MotionVL: Visual-Language Supervision for Guiding Reinforcement Learning in Humanoid Motion

Yan Luo, Zhenhua Xiong, *Member, IEEE* and Han Ding, *Member, IEEE*

***Abstract*—Reward function design remains a fundamental challenge in reinforcement learning for humanoid robots, where handcrafted rewards often fail to capture human-like behavior, limit generalization, and require costly manual tuning. Addressing this bottleneck is essential for enabling scalable, adaptive, and interpretable control. In this work, we introduce MotionVL, a novel framework for humanoid reinforcement learning that leverages multimodal large models to automate reward generation and semantic supervision. A Vision-Language Model (VLM) encodes visual observations into structured natural language descriptions, while a Large Language Model (LLM) generates task-aligned reward functions based on these descriptions and the given instructions. This closed-loop design supports dynamic reward optimization and behavior alignment. Experimental results, including the deployment of straight-leg walking on a humanoid robot, demonstrate significant gains in efficiency, generalization, and human-likeness compared to traditional reward designs, highlighting a new paradigm for language-informed, scalable robot learning.**

***Index Terms*—Reinforcement learning, generative reward, large language models, multimodal supervision.**

I. Introduction

HUMANOID robots are widely recognized as the best vehicle for embodied intelligence due to their human-like morphology and powerful locomotion capabilities. Current research on motion control for humanoid robots is divided into two main categories: traditional model-based control methods, represented by model predictive control (MPC) [1], [2], etc., and model-free data-driven methods, such as reinforcement learning (RL) [3]–[6]. The latter have attracted much attention for their ability to optimize control strategies through interaction, especially with the emergence of NVIDIA's Isaac series of high-performance simulation platforms [7], [8] , which provide standardized training environments as well as the ability to accelerate strategy learning. In recent years, humanoid robots have achieved learning training of many skills through artificially designed reward functions and improved anti-jamming stability under multiple complex terrains [9]–[11].

The design of reward functions is currently the most central part of RL-based systems. For the task of humanoid gait walking, the reward function usually considers factors such as stability, energy efficiency, gait coordination, tracking speed, etc. [12], [13]. Taking the open source reinforcement learning framework of ENGINEAI as an example, the traditional reward function is designed and optimized manually, in which the robot can walk with bent knees, but cannot do walking with human-like gait. Such manually designed reward functions face two key limitations: (i) the design often neglects to capture motion details, such as straight-knee motion, resulting in behaviors that deviate from the natural human gait; and (ii) the design process needs to be task-specific and relies on the developer's experience, which requires a lot of testing and tuning, impeding generalization and large-scale applications.

Recent advances in multimodal large models (MLMs), particularly large language models (LLMs) and vision-language models (VLMs), are reshaping reinforcement learning (RL) by enabling language-conditioned reward design, policy learning, and task generation. Foundational work such as RLAIF [14] uses AI-generated preferences to train reward models, reducing reliance on human annotations. EUREKA [15] demonstrates that LLMs can synthesize structured reward code from natural language instructions, facilitating scalable reward specification. RoboGen [16] leverages LLMs for procedural task generation but lacks real-world physical fidelity. Video2Policy [17] directly learns policies from human videos, though scene reconstruction remains insufficiently realistic. LLM^3 [18] introduces failure-aware reasoning into planning, yet still relies on manually defined symbolic abstractions and assumes access to accurate motion primitives. VLM-RL [19] proposes a vision-language RL framework for safe driving, but lacks real-world validation. Language-guided locomotion [20] explores LLM-generated gaits in simulation, without VLM integration or online adaptation. These works highlight the potential of LLM/VLM-augmented RL.

Despite these promising advances, several challenges remain. In humanoid control, reward functions derived from large models often lack grounding in the agents actual behavior, leading to limited interpretability and suboptimal performance. Without incorporating visual observations of the agents motion, it is difficult to detect and correct semantic misalignments during training.

To this end, we introduce MotionVL, a reinforcement learning framework that incorporates semantic feedback from vision-language models to inform policy optimization. Rather than relying solely on predefined or externally scripted rewards, MotionVL constructs a dynamic reward signal by

This work was supported by Jiangsu Provincial Science and Technology Department Program Special Funds (Basic Research on Frontier Leading Technologies) Project SBK2023050162. *(corresponding author: Zhenhua Xiong).*

Y. Luo, Z. Xiong and H. Ding are with the School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: luoyan99@sjtu.edu.cn; aalon@sjtu.edu.cn; wujh@sjtu.edu.cn; mexiong@sjtu.edu.cn; hding@sjtu.edu.cn).

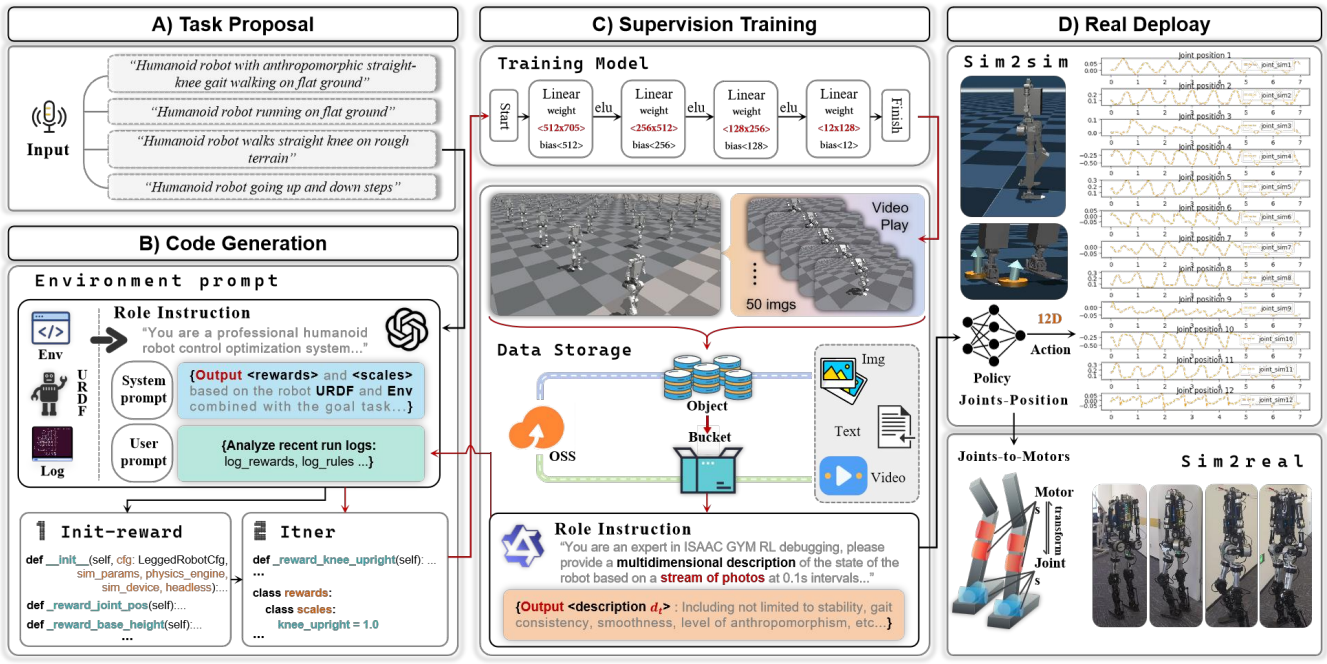


Fig. 1. **MotionVL** structure diagram. The proposed system comprises four stages: (A) **Task Proposal**: Natural language inputs are used to describe the desired humanoid robot behavior tasks, such as walking on flat or uneven terrain. (B) **Code Generation**: Based on the robot URDF, environment configuration, and recent log data, a language model generates and modifies the reward function code, incorporating task-specific rewards and scaling factors. (C) **Supervision Training**: A multi-layer neural network model is trained using simulation data. Robot behaviors are rendered as video frames at 0.1s intervals and stored in an object storage service (OSS). Experts generate multidimensional descriptors ψ from these videos to provide feedback for improving training. (D) **Real Deploy**: The trained policy generates a 12-dimensional action vector representing joint positions. The policy is validated in simulation (*Sim2Sim*) and then deployed to a physical robot (*Sim2Real*) via a joints-to-motors translation module. Joint trajectories demonstrate smooth and anthropomorphic motion transfer from simulation to real-world execution.

interpreting the agents behavior through visual observations. These observations are translated into structured descriptions via a Vision-Language Model (VLM), and further grounded in task intent using a Large Language Model (LLM). This enables closed-loop learning that enhances generalization, interpretability, and task alignment across diverse humanoid control tasks.

The remainder of this paper is organized as follows. Section II introduces the problem formulation and system architecture. Section III details the perception, reward generation, and policy optimization components. Section IV presents experimental results and ablation studies. Section V concludes with a discussion of future directions.

II. Problem Statement and System Structure

A. Problem Statement

In the design and optimization of reinforcement learning (RL) systems, constructing effective reward functions remains a fundamental challenge. Traditional methods face two major difficulties: first, reward signals in real-world environments are often inaccessible, noisy, or extremely sparse; second, handcrafted reward functions typically lack adaptability and generalization in complex, multimodal tasks. These limitations are formalized under the Reward Design Problem (RDP) framework [21], which models reward specification as a higher-level optimization problem:

$$\mathcal{P} = \langle \mathcal{M}, \mathcal{R}, \mathcal{A}_{\text{alg}}, \mathcal{F}_{\text{eval}} \rangle,$$

where $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T})$ defines the environment's state space $\mathcal{S}$, action space $\mathcal{A}$, and transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$; $\mathcal{R}$ denotes the space of candidate reward functions $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$; $\mathcal{A}_{\text{alg}}$ represents the learning algorithm that maps rewards to policies; and $\mathcal{F}_{\text{eval}}$ is a fitness function evaluating policy performance with limited interaction samples.

Despite increasing interest in automating reward design, most contemporary RL systems still rely on manually crafted reward functions $R_{\text{manual}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. These require substantial domain expertise and iterative tuning, often taking weeks or months to yield desirable behaviors. Moreover, such manually defined surrogates lack the generative flexibility and cross-task transferability of large pretrained models, resulting in repeated reward engineering for each new environment or skill.

Recent advances in vision–language models (VLMs) offer an appealing alternative by providing rich semantic feedback derived from visual observations. However, how to systematically integrate such perceptual supervision into policy learning remains an open question. In particular, there is a lack of unified frameworks that can leverage VLM-generated semantics to construct meaningful reward signals and guide policy optimization in a consistent, adaptive manner.

B. System Structure

The overall architecture of MotionVL is illustrated in Figure 1. It forms a closed-loop reinforcement learning system that integrates visual-language feedback for automatic reward

construction and policy supervision. The system consists of four main components: (A) Task Proposal, (B) Code Generation, (C) Supervision Training, and (D) Real Deployment.

a) Task Proposal: Users provide natural language instructions $\mathcal{I} \in \mathcal{L}_{\text{instruction}}$ to specify high-level humanoid motion goals, such as "walking with straight knees" or "climbing stairs." These instructions serve as the semantic anchor for subsequent reward generation and policy training.

b) Code Generation: Given the instruction $\mathcal{I}$, environment configuration $\mathcal{E}_{\text{config}}$, and robot model $\mathcal{U}_{\text{robot}}$, an LLM generates an initial training script. Based on role-specific prompting, it produces task-relevant reward functions and iteratively refines them using execution logs.

c) Supervision Training: During simulation, image streams are captured and interpreted by a VLM to produce semantic descriptions $\mathcal{D}_t$. The LLM uses $\mathcal{D}_t$ and instruction $\mathcal{I}$ to synthesize refined rewards. Policy learning is supervised by both physical and semantic signals, with all intermediate data stored for iterative refinement.

d) Real Deployment: The trained policy is validated in simulation and then deployed to the physical humanoid robot JT-1. Domain randomization is applied during training to reduce the sim-to-real gap. At deployment, the learned joint-angle trajectories are mapped directly to motor commands, enabling reliable execution on the real hardware.

III. Method

In this section, we introduce the core design of the **MotionVL** framework, a multimodal policy learning architecture that integrates structured semantic perception, language-conditioned reward generation, and semantically aligned policy optimization. MotionVL is designed to form a closed-loop learning system, enabling generalizable, interpretable, and goal-consistent behavior for complex humanoid control tasks.

A. LLM-Based Reward Design

Reward design is inherently intertwined with the semantics of the environment and the structural constraints of the robot. To address this challenge, MotionVL incorporates a code-level reward inference mechanism powered by a large language model (LLM), specifically GPT-4o. This mechanism synthesizes semantically grounded reward functions by directly interpreting the environment's source code and execution traces.

We define the LLM input as a tuple $\mathcal{C} = (\mathcal{E}_{\text{code}}, \mathcal{U}_{\text{robot}}, \mathcal{H}_{\text{exec}})$, where $\mathcal{E}_{\text{code}}$ represents the environment code (excluding hardcoded rewards), $\mathcal{U}_{\text{robot}}$ encodes the robot's structural specification (e.g., URDF or MJCF), and $\mathcal{H}_{\text{exec}} = \{(o_t, a_t, s_t)\}_{t=1}^{T_{\log}}$ contains temporal logs of observations $o_t \in \mathcal{O}$, actions $a_t \in \mathcal{A}$, and states $s_t \in \mathcal{S}$. These inputs collectively form the conditioning context for the language model $\text{LLM}(\cdot)$, which is guided by a task instruction prompt $\mathcal{P}_{\text{task}} \in \mathcal{L}_{\text{prompt}}$.

Unlike conventional methods that rely on manually defined task abstractions, MotionVL enables the LLM to reason over full simulation context. This is grounded in two key observations [15]: (i) Code-pretrained LLMs exhibit strong capabilities in understanding structured simulation logic, particularly in Python-like environments; (ii) Environment code and robot descriptions encode task-relevant semantics, including controllable variables, constraints, and objective structures.

The model generates two functional components. The first is an initialization function $f_{\text{init}} : \mathcal{C} \mapsto \mathcal{X}_{\text{semantic}}$, which extracts a semantically meaningful state representation $\mathcal{X}_{\text{semantic}} = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\} \subset \mathbb{R}^n$. The second is a reward synthesis function $f_{\text{reward}} : \mathcal{X}_{\text{semantic}} \mapsto \mathcal{W}_{\text{reward}}$, where $\mathcal{W}_{\text{reward}} = \{(k_i, w_i)\}_{i=1}^{m}$ is a structured reward dictionary with component labels k_i and normalized weights $w_i \in [0, 1]$. Each entry (k_i, w_i) corresponds to a semantically labeled component, such as `goal_reach`, `energy_efficiency`, or `trajectory_smoothness`.

This inference process can be described as a functional composition, where the language model first applies an initialization function f_{init} to the input context $\mathcal{C}$, extracting a set of semantically grounded state variables $\mathcal{X}_{\text{semantic}}$. The reward function f_{reward} is then applied to this representation, producing a structured reward set $\mathcal{W}_{\text{reward}}$. Formally, we have:

$$\mathcal{W}_{\text{reward}} = f_{\text{reward}}(f_{\text{init}}(\mathcal{C})) \mid \mathcal{P}_{\text{task}}$$

conditioned on the instruction prompt $\mathcal{P}_{\text{task}} \in \mathcal{L}_{\text{prompt}}$.

The resulting reward dictionary $\mathcal{W}_{\text{reward}}$ fulfills two complementary roles. First, it serves as a differentiable supervision signal for reinforcement learning-based policy optimization, enabling gradient-based updates in continuous control settings. Second, it provides an interpretable and semantically structured representation that can be leveraged by downstream modules, such as the Semantic Alignment Module, to enhance policy alignment with human-understandable goals.

This formulation abstracts away the need for manual reward engineering and supports scalable, task-agnostic deployment across diverse robotic environments.

B. VLM-Based Motion Feature Extraction

While the large language model (LLM) module in Section III-A generates structured reward functions by interpreting environment code and execution traces, it lacks direct perceptual access to the robot's visual behavior. To bridge this gap, we incorporate a vision-language model (VLM), based on Qwen-VL, that semantically interprets motion trajectories using natural language descriptions. This allows MotionVL to unify code-level reward synthesis with visual-semantic behavior analysis.

After training in Isaac Gym, the robot's performance is visualized through playback, resulting in a state trajectory $\tau = \{s_t\}_{t=1}^{T_{\text{traj}}}$. We capture visual observations at uniform intervals $\Delta t = 0.1$s, producing a video sequence $\mathcal{V} = \{v_1, v_2, \ldots, v_{N_{\text{frame}}}\}$, where each frame v_j depicts the robot's configuration at time $j \cdot \Delta t$.

To extract semantic feedback, we define the VLM inference process as a mapping:

$$\Phi_{\text{VLM}} : \mathcal{V} \mapsto \mathcal{D}_{\text{motion}} = \{d_k\}_{k=1}^{K_{\text{desc}}}, \quad d_k \in \mathcal{L}_{\text{descriptor}}$$

where $\mathcal{L}_{\text{descriptor}}$ is the space of natural language descriptors. Each d_k characterizes an interpretable motion trait, such as posture alignment (e.g., "the torso remains upright"),

joint symmetry (e.g., "the left and right knees move synchronously"), or motion irregularities (e.g., "the ankle exhibits delayed lifting").

To improve focus and reduce irrelevant outputs, we apply **region-aware prompt engineering**. Specifically, input prompts are augmented with spatial constraints to direct the model's attention toward key morphological componentsknees, ankles, hips, and torso. This enhances the relevance and consistency of extracted features without requiring fine-tuning. In addition, we insert temporal continuity cues to ensure the VLM captures motion flow across frames, rather than isolated postures.

We further discretize these outputs into a structured feature set:

$$\mathcal{F}_{\text{motion}} = \{(\ell_i, \alpha_i)\}_{i=1}^{M_{\text{feat}}}, \quad \ell_i \in \mathcal{L}_{\text{label}}, \alpha_i \in [0,1]$$

where $\mathcal{L}_{\text{label}}$ is a label space of interpretable motion descriptors, and α_i denotes a confidence score derived from the VLM's attention or output logits. This feature set serves as a compact and interpretable representation of motion behavior, grounded in visual observations.

Crucially, this mechanism serves not only as a perceptual encoder but also as a **reward supervisor** during training. The semantically structured feedback from the VLM informs the large language model (LLM), guiding the synthesis of behaviorally aligned reward functions. For example, in a "straight-leg walking" task, if the VLM consistently produces descriptions such as "left knee joint - bending angle - 32" and "right knee joint - bending angle - 29," while the instruction $\mathcal{I}$ specifies "maintain an upright walking posture," the LLM will tend to generate reward formulations that penalize joint bending more heavily. This enables the overall system to better reinforce behaviors that align with human-level expectations and task semantics.

Compared to handcrafted motion metrics, the VLM-based feature extraction and supervision offer greater flexibility, generalizability, and interpretability. The resulting feature set $\mathcal{F}_{\text{motion}}$ not only supports semantic alignment during reward generation, but also provides a transparent diagnostic layer for evaluating policy quality across diverse tasks and morphologies.

C. Semantic Alignment and Iterative Reward Optimization

While the LLM and VLM modules provide mechanisms to extract task-relevant semantics and behavior-level summaries, respectively, the quality of the final policy is critically dependent on the accuracy and effectiveness of the learned reward function. To this end, MotionVL employs an iterative optimization scheme that aligns the generated reward functions with both environment structure and behavioral semantics. This section outlines how reward function evolution is driven by policy feedback and semantic validation.

We consider two categories of tasks:

- **(i) Tasks with canonical or community-endorsed reward functions**: In such cases, we initialize the iterative optimization with the known reference reward R_{ref}, and treat it as the initial solution. The LLM refines this baseline over multiple rounds by incorporating motion feedback from the VLM (as described in Section III-B), producing progressively improved variants that align more closely with intended behavior.
- **(ii) Tasks with ambiguous or poorly defined objectives**: In the absence of a reliable reference reward, we adopt a sampling-based strategy inspired by EUREKA [15]. In each search iteration n, we sample $K_{\text{sample}} = 10$ candidate reward functions independently from the LLM:

$$R_1^{(n)}, R_2^{(n)}, \ldots, R_{K_{\text{sample}}}^{(n)} \sim \text{LLM}(\mathcal{C}, \mathcal{P}_{\text{task}}, \mathcal{M})$$

where $\mathcal{C}$ is the simulation context (Section III-A), $\mathcal{P}_{\text{task}} \in \mathcal{L}_{\text{prompt}}$ is the instruction prompt, and $\mathcal{M}$ is the world model (Section II-A). Each candidate is executed in simulation for a short training cycle (e.g., $T_{\text{eval}} = 300$ steps), and its performance is evaluated using semantic feedback from the VLM.

In both cases, we apply a mutation-based optimization loop to refine the reward function. Given a best-performing reward $R_{\text{best}}^{(n)}$ from iteration n, we generate new candidates by prompting the LLM with a reward mutation template, enriched with textual summaries from the VLM and statistical indicators from policy training. This encourages reward evolution toward behaviors that are not only high-performing, but also semantically aligned with task goals.

To enhance controllability and consistency of reward generation, we regulate the LLM's output diversity using a temperature parameter. Specifically, we set temperature $\theta_{\text{temp}} = 0.3$, which strikes a balance between creativity and stabilityallowing for nuanced reward variations while maintaining reproducibility and interpretability.

Each search run proceeds for a fixed number of iterations (typically $N_{\text{iter}} = 5$), with $K_{\text{sample}} = 10$ samples per iteration. Once a reward function with satisfactory semantic alignment is identified, we commit to full training with extended rollout horizons (e.g., $T_{\text{full}} = 1000$ or more steps). This approach enables MotionVL to bootstrap high-quality reward functions even in novel environments, and systematically refine them based on structured language feedback.

Overall, this reward evolution process bridges perception, language, and control in a unified loop: the VLM interprets behavior, the LLM adjusts reward formulations, and policy gradients close the training loopyielding policies that are both task-effective and semantically interpretable. The following algorithm (1) summarizes the core steps of the MotionVL framework.

D. Simulation-to-Real Transfer and Deployment

To deploy MotionVL-trained policies on physical robots, we bridge the simulation-to-reality gap through domain randomization and joint-to-motor space mapping.

1) Domain Randomization: We randomize key parameters to enhance robustness: physical properties (e.g., friction $\mu \in [0.2, 1.3]$), actuator dynamics (e.g., PD controller gains $\kappa_{\text{stiff}} \in [0.8, 1.2]$), joint properties (e.g., friction $f_{\text{joint}} \in [0.1, 3.0]$), external disturbances (random push forces every $T_{\text{push}} = 8$ seconds), and communication latency (e.g., $\tau_{\text{cmd}} \in [1, 10]$ timesteps).

Algorithm 1 MotionVL Framework

Input: Environment Code $\mathcal{E}_{\text{code}}$, Robot Specification $\mathcal{U}_{\text{robot}}$, Execution Logs $\mathcal{H}_{\text{exec}}$, Task Prompt $\mathcal{P}_{\text{task}}$
Output: Optimized Policy π_{opt}
Initialize LLM LLM and VLM Φ_{VLM}
Step 1: Reward Function Design
for each timestep t **do**
    Generate semantically grounded state $\mathcal{X}_{\text{semantic}}$ using f_{init}
    Generate reward function $\mathcal{W}_{\text{reward}}$ using f_{reward}
end for
Step 2: Motion Feature Extraction
for each frame v_j in $\mathcal{V}$ **do**
    Extract motion descriptors $\mathcal{D}_{\text{motion}}$ using VLM Φ_{VLM}
end for
Step 3: Reward Optimization
for each task **do**
    if task has canonical reward function **then**
        Use reference reward R_{ref}
    else
        for each iteration n **do**
            Sample K_{sample} reward functions from LLM
            Evaluate performance with VLM feedback
        end for
    end if
    Optimize reward function
end for
Step 4: Policy Optimization
Optimize policy π_{opt} using $\mathcal{W}_{\text{reward}}$

2) Joint-to-Motor Space Mapping: The joint angle targets $\boldsymbol{\theta}_{\text{target}}$ generated by the policy are mapped to motor commands $\boldsymbol{u}_{\text{motor}}$ for real-world deployment. The target position at timestep t is computed as:

$$\boldsymbol{p}_t^d = \boldsymbol{p}_t + \beta a_t,$$

where $\boldsymbol{p}_t$ is the current position, $a_t \in \mathcal{A}$ is the action, constrained to $[-1, 1]$, and β regulates motion speed. The torque at timestep t is computed as:

$$\tau_t = K_p \cdot (\boldsymbol{p}_t^d - \boldsymbol{p}_t) - K_d \cdot \dot{\boldsymbol{p}}_t,$$

where K_p and K_d are the stiffness and damping coefficients. Finally, motor commands are generated using the PD control law and feedforward compensation, and are clipped to ensure they remain within hardware limits.

IV. Experiments

A. Experimental Setup

To evaluate the effectiveness of our proposed RL-LLM+VLM framework for humanoid robot control, we conducted extensive experiments under a standardized simulation setting. All experiments were performed on a workstation running Ubuntu 20.04, equipped with an NVIDIA GeForce RTX 4080 GPU (16 GB VRAM), Intel Core i9-14900K CPU, and 64 GB of RAM. The system was configured to support GPU-accelerated simulation and training.

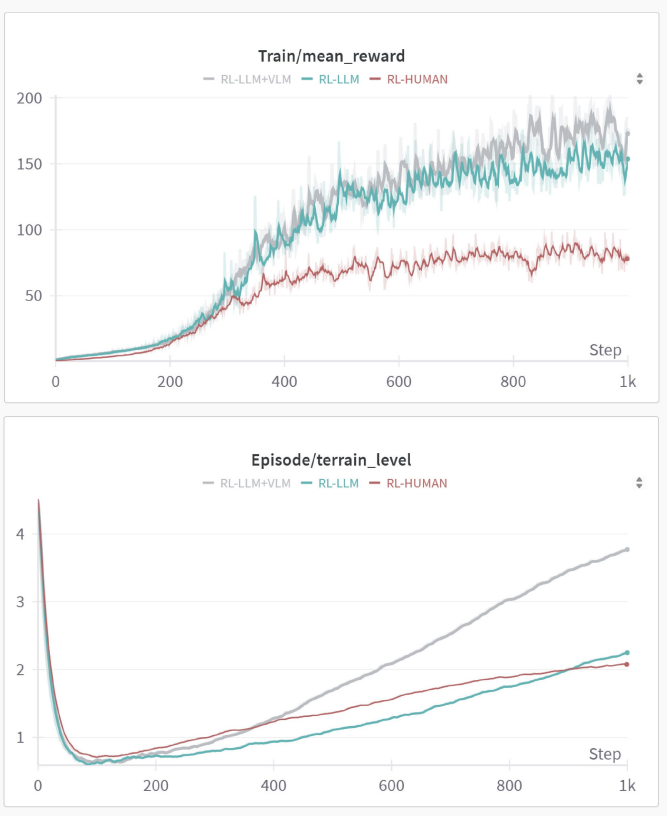


Fig. 2. Training performance under different reward design methods.

We employed Isaac Gym [7] as the primary high-throughput physics simulation environment, leveraging its capability to parallelize thousands of robot environments in a single GPU session. To assess sim-to-sim generalization performance, we also utilized the MuJoCo physics engine for additional validation tasks under varying physical dynamics. All models were implemented in Python using PyTorch 1.13, and the reinforcement learning algorithm was based on PPO (Proximal Policy Optimization) with standard hyperparameter tuning. Training for each model variant was conducted for 1000 steps, with results averaged over three independent seeds.

We compare three variants of our learning system: (1) a baseline reinforcement learning model (RL-Base) trained with conventional handcrafted reward functions; (2) an LLM-assisted variant (RL-LLM) where reward functions are generated from language descriptions; and (3) the full model (RL-LLM+VLM), which further incorporates multimodal perception and reward supervision through a vision-language model.

B. Ablation Study and Performance Analysis

Table I presents a detailed ablation study across the three aforementioned models. The RL-LLM+VLM framework outperforms both RL-Base and RL-LLM variants across the majority of performance metrics, validating the contribution of vision-language supervision and closed-loop alignment.

Policy Learning Efficiency.: RL-LLM+VLM achieves the highest mean noise standard deviation (**0.20810**), indicating a broader and more explorative policy distribution. This represents a 35.4% improvement over RL-Base and a 10.2%

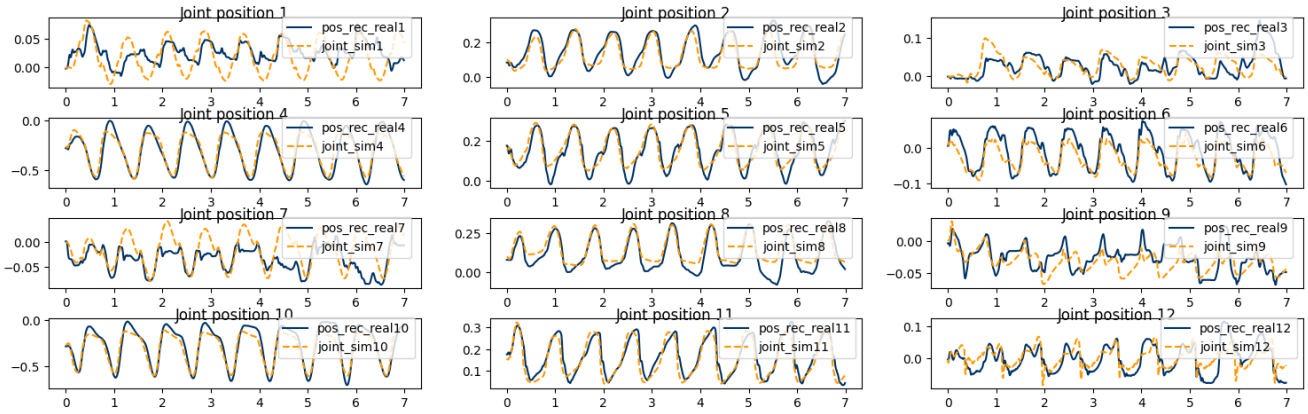


Fig. 3. Comparison of real (pos_rec_real) and simulated (joint_sim) joint angles (rad) over 7 seconds across 12 joints.

TABLE I
ABLATION STUDY OF RL-LLM+VLM AGAINST BASELINES. BOLD VALUES DENOTE THE BEST PERFORMANCE.

Category	Metric	RL-Base	RL-LLM	RL-LLM+VLM	UP(Base%)	UP(LLM%)
Policy	Mean noise std	$0.13950_{\pm 0.0003}$	$0.18890_{\pm 0.0006}$	$\mathbf{0.20810}_{\pm 0.0008}$	35.4	10.2
Training	Mean reward	$77.97000_{\pm 1.80}$	$147.84000_{\pm 5.10}$	$\mathbf{172.29000}_{\pm 13.14}$	89.6	16.5
	Episode length	$1773.27000_{\pm 12.73}$	$1769.71000_{\pm 71.90}$	$\mathbf{1784.19000}_{\pm 101.85}$	-0.2	0.8
Loss	Symmetric loss	$0.00679_{\pm 0.00015}$	$\mathbf{0.00711}_{\pm 0.00020}$	$0.00695_{\pm 0.00015}$	4.7	-2.3
	Learning rate	$0.00015_{\pm 0.00003}$	$\mathbf{0.00020}_{\pm 0.00006}$	$0.00013_{\pm 0.00003}$	33.3	-35.0
Episode Reward	Terrain level	$2.07700_{\pm 0.007}$	$2.24600_{\pm 0.008}$	$\mathbf{3.76700}_{\pm 0.010}$	8.1	67.7
	Tracking lin. vel	$0.41000_{\pm 0.043}$	$0.92800_{\pm 0.054}$	$\mathbf{1.13000}_{\pm 0.215}$	126.3	21.8
	Feet air time	$0.00720_{\pm 0.0002}$	$0.00820_{\pm 0.0007}$	$\mathbf{0.01050}_{\pm 0.0010}$	13.9	28.0
	Base height	$0.00056_{\pm 0.00006}$	$0.00315_{\pm 0.00012}$	$\mathbf{0.00601}_{\pm 0.00030}$	462.5	90.8
	Torques ($\times 10^{-7}$)	$\mathbf{-3.00800}_{\pm 0.195}$	$-3.26400_{\pm 0.147}$	$-3.50300_{\pm 0.059}$	-8.5	-7.3
	Vel mismatch	$0.23000_{\pm 0.23}$	$\mathbf{0.26000}_{\pm 0.28}$	$0.20000_{\pm 0.22}$	13.0	-23.1
	Tracking ang. vel	$0.65000_{\pm 0.66}$	$\mathbf{1.05000}_{\pm 1.13}$	$0.92000_{\pm 0.97}$	61.5	-12.4
	Track vel. hard	$\mathbf{-0.02000}_{\pm 0.02}$	$-0.02000_{\pm 0.02}$	$-0.03000_{\pm 0.03}$	0.0	-50.0
	Orientation	$0.42000_{\pm 0.44}$	$\mathbf{1.27000}_{\pm 1.38}$	$0.70000_{\pm 0.76}$	202.4	-44.9
	Low speed	$-0.22000_{\pm 0.21}$	$-0.30000_{\pm 0.34}$	$\mathbf{-0.22000}_{\pm 0.23}$	-36.4	26.7
	Knee distance	$0.15000_{\pm 0.15}$	$\mathbf{0.16000}_{\pm 0.18}$	$0.16000_{\pm 0.19}$	6.7	0.0
	Joint pos	$\mathbf{0.88000}_{\pm 0.90}$	$0.64000_{\pm 0.68}$	$0.37000_{\pm 0.41}$	-27.3	-42.2
	Foot slip	$\mathbf{-0.06000}_{\pm 0.06}$	$-0.19000_{\pm 0.21}$	$-0.19000_{\pm 0.20}$	-216.7	0.0
	Feet contact	$0.73000_{\pm 0.74}$	$1.05000_{\pm 1.14}$	$\mathbf{1.26000}_{\pm 1.38}$	43.8	20.0
	Feet distance	$0.15000_{\pm 0.15}$	$\mathbf{0.18000}_{\pm 0.19}$	$0.18000_{\pm 0.19}$	20.0	0.0
	Base acc	$\mathbf{0.10000}_{\pm 0.10}$	$0.08000_{\pm 0.09}$	$0.08000_{\pm 0.08}$	-20.0	0.0
	DoF vel	$\mathbf{-0.00010}_{\pm 0.00010}$	$-0.00012_{\pm 0.00014}$	$-0.00035_{\pm 0.00039}$	-20.0	-191.7
	DoF acc	$\mathbf{-0.00015}_{\pm 0.00016}$	$-0.00018_{\pm 0.00019}$	$-0.00042_{\pm 0.00047}$	-20.0	-133.3
	Action smooth	$\mathbf{-0.00710}_{\pm 0.0073}$	$-0.01140_{\pm 0.0123}$	$-0.01290_{\pm 0.0139}$	-60.6	-13.2
Additional Reward	Foot contact	–	$1.25700_{\pm 0.119}$	$\mathbf{1.26500}_{\pm 0.121}$	–	0.6
	Knee extension	–	$0.12000_{\pm 0.15}$	$\mathbf{0.13000}_{\pm 0.13}$	–	8.3
	Ankle stance	–	–	$0.00930_{\pm 0.0005}$	–	–
	Toe direction	–	–	$0.17000_{\pm 0.18}$	–	–

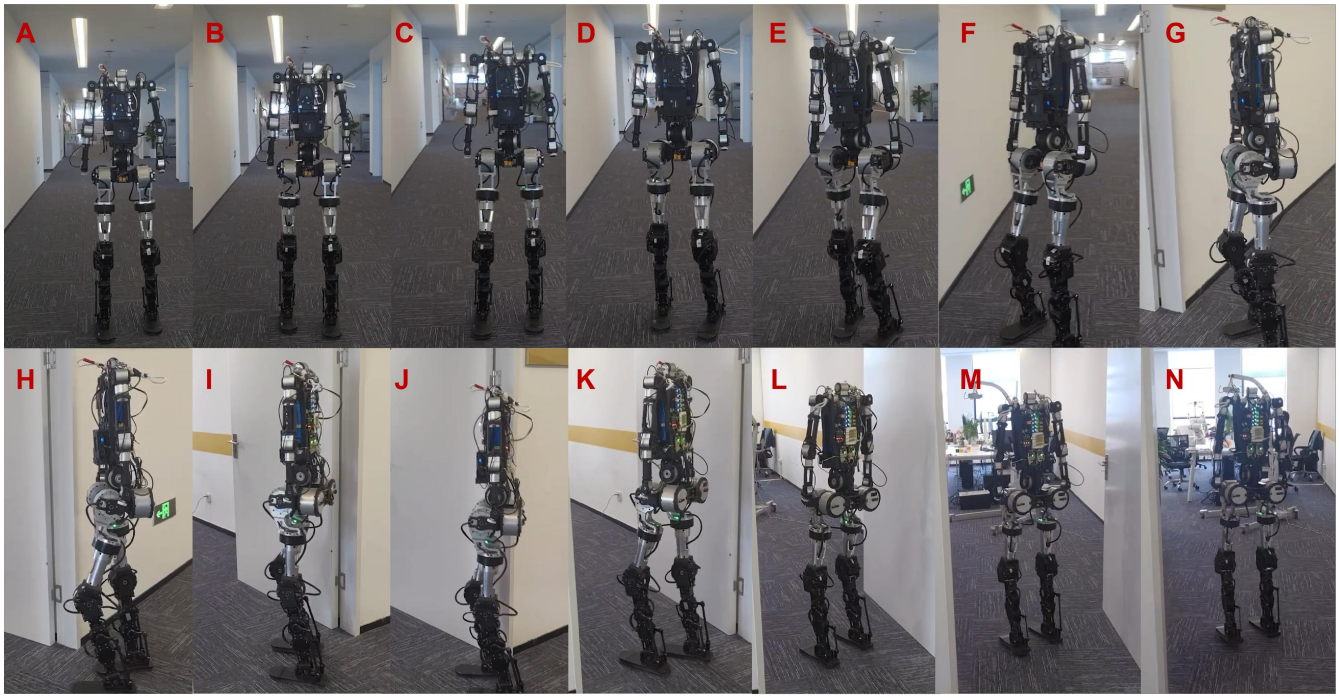
Note: All values shown as mean std. Bold marks the highest numerical value per row (not based on absolute value).

gain over RL-LLM, highlighting the benefit of incorporating semantic visual context into the learning process.

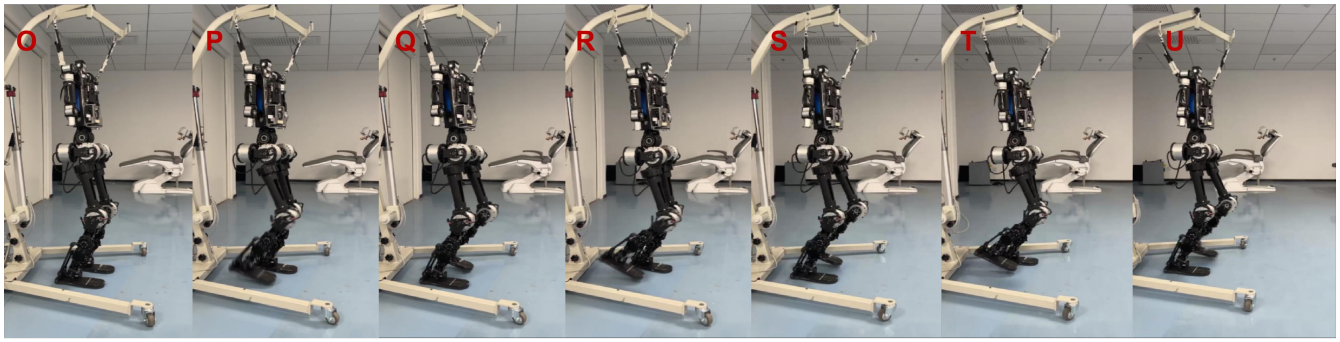
Training Dynamics.: In terms of training metrics, RL-LLM+VLM significantly improves the average episodic reward (**172.29**), nearly doubling that of RL-Base (77.97). The slight increase in average episode length suggests improved policy robustness and stability in long-horizon control tasks.

Loss Behavior.: The symmetric loss and learning rate metrics exhibit more nuanced trends. While RL-LLM yields the highest symmetric loss, RL-LLM+VLM maintains competitive values, indicating a better trade-off between expressiveness and regularization. Interestingly, RL-LLM's elevated learning rate may contribute to its faster early convergence, but RL-LLM+VLM converges to more stable and optimal policies.

Reward Decomposition.: A granular analysis of episodic reward components reveals key improvements in motion coordination and locomotion realism. Notably, RL-LLM+VLM significantly boosts terrain traversal capability (terrain level: **3.767**), linear velocity tracking (**1.130**), and base height stabilization (**0.00601**), compared to both baselines. These improvements confirm the system's ability to extract meaningful visual-linguistic feedback to shape complex gait patterns.



(a) Reinforcement learning in conjunction with VLM supervision



(b) Reinforcement learning using artificially designed reward functions

Fig. 4. (a) Robot walking in a knee-stretch gait and turning into the door of the room. (b) The robot walks forward in a basic gait.

Conversely, some metrics such as joint position fidelity and torque efficiency degrade slightly in the full model, suggesting a trade-off between biomechanical realism and task-directed agility. Moreover, orientation and angular velocity tracking exhibit fluctuations across variants, possibly due to the broader policy distribution induced by LLM-guided rewards.

Human-Like Gait Attributes.: Additional reward metricssuch as knee extension, toe direction, and ankle stancedemonstrate the full models ability to generate more human-like walking styles. For instance, the knee extension reward improves from 0.120 (RL-LLM) to **0.130** (RL-LLM+VLM), while newly introduced VLM-guided attributes (toe direction: 0.170, ankle stance: 0.0093) show that vision-language supervision enables the policy to implicitly optimize for biomechanically natural motions without requiring handcrafted tuning.

Qualitative Evaluation of Learned Gaits.: Fig. 4(a) illustrates the real-world deployment of the humanoid robot trained with our RL-LLM+VLM framework. The robot demonstrates a highly coordinated and human-like gait, featuring an upright torso, extended knees, and symmetric leg swinging. Beginning with straight-line walking (frames AD), it smoothly transitions into a turning maneuver (EG), and successfully executes a sharp left turn to navigate through a doorway (HN). Notably, this complex motion sequenceincluding real-time foot placement adjustment and balance preservation in a constrained corridoremerges without manual reward design or trajectory supervision. These results highlight the capability of vision-language guided reward shaping to induce biomechanically plausible, adaptable behaviors that generalize beyond training conditions.

Comparison with Basic Reinforcement Learning.: Fig. 4(b) presents a visual rollout of the baseline RL-Base model trained with handcrafted reward functions. The robot is able to walk forward in a relatively stable manner; however, the posture is noticeably crouched, with excessive knee bending and reduced step length. This ”squat-walking” gait is commonly observed in baseline RL agents due to reward misspecification, where simple velocity or stability-based objectives fail to promote natural locomotion. The contrast between (a) and (b) highlights the impact of semantically

grounded reward generationour RL-LLM+VLM policy not only achieves task completion but also promotes efficiency and human-likeness, validating the superiority of language-informed supervision.

These findings affirm the potential of integrating LLMs and VLMs into reinforcement learning pipelines for scalable, interpretable, and generalizable reward function design in humanoid robotics.

V. Conclusion

In this work, we presented the MotionVL framework, a novel reinforcement learning approach that integrates large language models (LLMs) and vision-language models (VLMs) to automate reward function generation and enforce semantic alignment in humanoid robot control. By employing a closed-loop structure wherein VLMs translate visual observations into structured semantic descriptions and LLMs synthesize task-consistent reward functions, MotionVL eliminates the need for manual reward engineering and enhances task interpretability. Furthermore, we introduced a semantic alignment mechanism that aligns policy representations and behaviors with multi-modal feedback, promoting both generalization and human-likeness.

Experimental results show that MotionVL outperforms traditional reinforcement learning baselines in terms of learning efficiency, reward decomposition, and gait realism. Ablation studies further highlight the complementary effects of language-guided supervision and multimodal perception in shaping complex locomotion behaviors. These findings suggest that the integration of foundation models can effectively bridge the gap between high-level task objectives and low-level control policies in embodied agents.

In future work, we aim to extend our approach to real-world deployments and investigate its applicability to a broader range of manipulation and interaction tasks. We also plan to explore adaptive prompting strategies and incorporate additional sensory modalities to further improve robustness and semantic grounding in dynamic, unstructured environments. Additionally, we will explore the use of higher-performance or specialized vision-language models (VLMs) to enhance the construction of supervised reinforcement learning methods, potentially improving both efficiency and task performance in complex environments.

References

[1] N. Scianca, D. De Simone, L. Lanari, and G. Oriolo, "Mpc for humanoid gait generation: Stability and feasibility," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1171–1188, 2020.

[2] S. Katayama, M. Murooka, and Y. Tazaki, "Model predictive control of legged and humanoid robots: models and algorithms," *Advanced Robotics*, vol. 37, no. 5, pp. 298–315, 2023.

[3] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi9579, 2024.

[4] A. K. Shakya, G. Pillai, and S. Chakrabarty, "Reinforcement learning algorithms: A brief survey," *Expert Systems with Applications*, vol. 231, p. 120495, 2023.

[5] B. Singh, R. Kumar, and V. P. Singh, "Reinforcement learning in robotic applications: a comprehensive survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 945–990, 2022.

[6] D. Han, B. Mulyana, V. Stankovic, and S. Cheng, "A survey on deep reinforcement learning algorithms for robotic manipulation," *Sensors*, vol. 23, no. 7, p. 3762, 2023.

[7] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[8] Z. Zhou, J. Song, X. Xie, Z. Shu, L. Ma, D. Liu, J. Yin, and S. See, "Towards building ai-cps with nvidia isaac sim: An industrial benchmark and case study for robotics manipulation," in *Proceedings of the 46th international conference on software engineering: software engineering in practice*, 2024, pp. 263–274.

[9] P. Kormushev, S. Calinon, R. Saegusa, and G. Metta, "Learning the skill of archery by a humanoid robot icub," in *2010 10th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2010, pp. 417–423.

[10] J. García and D. Shafie, "Teaching a humanoid robot to walk faster through safe reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 88, p. 103360, 2020.

[11] I. Radosavovic, S. Kamat, T. Darrell, and J. Malik, "Learning humanoid locomotion over challenging terrain," *arXiv preprint arXiv:2410.03654*, 2024.

[12] X. Gu, Y.-J. Wang, and J. Chen, "Humanoid-gym: Reinforcement learning for humanoid robot with zero-shot sim2real transfer," *arXiv preprint arXiv:2404.05695*, 2024.

[13] I. J. Silva, D. H. Perico, T. P. Homem, C. O. Vilao, F. Tonidandel, and R. A. Bianchi, "Using reinforcement learning to improve the stability of a humanoid robot: Walking on sloped terrain," in *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*. IEEE, 2015, pp. 210–215.

[14] H. Lee, S. Phatale, H. Mansoor, K. R. Lu, T. Mesnard, J. Ferret, C. Bishop, E. Hall, V. Carbune, and A. Rastogi, "Rlaif: Scaling reinforcement learning from human feedback with ai feedback," 2023.

[15] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, "Eureka: Human-level reward design via coding large language models," *arXiv preprint arXiv:2310.12931*, 2023.

[16] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Y. Wang, K. Fragkiadaki, Z. Erickson, D. Held, and C. Gan, "Robogen: Towards unleashing infinite data for automated robot learning via generative simulation," *arXiv preprint arXiv:2311.01455*, 2023.

[17] W. Ye, F. Liu, Z. Ding, Y. Gao, O. Rybkin, and P. Abbeel, "Video2policy: Scaling up manipulation tasks in simulation through internet videos," *arXiv preprint arXiv:2502.09886*, 2025.

[18] S. Wang, M. Han, Z. Jiao *et al.*, "Llm^3: Large language model-based task and motion planning with motion failure reasoning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 12 086–12 092.

[19] Z. Huang, Z. Sheng, Y. Qu *et al.*, "Vlm-rl: A unified vision language models and reinforcement learning framework for safe autonomous driving," *arXiv preprint arXiv:2412.15544*, 2024.

[20] S. Sun, C. Li, Z. Zhao *et al.*, "Leveraging large language models for comprehensive locomotion control in humanoid robots design," *Biomimetic Intelligence and Robotics*, vol. 4, no. 4, p. 100187, 2024.

[21] S. Singh, R. L. Lewis, and A. G. Barto, "Where do rewards come from," in *Proceedings of the annual conference of the cognitive science society*. Cognitive Science Society, 2009, pp. 2601–2606.