Biomimetic Intelligence and Robotics 4 (2024) 100187

Contents lists available at ScienceDirect

Biomimetic Intelligence and Robotics

journal homepage: www.elsevier.com/locate/birob

Research Article

# Leveraging large language models for comprehensive locomotion control in humanoid robots design

Shilong Sun [a,b,*], Chiyao Li [a], Zida Zhao [a], Haodong Huang [a], Wenfu Xu [a,b,c,1]

[a] School of Mechanical Engineering and Automation, Harbin Institute of Technology, Shenzhen 518055, China
[b] Guangdong Provincial Key Laboratory of Intelligent Morphing Mechanisms and Adaptive Robotics, Shenzhen 518055, China
[c] Key University Laboratory of Mechanism & Machine Theory and Intelligent Unmanned Systems of Guangdong, Shenzhen 518055, China

ARTICLE INFO

*Article history:*
Received 23 July 2024
Revised 20 September 2024
Accepted 5 October 2024
Available online 16 October 2024

*Keywords:*
Humanoid robots
Large language models
Locomotion control
Reinforcement learning

ABSTRACT

This paper investigates the utilization of large language models (LLMs) for the comprehensive control of humanoid robot locomotion. Traditional reinforcement learning (RL) approaches for robot locomotion are resource-intensive and rely heavily on manually designed reward functions. To address these challenges, we propose a method that employs LLMs as the primary designer to handle key aspects of locomotion control, such as trajectory planning, inverse kinematics solving, and reward function design. By using user-provided prompts, LLMs generate and optimize code, reducing the need for manual intervention. Our approach was validated through simulations in Unity, demonstrating that LLMs can achieve human-level performance in humanoid robot control. The results indicate that LLMs can simplify and enhance the development of advanced locomotion control systems for humanoid robots.



## 1. Introduction

Humanoid robot technology stands at the forefront of artificial intelligence and robotics, garnering significant attention and research interest due to its advanced capabilities and diverse applications. These sophisticated robots, meticulously designed to mimic human form and function, hold immense potential across a wide range of scenarios. In home assistance [1], humanoid robots can perform everyday tasks, providing support for the elderly and disabled, thereby enhancing quality of life. Industrial automation [2] can streamline manufacturing processes, increase efficiency, and reduce the risk of human error. The logistics sector benefits from their ability to handle complex sorting and delivery tasks with precision and speed [3]. Additionally, in hazardous environment exploration, humanoid robots can operate in conditions that are unsafe for humans, such as disaster sites or space missions, ensuring safety while performing critical operations. The versatility and adaptability of humanoid robots make them a pivotal focus in the ongoing evolution of AI and robotic technologies.

In recent years, significant developments have been made in the control methods for humanoid robots. Model-based control approaches, such as zero moment point (ZMP) [4], model predictive control (MPC) [5], and whole body control (WBC) [6], have been widely applied in locomotion control, often requiring analysis based on dynamic models like the table-cart model [7], inverted pendulum model [8], and multi-link models [9]. However, these methods entail complex computations and may introduce model errors. With the rapid advancement of machine learning and deep learning, learning-based methods, including RL [10,11] and imitation learning [12,13], have gradually been integrated into various robot RL tasks. RL, in particular, has found extensive application in the control of legged robots. It eliminates the need for explicit dynamic models and offers an effective end-to-end approach, demonstrating superior generalization and robustness. Nevertheless, RL heavily relies on reward design, which remains a challenging task, and its stability and reliability in complex environments are ongoing challenges.

Currently, large language models (LLMs) have been proven to play a significant role in various control tasks for robots. LLMs possess robust semantic understanding, multi-task processing, and code-writing capabilities, which can provide high-level task planning for robotic tasks and feedback for policy learning techniques [14]. Michael et al. [15] combined LLMs with the availability functions of robotic skills, enabling robots to execute highly abstract natural language instructions. Vemprala et al. [16] provided design principles for using LLMs in robotics and demonstrated how LLMs can help quickly extend robotic functions to different robotic task environments, allowing intuitive control of robotic arms, drones, home assistant robots, and more platforms

* Corresponding author at:
*E-mail address:* sunshilong@hit.edu.cn (S. Sun).

[1] Given his role as Associate Editor of this journal, Wenfu Xu had no involvement in the peer-review of this article and had no access to information regarding its peer-review. This article was handled by Prof. Mingguo Zhao.

https://doi.org/10.1016/j.birob.2024.100187

through language. Jacky et al. [17] used hierarchical code generation, enabling LLMs to recursively define unknown functions, a method that can express various robotic control strategies such as feedback controllers and vision-based grasping and placing.

However, translating natural language directly into action sequences for complex tasks remains challenging due to the hardware-dependent nature of low-level robot operations [18]. Many researchers have introduced intermediate interfaces to facilitate LLMs' understanding of low-level commands. For instance, Tang et al. [19] used the robot's foot contact patterns as an interface, allowing humans to issue commands through natural language for corresponding control. Moreover, utilizing reward functions as interfaces for LLMs to generate action sequences is a commonly adopted method due to the semantic richness and flexibility of reward functions [20–22]. Recent studies have shown that designing well-crafted prompts, such as texts, examples, and APIs, can significantly improve LLMs' success rates in generating code for low-level operations [23].

Most current efforts apply LLMs to specific aspects of robot control, capitalizing on their capabilities in tasks such as natural language processing [24], command interpretation, and high-level decision-making [25]. However, when it comes to complex, high-dimensional systems like humanoid robots, achieving effective locomotion control presents a multifaceted challenge. This process involves several intricate and interdependent stages, including precise trajectory planning [26], dynamic action sequence generation [27], and the creation of sophisticated reward functions [28]. Each of these tasks demands a deep understanding of robotics, advanced programming skills, and the ability to handle complex computational processes. Trajectory planning requires the robot to determine optimal paths for movement, considering factors such as balance [29], coordination [30], and efficiency [31]. This involves sophisticated algorithms that can adapt to changing environments and ensure smooth, continuous motion. Action sequence generation translates these trajectories into executable commands that control the robot's joints and actuators. This step requires precise timing and synchronization to maintain stability and achieve desired movements. Finally, the design of reward functions [20] is crucial for RL, guiding the robot to learn from its actions and improve its performance over time. These functions must be carefully crafted to balance immediate performance with long-term learning objectives.

This work aims to demonstrate how LLMs can significantly alleviate the burden on engineers by autonomously managing these critical and challenging tasks. By leveraging the advanced capabilities of LLMs, such as their robust semantic understanding [32], multi-task processing [33], and code generation abilities [23], we can streamline the development process for humanoid robots. Instead of manually designing and coding each aspect of locomotion control, engineers can use LLMs to generate and refine the necessary code. This not only reduces the need for extensive manual intervention but also accelerates the development cycle and enhances the overall efficiency of creating advanced robotic systems. Moreover, the integration of LLMs in locomotion control enables continuous improvement and adaptation. As the LLMs receive feedback and learn from the robot's performance, they can optimize the generated code, improving the robot's ability to navigate complex environments and perform tasks with greater precision and reliability. This iterative process of refinement and feedback ensures that the robot's control systems are constantly evolving, leading to more robust and effective locomotion capabilities. Therefore, applying LLMs to handle the comprehensive control tasks of humanoid robots represents a significant advancement in the field of robotics. This approach not only alleviates the workload on engineers but also enhances the development and deployment of sophisticated robotic systems. By harnessing the power of LLMs, we can achieve more efficient, adaptable, and scalable solutions in humanoid robot technology, ultimately paving the way for broader applications and innovations in this exciting field.

To address the issues above, we leverage the advanced capabilities of the latest LLMs (GPT-4o), which excel in semantic understanding, code writing, and zero-shot generation. By providing well-formulated prompts to the LLMs, we use the model as the primary designer to complete a comprehensive locomotion control task for a humanoid robot, as outlined in Fig. 1. LLMs assume three roles crucial to the locomotion control task: trajectory planner, inverse kinematics solver, and reward designer, each of which is pivotal and challenging. Users can evaluate the code generated by LLMs and provide appropriate feedback and refined prompts, enabling LLMs to output correct executable code.

The main contributions of this work can be summarized as:

- We proposed a method for utilizing LLMs to replace engineers in completing all critical stages of a locomotion control task for a humanoid robot.
- We designed various prompts and input them into the LLMs, generating code capable of successfully executing all critical stages of the task.
- We validated our method through simulations, demonstrating that LLMs can achieve human-level performance in this control task.

It should be noted that all processes were achieved by carefully crafting and setting reasonable prompts for the LLMs. These prompts were designed to guide the LLMs through various stages of the control tasks. The models could then autonomously generate and refine the necessary code, optimizing the results based on feedback provided by the users. This process eliminated the need for manual code adjustments, thereby streamlining the development workflow and significantly reducing the time and effort traditionally required for such tasks. The iterative prompt-based approach ensured that the LLMs could adapt and improve their outputs, leading to efficient and effective locomotion control for the humanoid robot.

The structure of this paper is as follows: Section 2 introduced the related work about the RL and LLMs in legged robots control. Section 3 describes the overall proposed control framework, the robot model used, the process of prompts design, and a comparative description. Section 4 describes the formulation of the RL problem. Section 5 presents the simulation results and analysis. Section 6 provides a summary of the entire paper and looks forward to future work.

## 2. Related work

In this section, we delve into the extensive body of prior work focused on utilizing RL methods to control legged robots, as well as the innovative application of language models in various robotic tasks. The field of RL has seen significant advancements, providing robust frameworks for enabling autonomous and adaptive behaviors in legged robots. Researchers have explored numerous RL-based approaches, addressing challenges such as dynamic balance, terrain adaptability, and efficient locomotion strategies. These studies have demonstrated the potential of RL to replace traditional model-based control methods, offering more flexible and scalable solutions for complex robotic systems.

Simultaneously, the integration of language models into robotic tasks has opened new avenues for enhancing robot autonomy and interaction. LLMs, with their advanced natural language processing capabilities, have been employed to facilitate high-level planning, command interpretation, and even code generation for robotic control. This dual focus on RL and LLMs represents

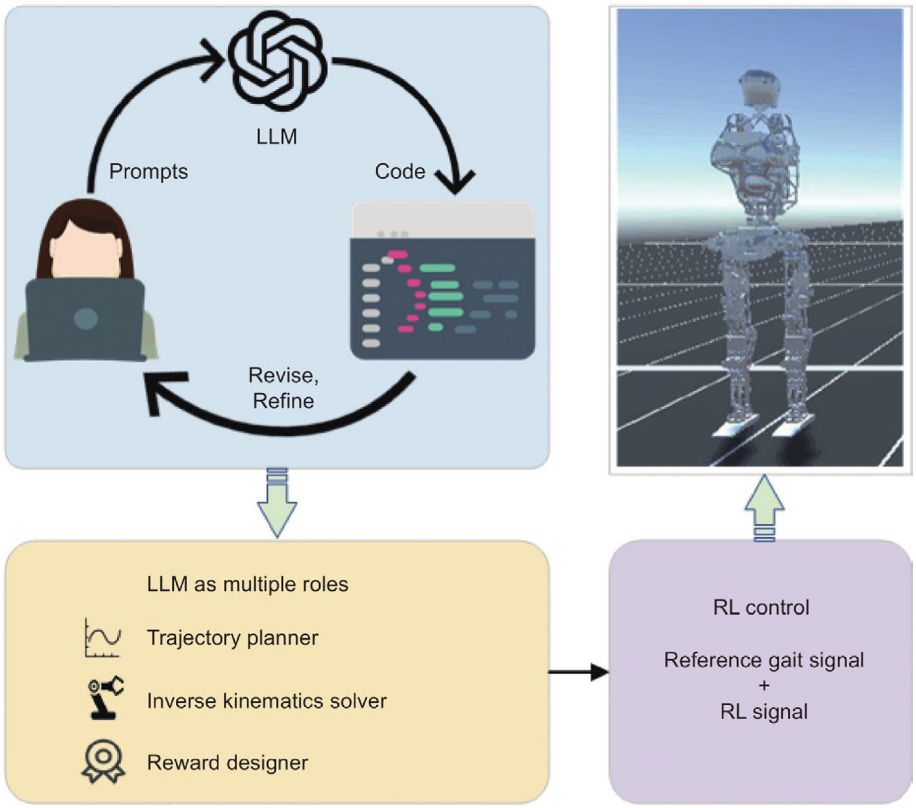



**Fig. 1.** Overview of Our Work. LLMs undertake three main stages in a humanoid robot control task: trajectory planning, inverse kinematics solving, and reward function design. The steps guiding LLMs in accomplishing the task are provided (blue). The results generated by LLMs (yellow) are input into the control layer, which uses a method combining reference gait signals and RL signals (purple).

a convergence of machine-learning techniques aimed at advancing the capabilities of legged robots. By leveraging the strengths of both RL and LLMs, researchers are pushing the boundaries of what robots can achieve, paving the way for more intuitive, adaptable, and intelligent robotic systems.

### *2.1. RL control for legged robots*

RL [34–36] has emerged as a powerful technique for controlling legged robots, offering a data-driven approach that bypasses the need for explicit dynamic models. Unlike traditional model-based methods, which often involve complex computations and potential model inaccuracies, RL provides an end-to-end solution that can adaptively learn from interactions with the environment. This adaptability makes RL particularly well-suited for the dynamic and unpredictable nature of legged robot locomotion. In recent years, significant advancements have been made in applying RL to various types of legged robots [37], demonstrating superior generalization, robustness, and the ability to perform complex locomotion tasks. This subsection reviews the current state of RL in legged robot control, highlighting critical methodologies, challenges, and breakthroughs in the field.

Currently, RL has been widely applied to the motion control of various legged robots. Ilija et al. [34] proposed a learning-based method, using a teacher–student policy during the training process and employing a causal Transformer model as the neural network controller, which takes proprioceptive observations and historical actions as inputs to predict the next action. This method enables the real humanoid robot Digit to walk in various outdoor environments. Xie et al. [38] described the characteristics of policies through deterministic action stochastic state (DASS) tuples, separately training three expert policies for walking forward, walking in place, and walking backward, and distilled them into a single policy that handles all three tasks. This unified policy demonstrated stable walking with different gait styles and speeds on the bipedal robot Cassie. Lokesh et al. [39] proposed learning a linear policy for walking and navigating the robot Digit on different terrains. This method achieves transfer from simulation to reality without any adjustment or use of dynamic randomization. The Swiss Federal Institute of Technology in Zurich (ETH) developed the ANYmal robot [40–42], which has been trained to integrate proprioceptive and exteroceptive sensing. It can utilize sensory information from exteroceptive sensors to quickly predict terrain, while still relying on proprioceptive sensing to adapt to complex environments when external sensing is unreliable. ANYmal can traverse various off-road terrains such as mud, sand, and snow, and even completed a hiking trip in the Alps. In recent publications, ANYmal demonstrated agile navigation and successfully completed various parkour challenges [43]. Tan et al. [44] enhanced simulation fidelity by developing precise actuator and delay models and by improving controller robustness through physical parameter randomization. Deepali et al. [45] proposed a new hierarchical RL method for quadrupedal locomotion tasks, demonstrating the automatic decomposition of complex tasks through the transfer of low-level policies, effectively adapting to new tasks. They tested the proposed framework on the physical robot Minitaurs for path-following tasks. Yang et al. [46] introduced a multi-expert learning architecture (MELA), a hierarchical control framework capable of generating adaptive motion skills achieving extensive motor expertise. The proposed method was successfully demonstrated on the robot Jueying, achieving multiple skillful motions.

### 2.2. LLMs to robot tasks

LLMs [47,48] have increasingly become a focal point in the realm of robotics, demonstrating significant potential in enhancing both the reasoning and planning capabilities required for various robotic tasks. These models, known for their robust semantic understanding, multi-task processing, and code generation abilities, offer innovative solutions to complex control and planning problems that have traditionally challenged roboticists. In this subsection, we delve into the application of LLMs in robotic tasks, exploring how their integration can revolutionize task execution, from high-level planning to low-level motion control. We will examine the advancements made in using LLMs for autonomous driving, task, and motion planning, and their recent applications in directly controlling robotic actions. Through this exploration, we aim to highlight the transformative impact of LLMs on the efficiency, adaptability, and overall capability of robotic systems.

LLMs have already been extensively researched in the reasoning and planning of robotic tasks [49–52]. This is particularly evident in the fields of robotic arm control and task planning [53–55]. LLMs can transform high-level human instructions into specific operational steps for robotic arms by understanding and generating natural language [56,57], reducing the need for complex programming. Moreover, LLMs can enhance the adaptability and flexibility of robotic arms by learning the context of different tasks [58]. This capability provides a more intuitive and convenient solution for human–machine collaboration and intelligent automation. In addition, LLMs are also applied to solve other complex task planning problems. Wang et al. [59] proposed a novel LLM-driven task and motion planning (TAMP) framework called LLM3, which leverages the reasoning and planning capabilities of pre-trained LLMs to generate symbolic action sequences and continuous action parameters. It iteratively optimizes the planning scheme based on feedback from motion planning, thereby eliminating the need for manually designed domain-specific interface modules required by traditional TAMP methods. Zhao et al. [60] utilized the common sense knowledge of LLMs to guide search algorithms (such as Monte Carlo Tree Search) and applied it to everyday task planning in home environments.

Recently, an increasing number of researchers have applied LLMs to the action control of robots [61–65], not just to high-level task planning. Early work typically used standardized text templates to translate language into robot commands, but LLMs cannot directly generate low-level robot motion commands, such as joint position targets. To address this issue, many researchers leverage the code generation capabilities of LLMs to extend low-level primitive skills to the full expressive power of code [17,50,66–68]. These methods often design an intermediate interface to connect the LLM and robot commands [69,70].

## 3. Proposed method

### 3.1. Control framework

Our control framework is illustrated in Fig. 2. We utilize the state-of-the-art LLMs (ChatGPT-4-turbo) as the primary designer for locomotion control of a humanoid robot. It assumes three roles: trajectory planner, inverse kinematics (IK) solver, and reward designer. Initially, users input prompts to the LLMs acting as the trajectory planner, which outputs trajectories to the subsequent LLMs acting as the IK solver. Users also provide relevant prompts to the IK solver, which outputs a time series of joint angles for the robot's 12-leg joints. Subsequently, the Reward designer generates rewards based on given prompts. Our control framework utilizes the angle sequences computed by LLMs and rewards generated through training to achieve forward stable walking locomotion control for the humanoid robot.

We adopt a control approach that combines weighted references from gait signals with RL action signals. Introducing reference signals in RL has proven effective for facilitating training, such as imitation learning [71] and instruction learning [72]. The reference signal we employ consists of joint angle sequences for the legs derived from inverse kinematics, representing an ideal locomotion pattern obtained from pre-planned trajectory solutions. The RL component achieves task objectives through reward settings. Here, the signal output from RL is multiplied by a carefully set weight, balancing the influence of RL and reference signals. Incorporating reference gait signals enables the robot to perform relatively reasonable actions early in the learning process, significantly reducing the time spent exploring random actions and accelerating overall learning. Meanwhile, RL retains exploration space to fine-tune the reference signals, maintaining body balance and generating more humanoid-like locomotion. Finally, we employ a PD control method to regulate the robot's joint movements.

### 3.2. Robot model

In this paper, we leverage the HIT Hydraulic Robot with a total of 14 joints, with its center of mass located at the pelvis. Each leg of the robot comprises six degrees of freedom, including three hip joints, one knee joint, and two ankle joints. Additionally, there are two joints at the waist. Specific details of the robot model are depicted in Fig. 2 (see Fig. 3).

### 3.3. Prompts design for humanoid control

ChatGPT, trained on extensive large-scale data, excels in both language understanding and generation. By providing carefully designed prompts, we can significantly enhance the success rate of LLMs in generating accurate and contextually appropriate responses.

In this work, we leverage ChatGPT-4-turbo to fulfill three critical roles in the locomotion control tasks of humanoid robots: trajectory planner, inverse kinematics solver, and reward designer. Each role is assigned specific, meticulously crafted prompts to guide the LLM in performing its tasks effectively. The trajectory planner is responsible for generating optimal movement paths for the robot, the inverse kinematics solver computes the necessary joint angles to achieve these movements, and the reward designer formulates the reward functions to optimize the robot's learning process through RL.

By using carefully designed prompts, we can fully leverage the LLMs' excellent programming and design capabilities across various tasks. Additionally, through user feedback, LLMs can optimize the output, thereby providing high-quality solutions for complex humanoid robot locomotion challenges. This approach not only streamlines the development process but also demonstrates the potential of LLMs to handle sophisticated engineering tasks with minimal manual intervention autonomously.

#### 3.3.1. Trajectory planner

Trajectory planning for the robot aims to provide precise target positions and orientations for solving the inverse kinematics, which in turn determines the sequence of joint angles to be used as reference signals. To achieve this, we leverage LLMs to generate the position change curves for the center of mass (COM) and the left and right ankles in the x, y, and z directions within the world coordinate system. For the target orientations, we simplify the leg joint solutions by assuming that both feet remain level with respect to the transverse plane of the torso. Our primary task is to generate a trajectory that enables the robot to walk forward effectively and efficiently.

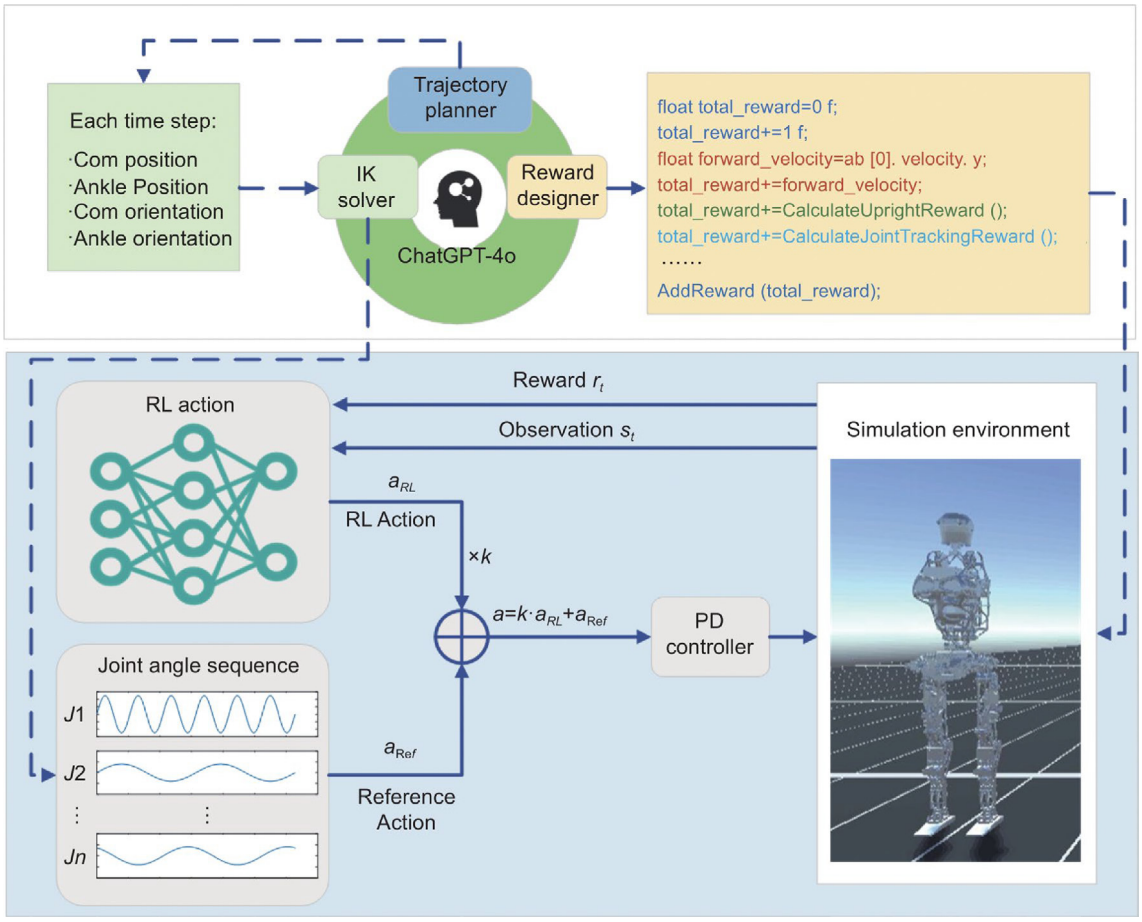
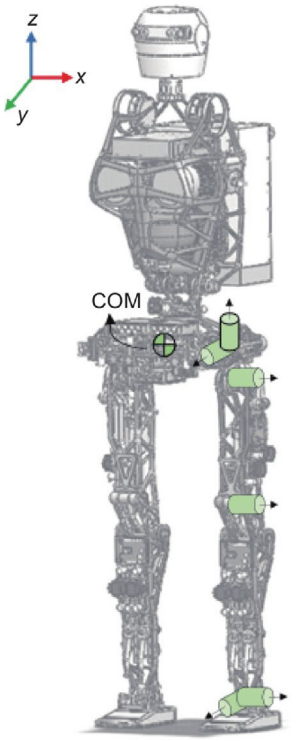



**Fig. 2.** LLMs-assisted for Humanoid robot control framework.




| Physical parameters | | Range of joint angles | |
|---|---|---|---|
| Height | 165 cm | Joint | Degrees (°) |
| Link | Mass (kg) | Hip (roll) | (−15, 15) |
| Body | 22.70 | Hip (yaw) | (−15, 15) |
| Waist | 2.28 | Hip (pitch) | (−45, 30) |
| Pelvis | 5.58 | Knee (pitch) | (0, 90) |
| Hip base | 0.46 (×2) | Ankle (pitch) | (−60, 60) |
| Hip | 0.60 (×2) | Ankle (roll) | (−15, 15) |
| Thigh | 5.63 (×2) | Waist (pitch) | (−1, 1) |
| Leg | 3.95 (×2) | Waist (yaw) | (−1, 1) |
| Ankle | 0.20 (×2) | | |
| Foot | 1.01 (×2) | | |

**Fig. 3.** Robot model, physical parameters, and range of joint angles.

```
<Locomotion Task Description>

1.A humanoid robot needs to move forward at a speed of 0.25 m/s.

2.Assume the robot's COM is located at the center of the pelvis and remains at a constant height throughout the locomotion process.

3. Please design the trajectories for the robot's COM and its left and right ankle joints. Also, generate MATLAB code for this task.

<Robot Dimension Information>

1.The distance between the two hip joints is 0.374 meters.

2.The robot walks with bent knees, and the COM remains at a height of 0.6 meters.

<Gait Description>

1.Stride length: 0.4 meters; Step height: 0.1 meters; Walking cycle: 1.6 seconds; Phases: divided into stance phase and swing phase.

2.The left leg initiates the movement. Initially, both feet are positioned on the same horizontal line.

<Trajectory Constraints>

1.The origin of the WCS is assumed at the robot's initial position where the COM projects onto the ground, with x as forward, y as lateral, and z as vertical direction.

2.The generated trajectories are in the absolute positions of the COM, left and right ankle joints in the WCS's x, y, z directions, not relative positions.

3.The COM follows a sinusoidal-like curve in the y-direction, with a bias towards the supporting leg during walking, and the amplitude in the y-direction is set to one-fourth of the distance between the two hip joints.

4.Ensure the generated trajectories are as smooth as possible.

Note: COM stands for Center of Mass, and WCS stands for World Coordinate System.
```

**Fig. 4.** Prompts for trajectory planner.

In designing the prompts, we first included a description of the motion task, such as walking forward at a speed of 0.25 m/s while maintaining a constant center of mass height, and clearly specified the need to generate the trajectories for the robot's center of mass and left and right ankles. Additionally, we provided relevant robot dimensions, such as the distance between the hip joints (which is also the distance between the ankles) and the height of the pelvis (i.e., the center of mass height) during walking. These pieces of information are essential parameters and prerequisites for trajectory planning. Furthermore, gait planning is typically required before trajectory planning. However, LLMs cannot flexibly perform gait planning based on basic task requirements as humans do. Therefore, we supplemented the prompts with information about gait planning, including step length, step height, and gait cycle. Although LLMs were able to generate trajectories for the center of mass and ankles after receiving these prompts, there were still some errors in the output. To correct these errors, we continued to provide more relevant information to the LLMs, such as constraints on the output trajectories, gradually guiding them to adjust and optimize the results without any manual intervention. After several iterations of corrections, we developed a set of prompts that enabled the LLMs to accurately generate the trajectory curves. The designed prompts consisted of four parts: a detailed description of the locomotion task, the critical dimensions and parameters of the robot, a comprehensive description of the gait, and the specific constraints for the trajectory. The specific details of the prompts are shown in Fig. 4. While the prompts are highly detailed, they do not prescribe the exact type of curve to be used, except for the $y$-direction trajectory of the COM. Instead, the LLMs autonomously generate all other trajectories, ensuring adaptability and flexibility in the trajectory planning process.

The prompts in Fig. 4 represent a refined version that has been modified to near perfection through iterative testing and adjustments. However, even with these optimized prompts, there may still be a need to fine-tune the generated trajectory in specific directions. This can be achieved by continuing to input additional prompts to correct any deviations or errors. After this optimization process, the resulting trajectory, as depicted in Fig. 5, demonstrates the effectiveness of using LLMs for sophisticated trajectory planning tasks. The LLMs' ability to autonomously generate and refine these trajectories showcases their potential to handle complex robotic control challenges with minimal manual intervention, ultimately streamlining the development process and enhancing the robot's performance.

### 3.3.2. Inverse kinematics solver

An inverse kinematics solver has always been a commonly used and critical control component in legged robots [73], responsible for calculating the required joint angles to achieve specific positions and orientations of the feet. Common methods for inverse kinematics include analytical methods and numerical methods. In this paper, we employ the numerical approach known as the Newton–Raphson iteration method based on the Jacobian matrix to solve inverse kinematics.

Taking the left leg as an example, we establish the base coordinate system at the center of mass, denoted as 0, and the coordinate system of the left ankle joint is denoted as 6. By using the Denavit–Hartenberg (DH) method and analyzing the link coordinate systems of the leg, we can calculate the pose of the ankle joint at each time step relative to the base coordinate system as.

$$T_{current} = {}^{0}T_{6}(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) \tag{1}$$

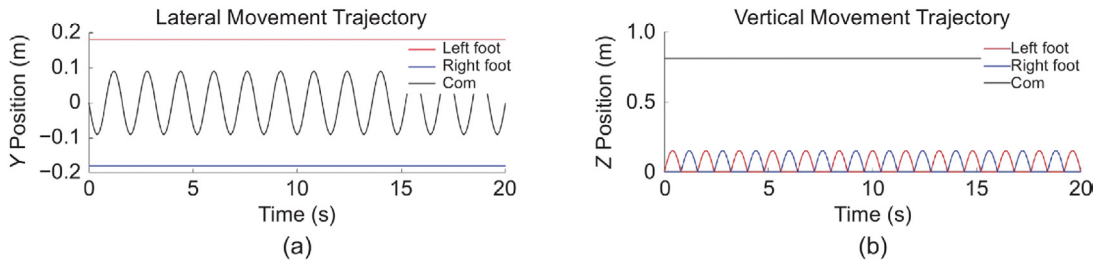



**Fig. 5.** Lateral and vertical movement trajectory generated by LLMs. (a) Lateral movement trajectory. (b) Vertical movement trajectory.

```
<Given Conditions>
1.A humanoid robot with known com and left ankle trajectory, where the position changes over time and the posture is known. From the trajectory, the target position and posture of the foot at each time step can be calculated.
2.The robot's single leg has 6 degrees of freedom.
<Task Description>
    Using the Jacobian matrix-based Newton-Raphson iterative method to solve the inverse kinematics for the humanoid robot's left foot. Generate MATLAB code for this.
<DH Parameters for the Leg>
    The forward kinematics parameters are as follows:
    a = [0, 0, 0.08, 0.37, 0.33, 0];
    d = [0, 0, 0, 0, 0, 0];
    alpha = [0, pi/2, -pi/2, 0, 0, pi/2];
    theta=[ theta1, theta2, theta3, theta4, theta5, theta6].
<Relevant Formulas>
1.Jacobian Matrix in LaTeX format.
2. Newton-Raphson iterative formula in LaTeX format.
```

**Fig. 6.** Prompts for IK solver.

where $T_{current}$ is a function of the six joint variables $\theta(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ of the leg. Given the desired end-effector pose (the ankle joint of the robot), all joint positions of the leg can be solved through inverse kinematics algorithms. We express all changes in end-effector poses in the base coordinate system.

$$T_{target} = {}^{0}T_6 = ({}^{W}T_0)^{-1}{}^{W}T_6 \tag{2}$$

where $T_{target}$ represents the end-effector pose relative to the base coordinate system, ${}^{W}T_0$ represents the centroid coordinate system pose relative to the world coordinate system, and ${}^{W}T_6$ represents the ankle joint pose relative to the world coordinate system.

For small changes in joint angles $\partial q$, the infinitesimal change in end-effector pose is denoted as $(\partial p, \partial w)$

$$\begin{bmatrix} \delta p \\ \delta w \end{bmatrix} = J\delta q \tag{3}$$

where $J$ represents the Jacobian matrix. Let $f_k(\theta) = T_{current}(i,j) - T_{end}(i,j)(k = 1, 2, \ldots, 12)$, due to the homogeneous transformation matrix having a rotation matrix with nine elements in the upper left corner and a positional vector with three elements in the upper right corner, subtracting these corresponding 12 elements results in 12 equations, which represent the change in pose. The Jacobian matrix can be calculated by

$$J(\theta) = \begin{bmatrix} \frac{\partial f_1(\theta)}{\partial \theta_1} & \frac{\partial f_1(\theta)}{\partial \theta_2} & \frac{\partial f_1(\theta)}{\partial \theta_3} & \frac{\partial f_1(\theta)}{\partial \theta_4} & \frac{\partial f_1(\theta)}{\partial \theta_5} & \frac{\partial f_1(\theta)}{\partial \theta_6} \\ \frac{\partial f_2(\theta)}{\partial \theta_1} & \frac{\partial f_2(\theta)}{\partial \theta_2} & \frac{\partial f_2(\theta)}{\partial \theta_3} & \frac{\partial f_2(\theta)}{\partial \theta_4} & \frac{\partial f_2(\theta)}{\partial \theta_5} & \frac{\partial f_2(\theta)}{\partial \theta_6} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_{12}(\theta)}{\partial \theta_1} & \frac{\partial f_{12}(\theta)}{\partial \theta_2} & \frac{\partial f_{12}(\theta)}{\partial \theta_3} & \frac{\partial f_{12}(\theta)}{\partial \theta_4} & \frac{\partial f_{12}(\theta)}{\partial \theta_5} & \frac{\partial f_{12}(\theta)}{\partial \theta_6} \end{bmatrix} \tag{4}$$

Due to $J \in R^{12\times 6}$, direct inversion is not feasible. Therefore, its pseudoinverse is used, yielding the iterative formula for joint angles:

$$\theta_{n+1} = \theta_n + J^{+}\begin{bmatrix} \delta p \\ \delta w \end{bmatrix} \tag{5}$$

where $J^{+}$ denotes the pseudoinverse of $J$. Iteration stops when the error in the end-effector pose is sufficiently small.

For the complex IK-solving process, we utilized LLMs to generate code. First, we provided some known conditions, such as the positions and orientations of the robot's center of mass and ankles during motion, and that each leg has six degrees of freedom. Next, we described the task: please use the Newton–Raphson method to solve the inverse kinematics for the humanoid robot's legs. After inputting this information into the LLMs, we obtained a basic code framework. This framework appeared comprehensive, outlining all the steps of the inverse kinematics algorithm, including setting the target positions and orientations, computing the forward kinematics for the legs, iterating the inverse kinematics solution, and providing examples of Jacobian matrix calculations. However, this was just a code framework and was not executable. The LLMs needed more relevant knowledge to generate accurate results, so we continued to provide the DH parameters for the robot's legs, specific formulas for the Jacobian matrix, and the angular iteration formulas in the Newton–Raphson method. To ensure LLMs could recognize the formulas, we input them in LaTeX format. Therefore, the final prompts for solving this task were divided into four parts: given conditions, task description, DH parameters for the leg, and relevant formulas. The refined prompts are depicted in Fig. 6.

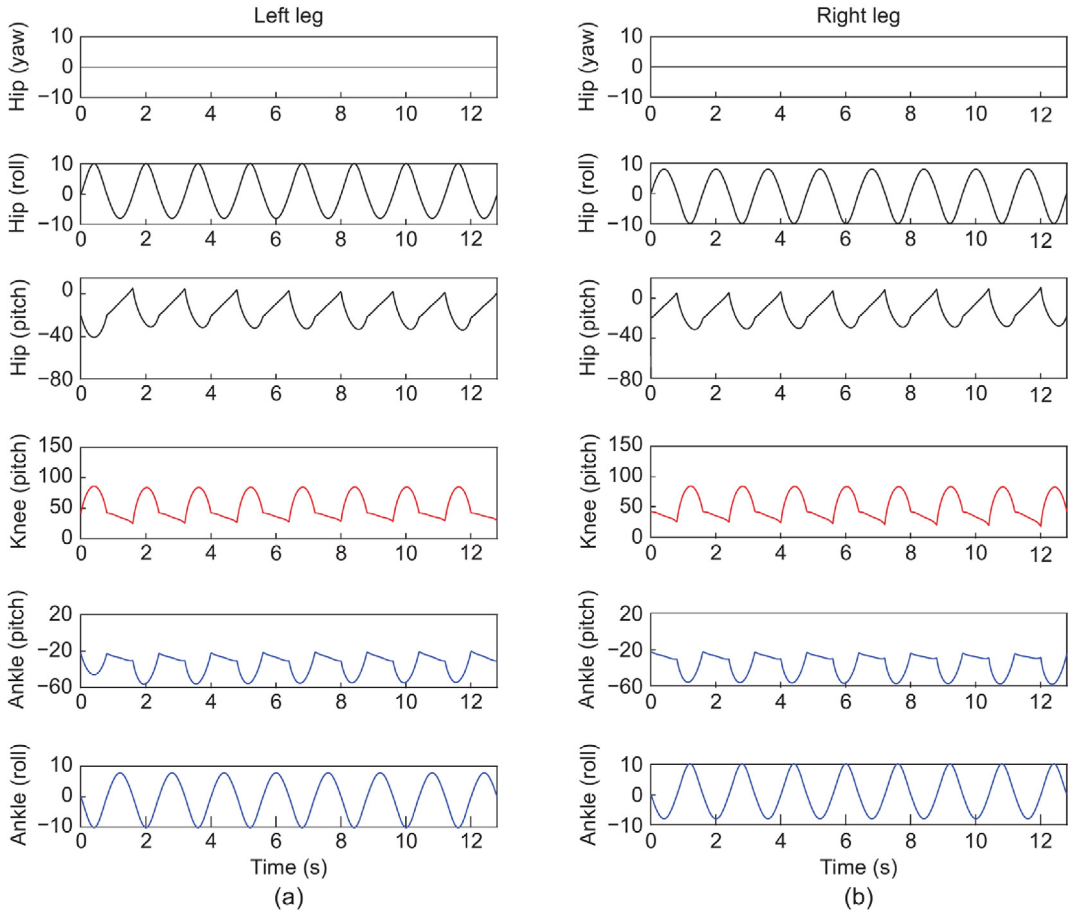



**Fig. 7.** The Joint Angle Sequence of the Legs. (a) Left leg angles. (b) Right leg angles.

Although no specific steps were provided, the LLMs generated a complete numerical solving framework based on their internal knowledge, including the setting of target positions and postures, forward kinematics calculations, the iterative process for inverse kinematics solving, and the computation of the Jacobian matrix. Additionally, the LLMs automatically specified the solving precision and the number of iterations. This showcases the LLMs' exceptional abilities in code generation and algorithm design.

Using the prompts provided above, we generated code and imported the trajectory generated by LLMs in the previous step into the code, resulting in the sequence of joint angles for the left and right legs, as shown in Fig. 7. The figure effectively illustrates the coordinated and periodic movements of a humanoid robot's hip, knee, and ankle joints during a walking gait cycle. The distinct oscillation patterns across different joints and axes underscore the complexity of achieving stable and efficient locomotion.

### 3.3.3. Reward designer

Designing a reward function for RL control methods has always been a challenging issue due to the complexity and nuance required to guide the learning process effectively [11,74]. To explore the capabilities of state-of-the-art LLMs in this domain, we tested whether they could autonomously design a reward function capable of achieving human-level performance.

In RL control methods, setting the reward function is crucial for guiding the robot's behavior, as it formalizes various actions of the robot. Therefore, when designing the prompts, we first described the robot's behavioral goals, including moving forward while maintaining body balance, avoiding falls, and tracking the reference angle sequences for the leg joints. During RL training, the robot's movement is essentially controlled by adjusting the leg joints, so specifying the number of joints on the robot is important information in the prompts. Additionally, the directions represented by the components of the robot's body coordinate system are crucial for reward design. For example, the reward function can be set to reward the robot's forward movement speed and penalize speeds in other directions. Different simulation software has different environment code, so we included the environment code framework for Unity's RL module and related APIs in the prompts. Finally, we provided an example of setting survival rewards for the robot. Through this example, the LLMs can use their few-shot learning capability to generate other reward functions consistent with this format. This approach effectively helps the model understand and generate reward function designs suitable for RL environments. Therefore, the prompts for the reward design phase were ultimately composed of four parts: task description, robot information and direction explanation, environment code and related APIs, and example of output format. The specific details of the prompts are shown in Fig. 8. By inputting these prompts, the LLMs were able to generate a remarkably reasonable reward function on the first attempt. Part of this generated code is shown in Fig. 9. We described the robot's desired movement goals in the prompts, such as maintaining balance, walking forward, and avoiding falls, and LLMs were able to generate reward functions corresponding to these movement goals. The design of the reward function is a critical and challenging aspect of RL control methods, and LLMs fully demonstrated their design capabilities in this task.

Initially, we used the reward function generated by the LLMs to train the robot. However, the early training results were not

```
<Task Description>

1.Design a reward function for training a humanoid robot to walk forward and maintain balance using RL.

2.The robot's leg joints should follow inverse kinematics joint angle sequences.

3.End the episode if the robot shows a significant fall tendency.

<Robot Information and Direction Explanation>

1.Robot Information: The robot has 14 joints—12 in the legs and 2 in the waist (pitch and yaw).

2.Direction Information: In Unity, for the robot's body coordinate system, the y-axis is forward, the z-axis is height, and the x-axis is lateral.

<Environment Code and Related API>

1. Environment Code

void FixedUpdate() // Setting the reward function

{ ...

  EndEpisode(); // Function to end the episode

  ...

  AddReward(total_reward); // Function to add the total reward

}

2. API

1)All API indices [0] represent the x-direction, [1] the y-direction, and [2] the z-direction. For example, body.eulerAngles[0] represents the Euler angle in the x-direction.

2)"ab"is an array of ArticulationBody components. ab[0] (pelvis) is the root of the robot's articulated body. ab[1] to ab[14] represent other parts of the robot.

3)"velocity": API for a component's speed; "position": API for a component's position; "jointPosition": Represents joint rotation angles. In Unity, revolute joints rotate around the x-axis by default, so the index after jointPosition is [0].


<Example of Output Format>

var live_reward = 1f;
```

**Fig. 8.** Prompts for reward designer.

satisfactory. We observed that the hyperparameters preceding each reward designed by the LLMs were identical, which necessitated further adjustments. To fully leverage the capabilities of the LLMs in this task, we adopted an iterative feedback approach rather than direct manual adjustments. We observed various situations during training, described them in natural language, and provided this feedback to the LLMs.

For instance, the robot initially struggled to maintain balance, which led to tremendous negative rewards for failing to keep the body upright, resulting in significant negative rewards at the beginning of training. We communicated these issues to the LLMs, which responded by not only modifying the hyperparameters for the target reward function but also automatically adjusting other related hyperparameters. This iterative process of refinement allowed the LLMs to optimize the reward functions effectively, enhancing the overall training performance. This approach highlights the potential of LLMs to autonomously manage complex tasks and improve through iterative feedback, significantly reducing the manual effort typically required in designing sophisticated RL reward functions.

### *3.3.4. Comparative description*

For the three key aspects of the control task discussed in this paper, we conduct a detailed comparison between the results generated by LLMs and those designed by human experts. Firstly, to better simulate the trajectory of the foot during walking, human experts designed the foot lift trajectory as a composite cycloid, which closely replicates the natural, efficient movement seen in human gait. In contrast, the LLMs automatically generated a simpler sine curve. While the cycloid trajectory is more complex and accurate, the sine curve represents a computationally efficient approach, demonstrating the practicality of using LLMs for rapid prototyping. Both methods employed numerical techniques for solving inverse kinematics, essential for determining the joint angles needed to achieve specific foot positions and orientations. This reveals that LLMs can effectively handle sophisticated mathematical modeling, producing results comparable to those crafted by human expertise.

In the design of reward functions for RL, LLMs generated reward types nearly identical to those devised by human experts, showcasing their ability to understand and replicate complex decision-making processes. However, there were differences in the specific forms and weights of the reward functions. Human experts might tailor these parameters based on nuanced understanding and experience to optimize performance metrics, while LLMs, although accurate, might generate more generic reward structures that require further fine-tuning.

This comparison underscores the potential of LLMs to autonomously manage critical aspects of humanoid robot control autonomously, offering a balance between efficiency and accuracy. This suggests a significant reduction in the workload for engineers without compromising the quality of the results.

The detailed comparisons are summarized in Table 1 and Table 2. Here, $H$ represents the step height, $T$ is the walking cycle, $T_m$ indicates the time spent in the support phase. $v_{forward}$ represents the API for forward speed, $v^*$ is a specific speed value.

```
void FixedUpdate()
{
    float total_reward = 0f;
    // Survival reward
    total_reward += 1f;
    // Forward reward
    float forward_velocity = ab[0].velocity.y;
    // Upright reward
    total_reward += CalculateUprightReward();
    // Angle tracking reward
    total_reward += CalculateJointTrackingReward();
    // Check for fall
    if ((pelvis.position-initial_position).z < 0.3f)
    AddReward(total_reward);
    }
```

**Fig. 9.** Excerpt of the reward code generated by LLMs.

**Table 1**
Foot lift trajectory and reward design by human experts.

| Foot lift trajectory | Reward design | | | | |
|---|---|---|---|---|---|
| | Survival reward | Forward reward | Upright reward | Angle tracking reward | Preventing falls |
| Hypocycloid: $z_H(t) = \frac{H}{2\pi}(1 - \cos(2\pi \frac{t}{T_m}))$ | 1 | $e^{-\lvert v_{forward} - v^* \rvert}$ | $-0.01(\lvert\theta_{pitch}\rvert + \lvert\theta_{roll}\rvert + \lvert\theta_{yaw}\rvert)$ | $1.2 - 0.1 \cdot \sum_{i=1}^{14}(q_i - q_i^{ref})$ | If falling, end episode |

**Table 2**
Foot lift trajectory and reward design by LLMs.

| Foot lift trajectory | Reward design | | | | |
|---|---|---|---|---|---|
| | Survival reward | Forward reward | Upright reward | Angle tracking reward | Preventing falls |
| Sine curve: $z_L(t) = H \sin(\frac{2\pi}{T}(t - t_0))$ | 1 | $1 \cdot v_{forward}$ | $1 - (\lvert\frac{\theta_{pitch}}{180^\circ}\rvert + \lvert\frac{\theta_{roll}}{180^\circ}\rvert + \lvert\frac{\theta_{yaw}}{180^\circ}\rvert) \times \frac{1}{3}$ | $1.5 - \sum_{i=1}^{14}(q_i - q_i^{ref})$ | If falling, $-1$, end episode |

$\theta_{pitch}$, $\theta_{roll}$, and $\theta_{yaw}$ are the body Euler angles, $q$ represents the current joint angle, and $q_i$ is the reference angle.

### 3.4. RL problem formulation

We describe the RL control method as a Markov decision process (MDP). A Markov decision process is typically described using a quintuple $\langle S, A, P, R, \gamma \rangle$. Here, $S$ represents the state space, $A$ represents the action space, $P$ denotes the state transition matrix of the environment, where $P(s'|s, a)$ indicates the probability of transitioning to the next state $s'$ when taking action $a$ in state $s$. $R$ represents the reward function, and $\gamma$ is the discount factor. The goal of a Markov Decision Process is to optimize a policy $\pi(a|s)$ to maximize the cumulative reward obtained throughout the learning process. The cumulative reward can be formally expressed as: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

#### 3.4.1. Hyperparameter and neural network

The RL algorithm used in this paper is the proximal policy optimization (PPO) algorithm, which is suitable for high-dimensional continuous action spaces and offers high stability [75]. Detailed parameters of the PPO algorithm are shown in the Table 3.

**Table 3**
Parameters of the PPO algorithm.

| Parameter | | Value |
|---|---|---|
| Hyperparameter | Batch_size | 2048 |
| | Buffer_size | 20480 |
| | Learning_rate | 0.0003 |
| | Beta | 0.005 |
| | Epsilon | 0.2 |
| | Lambd | 0.95 |
| | Num_epoch | 3 |
| | Learning_rate_schedule | Linear |
| Neural network | Normalize | True |
| | Hidden_units | 512 |
| | Num_layers | 3 |
| | Vis_encode_type | simple |
| | Goal_conditioning_type | Hyper |

#### 3.4.2. Observation

The observations during the RL process in this paper are designed as follows:

$$O_t = [J_P \; J_V \; \theta \; \omega \; v] \tag{6}$$

The meaning and dimension of each variable in the observation space are shown in Table 4. Obviously, the size of the observation space is 37.

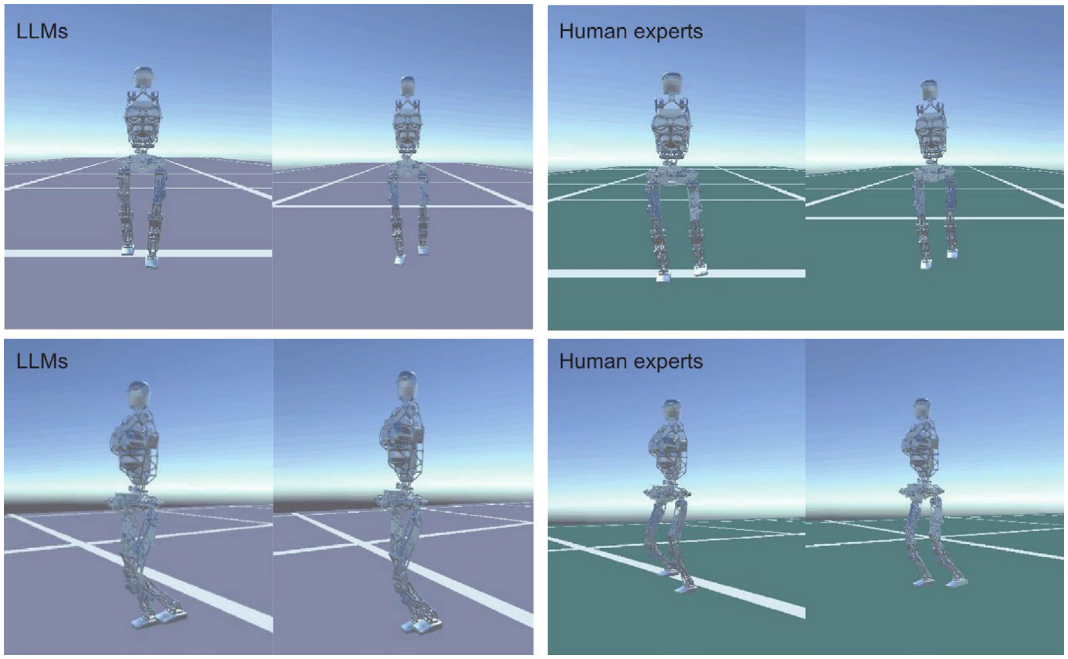



**Fig. 10.** Comparison of the performance simulation results designed by LLMs and human experts.

**Table 4**
The meaning and dimension of each variable in the observation space.

| Observation | Meaning | Dimension |
|---|---|---|
| $J_P$ | The vector of joint angles around the axes | 14 |
| $J_V$ | The vector of joint angular velocities | 14 |
| $\theta$ | The Euler angles of the robot's body | 3 |
| $\omega$ | The angular velocity of the robot's body | 3 |
| $v$ | The velocity of the robot's body | 3 |

#### 3.4.3. Action

The action space consists of changes in the angles of the 14 joints controlling the entire robot. Notably, the control signals used in this paper are not solely the output of the RL policy network. Instead, the robot's control is achieved by combining the weighted signals from RL with the reference gait signals derived from the inverse kinematics algorithm. The reference gait signals provide a basic movement pattern, while RL enables random exploration in the action space, fine-tuning the reference signals and maintaining body balance. The control framework shown in Fig. 2 illustrates the composition of the control signals as follows:

$$a = k \cdot a_{RL} + a_{\mathrm{Ref}} \tag{7}$$

where $a_{RL}$ is the control signal output by the RL policy network, and $a_{\mathrm{Ref}}$ is the reference joint angle sequence obtained through inverse kinematics. $k$ is the weighting coefficient, and the value of $k$ affects the training performance. If $k$ is too small, RL's exploration space will be limited. If $k$ is too large, it will reduce the convergence speed of learning, and the reference gait will not effectively contribute to the robot's control. Therefore, after continuous adjustments based on the training results, we set $k = 0.5$.

## 4. Results and discussions

We conducted the simulation in Unity ML-Agents. The time step for each action was set to 0.01 s. All training was performed using 16 agents copies in parallel to accelerate the training process. We fixed the robot's upper body posture in the simulation.

To validate whether the LLMs approach achieves human-level performance, we compared our proposed method with the human expert's approach across three aspects: learning effectiveness, smoothness of locomotion, and trajectory tracking error.

### 4.1. Learning efficiency

Fig. 10 illustrates the walking effect observed after the training phase. The simulation results demonstrate that the LLM-based method successfully enables the robot to walk forward, maintaining body balance without falling, comparable to the performance of the human-designed method. This indicates that the LLMs can effectively replicate the complex locomotion dynamics required for stable walking in humanoid robots.

Fig. 11 presents the reward curve after training. Although the LLMs achieve higher final reward values than those achieved by human experts, these values are not directly comparable due to the different weights assigned in the reward functions. However, the key insight from this comparison lies in the convergence behavior of the reward curves. The LLMs' reward curve converges more rapidly, suggesting that the LLMs are capable of optimizing behavior more quickly. This rapid convergence points to the efficiency of LLMs in learning and adapting to the control tasks. Nonetheless, the reward curve for the LLMs exhibits greater fluctuation, indicating a trade-off between speed of optimization and stability. The increased variability suggests that while LLMs can quickly reach high-performance levels, they might require additional tuning or stabilization mechanisms to achieve the same level of consistent performance as human-designed systems. This highlights an area for further refinement in the application of LLMs for humanoid robot control.

### 4.2. Smoothness of locomotion

To evaluate the stability of the motion, we compared the forward speed and joint angle fluctuations during the motion process using both the LLM-based and human-designed methods. As shown in Fig. 12, the speeds of both methods fluctuate around 0.25 m/s, successfully maintaining a constant forward walking

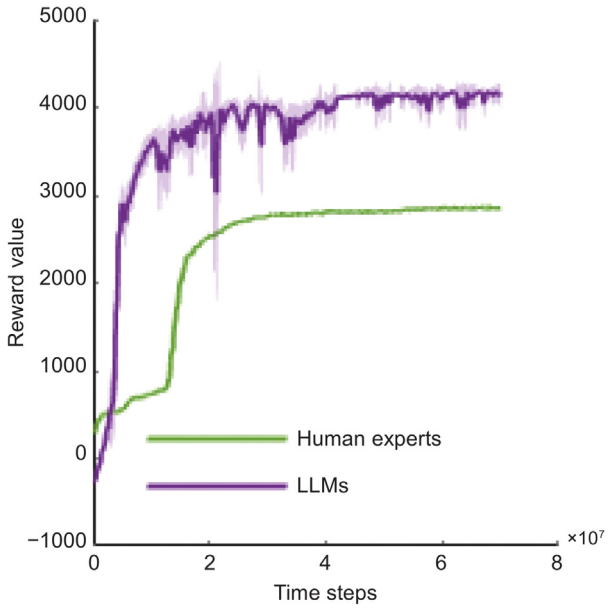



**Fig. 11.** Comparison of the reward curve between LLMs and human experts.

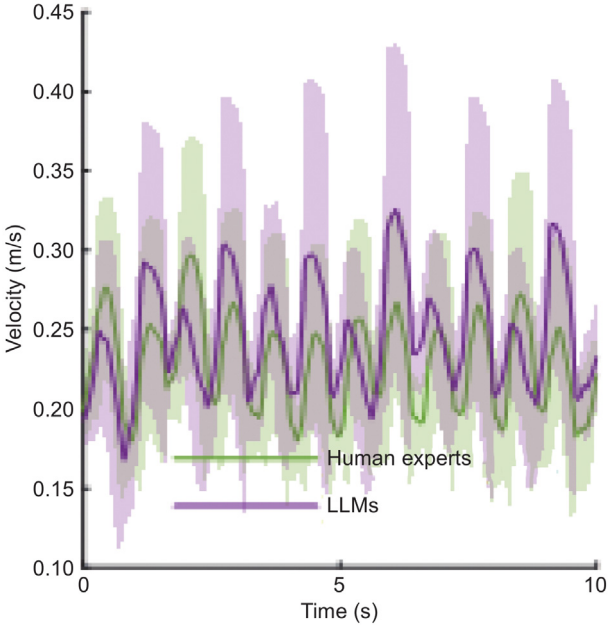



**Fig. 12.** Comparison of the humanoid speed curve between the LLMs and human experts.

speed. This consistency in speed indicates that both methods are effective in achieving stable locomotion.

Fig. 13 illustrates the joint angle variations generated by the human expert, while Fig. 14 depicts the joint angle variations produced by the LLMs. We observed that all angle fluctuations in both methods remain within the normal range of joint angles, with no significant sudden changes. This observation suggests that the LLMs can generate smooth and stable motions comparable to human-level performance. Notably, the LLM-based approach results in smoother actual angle change curves for the robot, which is crucial for ensuring fluid and stable motion when transitioning from simulation to real-world applications. The smoothness of these curves is indicative of the LLMs' ability to refine and optimize motion control strategies, reducing the risk of abrupt movements that could destabilize the robot. This enhanced smoothness not only improves the overall performance but also facilitates a more seamless transition from virtual simulations to practical, real-world robot operations. The findings underscore the potential of LLMs to effectively manage and fine-tune complex robotic control tasks, paving the way for more advanced and reliable humanoid robot systems.

### *4.3. Trajectory tracking error*

Finally, we wanted to evaluate whether the motion generated by the LLMs effectively tracks the pre-planned trajectory. Since the planned trajectory is converted into joint angles in the robot's body space by the IK algorithm, we measure the trajectory tracking error by evaluating the error between the actual joint angles and the reference angle sequences. Figs. 15 and 16 show the trajectory tracking errors. These two methods have error values around 0–5 degrees, which is an acceptable range for normal motion.

## 5. Conclusion

In this work, we propose a method where LLMs act as the primary designers to accomplish locomotion control tasks for a humanoid robot. Our control method is designed to use the reference angle sequences obtained from inverse kinematics as reference signals, combined with weighted RL signals to form the final control signals. LLMs complete three key tasks: trajectory planning, inverse kinematics solving, and reward function design. For each task, we designed prompts with different components to input into LLMs, which then generated the corresponding code and optimized it based on feedback. We validated the locomotion generated by LLMs in Unity and used meticulously designed human expert locomotion as a comparison. The results demonstrated that LLMs could successfully execute a forward walking task for a humanoid robot, achieving human-level performance. Furthermore, they highlight the potential of LLMs to significantly reduce the workload on engineers and streamline the development process for humanoid robot control systems.

We only validated the forward walking task in this work. In the future, we plan to have LLMs complete a variety of walking tasks, such as turning and stair climbing, and apply them to real robots.

## CRediT authorship contribution statement

**Shilong Sun:** Writing – review & editing, Supervision, Funding acquisition, Formal analysis, Conceptualization. **Chiyao Li:** Writing – original draft, Methodology. **Zida Zhao:** Validation, Investigation, Formal analysis, Data curation. **Haodong Huang:** Validation, Software, Formal analysis, Conceptualization. **Wenfu Xu:** Writing – review & editing, Supervision, Funding acquisition.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work was supported in part by the Guangdong Basic and Applied Basic Research Foundation, China (2024A1515012041), and in part by the Shenzhen Higher Education Stability Support Plan, China (GXWD20231130195340002), and in part by the Program of Shenzhen Peacock Innovation Team, Guangdong, China (KQTD20210811090146075), and the Basic Research Program of Shenzhen, China (JCYJ20220818102415034).

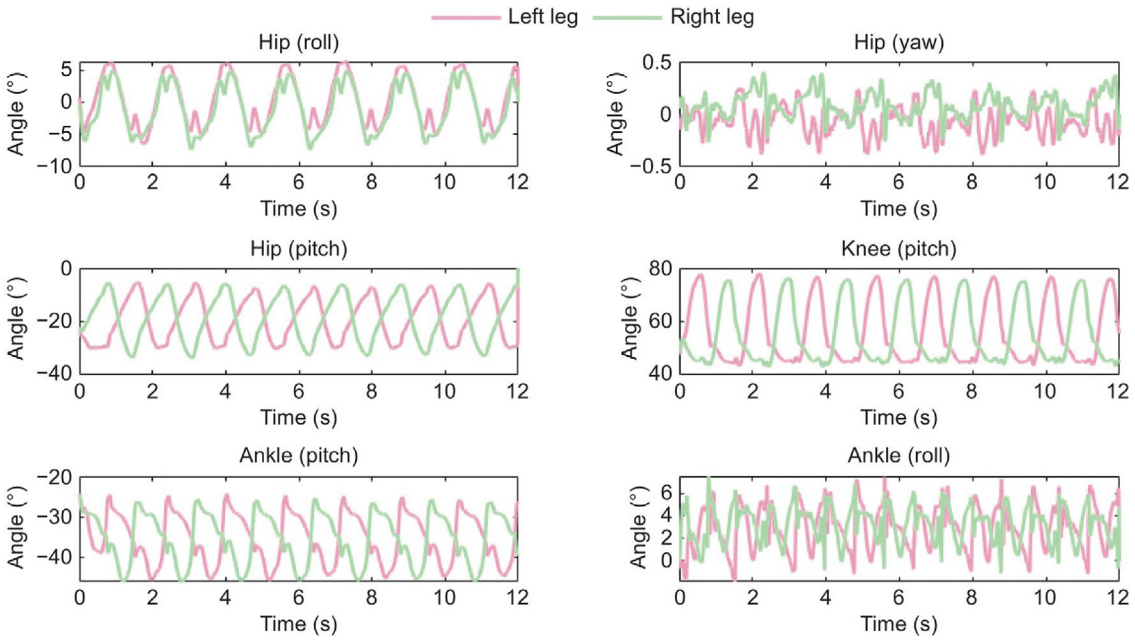



**Fig. 13.** Joint angle variations of the left and right leg using the human expert's method.

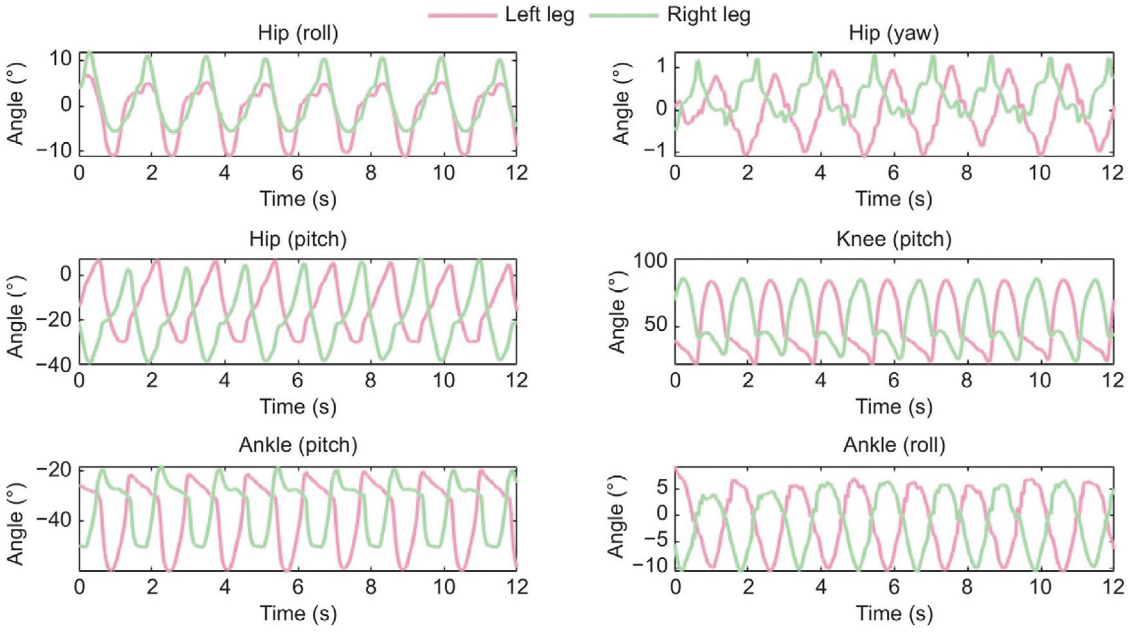



**Fig. 14.** Joint angle variations of the left and right leg using the LLM-based method.

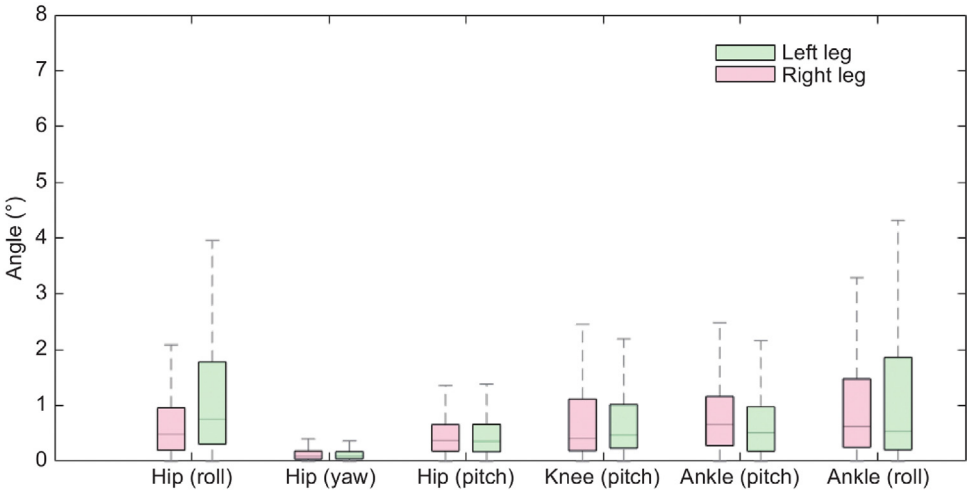



**Fig. 15.** Tracking errors of leg joint angles using the human expert's method.

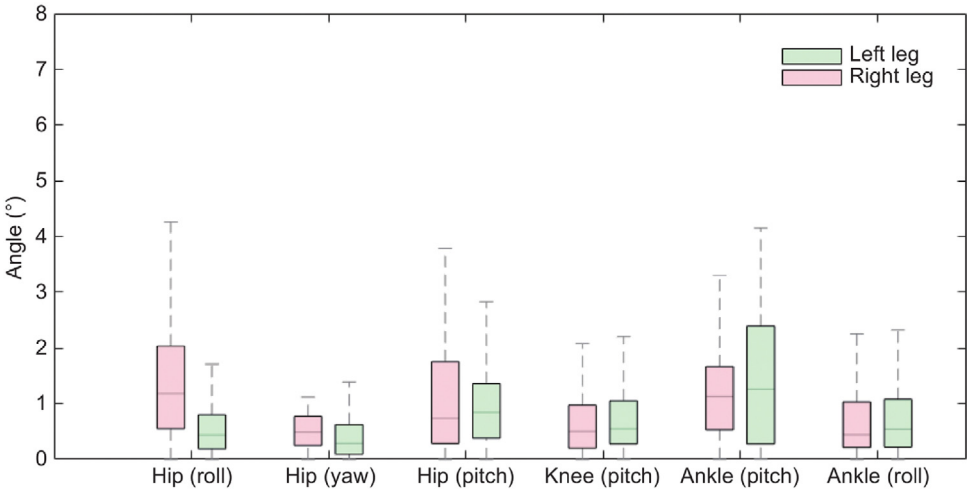



**Fig. 16.** Tracking errors of leg joint angles using the LLM-based method.

## Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.birob.2024.100187.

## References

[1] A. Joseph, B. Christian, A.A. Abiodun, F. Oyawale, A review on humanoid robotics in healthcare, MATEC Web Conf. 153 (2018) 02004.
[2] K. Darvish, et al., Teleoperation of humanoid robots: A survey, IEEE Trans. Robot. 39 (3) (2023) 1706–1727.
[3] M. Chignoli, D. Kim, E. Stanger-Jones, S. Kim, The MIT humanoid robot: Design, motion planning, and control for acrobatic behaviors, in: 2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids), IEEE, 2021, pp. 1–8.
[4] M. Konishi, K. Kojima, K. Okada, M. Inaba, K. Kawasaki, ZMP feedback balance control of humanoid in response to ground acceleration, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023, pp. 8525–8531.
[5] K. Ishihara, T.D. Itoh, J. Morimoto, Full-body optimal control toward versatile and agile behaviors in a humanoid robot, IEEE Robot. Autom. Lett. 5 (1) (2020) 119–126.
[6] E. Dantec, et al., Whole-body model predictive control for biped locomotion on a torque-controlled humanoid robot, in: IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids), 2022, pp. 638–644.
[7] C. Piazza, C.D. Santina, G. Grioli, A. Bicchi, M.G. Catalano, Analytical model and experimental testing of the SoftFoot: An adaptive robot foot for walking over obstacles and irregular terrains, IEEE Trans. Robot. 40 (2024) 3290–3305.
[8] S. Hanasaki, Y. Tazaki, H. Nagano, Y. Yokokohji, Running trajectory generation including gait transition between walking based on the time-varying linear inverted pendulum mode, in: IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids), 2022, pp. 851–857.
[9] T. Ishigaki, K. Yamamoto, Dynamics computation of a hybrid multi-link humanoid robot integrating rigid and soft bodies, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 2816–2821.
[10] J. Siekmann, Y. Godse, A. Fern, J. Hurst, Sim-to-real learning of all common bipedal gaits via periodic reward composition, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2021, pp. 7309–7315.
[11] S.H. Jeon, S. Heim, C. Khazoom, S. Kim, Benchmarking potential based rewards for learning humanoid locomotion, in: Presented At the 2023 IEEE International Conference on Robotics and Automation (ICRA), 2023.

[12] M. Seo, et al., Deep imitation learning for humanoid loco-manipulation through human teleoperation, in: 2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids), IEEE, 2023, pp. 1–8.
[13] M. Kobayashi, J. Yamada, M. Hamaya, K. Tanaka, Lfdt: Learning dual-arm manipulation from demonstration translated from a human and robotic arm, in: IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids), 2023, pp. 1–8.
[14] R. Firoozi, et al., Foundation models in robotics: Applications, challenges, and the future, 2023, arXiv preprint arXiv:2312.07843.
[15] M. Ahn, et al., Do as i can, not as i say: Grounding language in robotic affordances, 2022, arXiv preprint arXiv:2204.01691.
[16] S. Vemprala, R. Bonatti, A. Bucker, A. Kapoor, Chatgpt for robotics: Design principles and model abilities, Microsoft Auton. Syst. Robot. Res. 2 (2023) 20.
[17] J. Liang, et al., Code as policies: Language model programs for embodied control, in: IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2023, pp. 9493–9500.
[18] Y. Chen, J. Arkin, Y. Zhang, N. Roy, C. Fan, Autotamp: Autoregressive task and motion planning with llms as translators and checkers, 2023, arXiv preprint arXiv:2306.06531.
[19] Y. Tang, W. Yu, J. Tan, H. Zen, A. Faust, T. Harada, Saytap: Language to quadrupedal locomotion, 2023, arXiv preprint arXiv:2306.07580.
[20] M. Kwon, S.M. Xie, K. Bullard, D. Sadigh, Reward design with language models, 2023, arXiv preprint arXiv:2303.00001.
[21] Y.J. Ma, et al., Eureka: Human-level reward design via coding large language models, 2023, arXiv preprint arXiv:2310.12931.
[22] W. Yu, et al., Language to rewards for robotic skill synthesis, 2023, arXiv preprint arXiv:2306.08647.
[23] M.G. Arenas, et al., How to prompt your robot: A promptbook for manipulation skills with code as policies, in: Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023, 2023.
[24] J. Shen, Y. Liu, X. Zhang, D. Hong, Optimized jumping of an articulated robotic leg, in: 2020 17th International Conference on Ubiquitous Robots (UR), 2020, pp. 205–212.
[25] M. Saveriano, F.J. Abu-Dakka, A. Kramberger, L. Peternel, Dynamic movement primitives in robotics: A tutorial survey, Int. J. Robot. Res. 42 (13) (2023) 1133–1184.
[26] M. Jin, et al., Low-centroid crawling motion for humanoid robot based on whole-body dynamics and trajectory optimization, in: 2022 7th International Conference on Robotics and Automation Engineering (ICRAE), 2022, pp. 199–205.
[27] S. Schaal, Dynamic movement primitives-a framework for motor control in humans and humanoid robotics, in: Adaptive Motion of Animals and Machines, Springer, 2006, pp. 261–280.
[28] S.H. Jeon, S. Heim, C. Khazoom, S. Kim, Benchmarking potential based rewards for learning humanoid locomotion, in: 2023 IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 9204–9210.
[29] F.A. Koolen, Balance Control and Locomotion Planning for Humanoid Robots using Nonlinear Centroidal Models, Massachusetts Institute of Technology, 2020.
[30] G. Colin, J. Byrnes, Y. Sim, P.M. Wensing, J. Ramos, Whole-body dynamic telelocomotion: A step-to-step dynamics approach to human walking reference generation, in: 2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids), 2023, pp. 1–8.
[31] C. Mastalli, et al., Crocoddyl: An efficient and versatile framework for multi-contact optimal control, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 2536–2542.
[32] M. Shridhar, L. Manuelli, D. Fox, Cliport: What and where pathways for robotic manipulation, in: Presented At the Proceedings of the 5th Conference on Robot Learning, Proceedings of Machine Learning Research, 2022.
[33] Z. Li, X.B. Peng, P. Abbeel, S. Levine, G. Berseth, K. Sreenath, Robust and versatile bipedal jumping control through multi-task reinforcement learning, 2023, arXiv preprint arXiv:2302.09450.
[34] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, K. Sreenath, Real-world humanoid locomotion with reinforcement learning, Science Robotics 9 (89) (2024) eadi9579.
[35] W. Zhao, J.P. Queralta, T. Westerlund, Sim-to-real transfer in deep reinforcement learning for robotics: a survey, in: 2020 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2020, pp. 737–744.
[36] J. Peters, S. Vijayakumar, S. Schaal, Reinforcement learning for humanoid robotics, in: Proceedings of the Third IEEE-RAS International Conference on Humanoid Robots, 2003, pp. 1–20.
[37] T.-Y. Yang, T. Zhang, L. Luu, S. Ha, J. Tan, W. Yu, Safe reinforcement learning for legged locomotion, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2022, pp. 2454–2461.
[38] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, M. van de Panne, Iterative reinforcement learning based design of dynamic locomotion skills for cassie, 2019, arXiv preprint arXiv:1903.09537.
[39] L. Krishna, G.A. Castillo, U.A. Mishra, A. Hereid, S. Kolathaya, Linear policies are sufficient to realize robust bipedal walking on challenging terrains, IEEE Robot. Autom. Lett. 7 (2) (2022) 2047–2054.
[40] J. Hwangbo, et al., Learning agile and dynamic motor skills for legged robots, Science Robotics 4 (26) (2019) eaau5872.
[41] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter, Learning quadrupedal locomotion over challenging terrain, Science Robotics 5 (47) (2020) eabc5986.
[42] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter, Learning robust perceptive locomotion for quadrupedal robots in the wild, Science Robotics 7 (62) (2022) eabk2822.
[43] D. Hoeller, N. Rudin, D. Sako, M. Hutter, Anymal parkour: Learning agile navigation for quadrupedal robots, 2023, arXiv preprint arXiv:2306.14874.
[44] J. Tan, et al., Sim-to-real: Learning agile locomotion for quadruped robots, 2018, arXiv preprint arXiv:1804.10332.
[45] D. Jain, A. Iscen, K. Caluwaerts, Hierarchical reinforcement learning for quadruped locomotion, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2019, pp. 7551–7557.
[46] C. Yang, K. Yuan, Q. Zhu, W. Yu, Z. Li, Multi-expert learning of adaptive legged locomotion, Science Robotics 5 (49) (2020) eabb2174.
[47] C.E. Mower, et al., ROS-llm: A ROS framework for embodied AI with task feedback and structured reasoning, 2024, arXiv preprint arXiv:2406.19741.
[48] F. Stella, C. Della Santina, J. Hughes, How can LLMs transform the robotic design process? Nat. Mach. Intell. (2023).
[49] D. Honerkamp, M. Buchner, F. Despinoy, T. Welschehold, A. Valada, Language-grounded dynamic scene graphs for interactive object search with mobile manipulation, 2024, arXiv preprint arXiv:2403.08605.
[50] I. Singh, et al., Progprompt: Generating situated robot task plans using large language models, in: 2023 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2023, pp. 11523–11530.
[51] Y. Kant, et al., Housekeep: Tidying virtual households using commonsense reasoning, in: European Conference on Computer Vision, Springer, 2022, pp. 355–373.
[52] J. Zhang, et al., FLTRNN: Faithful long-horizon task planning for robotics with large language models, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 6680–6686.
[53] M. Han, Y. Zhu, S.-C. Zhu, Y.N. Wu, Y. Zhu, InterPreT: Interactive predicate learning from language feedback for generalizable task planning, in: Robotics: Science and Systems, 2024.
[54] G. Chen, et al., Language-augmented symbolic planner for open-world task planning, in: Robotics: Science and Systems, 2024.
[55] W. Xia, et al., Kinematic-aware prompting for generalizable articulated object manipulation with llms, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 2073–2080.
[56] K. Shirai, et al., Vision-language interpreter for robot task planning, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 2051–2058.
[57] Z. Zhou, J. Song, K. Yao, Z. Shu, L. Ma, Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 2081–2088.
[58] X. Liu, P. Li, W. Yang, D. Guo, H. Liu, Leveraging large language model for heterogeneous ad hoc teamwork collaboration, in: Robotics: Science and Systems, 2024.
[59] S. Wang, et al., LLM^3: Large language model-based task and motion planning with motion failure reasoning, 2024, arXiv preprint arXiv:2403.11552.
[60] Z. Zhao, W.S. Lee, D. Hsu, Large language models as commonsense knowledge for large-scale task planning, Adv. Neural Inf. Process. Syst. 36 (2024).
[61] S. Tellex, N. Gopalan, H. Kress-Gazit, C. Matuszek, Robots that use language, Annu. Rev. Control. Robot. Auton. Syst. 3 (1) (2020) 25–55.
[62] H. Kress-Gazit, G.E. Fainekos, G.J. Pappas, Translating structured english to robot controllers, Adv. Robot. 22 (12) (2008) 1343–1359.
[63] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, H. Ben Amor, Language-conditioned imitation learning for robot manipulation tasks, Adv. Neural Inf. Process. Syst. 33 (2020) 13139–13150.
[64] J.Y. Chai, Q. Gao, L. She, S. Yang, S. Saba-Sadiya, G. Xu, Language to action: Towards interactive task learning with physical agents, in: IJCAI, vol. 7, 2018, pp. 2–9.
[65] T.M. Howard, S. Tellex, N. Roy, A natural language planner interface for mobile manipulators, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 6652–6659.
[66] J. Austin, et al., Program synthesis with large language models, 2021, arXiv preprint arXiv:2108.07732.
[67] Y. Li, et al., Competition-level code generation with alphacode, Science 378 (6624) (2022) 1092–1097.

[68] F. Alet, et al., A large-scale benchmark for few-shot program induction and synthesis, in: International Conference on Machine Learning, PMLR, 2021, pp. 175–186.
[69] A. Bucker, et al., Latte: Language trajectory transformer, in: 2023 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2023, pp. 7287–7294.
[70] K. Lin, C. Agia, T. Migimatsu, M. Pavone, J. Bohg, Text2motion: From natural language instructions to feasible plans, Auton. Robots 47 (8) (2023) 1345–1365.
[71] X.B. Peng, Z. Ma, P. Abbeel, S. Levine, A. Kanazawa, Amp: Adversarial motion priors for stylized physics-based character control, ACM Trans. Graph. (ToG) 40 (4) (2021) 1–20.
[72] L. Ye, J. Li, Y. Cheng, X. Wang, B. Liang, Y. Peng, From knowing to doing: Learning diverse motor skills through instruction learning, 2023, arXiv preprint arXiv:2309.09167.
[73] P.A. Bhounsule, D. Torres, E.H. Hinojosa, A. Alaeddini, Task-level control and poincaré map-based sim-to-real transfer for effective command following of quadrupedal trot gait, in: 2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids), 2023, pp. 1–8.
[74] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[75] B. Zhao, H. Dong, Y. Wang, T. Pan, PPO-TA: Adaptive task allocation via proximal policy optimization for spatio-temporal crowdsourcing, Knowl.-Based Syst. 264 (2023) 110330.