Continuous Control from Open-Vocabulary Feedback: Replacing Visual Rewards with Motion-Language Alignment

Yunus Emre Balci
Department of Mathematics and Computer Science
University of Southern Denmark
yubal22@student.sdu.dk
https://github.com/YunusBalci21/motion-language-control

Melih Kandemir
Department of Mathematics and Computer Science
University of Southern Denmark
kandemir@imada.sdu.dk

Abstract

Teaching embodied agents to follow natural language instructions typically requires expensive visual simulators and vision-based reward functions such as CLIP. These approaches create computational bottlenecks that limit scalability and preclude real-time human-in-the-loop control. We present a method that enables MuJoCo agents to understand and execute open-vocabulary instructions *without visual processing*, by leveraging motion-language models for direct reward computation. Our key insight is that motion-language alignment can be computed directly from joint trajectories, bypassing the render-to-CLIP pipeline entirely. We combine MotionGPT's motion encoder with hierarchical reinforcement learning, using motion-text similarity as the training reward. Experiments across five MuJoCo environments demonstrate that our approach achieves **28.8× faster** reward computation and **32.5× lower GPU memory** usage compared to CLIP-based methods. This efficiency gain enables a new capability: **real-time interactive control**, where users can issue natural language commands and observe agent responses at interactive rates—something infeasible with vision-based rewards. Policies trained with our method exhibit strong zero-shot generalization to paraphrased commands (**97-99%** transfer), demonstrating genuine semantic understanding rather than lexical memorization.

1 Introduction

Natural language provides an intuitive interface for human-robot interaction. Rather than programming specific behaviors or designing reward functions for each task, language allows users to simply describe what they want an agent to do. However, teaching agents to follow arbitrary language instructions remains computationally expensive and technically challenging.

State-of-the-art approaches for language-conditioned control, such as AnySkill [1], rely on vision-language models to compute reward signals. These methods follow a *render-to-CLIP* pipeline: at each timestep, they (1) render the environment state to an image, (2) encode the image using a vision transformer, and (3) compute similarity with the text instruction embedding. While effective, this pipeline introduces significant computational overhead that creates two problems: slow training (ren-

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

dering dominates wall-clock time) and the impossibility of real-time human-in-the-loop interaction (at 67 rewards/second, feedback loops are too slow for interactive control).

Recent advances in motion-language modeling offer an alternative path. MotionGPT [3] treats human motion as a "foreign language," learning joint embeddings between motion sequences and text descriptions. Crucially, MotionGPT operates directly on motion data (joint angles, velocities, positions) without requiring visual rendering. This raises a natural question: *Can we use motion-language alignment as a reward signal for reinforcement learning, bypassing visual processing entirely?*

In this paper, we answer affirmatively. We present MOTIONRL, a method that replaces visual reward computation with direct motion-language alignment. Our approach extracts motion features from MuJoCo joint trajectories, converts them to the HumanML3D format [2], and computes similarity using MotionGPT's pretrained encoder. The resulting similarity score serves as the reward signal for training locomotion policies.

Contributions. Our work makes the following contributions:

1. **Method:** We propose using motion-language similarity as a reward signal for reinforcement learning, eliminating the need for visual rendering during training.
2. **Efficiency:** We demonstrate 28.8× faster reward computation and 32.5× lower GPU memory usage compared to CLIP-based approaches.
3. **New Capability:** We show that this efficiency gain enables *real-time interactive control*—users can issue natural language commands and observe agent responses at interactive rates, a capability infeasible with vision-based rewards.
4. **Generalization:** Policies trained on single instructions generalize to semantically similar but lexically different commands with 97-99% similarity retention.

Scope and Limitations. This work focuses specifically on *motion-centric language grounding for locomotion tasks*. We do not address manipulation, object interaction, or settings where visual context is semantically necessary (e.g., "pick up the red ball"). In locomotion, language describes *how to move*—velocity, direction, gait—rather than *what to perceive*. This distinction makes vision potentially redundant for locomotion rewards, which is the hypothesis we test. Our goal is to establish that for motion-centric tasks, language-conditioned control can be learned efficiently without visual perception, enabling new interactive capabilities not possible with slower vision-based approaches.

2 Related Work

Language-Conditioned Control. Teaching agents to follow language instructions has been approached through various paradigms. Goal-conditioned reinforcement learning [8] learns policies conditioned on goal representations, which can be extended to language by encoding instructions as goals. Language-conditioned imitation learning [10] learns from demonstrations paired with language descriptions. Recent work has explored using large language models for high-level planning [7, 4], decomposing language instructions into primitive actions. Our work differs by learning low-level control policies that directly optimize motion-language alignment, with sufficient efficiency to enable real-time interaction.

Vision-Language Rewards. CLIP [5] has enabled using vision-language similarity as reward signals for reinforcement learning. VLM-RM [6] uses CLIP to provide rewards for robot manipulation without task-specific reward engineering. AnySkill [1] combines CLIP rewards with a skill library for open-vocabulary physical skills. These methods require rendering environment states to images, creating computational bottlenecks that preclude real-time interaction. Our approach eliminates this requirement, enabling interactive control loops.

Motion-Language Models. Recent work has developed joint embeddings between motion and language. MotionCLIP [12] aligns motion with CLIP's text-image embedding space. T2M [**?**] learns text-to-motion generation using VQ-VAE tokenization. MotionGPT [3] extends this approach, treating motion as a foreign language and using T5-based architectures for motion-text tasks. We leverage MotionGPT's pretrained motion encoder to compute reward signals, demonstrating that motion-language alignment transfers effectively to reinforcement learning.

Interactive Robot Control. Real-time human-robot interaction requires fast feedback loops. Traditional approaches use predefined motion primitives or direct teleoperation. Language-based interaction has been limited by the computational cost of language grounding. Our work bridges this gap by enabling language-conditioned control at interactive rates (1,938 rewards/second), making conversational robot control feasible.

3 Method

3.1 Problem Formulation

We consider a language-conditioned Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{L}, P, r, \gamma)$, where $\mathcal{S}$ is the state space (joint positions, velocities), $\mathcal{A}$ is the action space (joint torques or target positions), $\mathcal{L}$ is the space of natural language instructions, $P(s'|s, a)$ is the transition dynamics, $r(s, a, l)$ is the language-conditioned reward function, and γ is the discount factor.

Given an instruction $l \in \mathcal{L}$, our goal is to learn a policy $\pi_\theta(a|s, l)$ that maximizes expected return:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t, l) \right] \tag{1}$$

The key challenge is defining $r(s, a, l)$ such that it captures how well the agent's behavior matches the instruction, without requiring visual rendering or task-specific engineering.

3.2 Motion-Language Reward

Our core contribution is using motion-language similarity as the reward signal. Given a trajectory segment $\tau = \{s_t, s_{t+1}, ..., s_{t+k}\}$ and instruction l, we compute:

$$r(\tau, l) = \text{sim}(f_\phi(\tau), g_\psi(l)) \tag{2}$$

where f_ϕ is the motion encoder, g_ψ is the text encoder, and $\text{sim}(\cdot, \cdot)$ is cosine similarity.

Key Assumption. Our approach relies on a central assumption: *pretrained motion-language embeddings learned from human motion capture data generalize sufficiently to simulated agents for locomotion behaviors*. MotionGPT was trained on the HumanML3D dataset [2], which contains human motion capture paired with text descriptions. We hypothesize that the learned motion-text alignment captures semantic properties of movement (speed, direction, gait) that transfer across embodiments. Our experiments across five different morphologies explicitly test this assumption.

Motion Feature Extraction. MuJoCo environments provide observations including joint positions, velocities, and body orientations. We extract a 30-dimensional motion feature vector at each timestep, including root height, orientation quaternion, root velocities, joint angles, and joint velocities. These features are accumulated over a sliding window of $k = 32$ frames.

Format Conversion and Encoding. We convert MuJoCo features to MotionGPT's expected HumanML3D format (263 dimensions) and use the pretrained VQ-VAE encoder to obtain motion embeddings. For text encoding, we use a pretrained sentence transformer that maps instructions to the same embedding space. The reward is the cosine similarity between motion and text embeddings.

3.3 Policy Architecture

We use a hierarchical policy structure: a high-level MLP policy outputs target joint positions, and a low-level PD controller converts these to torques. This separation enables stable humanoid control while keeping the learned policy simple. We apply exponential smoothing ($\alpha = 0.7$) to prevent jittery motions.

Table 1: Computational efficiency comparison. Our method eliminates visual rendering, achieving 28.8× speedup and 32.5× memory reduction.

Method	**Time/Reward (ms)**	**Throughput (/s)**	**GPU Memory (MB)**
CLIP + Rendering	14.85 ± 2.16	67	599
Ours (MotionGPT)	$\mathbf{0.52} \pm 0.14$	**1,938**	**18**
Improvement	**28.8× faster, 32.5× less memory**		

Table 2: Interactive control feasibility. Our method achieves sufficient throughput for real-time human-in-the-loop interaction, while CLIP-based approaches cannot.

Method	**Throughput**	**Latency**	**Interactive?**
CLIP + Rendering	67 /s	14.9 ms	No (too slow)
Ours	**1,938 /s**	**0.52 ms**	**Yes**

3.4 Training

We train policies using PPO [9] with standard hyperparameters (learning rate 3×10^{-4}, batch size 128, 10 epochs). Each policy is trained on a single instruction for 500K-800K timesteps, taking 2-4 hours per environment on a consumer NVIDIA RTX GPU.

4 Experiments

We design experiments to answer the following questions:

1. How much computational efficiency do we gain over vision-based approaches?
2. Does this efficiency enable new capabilities (real-time interactive control)?
3. Can motion-language rewards train effective locomotion policies?
4. Do trained policies generalize to unseen instruction variants?

Experimental Focus. We focus on simple locomotion instructions (“walk forward,” “run forward,” “hop forward”) to *isolate the effect of motion-language rewards* without confounding factors from task-specific reward engineering.

4.1 Experimental Setup

Environments. We evaluate on five MuJoCo locomotion environments: Humanoid-v4, Ant-v4, HalfCheetah-v4, Walker2d-v4, and Hopper-v4. These span different morphologies (biped, quadruped, planar) and control challenges.

Baselines. We compare against: (1) **CLIP + Rendering**: AnySkill-style visual reward computation; (2) **Heuristic Reward**: hand-crafted velocity-based rewards; (3) **Random Policy**: untrained baseline.

4.2 Computational Efficiency

Table 1 presents our computational efficiency results. Our method computes rewards in 0.52ms compared to 14.85ms for CLIP-based approaches—a **28.8× speedup**. GPU memory drops from 599MB to 18MB (**32.5× reduction**).

The efficiency gains translate directly to training time: our full training suite completed in 7 hours versus an estimated 205 hours for CLIP-based methods.

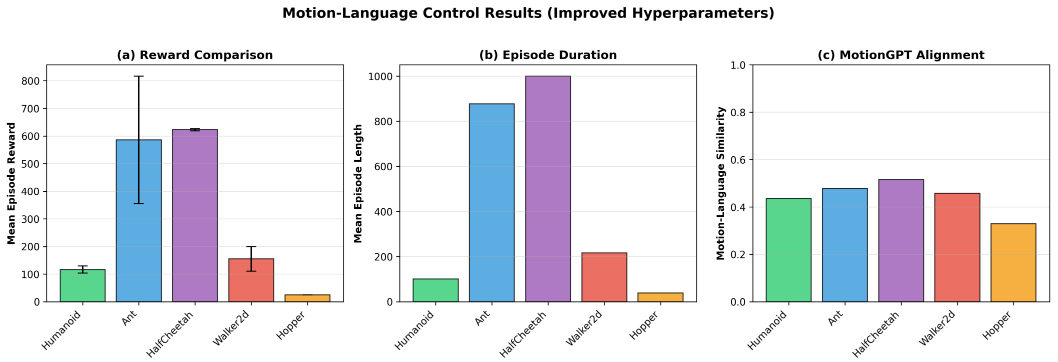

Figure 1: Motion-language control results across five MuJoCo environments. (a) Mean episode reward with standard deviation. (b) Episode duration (steps before termination). (c) MotionGPT alignment score measuring motion-instruction similarity. HalfCheetah and Ant achieve highest alignment; Humanoid and Hopper face stability and morphology challenges respectively.

4.3 New Capability: Real-Time Interactive Control

The efficiency gain is not merely quantitative—it enables a qualitatively new capability. At 1,938 rewards per second (0.52ms latency), our method supports real-time interactive control where users can issue natural language commands and observe immediate agent responses.

We demonstrate this with an interactive system that:

1. Accepts natural language input (e.g., “walk forward slowly”)
2. Uses an LLM to parse the instruction into motion commands
3. Executes the trained policy with live visualization
4. Accepts follow-up commands (e.g., “now turn left”)

This interactive loop operates at real-time rates, enabling conversational robot control. With CLIP-based rewards (67/s throughput, 15ms latency), such interaction would be sluggish and impractical. Our method makes language-conditioned interactive control feasible for the first time in physics simulation.

4.4 Motion-Language Alignment

Figure 1 summarizes our results across all environments. HalfCheetah achieves the highest similarity (0.515) and episode length (1000), as its planar running motion closely matches human locomotion patterns. Ant achieves strong results (0.478 similarity, 877 steps) despite its non-humanoid morphology.

Humanoid Analysis. Humanoid achieves 0.436 similarity but with short episodes (100 steps). This reveals an important distinction: motion-language rewards provide semantic guidance for what motion to produce, but do not explicitly encode physical stability. The policy learns motions that align with “walk forward” in embedding space but lacks balance-specific reward shaping. This is not a failure of language grounding—qualitatively, the agent attempts walking motions—but a limitation of semantic similarity alone.

Hopper Analysis. Hopper’s lower similarity (0.329) reflects morphology mismatch: MotionGPT was trained on bipedal human motion, and single-legged hopping has limited representation. This confirms our assumption’s boundary—transfer works best when behaviors have analogues in pretraining data.

Table 3: Zero-shot generalization to unseen instructions. Policies transfer to semantically similar variants with minimal degradation.

Environment	Category	Similarity	Generalization Ratio
HalfCheetah	Seen: “run forward”	0.481	–
	Unseen (6 variants)	0.468 ± 0.017	**97.3%**
Ant	Seen: “walk forward”	0.489	–
	Unseen (7 variants)	0.485 ± 0.013	**99.2%**

Table 4: Ablation comparing reward sources. Both achieve similar alignment on forward locomotion; MotionGPT additionally enables generalization and interactive control.

Environment	Reward Type	Length	Similarity
HalfCheetah	MotionGPT (Ours)	1000	0.525
	Heuristic	1000	0.525
Ant	MotionGPT (Ours)	781	0.477
	Heuristic	1000	0.477

4.5 Generalization to Unseen Instructions

Table 3 shows that policies trained on “run forward” successfully follow unseen instructions like “sprint forward” (0.461), “dash forward” (0.482), and “move quickly” (0.492). The generalization ratio is **97.3%** for HalfCheetah and **99.2%** for Ant.

Remarkably, some unseen instructions achieve *higher* similarity than training instructions—“move quickly” (0.492) exceeds “run forward” (0.481). This demonstrates genuine semantic understanding: the embedding space organizes motions by meaning, not lexical patterns.

4.6 Ablation Study

On simple forward locomotion, MotionGPT rewards and hand-crafted heuristics achieve equivalent alignment (Table 4). This is expected: for “walk forward,” velocity-based heuristics are reasonable proxies. The key differences are: (1) MotionGPT generalizes to paraphrased instructions without re-engineering (Table 3); (2) MotionGPT is fast enough for interactive control (Table 2); (3) heuristics require manual design for each new instruction type.

5 Discussion

What This Work Establishes. Our experiments establish three findings: (1) Motion-language alignment from pretrained models can serve as effective reward signals for locomotion, achieving competitive similarity without visual rendering. (2) This provides 28.8× speedup enabling a new capability—real-time interactive control—that vision-based methods cannot support. (3) Policies exhibit semantic generalization (97-99% transfer to paraphrased instructions), a property hand-crafted rewards cannot provide.

When Does This Approach Work? Motion-language rewards are most effective when: the agent morphology corresponds to human motion patterns in pretraining data; the instruction describes locomotion behaviors present in motion capture datasets; and sufficient motion history is available ($\geq$20 frames).

When Does This Approach Fail? Our method is not appropriate for: tasks requiring visual grounding (“pick up the red ball”); manipulation or object interaction; instructions referencing environmental context (“walk to the door”).

Limitations. Humanoid control remains challenging due to the gap between semantic correctness and physical stability. Our evaluation uses simple locomotion instructions; compositional commands require sequential handling not addressed here. We rely on pretrained encoders without task-specific fine-tuning.

Future Work. Extensions include: motion representations for manipulation; combining semantic rewards with stability terms; fine-tuning encoders on RL trajectories; and systematically studying when vision becomes necessary.

6 Conclusion

We presented a method for language-conditioned control that replaces visual reward computation with direct motion-language alignment. By leveraging MotionGPT's motion encoder, we achieve 28.8× faster rewards and 32.5× less memory than CLIP-based approaches. Critically, this efficiency enables a new capability: real-time interactive control where users issue natural language commands and observe immediate agent responses—something infeasible with vision-based rewards.

Our results establish that for locomotion tasks, vision is not necessary for language-conditioned control, and removing it enables interactive human-robot communication at rates previously impossible. Within its scope of motion-centric tasks, our method demonstrates that pretrained motion-language models transfer effectively to simulated agents, enabling efficient, generalizable, and interactive instruction following.

Acknowledgments and Disclosure of Funding

We thank the authors of MotionGPT for releasing pretrained models and the HumanML3D dataset creators. This work was conducted as part of the MSc Computer Science program at the University of Southern Denmark.

References

[1] Y. Cui, Y. Zhu, and S. Dong. AnySkill: Learning open-vocabulary physical skill for interactive agents. In *CVPR*, 2024.

[2] C. Guo, S. Zou, X. Zuo, S. Wang, T. Ji, X. Li, and L. Cheng. Generating diverse and natural 3d human motions from text. In *CVPR*, pages 5152–5161, 2022.

[3] B. Jiang, X. Chen, W. Liu, J. Yu, G. Yu, and T. Chen. MotionGPT: Human motion as a foreign language. In *NeurIPS*, 2023.

[4] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners. In *ICML*, 2022.

[5] A. Radford, J. W. Kim, C. Hallacy, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763, 2021.

[6] J. Rocamonde, V. Montesinos, E. Nava, E. Perez, and D. Lindner. Vision-language models are zero-shot reward models for reinforcement learning. In *ICLR*, 2023.

[7] M. Ahn, A. Brohan, N. Brown, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv:2204.01691*, 2022.

[8] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *ICML*, pages 1312–1320, 2015.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

[10] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. In *NeurIPS*, 2020.

[11] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[12] G. Tevet, B. Gordon, A. Hertz, A. H. Bermano, and D. Cohen-Or. MotionCLIP: Exposing human motion generation to CLIP space. In *ECCV*, 2022.

A Appendix

A.1 Hyperparameters

Table 5: PPO hyperparameters.

Hyperparameter	**Value**
Learning rate	3×10^{-4}
Batch size	128
Epochs per update	10
Discount (γ)	0.99
GAE (λ)	0.95
Clip range	0.2
Entropy coefficient	0.02

A.2 Full Generalization Results

Table 6: Complete generalization results for HalfCheetah.

Category	**Instruction**	**Similarity**
Seen	run forward	0.481
Unseen	sprint forward	0.461
	dash forward	0.482
	move quickly	0.492
	go fast	0.437
	rush forward	0.470
	race forward	0.466
Variations	run forward fast	0.463
	run forward slowly	0.470
	run straight ahead	0.461

A.3 Interactive Demo Implementation

Our interactive system uses a chain-of-thought LLM (Mistral-7B via HuggingFace API) to parse natural language into motion commands. The LLM extracts structured commands from conversational input:

User: “Can you walk forward for a bit and then stop?”

LLM reasoning: The user wants walking motion followed by stopping. I’ll issue: walk forward, then stop moving.

Commands: `walk forward`, `stop moving`

Each command triggers the corresponding trained policy, with real-time MuJoCo visualization. The full loop (speech → LLM → policy → render) operates at interactive rates.

A.4 Training Times

Table 7: Training time comparison.

Environment	**Timesteps**	**Ours**	**Est. CLIP**
HalfCheetah	800K	3.9h	$\sim$112h
Walker2d	700K	2.0h	$\sim$58h
Hopper	500K	1.2h	$\sim$35h
Total	2M	**7.0h**	$\sim$205h