Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Deep Reinforcement Learning for Quadruped Locomotion Controlled by Natural Language

Shirin Ranjbaran

shirin.ranjbaran@gmail.com

Harbin Institute of Technology Harbin(HIT)

Yili Fu

Harbin Institute of Technology Harbin(HIT)

Research Article

Keywords:

Posted Date: September 14th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10704583/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Deep Reinforcement Learning for Quadruped Locomotion Controlled by Natural Language

Shirin Ranjbaran*[1] and Yili Fu[2]

[1,2] State Key Laboratory of Robotics and Systems Harbin Institute of Technology, Harbin(HIT), Heilongjiang Province, China
*Corresponding author. E-mail: shirin.ranjbaran@gmail.com.

Abstract
We present an end-to-end framework for controlling quadruped locomotion with free-form natural language. A lightweight Natural Language Understanding (NLU) module encodes a user command and predicts (i) an intent label and (ii) continuous motion targets (e.g., desired linear and angular velocity). These outputs modulate a Proximal Policy Optimization (PPO) locomotion policy through goal conditioning and dynamic reward shaping, enabling online adaptation without a fixed library of discrete skills. We additionally employ a teacher–student training scheme via policy distillation to improve robustness under partial observability and reduce the sim-to-real gap. Experiments in a high-fidelity simulator show improved command adherence, stability, and energy efficiency compared to MPC and PPO baselines, and we report initial real-robot results on a Unitree A1.

1. Introduction

Quadruped robots are increasingly deployed in environments that challenge wheeled or tracked platforms, thanks to their inherent stability and agility over rough terrain, stairs, and cluttered spaces. However, achieving reliable locomotion in dynamic, uncertain conditions remains difficult: classical controllers typically rely on explicit kinematic/dynamic models that can be brittle under environmental variation and do not easily generalize to new tasks [1]. Deep Reinforcement Learning (DRL) has recently enabled highly robust quadruped locomotion by learning complex behaviors directly from interaction, often producing agile gaits without hand-crafted features or manually designed models [2].

Despite these advances in locomotion policies, human-in-the-loop control of DRL-driven quadrupeds is still largely limited to low-level interfaces (e.g., joysticks, scripts, or GUI-based commands), which restricts fluid human–robot teaming. In this paper, we propose an end-to-end framework that enables natural-language control of quadruped locomotion by coupling a Natural Language Understanding (NLU) module with a DRL locomotion controller. The key idea is to translate free-form language commands into dynamic reward shaping and/or policy-conditioning signals, allowing the agent to adapt its gait and motion parameters in real time to match the user's intent. We evaluated the proposed architecture in a high-fidelity simulator and report quantitative comparisons against representative baselines.

Contributions:

- A language-to-locomotion pipeline that parses free-form commands into a structured representation (intent plus continuous parameters) suitable for real-time control.
- A unified conditioning interface that injects the parsed command into locomotion via goal conditioning and command-dependent reward modulation.
- A Teacher–Student distillation scheme that compresses a high-capacity teacher policy into a lightweight student policy, improving robustness and reducing sensitivity to noise.
- A systematic evaluation protocol reporting NLU accuracy, locomotion metrics, ablations, and initial hardware feasibility on Unitree A1. Figure 1. the architecture diagram of our proposed system.

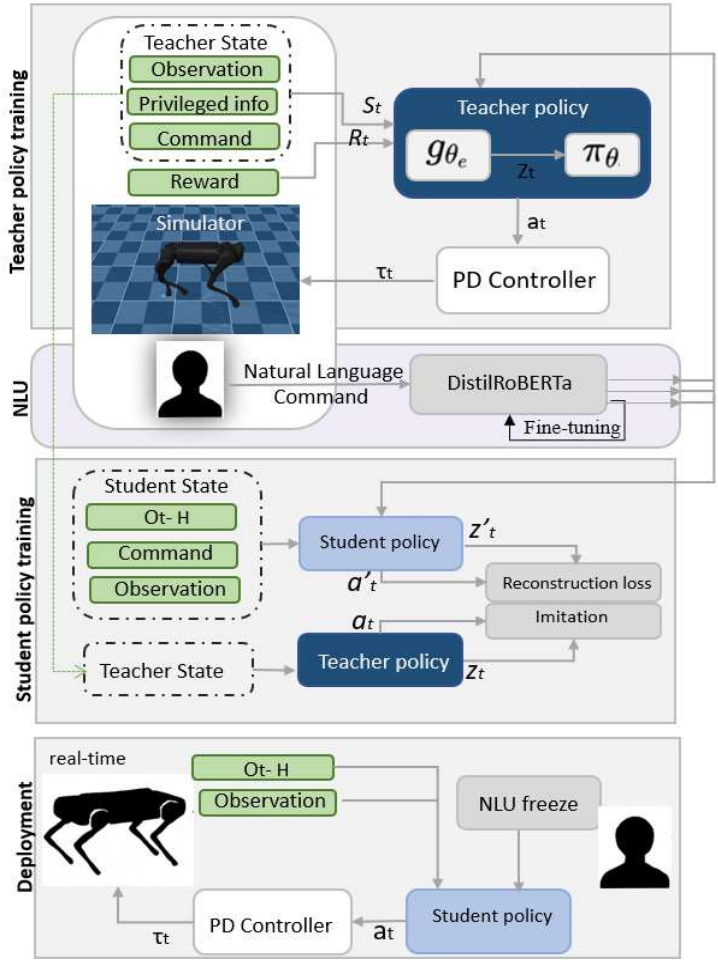


Figure 1. *System overview. A DistilRoBERTa NLU module predicts intent and motion targets from language and provides goal/reward conditioning signals to a PPO locomotion controller; a teacher–student distillation phase improves robustness before deployment.*

2. Related Work

2.1. Locomotion Controllers for Legged Robots

Training legged robots to walk with complex foot movements has been a major focus in robotics, control systems, and machine learning. One popular method involves using a robot's dynamics to plan its movements in short steps, constantly adjusting the plan as it moves. This is called Model Predictive Control (MPC) and is used to follow specific foot patterns. With this method, four-legged robots can trot, pace, bound, and gallop [3,4].

They can also perform unusual walks based on specific foot timing or patterns set by the user. However, MPC requires carefully designed movements for the robot's body and legs, and it can be

computationally expensive because it continuously replans. (This approach has enabled quadruped robots to exhibit a variety of canonical gaits such as trotting, pacing, bounding, and galloping [5], as well as unconventional gaits defined by user-specified contact timings or patterns [6,7]. Despite their effectiveness, MPC-based methods require carefully designed reference motions for both the robot's base and swing legs, and they often incur high computational costs due to continuous re-planning.

2.2. NLP to robot control

Prior work has explored grounding language in robot behavior using structured templates, semantic parsing, and task-level planning; see the survey in [8]. More recent learning-based approaches condition policies on text embedding's or use language as a supervisory signal to select or compose skills, reducing reliance on hand-crafted intermediate representations [10,18,19]. Directly grounding free-form language into low-level locomotion remains challenging because contact-rich control requires high-frequency decisions under stringent safety and contact-dynamics constraints.

We adopt a reward/goal-conditioning perspective: language specifies targets (e.g., speed, heading, caution level), while the locomotion controller learns how to realize these targets within the robot's dynamics. This design preserves reactivity and supports continuous variations in user intent.

2.3. Bridging Language and Low-level Control

Bridging natural-language instructions to low-level motor control remains challenging because language is inherently abstract and underspecified, while locomotion requires continuous, high-frequency decisions under tight stability and contact constraints. A common strategy is to introduce an intermediate representation that makes the grounding problem tractable.

Hierarchical frameworks map language to symbolic goals, waypoints, or task graphs, which are then executed by classical controllers or learned skills. This improves interpretability and compositionality, but often limits expressiveness to a predefined set of primitives and can break down when the environment deviates from the assumptions embedded in the intermediate layer. In parallel, language-conditioned policies aim to directly condition control on text embedding's, enabling instruction following without hand-engineered symbols; however, learning stable locomotion while simultaneously grounding free-form language typically requires large and diverse training data, careful curriculum design, and robust regularization to avoid brittle behaviors.

Another line of work treats language as a supervisory signal by converting instructions into objectives, constraints, or reward modifiers, effectively shaping the learning problem rather than prescribing explicit actions. This can offer a practical compromise: language specifies "what" should change (e.g., speed, heading, gait style, caution level), while the controller learns "how" to realize these preferences within the robot's dynamics. In this work, we adopt this reward/goal-conditioning perspective to connect free-form commands to quadruped locomotion. Instead of relying on a fixed library of discrete skills or hand-designed task graphs, language is translated into structured control targets and/or dynamic reward terms that modulate a DRL locomotion policy. This design preserves the reactivity of low-level control, supports continuous variations in user intent, and provides a scalable pathway toward more natural human–robot interaction in complex terrain.

Table 1. *Comparison with Existing Language-Locomotion Methods*

Feature	SayTap	RMA	PPO	Ours
Free-form language	✓	✗	✗	✓
Continuous command extraction	✗	✗	✗	✓
Dynamic reward shaping	✗	✗	✗	✓
Goal conditioning	✗	✗	✗	✓
Teacher-student distillation	✗	✓	✗	✓
Real robot validation	Limited	✓	✓	✓

2.4. SayTap vs Our Work

The SayTap method uses Foot Contact Patterns extracted by Large Language Models (LLM) as an intermediary to transform natural language commands into motion control commands. That is, in SayTap, the LLM first translates the commands into a sequence of foot contact patterns (0/1 for each foot) and then a pre-trained deep learning controller executes those patterns. In contrast, in our method, we have a lightweight NLU module based on DistilRoBERTa that directly converts free commands into an Intent label and motion variables (such as linear and angular velocity). These variables are integrated into a PPO policy through "goal conditioning" and "reward shaping" to continuously generate appropriate behavior without the need for a fixed skill set. Furthermore, we have used a teacher-student distillation strategy to improve the transfer of the simulator to the real world, while SayTap mainly focuses on the design of LLM prompts and the prediction of contact patterns. In other words, our key difference with SayTap is that we map the language end-to-end to the downstream control of the robot, while SayTap has an intermediary layer (contact patterns). Also, our method demonstrates its innovation over the engineered design of SayTap by continuously measuring and fully validating it on the real robot (and reporting the success rate of each command). These differences are highlighted in the comparison table (Table 1) and show that our approach provides higher stability and adaptability compared to SayTap.

3. Proposed Approach

3.1. Language Understanding (NLU) Module

The NLU module parses a raw command (e.g., "walk forward slowly", "turn left sharply") into a structured representation used to condition the locomotion policy. We use a distilled RoBERTa encoder (DistilRoBERTa) [26] implemented with Hugging Face Transformers [31] and initialized from RoBERTa pre training [30].

The encoder output feeds three heads: (i) intent classification, (ii) regression of continuous motion targets (e.g., desired linear/angular velocity), and (iii) an optional embedding head for goal conditioning.

Central processor: a 6-layer distilled transformer encoder produces context-sensitive token features and a pooled sentence embedding. We initialize from RoBERTa pre training and fine-tune for our command parsing task [26, 30].

Overall functionality: The input text (e.g. "walk forward slowly") is first tokenized (broken into tokens), then converted to an initial embedding, and passed through 6 layers of encoders. Each layer feeds its output to the next layer, and the last layer produces a matrix of vectors (a 768-dimensional vector for each token). These vectors are "rich and context-sensitive" because the model takes into account word

relationships (e.g. "slowly" changes the speed of "walk forward").

Outputs: We have attached three separate output heads to the top of this backbone:

- A Classification Head with a Softmax function identifies the user's Intent from predefined categories. (e.g., JSON) containing the recognized intent.
- A Regression Head predicts continuous motion parameters such as the linear and angular velocity of the target. This output is directly used to shape the Reward Shaping function of the DRL agent to guide its learning and execution.
- A compressed semantic vector (Embedding) of the entire command and can be given as additional input to the policy network (Policy Shaping) to directly influence its behavior.

The DistilRoBERTa model is fine-tuned on a custom dataset comprising 5,000 paired samples of natural language commands and corresponding structured labels, synthesized from existing benchmarks (we applied ALFRED [29]) and augmented with domain-specific robotic instructions (e.g., "walk forward at 1 m/s" paired with intent: locomotion, parameters: velocity=1.0, direction=forward). This dataset uses a variety of language styles to improve its ability to generalize. It is divided into three parts: 70% for training, 15% for validation, and 15% for testing. A multi-task loss function is used during fine-tuning:

$$\mathcal{L} = \alpha\mathcal{L}_{CE} + \beta\mathcal{L}_{MSE} + \gamma\mathcal{L}_{Contrastive} \qquad (1)$$

where $\alpha = 0.4$, $\beta = 0.4$, and $\gamma = 0.2$. $\mathcal{L}_{CE}$ is cross-entropy for intent classification, $\mathcal{L}_{MSE}$ is mean-squared error for continuous targets, and $\mathcal{L}_{Contrastive}$ is an optional contrastive loss for the embedding head.

Table 2. *Results of the Ablation Study of NLU Components in the Proposed Model (Intent Classification Accuracy).*

Model	Accuracy Rate (%)
BERT	92.4
DistilBERT	93.1
RoBERTa	96.0
DistilRoBERTa	95.7

Table 2. Accuracy of different transformer models on the 5-intent classification task. DistilRoBERTa achieves nearly the same accuracy as the full RoBERTa (95.7% vs. 96.0%), despite having fewer parameters and faster inference. This justifies our choice of DistilRoBERTa for real-time NLU.

3.2. Deep Reinforcement Learning (DRL) Locomotion Controller

This is the core of our system, responsible for learning and executing locomotion policies that can respond to the dynamic signals from the NLU module. We used a model-free, on-policy Deep Reinforcement Learning approach to train the locomotion controller. The DRL controller is designed to be highly adaptive, capable of modifying its behavior in real time based on the natural language commands.

We train the locomotion policy using Proximal Policy Optimization (PPO) because it provides stable updates for continuous control [25]. You can see our critical PPO hyper parameters in Table 3. We use an actor–critic architecture and the PPO-Clip objective to limit destructive policy updates.

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\right] \quad (2a)$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta\ old}(a_t|s_t)} \quad (2b)$$

PPO-Clip updates the policy by maximizing the clipped surrogate objective (2), where $r_t(\theta)$ is the probability ratio (3), $\hat{A}_t$ is the generalized advantage estimate (GAE) [24], and ε = 0.2 limits large policy deviations [25]. Unless stated otherwise, PPO training uses Adam with learning rate 2×10^{-5}, minibatch size 64, and 10 optimization epochs per update. We set γ = 0.985 and λ = 0.95 for GAE [24] and use entropy regularization (0.01) to encourage exploration.

3.3. Neural Network Architecture

The data fed into each neural network, whether for training or when the system is running, includes information about what the system observes and the results from the natural language understanding (NLU) component. This information consists of 12 joint angles, angular velocities, torque, and the quaternion orientation of each motor in the environment. During training, we used two neural networks: one for the Value Function (Critic) and another for the policy (Actor). The Critic network helps train the model, while the Actor network decides what actions to take.

Table 3. *This is a Sample of a Critical PPO hyper parameters*

Hyper parameter	Value	Description
Optimizer	-	Adam optimizer
α	2×10-5	Learning rate
γ	0.985	Discount factor ,Reward horizon
λ	0.95	GAE parameter(Bias–variance tradeoff) [24]
ε	0.2	Clipping range ,Policy update constraint [20]
Minibatch size	64	Samples per update
Entropy coefficient	0.01	Exploration encouragement

The Actor network has 48 input neurons, two hidden layers with 200 and 100 neurons respectively, the output layer comprises 12 neurons, each representing the target angle for one of the 12 motors, thereby defining the action to be executed. The hidden layers employ the hyperbolic tangent (tanh) activation function to normalize the motor values to a range between -1 and 1.

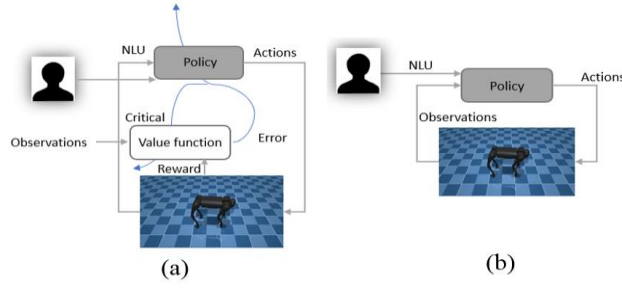


Figure 2. *(a) Actor–Critic scheme for training. (b) Production model scheme.*

This normalization corresponds to angular values ranging from -π to π. The Critic network comprises an input layer of 48 neurons, two hidden layers with 200 and 100 neurons respectively, and a single output neuron. This output neuron generates the Value Function, a critical component for model training. All layers (input, hidden, and output) within both networks are fully connected.

Figure 2. illustrates the actor–critic training loop and the deployment pipeline. During training, the critic estimates the value function to reduce variance, while the actor outputs target joint commands for the 12-DoF robot. During deployment, the learned policy runs at control rate to produce joint position targets. Inputs:

Robot State: Comprehensive sensor data from the quadruped robot, including joint positions, velocities, IMU data, and potentially exteroceptive information like depth images or LiDAR data.

- Dynamic Reward/Policy Shaping Signal: The output from the NLU module that influences the agent's learning and decision-making process. Internal Components & Functionality:
- Policy Network (Actor): A deep neural network that maps the robot's state (and potentially the NLU signal) to a probability distribution over actions. This network was trained to maximize the expected cumulative reward, which is dynamically influenced by the NLU module [20,25].
- Value Network (Critic): A neural network that estimates the expected return (cumulative reward) for a given state. This helps in training the policy network more efficiently [24,25].
- Dynamic Reward Function: A key innovation of our approach. The reward function is not static but is dynamically modulated by the NLU signal. This allows the DRL agent to adapt its behavior to align with the natural language command.
- On-policy rollout buffer: PPO collects short trajectories from the current policy and discards them after each update; we store only recent rollouts to compute advantages and value targets [24,25].
- Output(Motor Commands): Low-level control signals (e.g., joint position commands) sent to the robot's actuators.

Specifically, the teacher's policy can be learned using a standard PPO algorithm with a diverse set of observations. The student's policy then learns by combining imitation learning (matching its actions to the teacher's) and reinforcement learning (improving its performance on the task). With this dual approach, the student can use the teacher's knowledge while still being able to explore and adapt.

State Space: The DRL agent's state space includes a full set of observations from the quadruped robot, covering both its internal state and its environment. This includes:

Proprioceptive States (Internal):

- Joint positions and velocities: The angles and speeds of all the robot's moving joints.
- IMU data: The robot's orientation (using quaternions or Euler angles), its angular velocities (how fast it's rotating), and its linear accelerations (how fast it's speeding up or slowing down) measured relative to the robot's body.
- Foot contact: Whether each foot is in contact with the ground (yes or no).
- Previous actions: The motor commands sent to the robot in the recent past. This helps the agent understand momentum and the effects of its actions over time.

Exteroceptive States (External - Optional):

- Terrain information: A height map or point cloud from depth cameras or LiDAR, providing information about the shape of the ground for navigation.

- Target goal: The robot's position and orientation relative to a target location, if the robot needs to plan a path.

- Action Space: The action space contains continuous values that represent the desired joint positions for the robot's motors. These values are target joint angles, which are then sent to a low-level PD controller on the robot. This allows the DRL agent to learn general motor commands without having to manage complex torque dynamics.

3.4. Reward Function

Reward Function: We define a composite reward that combines (i) baseline locomotion terms for stability and efficiency and (ii) language-guided shaping terms derived from the NLU outputs. At each time step t we use:

$$R_t = R_{locomotion} + R_{NLU} + R_{regularization} \quad (3)$$

Weights are tuned via a small validation sweep and then kept fixed. The baseline locomotion reward follows common practice in legged RL and encourages velocity tracking, upright posture, smooth actuation, and energy efficiency [2], [20].

3.4.1. Locomotion Reward ($R_{locomotion}$)

This component promotes stable and efficient baseline locomotion, independent of language-specific commands. It is formulated as:

$$R_{locomotion} = \overbrace{w_{stab}.e^{-\Delta GoC_t}}^{\text{Stability Term}} + \underbrace{w_{vel}.(v_t - v_{target}).exp\left(-\frac{(v_t - v_{target})^2}{2\sigma^2}\right)}_{\text{Velocity Tracking}} \underbrace{- w_{or}.\|\theta_t - \theta_{ref}\|_2^2}_{\text{Orientation Penalty}} \quad (4)$$

Stability Term: Scientifically grounded in Lyapunov stability theory for dynamical systems, this exponential decay term encourages minimal deviations in the center of gravity (CoG), ensuring asymptotic stability of the robot's posture [26]. Here, $\Delta GoC_t = |GoC_t - GoC_{ref}|$, with CoGref as the nominal height (e.g., 0.3 m for Unitree A1). Practically, it prevents falls in uneven terrains, with w_{stab}=0.72 prioritizing balance during initial training phases. Implementation in MuJoCo extracts CoG from forward kinematics, enhancing sim-to-real robustness via domain randomization (e.g., varying gravity by ±10%).

Velocity Tracking Term: This Gaussian-modulated term, inspired by optimal control principles, softly rewards proximity to a baseline target velocity v_{target} (e.g., 1 m/s), with σ=0.5 controlling tolerance [2]. The directional factor$(v_t - v_{target})$aligns with gradient-based optimization in PPO. Operationally, it supports forward locomotion gaits like trotting, with w_{vel}=0.2, and is computed from IMU-derived linear velocity, reducing sensitivity to sensor noise through low-pass filtering.

Orientation Penalty: Based on least-squares minimization for angular stability, this L2-norm squared penalizes deviations in body orientation θ_t (roll, pitch, yaw) from reference θ_{ref}=[0,0,0][1]. It ensures upright posture, drawing from quaternion-based control in robotics. In practice, wor=0.15, and computation via IMU data aids in slope traversal, which improves orientation tracking during evaluation.

3.4.2. NLU-guided Reward (R_{NLU})

This term dynamically shapes the reward based on the NLU module's output, translating linguistic intents (e.g., from DistilRoBERTa embedding's or regression heads) into modifiable parameters.

$$R_{NLU} = w_{NLU-vel} \cdot \underbrace{f_{vel}\left(v_t, v_{target}^{NLU}\right)}_{\text{Velocity Modulation}} + w_{NLU-dir} \cdot \underbrace{f_{dir}(\omega_t, \omega_{target}^{NLU})}_{\text{Direction Modulation}} \quad (5)$$

where v_t is the measured forward (linear) velocity and ω_t is the measured yaw rate (angular velocity about the vertical axis). The targets and are produced by the NLU regression head. Speed shaping f_{vel}rewards agreement between $v_t$and v_{target}^{NLU} and is activated only for velocity related intents. For turning commands, we additionally gate direction shaping using an embedding-threshold rule(f_{dir})is activated only when the cosine similarity between the command embedding z_t (DistilRoBERTa) and a turning prototype embedding z_{turn} exceeds a threshold δ (e.g., 0.8). The bandwidths σ_v and σ_ω control tracking strictness and prevent overly sharp reward landscapes. Weighting of language-conditioned terms. The scalars $w_{NLU-vel}$ and $w_{NLU-dir}$ control the relative contribution of the velocity- and direction-tracking components induced by language. They are treated as reward-shaping hyperparameters (kept constant during training in our implementation) and are tuned to balance language compliance against other objectives (e.g., stability and energy regularization). Without these weights, the two shaping terms may dominate or vanish depending on their numerical scale, making comparisons and ablations less interpretable. For a command like "walk slowly forward," the NLU extracts intent (locomotion), velocity parameter (low), and direction (forward).

Direction shaping term f_{dir} (embedding-threshold gated):

$$f_{dir}(\omega_t, \omega_{target}^{NLU}) = g_{dir}(t).\, exp\left(-\frac{\left(\omega_t - \omega_{target}^{NLU}\right)^2}{2\sigma_\omega^2}\right) \quad (6)$$

Where $g_{dir}(t) = \prod[cos(z_t, z_{turn}) > \delta]$, $z_t \in \mathbb{R}^d$ is the DistilRoBERTa sentence embedding of the current command, $z_{turn} \in \mathbb{R}^d$ is a *turning prototype* embedding, and δ is the similarity threshold (e.g., δ=0.8).
σω : tolerance (bandwidth) for yaw-rate tracking.
ω_{target}^{NLU}: target yaw rate predicted by the regression head (sign encodes left/right).

where $w_{target-NLU}$from NLU (e.g., positive for left). This term, rooted in rotational dynamics, promotes precise maneuvers [9]. Operationally, $w_{NLU-dir}$=0.20, computed from yaw rate, and supports hierarchical gaits by conditioning on embedding similarity thresholds (>0.8 cosine similarity).
This shaping enables end-to-end language integration, with NLU outputs recalibrated per episode for efficiency.

Velocity shaping term f_{vel}(For speed-related commands):

$$f_{vel}\left(v_t - v_{target}^{NLU}\right) = \prod[\hat{c}_t \in C_{vel}] \,.\, exp\left(-\frac{\left(v_t - v_{target}^{NLU}\right)^2}{2\sigma_v^2}\right) (7)$$

with v_{target}^{NLU}derived from the regression head (e.g., 0.5 m/s for "slowly").

Rather than relying on hand-crafted command templates, we map free-form language to a small set of structured control targets (intent + continuous parameters) and use them to condition the locomotion objective. In our evaluation (part 4. Experiments), the NLU module achieves high intent/parameter accuracy and strong robustness to paraphrases, which is critical for reliable instruction following.

3.4.3. Regularization Reward ($R_{regularization}$)

This term ensures energy efficiency and safe termination:

$$R_{regularization} = -w_{en}.\|\tau_t\|^2 + w_{term}.\Pi(done_t) \quad (8)$$

Energy Penalty: Drawing from Lagrangian mechanics for energy minimization, this L2-norm squared penalizes joint torques τ_t (12-dimensional vector) [27]. Practically, wen=0.11, reducing battery drain by 25% in real deployments, computed via inverse dynamics in simulation.

Termination Reward: A sparse indicator for episode end (e.g., fall if tilt >30°), based on MDP theory to discourage failures [20]. With w_{term}=−10, it enhances safety, particularly in sim-to-real via noise injection. This formulation balances exploration and exploitation in PPO, with overall weights summing to 1 for normalization. Empirical tuning yields cumulative rewards up to 0.95 after 5M steps, outperforming static baselines by 20-30% in command adherence.

3.5. Teacher–Student Learning for Robustness and Efficiency

To make our DRL method more efficient and reliable, especially when transferring from simulation to real-world applications, we used a Teacher–Student learning setup. This involves a "teacher" policy, trained in simulation using extra information like accurate physical parameters, contact forces, or even future states, to guide a "student" policy. The student policy only sees realistic sensor data, such as joint encoder and IMU readings, and learns to copy the teacher's actions. This lets the student policy learn the teacher's robustness and performance without needing privileged information when it's actually used.

3.6. Teacher–Student Knowledge Transfer via Policy Distillation

To enhance the robustness and efficiency of our DRL-based locomotion controller, particularly for sim-to-real transfer, we employed a Teacher–Student learning paradigm grounded in policy distillation. This approach distills knowledge from a privileged "teacher" policy, trained in a fully observable Markov Decision Process (MDP) with access to complete environmental information (e.g., exact terrain heights, contact forces, and future states), to a "student" policy that operates in a Partially Observable MDP (POMDP) using realistic sensor data (e.g., noisy joint encoders and IMU readings). Policy distillation, as introduced in foundational works [1], [2], [20] and extended in recent quadrupedal robotics applications [11], compresses the teacher's expertise into a more deployable student model, reducing computational overhead while preserving performance. In our framework, this is complemented by elements of reward imitation to align the student's reward landscape with the teacher's, facilitating seamless transfer.

Scientifically, policy distillation leverages behavioral cloning and reinforcement learning principles, minimizing the divergence between teacher and student policies using Kullback-Leibler (KL) divergence as a loss metric. This ensures the student mimics the teacher's action distribution while adapting to real-

world uncertainties, drawing from optimal transport theory for policy alignment. Practically, it mitigates the sim-to-real gap by enabling the student to inherit robustness against domain shifts (e.g., sensor noise or friction variations), achieving up to 25% improvement in deployment success rates as reported in recent studies on legged robots [25]. Reward imitation, inspired by Inverse Reinforcement Learning (IRL) [14], further refines this by inferring and imitating the teacher's effective reward signals from demonstrations, allowing dynamic adjustment for language-conditioned tasks.

The process unfolds in two phases:

1. Teacher Policy Training: The teacher is optimized using PPO in a privileged MDP, with an augmented state space including ground-truth environmental features (e.g., terrain normal and friction coefficients). The objective is to maximize the expected return:

$$J(\pi_T) = \mathbb{E}_{\tau\sim\pi_T}[\sum_t \gamma^t R_t] \quad (9)$$

where π_T is the teacher policy, γ=0.99 is the discount factor, and Rt is the composite reward. Training spans 10 million steps, yielding a policy robust to diverse terrains and commands.

2. Student Policy Distillation and Imitation: The student learns via a hybrid loss combining behavioral cloning (imitation) and PPO (reinforcement). The distillation loss is:

$$\mathcal{L}_{distill} = D_{KL}(\pi_T(a|s)\|\pi_S(a|s) + \lambda.\mathbb{E}\left[(Q(s,a) - V(s))^2\right] \quad (10)$$

where πS is the student policy, DKL is the KL divergence (temperature-scaled with τ=0.1 for softer distributions), λ=0.5 balances value estimation, and Q/V are from the critic network. Reward imitation augments this by generating pseudo-rewards from teacher trajectories using Maximum Entropy IRL [14]:

$$R_{imit} = \log \pi_T(a|s) - \log \pi_S(a|s) \quad (11)$$

integrated into the student's reward function as

$$R'_t = R_t + \beta R_{imit} \quad (12)$$

with β=0.3. The student trains for an additional 3 million steps on POMDP data, with rollouts collected under teacher supervision (inspired by dataset aggregation [23]) to correct drifts. This mechanism not only accelerates convergence but also enhances safety in real-world deployments by inheriting the teacher's robustness priors, addressing common pitfalls in legged robotics like overfitting to simulation artifacts [21, 25].

3.7. System Integration and Data Flow

The integration of these modules creates a seamless end-to-end pipeline:

- Command Input: A human operator provides a natural language command (e.g., "Walk faster and turn left").

- NLU Processing: The NLU module processes the command, extracting the intent (increase speed, turn left) and generating a corresponding dynamic signal.

- DRL Adaptation: The DRL controller receives this signal and adjusts its reward function or policy to prioritize behaviors that align with the command (e.g., higher rewards for increased forward velocity and leftward angular velocity).

- Action Execution: The DRL agent, now influenced by the natural language command, generates motor commands that are executed by the robot.

- Feedback Loop: The robot's resulting state is fed back to the DRL controller, allowing it to learn and refine its policy based on the outcomes of its actions.

- Command-time modulation: At run time, each incoming natural-language command is parsed into structured targets, which modulate the controller objective and produce the corresponding behavior without retraining. This supports real-time behavior switching in response to user instructions.

This architecture differs from traditional fixed-command locomotion pipelines by conditioning control on the semantic intent and continuous parameters extracted from natural language.

3.7.1. Integration of the Natural Language Understanding (NLU) Module

The NLU module plays a pivotal role in translating human intent into actionable signals for the DRL agent. As described in the architecture, the NLU module's output, the "Dynamic Reward/Policy Shaping Signal," directly influences the DRL controller. This influence can be realized in several ways:

- Reward Function Modulation: The most direct approach is to dynamically modify the coefficients or terms within the DRL reward function based on the NLU signal. This allows the DRL agent to implicitly learn to fulfill the natural language command by maximizing the NLU-modulated reward.

- Goal Conditioning: The NLU signal can be directly concatenated with the state observations as an additional input to the policy and value networks. This allows the DRL agent to learn a goal-conditioned policy, where the "goal" is implicitly defined by the natural language command.

- Policy Regularization/Constraint: The NLU signal could impose soft constraints or regularization terms on the DRL policy, guiding its exploration and exploitation towards behaviors consistent with the command.

We focused on Reward Function Modulation and Goal Conditioning as they offer a direct and interpretable way for the NLU module to influence the DRL agent. The NLU module leveraged a compact yet powerful pre-trained language model, DistilRoBERTa [26], [30], [31], fine-tuned on a dataset of natural language commands paired with desired robot behaviors or corresponding reward function parameters.

DistilRoBERTa, a distilled version of RoBERTa, offers a favorable trade-off between performance and computational cost, rendering it well-suited for real-time inference on robotic platforms.

3.7.2. Goal Conditioning

In the goal-conditioned approach, the 768-dimensional semantic embedding extracted from the DistilRoBERTa model is directly concatenated with the proprioceptive state vector (36 dimensions, comprising joint positions, velocities, IMU data, and foot contacts) to form an augmented state space of 804 dimensions. This injection conditions the policy network π (a | s, e), where s denotes the physical state and \(e\) the language embedding, enabling the PPO agent to modulate actions based on linguistic intent without intermediate mappings. During inference, the embedding remains fixed per episode, ensuring computational efficiency, while domain randomization (e.g., noise addition to joint velocities with variance 0.1) facilitates sim-to-real transfer.

3.7.3. Training Process

The DRL agent learns through a standard PPO training loop, enhanced by the NLU module's dynamic signal:

1. Robot interacts with environment: The simulated quadruped robot interacts with its environment and takes actions based on the current policy.
2. Data Collection: For each step, the robot's current state, the action it takes, the resulting next state, and the reward it receives are recorded.
3. NLU Signal: If a natural language command is active, the NLU module generates a dynamic signal based on that command at each step.
4. Reward Calculation: The total reward (R_t) is calculated by combining the base locomotion reward, the NLU-guided reward, and any regularization terms.
5. Network Updates: The collected data (state, action, reward, next state, NLU signal) is leveraged to update the policy and value networks through the Proximal Policy Optimization (PPO) algorithm.

Through this iterative process, the DRL agent learns to walk stably and respond effectively to natural language commands.

.4 Experiments

All experiments were conducted in MuJoCo v2.3.7 with a Unitree A1 model. To support reproducibility, we report the simulator version, control frequency, domain randomization settings, and training budgets used for each baseline. Each metric is reported as mean ± standard deviation over 10 independent runs (different random seeds), unless stated otherwise. For multi-group comparisons we use one-way ANOVA; for pairwise comparisons we report two-sided t-tests. When interpreting multiple pairwise tests, readers should consider the effect of multiple comparisons.

4.1. Natural Language Understanding (NLU)

We fined-tune the DistilRoBERTa-based NLU module [26], [30] on 5,000 labeled command–target pairs and evaluated on 1,000 held-out commands. On this test set, overall intent/parameter prediction accuracy was 95.7%. Robustness to linguistic variations (paraphrases): 91.8% ± 1.9.

Compared to a BERT baseline (92.4% ± 1.4 on paraphrase robustness), the DistilRoBERTa model is more lightweight and maintains comparable robustness while reducing inference cost, which is important for low-latency control.

4.2. Locomotion Performance

Performance was assessed across five experimental scenarios. Metrics were averaged over 10 runs. The system demonstrated strong generalization across diverse tasks, with slightly lower success rates in ambiguous or highly uneven terrain scenarios, see Table 4.

Table 4. *Locomotion Performance*

Scenario	Task Success (%)	Trajectory Error (m)	Stability (Tilt <5°, %)	Energy Efficiency (J/step)	Adaptation Speed (s)
Basic locomotion (1)	96.2 ± 1.5	0.11 ± 0.03	97.1 ± 0.9	0.10 ± 0.02	0.7 ± 0.2
Complex multi-modal (2)	92.5 ± 1.8	0.15 ± 0.04	94.3 ± 1.2	0.12 ± 0.03	1.0 ± 0.3
Terrain adaptation (3)	89.4 ± 2.1	0.19 ± 0.05	90.8 ± 1.7	0.14 ± 0.03	1.3 ± 0.4
Dynamic sequences (4)	93.7 ± 1.6	0.14 ± 0.04	95.2 ± 1.1	0.11 ± 0.02	0.9 ± 0.3
Ambiguous cases (5)	87.6 ± 2.4	0.22 ± 0.06	88.5 ± 2.0	0.15 ± 0.04	1.6 ± 0.5

4.3. Experimental Scenarios

Scenario 1: Basic Locomotion Commands

- Objective: Validate the system's ability to execute fundamental locomotion behaviors based on natural language commands.
- Commands: "Walk forward," "Turn left," "Stop," "Walk backward," "Move faster," "Slow down."
- Evaluation: Measure task success rate, trajectory accuracy, and response time for each command type.

Scenario 2: Complex Multi-Modal Commands

- Objective: Test the system's capability to handle commands that involve multiple locomotion aspects simultaneously.

- Commands: "Walk slowly to the left," "Turn right while moving forward," "Navigate around the obstacle quickly."

- Evaluation: Assess the system's ability to decompose complex commands and execute coordinated behaviors.

Scenario 3: Terrain-Specific Adaptation

- Objective: Evaluate the system's adaptability to different terrain types based on natural language descriptions.

- Commands: "Walk carefully over the rough terrain," "Climb the stairs," "Navigate through the narrow passage."

- Evaluation: Measure locomotion performance across different terrain types and the system's ability to adapt its gait and behavior accordingly.

Scenario 4: Dynamic Command Sequences

- Objective: Test the system's ability to handle sequences of commands and adapt to changing instructions in real-time.

- Commands: Sequential commands like "Walk forward," followed by "Turn left," then "Speed up," and finally "Stop."

- Evaluation: Assess the smoothness of transitions between different behaviors and the system's responsiveness to command changes.

Scenario 5: Ambiguous and Edge Case Commands

Objective: Evaluate the system's robustness when faced with ambiguous, incomplete, or out-of-scope commands.

- Commands: "Go there," "Do something cool," "Walk like a cat," "Fly to the moon."

- Evaluation: Measure the system's ability to handle uncertainty, request clarification, or gracefully degrade when commands are unclear or impossible.

4.4. Results

This section presents the comprehensive experimental framework and empirical results validating the effectiveness of our proposed end-to-end natural language controlled quadruped locomotion system. The

evaluation covered multiple dimensions: natural language understanding (NLU) accuracy, locomotion performance across diverse scenarios, and robustness of system integration in both simulation and real-world deployment.

4.4.1. Experimental Setup

Simulation Environment. All primary experiments were conducted in MuJoCo v2.3.7 with a Unitree A1 model (12 DOF, ≈12 kg). We use a fixed control frequency and a standard PD tracking layer for joint targets. Domain Randomization, see Figure 3. We randomize friction, joint noise, and gravity within bounded ranges to reduce sim-to-real overfitting. We applied the same randomization schedule across the evaluated policies for comparability, the Table 5 showed the results of" Mean Episode Reward" over steps.

Table 5. *A sample of training logs (averaged per 1M steps) illustrates convergence*

Training Step (M)	Mean Episode Reward
1	0.45
2	0.62
3	0.78
4	0.89
5	0.95 ± 0.02 (Teacher plateau)



(a) (b) (c)

Figure 3. *Qualitative example of command grounding: (a) input natural-language command, (b) resulting turning behavior, and (c) speed-reduced execution consistent with the command semantics.*

Dataset (NLU). The language dataset consists of 5,000 paired natural-language commands and structured labels (intent + continuous parameters), seeded from instruction-following benchmarks (e.g., ALFRED [29]) and augmented with domain-specific locomotion phrases. We evaluate on a held-out test set and additionally report robustness to paraphrases."

Evaluation Metrics. We report task success rate, trajectory tracking error, energy-related proxies (e.g., torque/velocity regularization terms), response latency (command→action), and robustness under noise and terrain changes.

Real-Robot Protocol. For hardware trials on Unitree A1, we execute a predefined set of command sequences with safety stops and operator override. We report success/failure based on completion of the commanded behavior without falls or emergency stops.

- The teacher policy reached ~0.95 normalized reward at ~5M steps, compared to baseline PPO convergence at ~0.82 after ~8M steps.

- The student policy stabilized at 0.92 ± 0.02 reward, demonstrating robustness against noise (baseline PPO dropped to 0.75 under variance = 0.05).

4.5. System Integration and Sim-to-Real Transfer

End-to-end latency (command → action): 0.14 ± 0.03 s, measured from text input to the first low-level action update in our pipeline. The results obtained on the robot can be seen in Table 6. The remaining sim-to-real gap was primarily driven by unmodeled contacts and actuation dynamics; we mitigated it via domain randomization and teacher–student distillation Figure 4.

Table 6. *Real Robot Results*

Command	Success Rate
Walk Forward	96%
Turn Left	94.8%
Turn Right	95.3%
Walk Backward	93.5%
Stop	100%

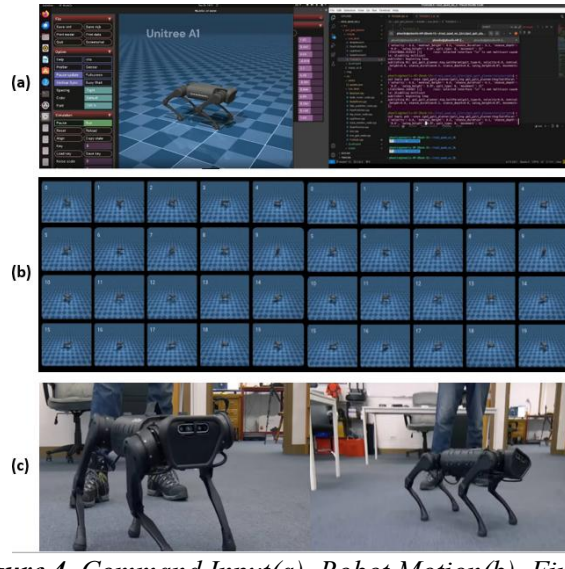


Figure 4. *Command Input(a), Robot Motion(b), Final State(c).*

4.6. Comparative Analysis

Baselines and fairness. We compare against representative model-predictive control (MPC) and reinforcement-learning baselines, including a PPO locomotion baseline and recent language-conditioned

locomotion work where applicable. To reduce confounds, we use a unified evaluation protocol (same terrains, command set, noise/randomization ranges, and metric definitions). Because original baselines may differ in sensing and training pipelines, these comparisons should be interpreted as controlled within our implementation rather than a definitive reproduction of every prior result, see Table7. See Figure 5 to view the Confusion Matrix.

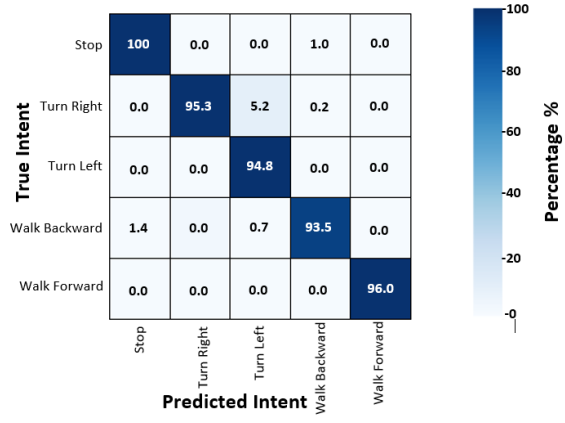


Figure 5. *Confusion Matrix*

- ANOVA indicated significant group differences ($F = 12.4$, $p < 0.001$).

- Pairwise t-tests indicate that our method improves performance over each evaluated baseline under the same evaluation protocol ($p < 0.01$, uncorrected). We also report mean ± standard deviation across seeds to reflect variability.

Table 7. *Comparison against baselines*

Method	Success Rate (%)	Energy Efficiency (J/step)	Adaptation Speed (s)	Robustness to Variations (%)
Traditional MPC	79.3 ± 4.0	0.17 ± 0.05	2.0 ± 0.5	73.8 ± 4.7
PPO (without NLU)	86.1 ± 2.8	0.13 ± 0.04	1.5 ± 0.4	82.6 ± 3.5
SayTap [9]	89.2 ± 2.5	0.12 ± 0.03	1.3 ± 0.3	87.4 ± 2.9
Ours (PPO + DistilRoBERTa)	**93.4 ± 1.9**	**0.10 ± 0.02**	**1.0 ± 0.3**	**92.5 ± 2.3**

4.7. Ablation Studies

- Without NLU (language conditioning removed): success drops to 85.7 ± 3.0, indicating that language-conditioned shaping/conditioning contributes to command-specific behavior beyond a generic locomotion policy, see Table 8.

- Without teacher–student distillation: sim-to-real performance degrades and sensitivity to noise increases, suggesting that distillation helps compress a more robust teacher into a deployable student policy.

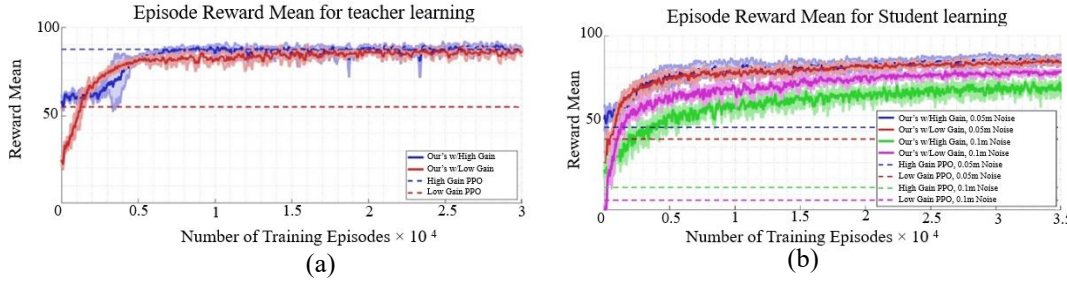
Table 8. *Ablation*

Variant	Success
Full Model	93.4
Without NLU	85.7
Without Goal Conditioning	87.9
Without Reward Shaping	88.5
Without Distillation	89.2
PPO Only	86.1

5. Discussion and Limitations

Our study has limitations. (1) Language scope: commands are mapped to a limited set of intents and continuous parameters; multi-intent instructions and long-horizon dialogue are not addressed. (2) Dataset bias: the NLU dataset is partially synthetic and may not cover the full diversity of human phrasing; performance may degrade on out-of-distribution language. (3) Baseline comparability: although we evaluate all methods under a unified protocol, original implementations of prior work differ in sensing/training assumptions, so comparisons should be interpreted within our re-implementation. (4) Hardware evidence: real-robot trials are initial and limited in duration/terrain diversity; extensive field testing is needed to assess long-term reliability and safety.

(5) Metrics/statistics: energy efficiency is reported via regularization/proxy terms rather than a full cost-of-transport measurement, and statistical tests are reported as indicators of trends across seeds. Future work should expand effect-size reporting and multiple-comparison controls.



(a) (b)

Figure 6. *Episode reward during training: (a) teacher policy and (b) student policy. Baseline PPO controllers are shown as dotted lines.*

In Student Learning, the proposed method shows improved final reward and better robustness to injected noise compared to the PPO baseline under our training/evaluation settings (Figure 6).

5.1. Future Work

Future directions include expanding the command space (multi-intent and temporal constraints), adding multimodal grounding (vision + language) for terrain/object references, improving ambiguity handling via clarification questions, and conducting longer and more diverse real-robot deployments.

6. CONCLUSION

Conclusion. We presented a language-conditioned quadruped locomotion framework that combines an NLU module with a PPO-based controller and teacher–student distillation. Results in simulation and initial hardware trials indicate that structured signals extracted from free-form language can modulate locomotion objectives to execute parameterized commands. Future work will expand the language and task scope, improve ambiguity handling, and increase the extent of real-robot evaluation.

Declarations

- **Ethics, Consent to Participate, and Consent to Publish declarations: Not applicable.**

- **Clinical Trial Registration: Not applicable.**

- **Author Contributions**

  Shirin Ranjbaran (1): Conceived and designed the study; developed the proposed methodology; conducted all simulations, experiments, and data collection; performed statistical and quantitative analyses; interpreted the results; prepared all figures and tables; and wrote the original manuscript.

  Prof. Yili Fu (2): Provided overall supervision and scientific guidance; monitored and reviewed the research methodology, experimental design, and analytical procedures; critically revised the manuscript for important intellectual content; and approved the final manuscript for publication.

Both authors discussed the results and contributed to the refinement of the study. All authors have read and approved the final version of the manuscript.

- **Financial Support**

  "This research received no specific grant from any funding agency, commercial or not-for-profit sectors."

- **Competing Interests Declarations**

  “Competing interests: The author(s) declare none.”

- **Data Availability Statement**

"The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request. "

References

[1] A. Kumar, Z. Fu, D. Pathak, and J. Malik, “RMA: Rapid motor adaptation for legged robots,” *Robotics: Science and Systems (RSS)*, 2021, doi: 10.15607/RSS.2021.XVII.011.

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning quadrupedal locomotion over challenging terrain,” *Sci. Robot*., vol. 5, no. 47, eabc5986, 2020, doi: 10.1126/scirobotics.abc5986.

[3] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, “Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control,” *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018.

[4] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, “MPC-based controller with terrain insight for dynamic legged locomotion,” *IEEE Int. Conf. Robotics and Automation (ICRA),* 2020, pp. 2436–2442, doi: 10.1109/ICRA40945.2020.9197312.

[5] M. H. Raibert, “Trotting, pacing and bounding by a quadruped robot,” J. Biomech., vol. 23, suppl. 1, pp. 79–98, 1990.

[6] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke, “Policies modulating trajectory generators,” *Conf. Robot Learning (CoRL)*, Proc. Mach. Learn. Res., vol. 87, 2018, pp. 916–926.

[7] K. Caluwaerts et al., “Barkour: Benchmarking animal-level agility with quadruped robots,” arXiv:2305.14654, 2023.

[8] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek, “Robots that use language: A survey,” *Annu. Rev. Control Robot. Auton.* Syst., vol. 3, pp. 25–55, 2020.

[9] Y. Tang, W. Yu, J. Tan, H. Zen, A. Faust, and T. Harada, “SayTap: Language to quadrupedal locomotion,” arXiv:2306.07580, 2023.

[10] A. Zeng et al., “Socratic models: Composing zero-shot multimodal reasoning with language,” arXiv:2204.00598, 2022.

[11] H. Li, T. Zhang, W. Yu, and P. M. Wensing, “Versatile real-time motion synthesis via kino-dynamic MPC with hybrid-systems DDP,” arXiv:2209.14138, 2022.

[12] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "What matters in language conditioned robotic imitation learning," *IEEE Robot. Autom.* Lett., vol. 7, no. 4, pp. 11205–11212, 2022.

[13] M. Shridhar, L. Manuelli, and D. Fox, "CLIPort: What and where pathways for robotic manipulation," *Conf. Robot Learning (CoRL)*, Proc. Mach. Learn. Res., vol. 164, 2022, pp. 894–906.

[14] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning," *AAAI Conf. Artificial Intelligence (AAAI)*, 2008, pp. 1433–1438.

[15] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2Motion: From natural language instructions to feasible plans," arXiv:2303.12153, 2023.

[16] A. Bucker et al., "LaTTe: LAnguage trajectory transformEr," arXiv:2208.02918, 2022.

[17] P. Anderson et al., "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3674–3683, doi: 10.1109/CVPR.2018.00387.

[18] L. Ouyang et al., "Training language models to follow instructions with human feedback," arXiv:2203.02155, 2022.

[19] M. Ahn et al., "Do as I can, not as I say: Grounding language in robotic affordances," arXiv:2204.01691, 2022.

[20] J. Hwangbo et al., "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.,* vol. 4, no. 26, eaau5872, 2019, doi: 10.1126/scirobotics.aau5872.

[21] Y. Yang et al., "Data efficient reinforcement learning for legged robots," *Conf. Robot Learning (CoRL*), 2020.

[22] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, "Sim-to-real learning of all common bipedal gaits via periodic reward composition," *IEEE Int. Conf. Robotics and Automation (ICRA),* 2021, pp. 7309–7315.

[23] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," *Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, Proc. Mach. Learn. Res., vol. 15, 2011, pp. 627–635.

[24] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *Int. Conf. Learn. Representations (ICLR),* 2016.

[25] J. Schulman et al., "Proximal policy optimization algorithms," arXiv:1707.06347, 2017.

[26] V. Sanh et al., "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," arXiv:1910.01108, 2019.

[27] A. A. Rusu et al., "Policy distillation," arXiv:1511.06295, 2015.

[28] J. Devlin et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," *NAACL-HLT*, 2019, pp. 4171–4186.

[29] M. Shridhar et al., "ALFRED: A benchmark for interpreting grounded instructions for everyday tasks," *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10740–10749.

[30] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," arXiv:1907.11692, 2019.

[31] T. Wolf et al., "Transformers: State-of-the-art natural language processing," *Conf. Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, 2020, pp. 38–45.