Four-Legged Gait Control via the Fusion of Computer Vision and Reinforcement Learning

Ignacio Dassori
Electrical Engineering Department
Universidad de Chile
Santiago, Chile
ignacio.dassori@ug.uchile.cl

Martin Adams
Electrical Engineering Department
Universidad de Chile
Santiago, Chile
martin@ing.uchile.cl

Jorge Vasquez
Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, PA, USA
jivasque@andrew.cmu.edu

***Abstract*—This article explores the integration of fully autonomous legged robots in obstacle filled environments, simultaneously addressing the challenges of navigation and control. Despite the potential of legged robots for dynamic tasks, their deployment in complex environments has been hindered by the difficulty of developing effective autonomous control systems. In particular, the motion planning problem is addressed in this article, by formulating it as a Partially Observable Markov Decision Process (POMDP) and applying Proximal Policy Optimization (PPO), a model-free Deep Reinforcement Learning (DRL) algorithm. To improve sample efficiency and real-world applicability, the proposed method incorporates a Central Pattern Generator (CPG) for motion planning and a Variational Autoencoder (VAE) for terrain representation, reducing the complexity of action and observation spaces. Referred to as the VAE-CPG architecture, its performance is demonstrated using the Unitree Laikago robot within the PyBullet simulation environment, aiming to show its effectiveness in simulated construction sites. Our findings indicate that by reducing the legged action space to periodic gait patterns and optimizing the gait based on sensory feedback, we achieve enhanced adaptability and efficiency. This work presents a viable means towards the deployment of autonomous legged robots and their improved efficiency in real applications.**

***Index Terms*—Robotics, Reinforcement Learning, Computer Vision, Central Pattern Generator, Variational Autoencoder, Gait optimization.**

I. Introduction

The aspiration to integrate fully autonomous robots into the construction industry promises a major shift in construction methodologies. However, the progression towards achieving full automation in construction robots has been gradual and incremental. Currently, these robots predominantly operate under semi-autonomous conditions, heavily reliant on human intervention for guidance and task-specific programming, which necessitates significant domain-specific knowledge. This scenario highlights a pivotal gap in the journey towards the envisioned level of autonomy.

Legged robots, with their agility and adaptability, are particularly well-suited for the unpredictable terrains of construction sites. Yet, devising effective control systems for these robots presents substantial hurdles. The challenges stem from the necessity to model their complex dynamics and navigate through extensive action spaces. Furthermore, the integration of visual systems for obstacle identification and terrain analysis adds another dimension of complexity, requiring advanced algorithms capable of concurrent visual processing and mechanical actuation.

Advances in Deep Reinforcement Learning (DRL) have shown potential in overcoming these obstacles, enabling legged robots to learn sophisticated control strategies through interactions within simulated environments [1]. Despite these advances, current approaches depend significantly on complex sensing technologies and expert knowledge. Model-free DRL, while effective for learning locomotion, faces limitations in sample efficiency and real-world applicability. Additionally, vision-based methods have yet to be extensively tested in environments that closely mimic the real world, revealing a research gap. Jain et al. [2] proposed an end-to-end vision-based RL architecture for learning dynamic leg movements within simulated environments. We expand on this by training with more visually complex settings and reducing image input dimensionality via the use of VAEs.

We address the motion planning issue for quadruped robots by conceptualizing it as a Partially Observable Markov Decision Process (POMDP) and employing Proximal Policy Optimization (PPO), a model-free RL algorithm [3]. Our strategy enhances the RL agent's sample efficiency and transferability to real-world scenarios by incorporating a Central Pattern Generator (CPG) for refined motion planning, thus narrowing the action and observation spaces [4], [5]. Applied to the Unitree Laikago robot in the PyBullet simulation framework [6], this article presents the VAE-CPG architecture, which strives to demonstrate improved efficiency and practical viability in environments designed to simulate construction sites. This paper outlines our proposed gait controller, followed by detailed experiments and results, demonstrating our approach's effectiveness in more realistic settings.

II. Related Work

Many different approaches for automatic gait optimization have been suggested to date, such as grid search and evolutionary algorithms [4]. Gait optimization is a basic, yet challenging problem for quadrupedal robots. Various automatic gait optimization methods have been used in locomotion to design gaits, including gradient descent methods [7], evolutionary algorithms [8], particle swarm optimization [9], and many

others [10], [11]. This article makes contributions in the following 3 areas:

A. Hierarchical Reinforcement Learning

There is extensive research exploring hierarchical approaches to reinforcement learning. Sutton et al. [12] proposed *options*, or temporarily extended actions. This is an early formulation of hierarchical RL, combining low-level options with high-level policies learning to choose appropriate options. [13], [14] used a policy gradient approach to learn the termination conditions of options. In [15], the low-level policy is trained with an auxiliary reward to encourage value changes of non-proprioceptive dimensions of state observation, such as translation toward a specific orientation for legged robots. While most early works focus on learning policies at different levels separately, end-to-end framework learning multiple levels of policy [16], [17] have also demonstrated feasibility recently.

Our work differs from these methods because we use the CPG rather than a low-level controlling policy. This reduces the effort to train another policy and allows continuous transitions between options. Therefore, our high-level policy doesn't have to choose from a fast or large gait, but can rather adapt to terrains by "defining" the appropriate speed and height of the gait. The smooth CPG action space also ensures the smooth transition between gaits.

B. Locomotion for Legged Robots

Locomotion skills are fundamentally important for legged robots to traverse various terrain environments. One line of work falls into the trajectory optimization domain. Hemker et al. [18] propose a motion optimization approach for humanoid robots. Specifically, they parameterize the walking trajectory and apply sequential surrogate optimization to solve for a fast and stable walking gait. Lizotte et al. [19] applies Bayesian optimization to a quadrupedal robot. They take into account two optimization criteria for a quadrupedal robot: one with respect to the maximum walking speed and the other for the maximum gait smoothness. Another line of research takes CPGs as an efficient approach. CPGs are proved to be capable of producing rhythmic movements such as swimming and walking based on environmental information [20], [21]. In the context of legged robots, Alexander et al. [22] demonstrates success in applying open-loop CPGs to a quadruped robot for fast locomotion on tough terrain. However, this work has limitations in generalizing to different terrains, and adapting its parameters according to the environment becomes crucial. Efforts have been made in designing closed-loop CPGs using sensory feedback [23], [24]. In this work, rather than closed-loop control, we propose a reinforcement learning network that learns to adapt CPG parameters based on the robot's proprioceptive and external states.

C. Terrain Representation

Visual data in the form of 2D images and heightmaps is a common approach to acquire information of the surrounding terrain. Many prior articles on RL involve learning dense representations from visual data, both 2D [25], [26] and 3D [27]. DeepLoco [28] has shown the effectiveness of using a terrain map features in learning high-level tasks. They use 2D heightmaps with a resolution of 32x32 pixels as input and apply convolutions to obtain a 128 dimension terrain representation. [29] explores various approaches of image latent representation and demonstrates that VAEs are one of the most efficient and stable representations for image-based RL. Therefore, in this article, we pretrain a VAE and apply it to encode the image to get a more compact representation of the terrain. This module simplifies perception tasks for our agents within the broader system.

III. Gait Controller

The complete structure of the proposed VAE-CPG architecture is presented in Figure 1, which is divided into two main modules. The first is a VAE, which receives RGB images from a camera positioned at the front of the robot, and transforms them into low dimensionality latent representations of the terrain. The second module is a fully connected policy network, which chooses the parameter values of a CPG and modulates its output by adding correction terms to the desired position of every motor. The CPG is a mathematical representation of a cyclical gait, and the trajectory it follows is determined by its parameters. This way, the policy adapts the robots gait to the current terrain and state by choosing optimal parameters, instead of directly yielding motor position commands.

A. Variational Autoencoder

Unlike the traditional autoencoder, where the encoder learns to map inputs to specific points in the latent space, a VAE's encoder is designed to map inputs to probability distributions over the latent space [30]. For instance, it may output the parameters of a Normal distribution (μ,σ^2) that represent the statistical properties of the latent space. Building on the probabilistic nature of the encoder, a latent vector z is then sampled from the learned distribution, utilizing a technique known as the reparameterization trick. This stochastic sampling gives the network its generative capabilities, as outputs are no longer deterministic. The decoder then utilizes this sampled latent vector to reconstruct the output. The introduction of sampling from the latent space enables the VAE to generate diverse and continuous outputs during both training and inference. The VAE's architecture can be seen in Figure 2.

While traditional Autoencoders utilize straightforward reconstruction error metrics to evaluate outputs, their use alone falls short in guiding the model to estimate meaningful distributions. Relying solely on reconstruction error might lead to an irregular latent space with limited generative capabilities. To address this, during training the objective is to minimize the loss function

$$loss = -D_{KL}(q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)}[\log p(x|z)]. \quad (1)$$

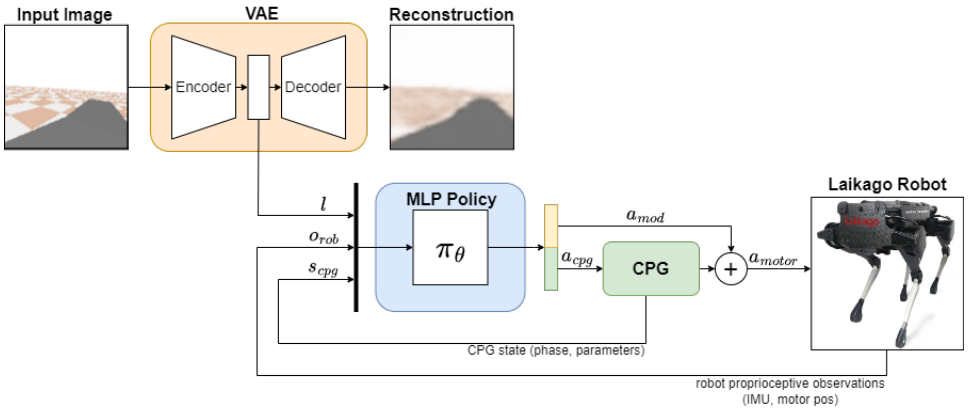

Fig. 1: Gait control pipeline overview.

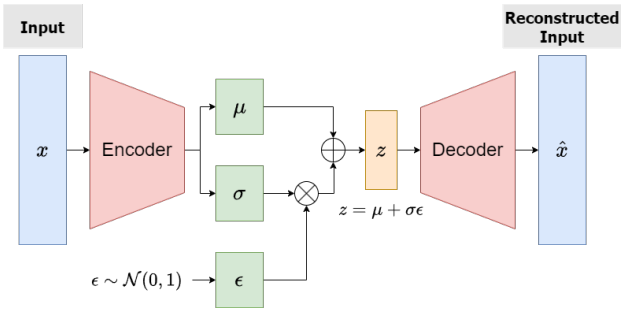

Fig. 2: The VAE architecture.

The first term on the RHS of 1 is known as the regularization term, where D_{KL} represents the Kullback-Leibler Divergence, $q_\phi(z|x)$ is the learned distribution of the latent variable z given by the encoder network with parameters ϕ, and $p_\theta(z)$ is a prior for z. This term measures how much the learned distribution diverges from the prior and penalizes it, encouraging the latent space to follow a certain structure. A Normal distribution is often used for $p_\theta(z)$. The second term is a reconstruction loss, where $q_\phi(z|x)$ is the likelihood of reconstructing the input data x given a latent sample z, decoded by the network with parameters ϕ.

B. Parameterized CPG Gait

Exploring the complete action space of the 12 on board motors of the Laikago can lead to a slow learning process, particularly since the legs start off with no sense of coordination. Introducing prior knowledge has the potential to expedite training by reducing the number of samples required to acquire essential skills. Notably, CPGs are widely used for generating repetitive, cyclic movements.

In our approach, we constrain the movements of the Laikago's hip and knee motors to follow periodic trajectories defined by Equations 2 to 8. These equations, obtained from [31], parameterize a gait specific to quadruped locomotion, providing a structured foundation for learning locomotion skills in a more efficient manner.

Four coupled oscillators with frequency f govern each one of the robot's legs, as shown in Figure 3. There is a fixed sequential $90°$ shift between their phases Θ_i given by

$$\Theta_i = (2\pi f \Delta t + \frac{\pi}{2} i) \bmod (2\pi), \quad i \in [0, 1, 2, 3], \tag{2}$$

in the order: front left (FL), back right (BR), front right (FR), back left (BL).

The gait is split into two stages, namely stance and swing. The ratio of the duration of each stage to the cycle is determined by the duty factor d, where a value close to 1 gives a longer stance and shorter swing, and vice-versa for values close to 0. In (3) a transform is applied to the phase in order to extend/shorten the stages duration in accordance with the duty factor.

$$\phi_i^h = \begin{cases} \frac{\phi_i}{2d} & \text{if } \Theta_i^h < 2\pi d \\ \frac{\phi_i + 2\pi(1-2d)}{2(1-d)} & \text{else.} \end{cases} \tag{3}$$

The target motor angle of the hip θ_i^h is calculated as

$$\theta_i^h = A_h \cos \phi_i^h + o_h, \tag{4}$$

where A_h is the hip oscillator amplitude, o_h is an offset which acts as the continuous component or center of oscillation and ϕ_i^h is the transformed phase of the hip.

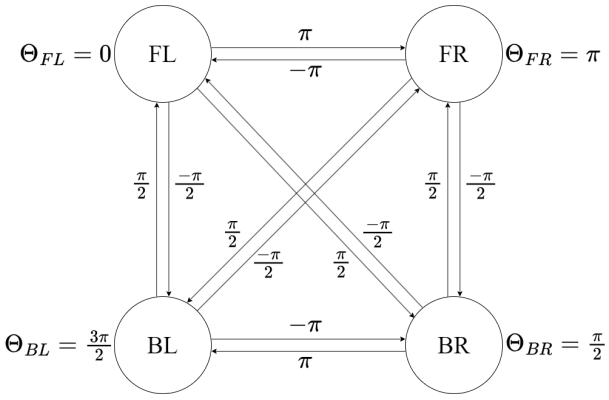

Fig. 3: Phase shift between coupled oscillators.

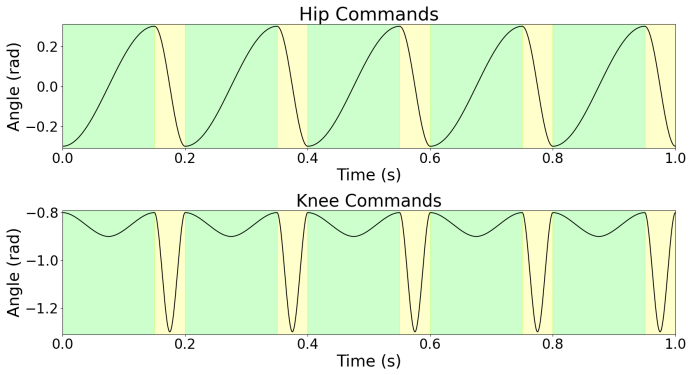

Fig. 4: Output of motor commands given by the CPG. Green and yellow areas indicate stance and swing, respectively. $f = 4$, $d = 0.75$, $A_h = 0.3$, $A_k^{st} = 0.1$, $A_k^{sw} = 0.5$.

The amplitude of the knee motor movements depends on the current stage of the gait. In general, the stance has small movements as the leg remains extended, while the swing contracts the leg with a broad movement to lift it from the ground. We include this prior knowledge by having a reduced value range for the stance amplitude, but ultimately leave it to be discovered by the agent during training. In our implementation, the hip and knee phases (ϕ_h, ϕ_k) are aligned

$$A_{k,i} = \begin{cases} A_k^{st} & \text{if } \phi_i^k < \pi \\ A_k^{sw} & \text{else,} \end{cases} \tag{5}$$

where $\phi_i^k = \phi_i^h$.

The piecewise cubic profile Γ_i described by

$$\phi_i' = 2\left(\frac{\phi_i^k}{2\pi} \bmod (0.5)\right), \tag{6}$$

$$\Gamma_i = \begin{cases} -16\phi_i'^3 + 12\phi_i'^2 & \text{if } \phi_i' < \frac{1}{2} \\ 16(\phi_i' - \frac{1}{2})^3 - 12(\phi_i' - \frac{1}{2})^2 + 1 & \text{else,} \end{cases} \tag{7}$$

serves as a tool for defining a smooth trajectory for the knee joint profile, where ϕ_i' denotes a transformed version of the knee phase resized to the interval $[0, 1]$. The cubic profile produces a periodic motion, oscillating between swing and stance phases for each stage's duration.

Finally, the target motor angles of the knees θ_i^k are calculated using

$$\theta_i^k = A_{k,i}\Gamma_i + o_k, \tag{8}$$

where the profile output Γ_i is scaled by the amplitude of the current gait stage and an offset o_k is added.

Utilizing these equations with a predefined set of parameters, results in the generation of periodic leg movements, exemplified in Figure 4. By smoothly varying parameters, the gait's form can be dynamically altered to allow the robot to traverse a changing environment.

C. Motor Control Policy

The policy of the VAE-CPG architecture is implemented as a two-layered fully connected Multi-Layer Perceptron (MLP) network, trained using the PPO algorithm provided by the Stable Baselines3 library [32]. The input, described in Table I, is a concatenation of several components: the latent vector obtained from the VAE, the CPG's phase and parameters, and the robot's state. This input is then fed into the MLP network, producing a set of four parameters along with a correction term for each motor.

PPO follows the Actor-Critic methodology, involving the training of two networks; namely the policy and the value function. In our architecture, the presented MLP network serves as the actor, responsible for decision-making. The Critic network, which estimates the value function, is not explicitly depicted, as its only relevant during the training phase. During training, the Critic network is used for computing the advantage and guiding policy updates by assessing the quality of chosen actions.

The segment of the input corresponding to the robot state contains information regarding orientation, angular velocities, and motor angles. The remaining segment of the input comprises the CPG's parameters that defined the legs' trajectories during the last timestep, along with the current phase of the CPG. For simplicity, we opt to use predetermined values for f, o_h, and o_k, and utilize eight of the twelve on board motors. Specifically, we choose to maintain the coronal plane motors in fixed positions.

	Input	**Description**
Latent Command	Latent Vector (x32):	Features extracted from RGB camera image by VAE.
Robot State	IMU Readings (x6):	Yaw, pitch, roll and angular velocities.
	Motor Positions (x8):	Current position of hip and knee motors.
CPG State	CPG Values (x5):	CPG phase and parameters

TABLE I: Description of the policy inputs and size.

The ranges of values for the action space shown in Table II were selected to afford the policy the flexibility to explore diverse strategies, while simultaneously imposing constraints, ensuring it operates within values that are sensible for the given problem. In defining these value ranges, we leverage our intuition about the problem and align them with practical considerations.

The correction terms provided by the policy adjust the motor positions generated by the CPG by incorporating residuals as necessary. These small adjustments are essential to achieve behaviors like steering, as the CPG assigns motor angles part of the same trajectory to all legs. This approach of modulating trajectory generators has demonstrated success in learning intricate control behaviors, as evidenced in [33].

	Action	Values
CPG Parameter	Frequency f	3Hz
	Amplitude A_h, A_{sw}	$[0.0, 0.5]$
	Amplitude A_{st}	$[0.0, 0.2]$
	Duty Factor d	$[0.5, 0.95]$
Correction Term	Residual (x8)	$[-0.1, 0.1]$

TABLE II: Action Space range of values and size.

IV. Experimental Setup

Our agent undergoes training within a virtual environment designed to simulate an interior setting, with the objective of accomplishing goal-reaching tasks. The simulation operates at a frequency of 500Hz, and at every 10 timesteps, a latent vector is derived from an RGB 512x512 pixels sized image captured from the robot's perspective. Each episode begins with the robot standing within a square room, with the goal placed randomly across the opposite wall. The goal position is such that it requires some degree of steering from the robot to be reached. We evaluate our method in two scenarios: a room without obstacles and a room with obstacles randomly distributed throughout. In both scenarios, the robot must reach the goal within a limited time frame. The obstacles include items commonly found on construction sites, such as traffic cones, ladders, concrete barriers, pallets, and cement bags (Figure 5, lower left image). We limit the maximum amount of obstacles per episode to three.

The reward signal consists of several terms detailed in Table III, each rewarding or penalizing specific behaviors. Terms r_1 and r_2 reinforce forward movement and movement in the goal's direction, respectively, both capped at a velocity of 1 m/s. These rewards are weighted based on the robot's progress towards the goal $w \in [0, 1]$. Initially, we promote forward walking more, gradually transitioning to rewarding steering in the later stages of an episode.

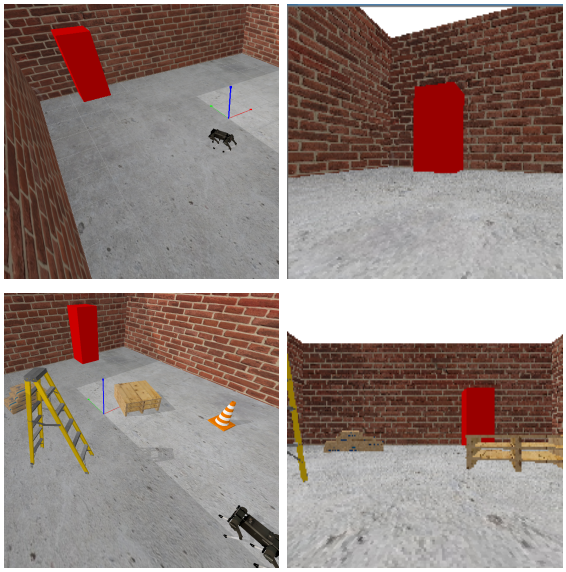

Fig. 5: Left: No-obstacle (upper images) and obstacle (lower images) goal finding tasks environments. Right: RGB camera feed from robot perspective.

Progress towards goal	$w = \lVert \frac{\vec{Q}}{\vec{G}} \rVert$
Capped forward velocity	$r_1 = \min(v_y, 1)$
Velocity in goal direction	$r2 = \min(\lVert v_x, v_y \rVert, 1) \cdot \cos(\theta_q - \phi_g)$
Termination reward	$r3 = \begin{cases} 1000 & \text{if goal reached} \\ 2000(w-1) & \text{else} \end{cases}$
Robot stability	$r4 = \lvert\omega_x\rvert + \lvert\omega_y\rvert$
Robot facing towards goal	$r5 = \lvert\theta_g - \phi_g\rvert^2$
Distance to obstacles	$\vec{d} = \left[\lVert\vec{Q} - \vec{o_0}\rVert, \ldots, \lVert\vec{Q} - \vec{o_n}\rVert\right]$
Obstacle proximity penalty	$r6 = \sum_i^n \max(1 - \vec{d_i}, 0)$
$R = (1-w)r_1 + wr_2 + r_3 - 0.3r_4 - 3r_5 - r_6$	

TABLE III: Reward signal components. $\vec{Q}, \vec{G}$ are the position vectors of the robot and goal. θ_q is the yaw of the robot. ϕ_g is the angle to the goal w.r.t. the robot's direction of motion.

An episode ends when the agent falls over, collides with an obstacle, reaches the goal, or exceeds the maximum number of timesteps. The termination term r_3 only applies when an episode concludes, having a positive value if the task was completed, and negative otherwise. The penalty diminishes proportionally based on the distance the agent traversed before termination. We promote more stable gaits and orientation towards the goal using r_4 and r_5, respectively. Finally, term r_6 penalizes proximity to obstacles within a 1-meter radius, guiding the robot to avoid them.

Each task is trained for a maximum of 25 million timesteps, which corresponds to approximately 14 hours of in-simulation time. We train using a RTX 4070 NVIDIA graphics card with 6 parallel agents, and hyperparameters detailed in Table IV. Our VAE was pretrained-trained using a dataset of over 2000 images collected within the simulated environment.

Parameter	**RL Policy**	**VAE**
Batch size	8192*6	4
Mini-batch size	1024*6	-
Max timesteps	8000	-
Clip range	0.2	-
Discount factor	0.99	-
Learning rate	0.0003	0.0001
Hidden layers	[2048, 1024]	[16, 32, 64, 128, 256]
Activation function	ReLU	ReLU
Output activation	-	Tanh
Latent dim	-	32
KLD weight	-	0.00025
Epochs	-	300
Dataset images	-	2048

TABLE IV: Hyperparameters of the VAE-CPG architecture.

V. Experimental Results

Using our method, we are able to learn policies capable of clearing both previously defined tasks consistently. The robot

successfully learns walking and steering behaviors, enabling it to move towards the goal while avoiding obstacles.

To demonstrate the advantages of employing a CPG to restrict the action space, we compare the performance of our strategy to the direct application of the baseline PPO algorithm to the motor angles. Figure 6 illustrates the evolution of the reward distribution throughout the training process for the task without the presence of obstacles. The policy generated by our

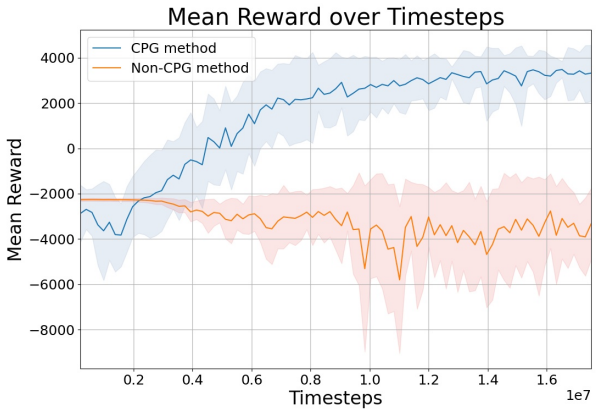

Fig. 6: Reward curves with standard deviation for the non-obstacle-based environment for the goal-reaching task, comparing the performance for the CPG and non-CPG strategies.

method converges to a solution at around 15 million timesteps, after experiencing over 3500 episodes. Its learned steering capabilities are displayed in the trajectory plots of Figure 7. For all goal positions on the opposite wall, the robot is able to reach it, executing a smooth curve within the XY-plane. Conversely, the non-CPG approach fails to learn meaningful policies within the given training time frame, being vastly outperformed by the VAE-CPG approach.

For the goal-reaching task in the presence of obstacles, our architecture once again considerably outperforms the baseline. As depicted in Figure 8, the CPG method steadily increases its rewards throughout the training process. The trajectories

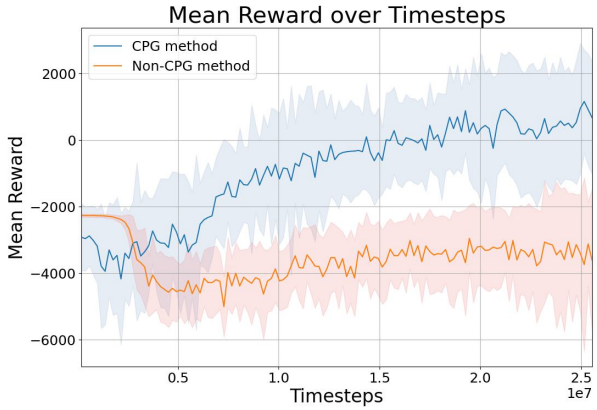

Fig. 8: Reward curves with standard deviation for the obstacle-based environment for the goal-reaching task, comparing performance for the CPG and non-CPG strategies.

presented in Figure 9 show that the robot alters its trajectory to account for the presence of obstacles, avoiding them in order to reach the goal.

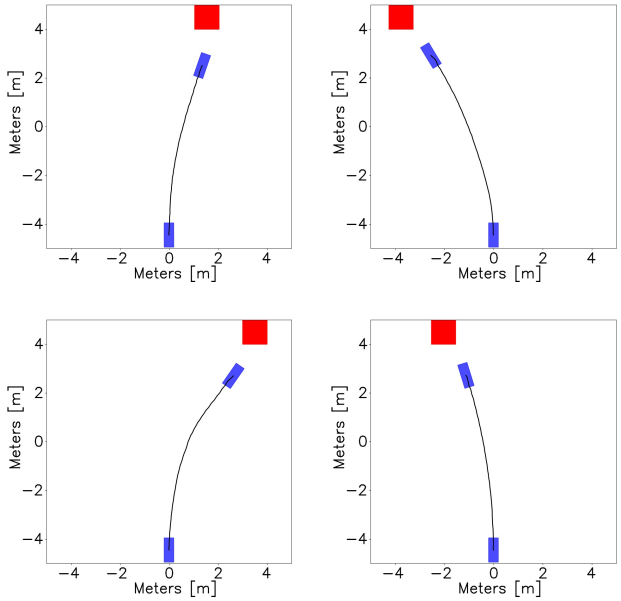

Fig. 7: Successful task completion in the no-obstacle-based environment. Red: Goal position. Blue: Robot initial and final positions. Black: Trajectory of the robot during episode.

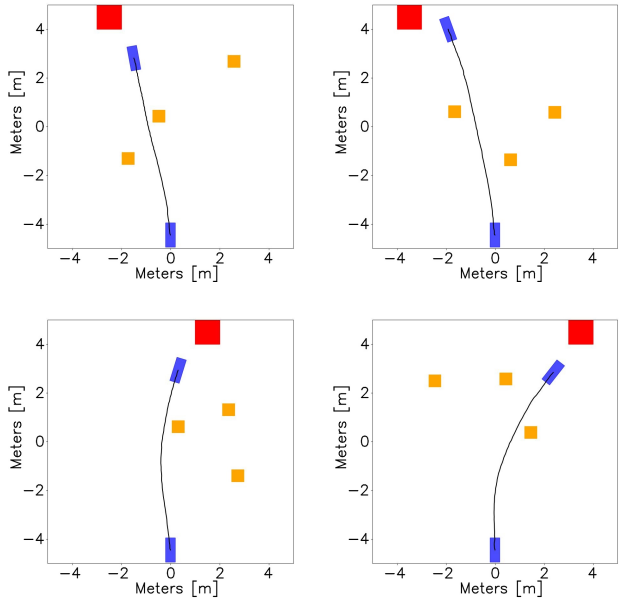

Fig. 9: Successful task completion in the obstacle-based environment. The orange squares show the locations of the obstacles.

In [2], the authors introduced an end-to-end hierarchical reinforcement learning architecture designed for learning quadruped motions based on visual inputs. They evaluated their method using a goal-finding task within a maze-like environment, akin to the obstacle-filled interior setting in our experiments. To showcase the efficiency of our approach, we compare both methods, as illustrated in Table V.

Architecture	**Episodes**	**Input Size**	**Pre-training**
VAE-CPG	6000	512x512x3	Yes
Hier. End-to-End	250000	16x16x1	No

TABLE V: VAE-CPG and Hierarchical End-to-End methods. A comparison of the number of episodes required to learn the policy, the input image dimensions, and the requirement of pre-training are shown.

We compare the episode counts needed by each method to achieve a policy capable of solving its designated task. The VAE-CPG architecture demonstrates significantly faster learning, as it only requires 6000 trials, compared with 250,000 trials required by the Hierarchical End-to-End method to acquire the requisite behaviors for task completion. However, this difference stems from the utilization of pre-training in our VAE-CPG method for its visual module, in contrast to the concurrent learning of terrain representation and locomotion primitives in the End-to-End method. The trade-off of having to collect a dataset of images and pre-train a VAE is balanced by the increase in sample efficiency. Additionally, the End-to-End method uses small 16x16 pixels depth images, which could hinder its ability to interpret more visually complex scenarios. In comparison, our method processes 512x512 pixels RGB images, and therefore has the potential to adapt to more realistic environments.

VI. Retraining and Limitations

The experimental setups in which our approach was tested, while they incorporate some random elements to diversify each episode, result in relatively controlled training environments given the fact that the image-processing module is pre-trained using images taken from within them. This impacts the ability of the model to generalize to unknown scenarios, which would require the retraining of policies in order to adapt to new environments. In such scenarios, Transfer Learning could be employed to shorten training times for new tasks by reusing learned policies [2], [34]. Due to the modularity of the VAE-CPG architecture, even if latent representations of new visual inputs need to be learned, the gait learning MLP policy can be reused. This allows training to start not from scratch, but rather with prior knowledge of how to walk, expediting the exploratory stage of the learning process.

A limitation of our method comes from one of the architecture's main features, that being the use of CPGs for action space reduction. Although the joint trajectories given by CPGs can accelerate the convergence to walking gaits, they also constrain the agent's ability to perform more complex movements. This limitation hinders the agent's capability to traverse more unstructured terrain or to learn additional behaviors, such as climbing over obstacles. To address this, a more complex trajectory generating model could be implemented, which gives each leg more autonomy over its movement, while still maintaining the notion of periodic movements.

VII. Conclusion

In this article, we have presented the VAE-CPG architecture for learning stable and adaptable quadruped gaits via the fusion of Variational Autoencoders and Deep Reinforcement Learning. State-of-the-art learning methods in this area lack sample efficiency, leading to long training times to achieve meaningful gait policies. This in turn hinders the implementation of legged vehicles in real environments, such as construction sites. This was demonstrated with the presented VAE-CPG architecture, since it required less training experience episodes than the hierarchical end-to-end vision-based approach (see Table V).

The presented VAE-CPG architecture was compared with, and outperforms, the baseline PPO algorithm in two navigation tasks. The first corresponds to goal reaching in an obstacle free environment, requiring the learning of simultaneous walking and steering. The second corresponds to an obstacle filled environment, in which the robot learns to simultaneously walk, steer and avoid the obstacles.

Further, the presented algorithm allows for the processing of images from relatively complex scenes in comparison with state-of-the-art implementations, which typically only allow for simplistic, low dimensionality images, such as purely depth images.

The implications of the concepts presented in this article allow for the learning of appropriate walking behaviors in more realistic, richer virtual environments.

Acknowledgments

The authors acknowledge "Agencia Nacional de Investigacion y Desarrollo" (ANID) Fondecyt project 1231658, the Department of Electrical Engineering, Universidad de Chile as well as the US Air Force Office of Scientific Research (AFOSR) grant 23IOS020 and ANID/PIA Project AFB180004.

References

[1] A. Kumar, N. Paul, and S. N. Omkar, "Bipedal walking robot using deep deterministic policy gradient," 7 2018. [Online]. Available: http://arxiv.org/abs/1807.05924

[2] D. Jain, A. Iscen, and K. Caluwaerts, "From pixels to legs: Hierarchical learning of quadruped locomotion," *arXiv preprint arXiv:2011.11722*, 2020.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 7 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[4] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty: An experimental comparison on a dynamic bipedal walker," *Annals of Mathematics and Artificial Intelligence*, vol. 76, pp. 5–23, 2 2016.

[5] M. Hutsebaut-Buysse, K. Mets, and S. Latré, "Hierarchical reinforcement learning: A survey and open research challenges," pp. 172–221, 3 2022.

[6] E. Coumans and B. Yunfei, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[7] R. Tedrake, T. W. Zhang, and H. S. Seung, "Stochastic policy gradient reinforcement learning on a simple 3d biped," 2004.

[8] D. Gong, J. Yan, and G. Zuo, "A Review of Gait Optimization Based on Evolutionary Computation," *Applied Computational Intelligence and Soft Computing*, vol. 2010, pp. 1–12, 2010.

[9] C. Tao, J. Xue, Z. Zhang, F. Cao, C. Li, and H. Gao, "Gait Optimization Method for Humanoid Robots Based on Parallel Comprehensive Learning Particle Swarm Optimizer Algorithm," *Frontiers in Neurorobotics*, vol. 14, 1 2021.

[10] W. Lacarbonara, B. Balachandran, M. J. Leamy, J. Ma, J. A. Tenreiro, M. Gabor, and S. Editors, "Advances in Nonlinear Dynamics," Tech. Rep. [Online]. Available: https://link.springer.com/bookseries/16666

[11] Z. Sun, H. Li, J. Wang, and Y. Tian, "A Gait Optimization Smoothing Penalty Function Method for Bipedal Robot via DMOC," in *IFAC-PapersOnLine*, vol. 48, no. 28. Elsevier B.V., 2015, pp. 1148–1153.

[12] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[13] G. Comanici and D. Precup, "Optimal policy switching algorithms for reinforcement learning," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, 2010, pp. 709–714.

[14] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[15] K. Marino, A. Gupta, R. Fergus, and A. Szlam, "Hierarchical rl using an ensemble of proprioceptive periodic policies," in *International Conference on Learning Representations*, 2018.

[16] K. Y. Levy and N. Shimkin, "Unified inter and intra options learning using policy gradient methods," in *European Workshop on Reinforcement Learning*. Springer, 2011, pp. 153–164.

[17] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," *Advances in neural information processing systems*, vol. 31, 2018.

[18] T. Hemker, M. Stelzer, O. von Stryk, and H. Sakamoto, "Efficient walking speed optimization of a humanoid robot," *The International Journal of Robotics Research*, vol. 28, no. 2, pp. 303–314, 2009.

[19] D. J. Lizotte, T. Wang, M. H. Bowling, D. Schuurmans *et al.*, "Automatic gait optimization with gaussian process regression." in *IJCAI*, vol. 7, 2007, pp. 944–949.

[20] M. MacKay-Lyons, "Central pattern generation of locomotion: a review of the evidence," *Physical therapy*, vol. 82, no. 1, pp. 69–83, 2002.

[21] M. R. Dimitrijevic, Y. Gerasimenko, and M. M. Pinter, "Evidence for a spinal central pattern generator in humans a," *Annals of the New York Academy of Sciences*, vol. 860, no. 1, pp. 360–376, 1998.

[22] A. Sproewitz, L. Kuechler, A. Tuleu, M. Ajallooeian, M. D'Haene, R. Moeckel, and A. Ijspeert, "Oncilla robot: a light-weight bio-inspired quadruped robot for fast locomotion in rough terrain," in *5th International Symposium on Adaptive Motion of Animals and Machines*, 2011.

[23] L. Righetti and A. J. Ijspeert, "Pattern generators with sensory feedback for the control of quadruped locomotion," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 819–824.

[24] G. Sartoretti, S. Shaw, K. Lam, N. Fan, M. Travers, and H. Choset, "Central pattern generator with inertial feedback for stable locomotion and climbing in unstructured terrain," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5769–5775.

[25] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker, "Universal correspondence network," *Advances in neural information processing systems*, vol. 29, 2016.

[26] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song, "Densephysnet: Learning dense physical object representations via multi-step dynamic interactions," *arXiv preprint arXiv:1906.03853*, 2019.

[27] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in rgb-d images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2930–2937.

[28] X. B. Peng, G. Berseth, K. Yin, and M. van de Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)*, vol. 36, no. 4, 2017.

[29] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 674–10 681.

[30] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations*, 2014.

[31] S. Gay, J. Santos-Victor, and A. Ijspeert, "Learning robot gait stability using neural networks as sensory feedback function for central pattern generators," 2013, pp. 194–201.

[32] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, pp. 1–8, 2021.

[33] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke, "Policies modulating trajectory generators," 10 2019. [Online]. Available: http://arxiv.org/abs/1910.02812

[34] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13 344–13 362, 2023.