

[9] A. Li Noce, L. Patanè, and P. Arena, "A stable and safe method for two-leg balancing of a quadruped robot using a neural-network-based controller," *Robotics and Autonomous Systems*, vol. 186, p. 104901, 2025.

[10] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *CoRR*, vol. abs/1811.06711, 2018.

[11] M. Chignoli and P. M. Wensing, "Variational-based optimal control of underactuated balancing for dynamic quadrupeds," *IEEE Access*, vol. 3, pp. 49 785–49 797, 2020.

[12] Unitree Robotics, "Unitree Go2: Agile Quadruped Robot Demonstration," https://youtu.be/6zPvT0ig1VM, 2023, youTube video, official Unitree Robotics channel.

[13] H. Yin, P. Seiler, M. Jin, and M. Arcak, "Imitation learning with stability and safety guarantees," *IEEE Control Systems Letters*, vol. 6, no. 1, pp. 409–414, 2022.

[14] F. Bullo and A. D. Lewis, *Geometric Control of Mechanical Systems*, ser. Texts in Applied Mathematics. New York-Heidelberg-Berlin: Springer Verlag, 2004, vol. 49.

[15] K. Kim, E. Patron, and R. Braatz, "Standard representation and unified stability analysis for dynamic artificial neural network models," *Neural Networks*, vol. 98, 12 2017.

[16] H. Yin, P. Seiler, and M. Arcak, "Stability analysis using quadratic constraints for systems with neural network controllers," *IEEE Transactions on Automatic Control*, vol. 67, no. 2, pp. 1980–1987, 2022.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, 2011.

[18] Open-Source-Repository, "MIT-biomimetics group, cheetah software," https://github.com/mit-biomimetics/Cheetah-Software.git, 2022.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.