Agile Humanoid Locomotion by Learning from Human Demonstrations

Kangning Yin[1], Yingguang Xing[2], Yao Mu[3], Jianyu Chen[4], Zheng Tian[1]

***Abstract*— Recent advances in humanoid robotics have demonstrated the potential of reinforcement learning for locomotion control. However, existing approaches often oversimplify upper body control during policy learning, resulting in unnatural movements and poor robustness against upper-body disturbances. In this work, we present a framework that enables human-like agile motions, including walking, running, and jumping, while maintaining full-body coordination under external perturbations. We develop a motion optimization pipeline that refines motion capture data by explicitly considering the robot's joint limits and torque constraints, ensuring the generation of physically feasible reference trajectories. Additionally, we propose an Autoencoder-based latent representation scheme that encodes expert demonstrations into privileged observations for cumulative reward estimation. This approach improves policy alignment with expert distributions while maintaining computational efficiency during deployment. Experimental results demonstrate that our method enables a real humanoid robot to perform dynamic locomotion tasks with human-like whole-body coordination and enhanced robustness to upper-body disturbances.**

I. INTRODUCTION

Humanoid robots have gained increasing attention in recent years due to their human-like appearance, which enables them to perform a variety of tasks traditionally undertaken by humans. Locomotion is a critical and integral aspect of humanoid robot systems. However, the inherently unstable nature of humanoid robots, due to their higher center of mass, makes achieving effective locomotion much more challenging compared to other robotic systems, such as quadruped robots. Traditional humanoid locomotion often relies on model-based optimization algorithms, such as Model Prediction Control (MPC) [1] and Linear Quadratic Regulator (LQR) [2]. These algorithms typically construct a mathematical model to represent both the robot system and its environment, which has demonstrated considerable success in the past.

However, as humanoid robots are increasingly required to traverse more challenging terrains, traditional optimization algorithms have been shown to be inadequate in addressing these complexities. Reinforcement Learning (RL) has emerged as a promising alternative and has achieved significant success in the field of humanoid locomotion in recent years [3]–[5]. RL allows a humanoid robot to independently find the optimal behavior by interacting with its environment through a trial-and-error process. Instead of specifying the solution directly, the control task designer in RL offers feedback via a scalar objective function, which assesses the humanoid robot's performance at each step. This process makes our humanoid robot adapt to different environments easily while maintaining its stability. Numerous studies have focused on controlling the lower body of humanoid robots by reward designing, enabling them to walk, run, and jump across a variety of terrains robustly [3]–[5].

Focusing solely on controlling the lower body of humanoid robots by reward designing may limit their ability to reach their full potential. For instance, walking without arm movement can significantly affect a humanoid robot's ability to maintain balance when subjected to large external forces. Furthermore, relying solely on reward design to control humanoid robots may fall short in achieving complex, agile motions such as fast running or dancing. As a result, whole-body control for humanoid robots has become an increasingly prominent area of focus.

Various works have employed imitation learning to achieve whole-body control in humanoid robots. Zhang et.al [6] propose adversarial motion prior learning framework, which combines imitation learning with reinforcement learning on a full-size humanoid robot, enabling humanoid robot to walk and run with human-like movements. Exbody [7] further applies the AMP method to imitate upper-body movements in humanoid robots, enabling a range of motions such as dancing, hand waving. However, all of these approaches rely heavily on the quality of motion capture (MoCap) data. When the reference motion is difficult for a humanoid robot to replicate, it can hinder the policy learning process. Additionally, reinforcement learning policies driven solely by expert policy reward signals still face challenges such as slow convergence and instability. These issues mainly stem from the insufficient integration of guiding information embedded in expert trajectories during the learning process, making it difficult to accurately learn and evaluate the alignment with expert policies when estimating cumulative returns. For example, when learning agile motions such as running and jumping, understanding future dynamics and recalling past experiences can help the robot perform accurately, as the states change rapidly in these scenarios.

To address these challenges, we developed a multi-stage motion capture and policy learning pipeline. First, we utilize high-precision mocap equipment to record detailed motion sequences. We then employ a two-step retargeting process: applying Poselib in ASE [8] for initial motion adaptation, followed by Inverse Kinematics optimization in Pink [9] to ensure joint trajectories remain within the robot's physical constraints. Our comparative analysis shows that this combined approach achieves higher retargeting precision

[1] ShanghaiTech University, Shanghai, China
[2] RobotEra TECHNOLOGY CO.,LTD.
[3] The University of Hong Kong, Hong Kong, China
[4] Tsinghua University, Beijing, China

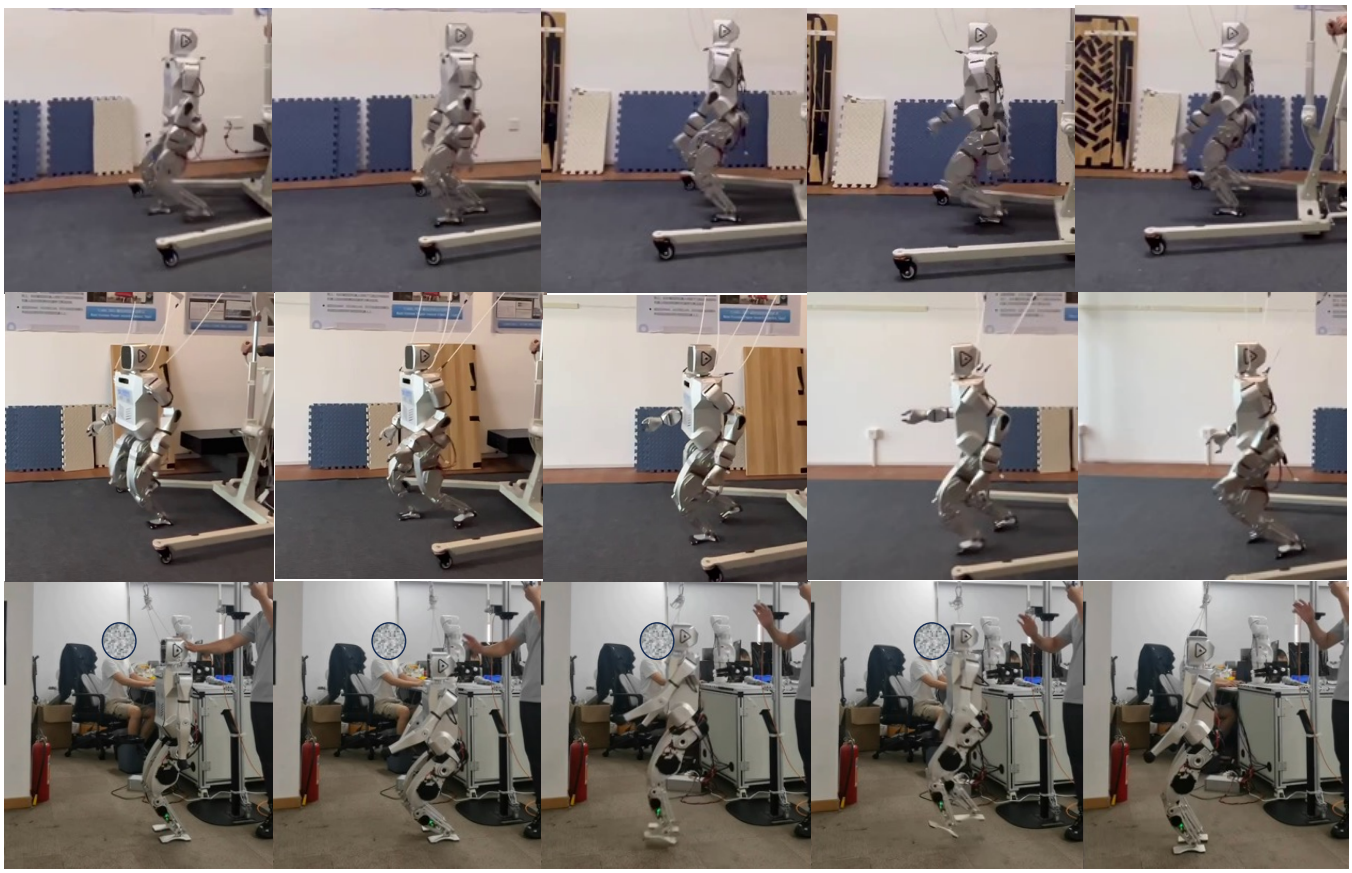

Fig. 1: Our framework can perform walk, run and jump in a human-like manner. The image above shows the side view of walk, run and jump respectively.

compared to direct application of Poselib to existing motion databases such as CMU Motion Capture dataset [10]. To effectively leverage these expert demonstrations, we implement an Autoencoder architecture [11] that compresses mocap sequences into a compact latent representation. The encoded expert motion data is then integrated into the critic network, allowing for more accurate cumulative reward estimation by considering both historical and future states during reinforcement learning. This facilitates more effective policy learning, enabling the robot to better replicate the demonstrated expert behaviors.

Our key contributions are summarized as follows: (i) We successfully deployed an agile policy based on the AMP framework for whole-body optimization on a real-world humanoid robot, enabling it to walk, run, and jump with human-like agility. The coordination between upper and lower limbs closely matches human behavior, while demonstrating robust performance against disturbances. (ii) We developed a mocap data retargeting technique that considers mechanical dynamic feasibility, significantly improving the quality of expert demonstrations and facilitating policy training. (iii) In our reinforcement learning approach, we utilize an Autoencoder to compress the mocap data and incorporate it into the privileged observation when estimating cumulative returns. This better helps align the robot's policy with expert policy distributions, significantly improving sample efficiency and training robustness.

II. Related Work

A. Legged Robot Locomotion

Legged robot locomotion is a popular topic in recent years. Among them the most studied areas are the quadruped robot locomotion, and the recently emerged humanoid robot locomotion. Quadruped robot, due the its high stability, is much more mature than humanoid robot locomotion. Lee et.al [12] is one of the most representing work, showing the capability of quadruped robots in maintaining stability in natural environments. TERT [13] is one of the first works trying to apply transformer to locomotion problem, showing the superiority of the high capacity for sequence modeling and the self-attention mechanism of Transformer. Some works integrate vision information into the robotics system, enabling robots to operate autonomously in remote and hazardous environments, even performing some agile motions around complex obstacles [14].

Many works train an end-to-end Reinforcement Learning(RL) controller for humanoid robots recently. One of the most representing works leverage periodic reward to train various gait patterns including walking, running, jumping and hopping [3]. Meanwhile, Duan et.al [4] enable robots to across complex environments such as climbing stairs and stepping across stones. Another type of work utilize the power of generative models such as transformer to generate trajectories, enabling the full-sized humanoid to walk in the

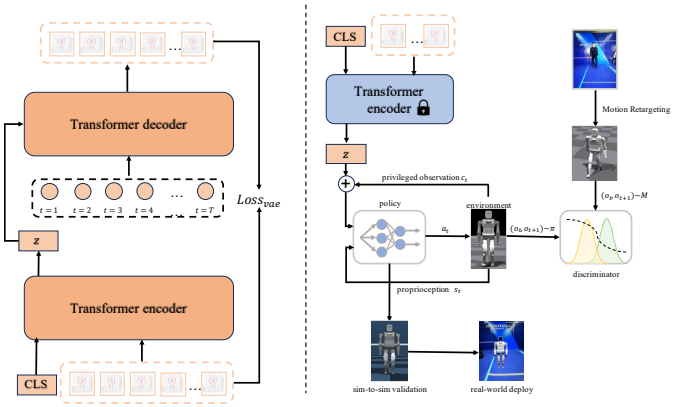

Fig. 2: Overall Framework: We begin by training a transformer-based Autoencoder to capture key features of the mocap data. During the RL training phase, the trained Autoencoder is used to compress both past and future mocap data into the privileged observation. Our RL framework is built on Adversarial Motion Priors (AMP).

real-world in zero-shot manner [15]. However, most of the work mentioned above focuses solely on controlling the lower body of humanoid robots. In contrast, our approach controls both the upper and lower body, aiming to train a more human-like and stable whole-body controller.

B. Imitation Learning for Humanoid Locomotion

Imitation learning is a crucial technique for humanoid locomotion, allowing humanoid robots to accurately replicate human motions that are challenging to achieve through traditional reward-based design alone. Existing frameworks for imitation learning for humanoid locomotion can be broadly categorized into two types. The first is to accurately mimic each frame of a human motion sequence. The representative work is deepmimic [16], which tracks the actuated joints, end effectors and base information of each frame. Li et.al [17] successfully deploy the framework on a real humanoid robot, allowing the humanoid to accomplish versatile motions like agile walking, running, jumping and even dancing. The second one is to imitate the style of human motion sequences using the generative adversarial imitation learning(GAIL) framework. The representative work is AMP [18], which incorporates a discriminator in an RL framework, enabling the RL policy to rollout sequences similar to the style of the human motion sequences. ASE [8] compress the skills into a latent space, allowing the framework to be more controllable. RACon [19] retrieves next framework target motion based on the current motion and control signal in a large dataset to serve as an guidance to the RL policy, enabling the policy to be more efficient and can switch between different motions. Zhang et.al [6] and ExBody [7] successfully transfer the AMP framework on a full-size humanoid robot, enabling the humanoid robot to perform various motions in a human manner. However, all of these works fail to incorporate future observations from the expert data during policy training.

III. PRELIMINARY

A. Humanoid Robot XBot-S

In this study, we employ the XBot-S robot model from RobotEra to conduct our experiments. The robot stands 1.2 meters tall, weighs 35 kilograms, and features 20 degrees of freedom, with 6 degrees allocated to each leg and 4 degrees to each arm.The XBot-S robot is engineered for exceptional dynamic performance, particularly in terms of torque and speed capabilities. Notably, its knee joint can generate a maximum motor torque of 250 N·m, coupled with a peak angular velocity of up to 12 rad/s. These specifications enable the robot to execute rapid, high-torque movements, essential for tasks requiring both power and precision.

Additionally, the robot's ankle is designed with two degrees of freedom—roll and pitch—which significantly enhance its flexibility. This multi-dimensional movement capability not only provides greater control potential but also allows for more human-like motion, closely mimicking the natural range of movement seen in human ankles. This advanced articulation offers improved balance and adaptability, particularly in dynamic and unstructured environments, making XBot-S an ideal platform for research and applications in humanoid robotics.

B. Reinforcement Learning

We utilize Reinforcement Learning (RL) to address our problem. Specifically, we model the problem as a discrete-time Partially Observable Markov Decision Process (POMDP), represented by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \mathcal{R}, \mathcal{O}, \gamma)$. Here, $\mathcal{S}$ denotes the state space, which encompasses all possible states of the environment. $\mathcal{A}$ represents the action space. T is the state transition function, which describes the transition from state s_t to state s_{t+1} under the action a_t, formally defined as $T(s_{t+1}|s_t, a_t)$. The reward function $R(s_t, a_t)$ assign a reward to the current state s_t when taking action a_t. $\mathcal{O}$ represents the partial observations of the humanoid robot, which are limited by the capabilities of the robot's sensors. Finally, $\gamma \in [0, 1]$ is the discount factor.

During training in the simulation, the robot receives partial observations o_t from its sensor at each timestep t. It then selects the action a_t according to the policy $\pi(a_t|o_t)$. After executing this action, the robot transitions to the next state s_{t+1} following the state transition function $T(s_{t+1}|s_t, a_t)$. Meanwhile, a reward is generated by the reward function $R(s_t, a_t)$, which is used to update the policy. The primary objective is to maximize the expected return:

$$J = \sum_{t}^{T} [\gamma_t R(s_t, a_t)] \tag{1}$$

IV. METHOD

Our primary objective is to integrate imitation learning into an RL policy to replicate human motions from mocap data. Using this RL policy, our humanoid robot is able to walk, run, and jump with human-like agility. In this section, we begin by introducing our improved motion retargeting process in Sec. IV-A. Then we introduce our imitation learning

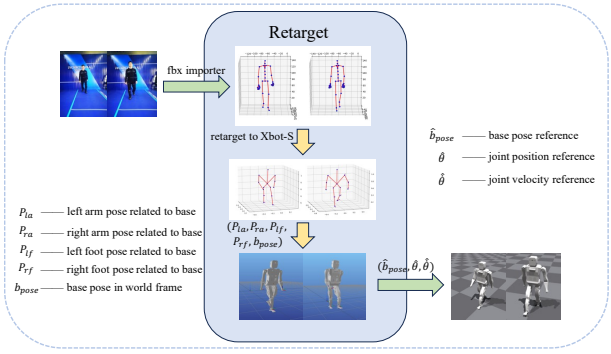

Fig. 3: Visualization of our Motion-Retargeting process

framework in Sec. IV-B. Following that, we describe how we use an Autoencoder [11] to compress mocap data into privileged observations in Sec. IV-C. Finally, we present our whole-body control using reinforcement learning in Sec. IV-D. Our overall framework is illustrated in Figure 2.

A. Humanoid Robot Reference Generation

In this work, alongside utilizing the CMU motion capture dataset [10], we conducted additional recordings using state-of-the-art motion capture equipment to capture the specific movements required for our study. The inclusion of these custom recordings significantly enriches our dataset, providing a more diverse range of motion samples. This, in turn, enhances the robustness and variability of the training process, allowing for more accurate and dynamic generation of movement styles.

Once a sufficient number of datasets have been collected, the next step is to map the motion capture data to the robot, ensuring that the robot's movements are both anthropomorphic and adapted to its mechanical constraints, such as motor position,velocity and joint limits. This process is outlined in Figure 3.

To be more specific, the raw motion capture data is first converted to fit the skeletal model used in our system through the method provided by ASE. The resulting transformation outputs the global base pose b_{pose} along with the pose of the four key end points relative to the base of the robot $(P_{la}, P_{ra}, P_{lf}, P_{rf})$.

From these outputs, the Python inverse kinematics(Pink) for articulated robot models, based on Pinocchio [20] is used to compute the corresponding joint angles, velocities, and base pose.

These calculated parameters are crucial for ensuring that the robot's motions not only replicate the captured human movements but also conform to the robot's physical properties, such as motor limitations and dynamics. Finally, these processed data points are transformed to the AMP(Adversarial Motion Prior) space, where they are utilized to generate fluid, anthropomorphic movements that closely resemble human motion while respecting the robot's physical constraints.

B. Imitation Learning from Motion Capture Data

In our imitation learning section, we build our framework on Adversarial Motion Priors (AMP) [18]. In our framework,

Algorithm 1 Retarget Process for Robot Motion

Require: Mocap data $D = \{v_1, v_2, \ldots, v_{\max}\}$ at different velocities

Ensure: Joint angles θ and joint velocities $\dot{\theta}$ for each velocity

1: **for** each $v \in D$ **do**

2: Record mocap data $P_{\text{mocap}} = \{p_1, p_2, \ldots, p_b\}$, where p_i represents the poses of end effectors and base in world coordinates

3: Use ASE network to map P_{mocap} to robot dimensions, yielding P_{robot}

4: Compute inverse kinematics for P_{robot}:

$$\{p_i\} \xrightarrow{\text{IK}} \{\theta, \dot{\theta}\}$$

5: Apply Gaussian filtering to $\{\theta, \dot{\theta}\}$:

$$\theta_{\text{filtered}} = \frac{1}{\sqrt{2\pi}\sigma} \sum_{k=-N}^{N} \theta_k e^{-\frac{k^2}{2\sigma^2}}$$

6: Store $\{\theta_{\text{filtered}}, \dot{\theta}_{\text{filtered}}\}$ as output for velocity v

7: **end for**

8: **return** Joint angles and velocities for all velocities $\{v_1, v_2, \ldots, v_{\max}\}$

we use a discriminator to distinguish whether the state originates from mocap data or the agent. Specifically, we first construct the AMP observation ${o_t}^D$ to better capture the style of the mocap data. The observation ${o_t}^D$ includes the local rotation and velocity of each joint, the linear, angular velocities, and the z-axis position of the base, as well as the positions of both hands and legs relative to the base. Observation transitions from the agent and those sampled from the mocap data are fed separately to the discriminator. We label the transitions from the mocap data as 'real' and those from the agent as 'fake.' The goal of the discriminator is to classify these inputs correctly: when mocap transitions are provided, the output should approach 1, whereas transitions from the agent should produce outputs closer to -1. As a result, the objective function can be formulated as follows:

$$\begin{aligned} \arg\min_D \; & \mathbb{E}_{d^{\mathcal{M}}({o_t}^D, o_{t+1}^D)} \left[\left(D({o_t}^D, o_{t+1}^D) - 1 \right)^2 \right] \\ & + \mathbb{E}_{d^{\pi}(s, s', {o_t}^D, o_{t+1}^D)} \left[\left(D({o_t}^D, o_{t+1}^D) + 1 \right)^2 \right]. \end{aligned} \tag{2}$$

Since discriminators often suffer from mode collapse, we follow AMP [18] by applying a gradient penalty to the mocap data transitions to ensure stability. This can be formulated as follows:

$$\arg\min_D \; \mathbb{E}_{d^{\mathcal{M}}({o_t}^D, o_{t+1}^D)} \left[\left\| \nabla D({o_t}^D, o_{t+1}^D) \right\|^2 \right]. \tag{3}$$

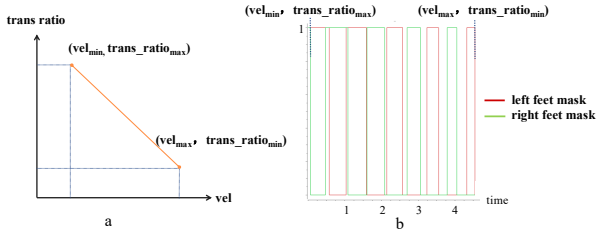

Fig. 4: Visualization of the gait function

The final objective function to train the discriminator is formulated as follows:

$$\begin{aligned}\arg\min_{D}\ & \mathbb{E}_{d^{\mathcal{M}}(o_t{}^D, o_{t+1}^D)}\left[\left(D(o_t{}^D, o_{t+1}^D)-1\right)^2\right] \\ & + \mathbb{E}_{d^{\pi}(s,s',o_t{}^D,o_{t+1}^D)}\left[\left(D(o_t{}^D, o_{t+1}^D)+1\right)^2\right] \\ & + w_{gp}\mathbb{E}_{d^{\mathcal{M}}(o_t{}^D,o_{t+1}^D)}\left[\left\|\nabla D(o_t{}^D, o_{t+1}^D)\right\|^2\right],\end{aligned} \tag{4}$$

where w_{gp} is the coefficient for the gradient penalty. Furthermore, during each epoch, we update the discriminator once and the RL policy five times. This approach enhances the RL policy's ability to generate states that more closely resemble human behavior. Following that in AMP [18] and Zhang et.al [6], the style reward at each timestep for the RL training is formulated as follows:

$$R_{\text{style}} = \max[0, 1 - 0.25\left(D(o_t^D, o_{t+1}^D) - 1\right)^2] \tag{5}$$

C. Compressing Mocap Data into Privileged Observations

Previous works employing the AMP framework on humanoid robots rely solely on historical data to learn the style of the mocap data, overlooking the importance of future states. However, enabling the policy to be aware of future expert states can improve the robot's posture and enhance robustness to a certain extent.

To address this, we first train a transformer [21]-based Autoencoder [11] to compress the mocap data into a latent space. The compressed mocap data is then incorporated into the privileged observation of the critic network for policy training. The motion sequence consists of a series of human motion frames $F = [F_1, F_2, ..., F_L]$, where L denotes the temporal length of the sequence and $F \in \mathbb{R}^{L\times C}$ with C representing the feature dimension of a single frame. A single motion frame here is structured the same way as those used in the discriminator. The motion sequence $F$is first mapped into a latent space via a linear projection, after which a [CLS] token z, representing the global feature of the entire sequence, is appended at the beginning of the sequence. The processed sequence $F_p = [z, F_1, F_2, ..., F_L]$ is then passed through a transformer-based encoder M_{enc}. The global feature z is subsequently fed into a transformer-based decoder M_{dec} to reconstruct the original sequence as F_{recon} . This process ensures that the global feature z effectively preserves the information of the entire sequence. The loss function for training the Autoencoder can be formulated as follows:

$$Loss_{AE} = MSE(F, F_{recon}). \tag{6}$$

During RL training, the encoder M_{enc} is kept fixed. To incorporate both historical and future observations, we select 10 past observations $[F_{t-10}, ..., F_{t-1}]$ and 10 future observations $[F_{t+1}, ..., F_{t+10}]$, based on the current observation F_t. The entire sequence $[F_{t-10}, ..., F_{t+10}]$ is then fed to the encoder M_{enc}, which outputs the global representation z_t. This global representation is added to the privileged observation of the critic network to enable the policy to be aware of both past and future states.

$$s_t = Concat(z_t, s_t) \tag{7}$$

D. Whole-Body Control through Reinforcement Learning

In the previous section, it was demonstrated that Adversarial Motion Priors (AMP) space facilitates the robot's ability to learn motion from mocap data. However, to achieve specific motion objectives, task-space guidance is essential. Our task space is structured into three key components:

- Robot Control Commands: This component encompasses the control commands necessary for managing the robot's movements and interactions within its environment.
- Gait Cycle Instructions: This part provides the directives for the entire gait cycle, ensuring the robot follows the desired locomotion pattern throughout its operation.
- Regularization Terms: The final component focuses on optimizing energy efficiency and applying smoothing techniques to the joint output pairs to ensure fluid and natural movement. Additionally, it ensures that the robot maintains a stable and natural pose throughout the motion.

For the robot control commands, we implement a reward function defined by the following formula:

$$R_{cmd} = \gamma_{xy}\, exp\left(\alpha_{xy}\left(\hat{v}_{xy} - v_{xy}\right)^2\right) + \gamma_{\omega}\, exp\left(\alpha_{\omega}\left(\hat{\omega} - \omega\right)^2\right) \tag{8}$$

γ and α is the weight of linear velocity and angular velocity part. The specific scale is in the reward appendix.

In the gait generation component, a gait generator is employed, which requires several parameters, including the phase indicator ϕ, transition ratio τ, and phase bias β. The gait generator outputs the stance mask for both feet. The phase indicator and transition ratio are constrained within the range of 0 to 1, while the swing ratio is implicitly defined as 1-τ

$$f_{\text{mask}} = \begin{cases} 1 & \text{if } (\text{clip}(\varphi + \beta, 0, 1)) < \tau \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Using the f_{mask}, we are able to effectively control the robot's gait by monitoring the number of foot contacts and the corresponding foot mask. As described in Table I, we formuate the gait reward as:

$$R_{gait} = (\mathrm{C}_c == \mathrm{C}_a), \tag{10}$$

where C_c is the expected number of contact feet and C_a is the actual number of contact.

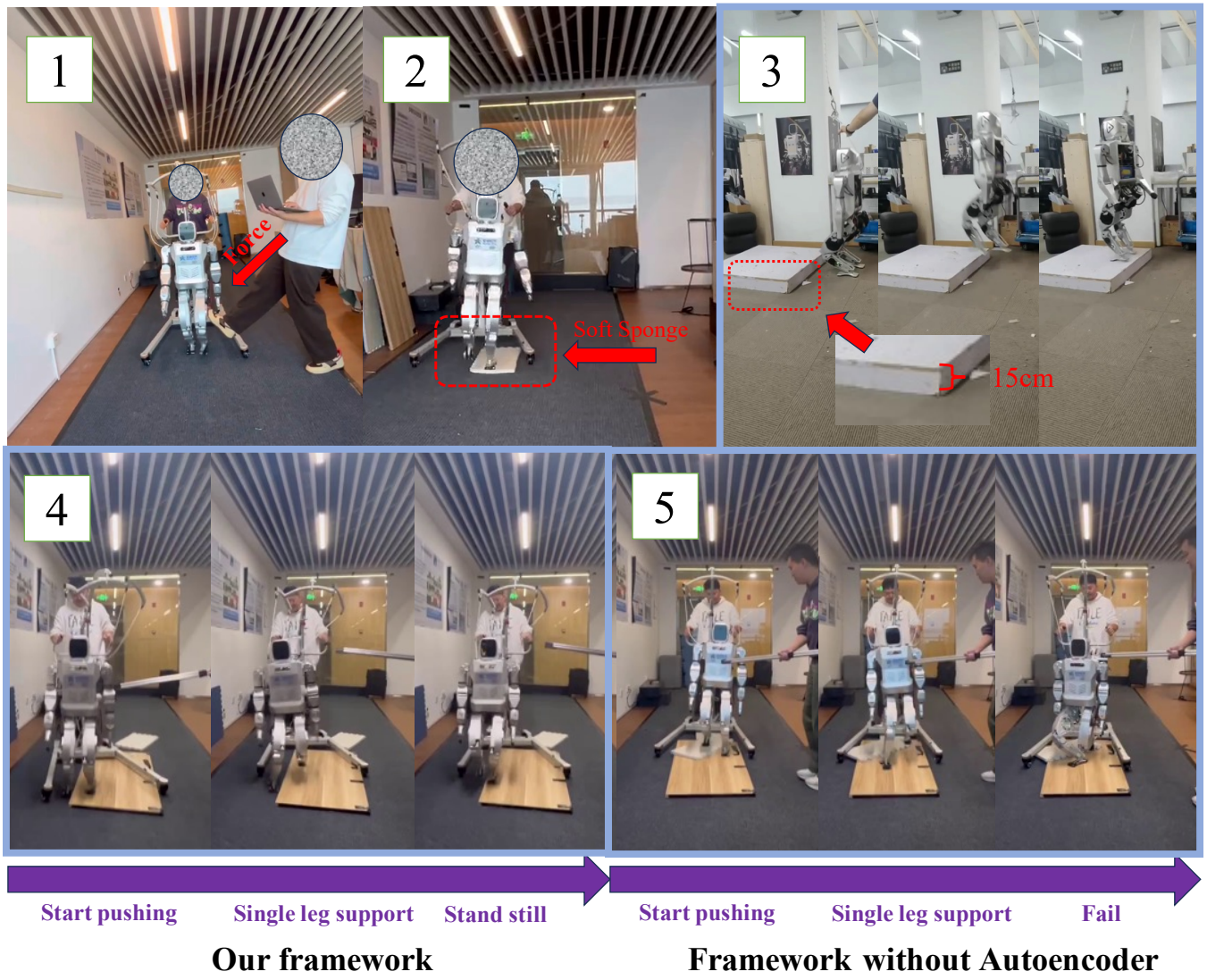

Fig. 5: Visualization of Our Real-World Experiments: The first picture shows our robot can maintain stability under external forces. The second picture shows that our robot can also keep balance while stepping on different surfaces. The third picture shows that our robot can jump onto 15cm stair. The last two pictures demonstrate that our framework is superior to that without Autoencoder.

Additionally, we have implemented adaptive adjustments to the gait as shown in Figure 4. Specifically, the transition ratio decreases as speed increases, which aligns with both intuition and the natural dynamics of human locomotion. Figure 4.a shows the linear relationship between the robot's velocity and the transition ratio in the gait of the humanoid robot XBot-S. At lower velocities, the transition ratio is higher, meaning that the swing phase is reduced, resulting in longer stance phases that enhance stability. In contrast, at higher velocities, the transition ratio decreases, which increases the swing phase duration. This adjustment enables the robot to cover more ground with each step at higher speeds, facilitating more efficient locomotion as velocity increases. And Figure 4.b depicts the foot contact patterns over time for both the left and right feet, represented by masks. The modulation of the transition ratio according to velocity influences the timing of the swing and stance phases. As the transition increases and the swing ratio decreases, the masks of the left and right feet will be 0 at the same time, which means that the robot has both feet in the air and has a running gait. This method can generate walking and running gaits according to the speed, which is a very simple and efficient method for gait control of humanoid robots.

For the regularization terms, as shown in Table I, we incorporate a feet distance term that ensures the feet remain within an acceptable range while moving. Additionally, we include an action smoothness term, which prevents the action from changing too abruptly during movement. The reward function is formulated as follows:

$$R_{reg} = w_{feet} R_{feet} + w_{action} R_{action}, \tag{11}$$

where w_{feet} and w_{action} is the scale of the feet distance reward and action smooth reward. The total reward to update

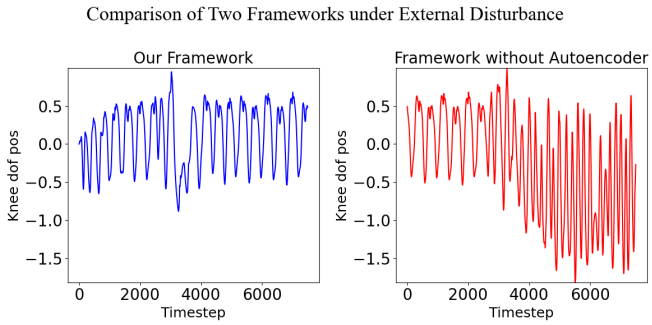

Fig. 6: Visualization of the knee dof position curve between our model and that without Autoencoder under external disturbance.

the policy is represented as follows:

$$R = R_{cmd} + R_{gait} + R_{reg}. \tag{12}$$

TABLE I: Robot Reward Functions

Reward	Equation	Scale
Tracking Linear Velocity	$exp\ (\alpha_{xy}\ (\hat{v}_{xy} - v_{xy})^2)$	1.2
Tracking Angular Velocity	$exp\ (\alpha_{\omega}\ (\hat{\omega} - \omega)^2)$	0.4
Feet Contact Number	$\mathrm{C}_c == \mathrm{C}_a$	0.8
Feet distance	$(\mathrm{Dis}_{min} < \mathrm{Dis}_f) \wedge (\mathrm{Dis}_f < \mathrm{Dis}_{max})$	0.15
Action Smooth	$\exp\left(\|a_t - 2a_{t-1} + a_{t-2}\|\right)$	0.01

V. EXPERIMENT

A. Implementation Details

All experiments are trained in the Isaac Gym simulation with 4,096 parallel environments using Proximal Policy Optimization (PPO) [22]. We present our PPO training parameters in Table II. Then the trained policy is transferred to Mujoco for sim-to-sim validation. The network architecture of the actor and critic is represented as a Multi-Layer Perceptron (MLP) with layers of sizes $[512, 256, 128]$ and $[768, 256, 128]$ respectively. The discriminator is also represented as a MLP with layers of sizes $[1024, 512]$. The motion encoder and decoder in the Autoencoder both consist of a 6-layer transformer encoder and decoder.

The action is represented as $a_t \in \mathbb{R}^{20}$, comprising the target position for each revolute joint. The target positions are then sent to a Proportional-Derivative (PD) controller, which converts them into torques for each joint. The observation is represented as $o_t \in \mathbb{R}^{71}$. The observation includes a periodic signal represented by the sine and cosine of the cycle time, along with a 3-dimensional command vector (v_x, v_y, v_{yaw}). This is followed by a 20-dimensional joint position vector, a 20-dimensional joint velocity vector, and a 20-dimensional action vector. The final 6-dimensional vector consists of the base angular velocity and base orientation. Domain randomization is the key factor to ensure sim-to-real success. As shown in Table III, Our domain randomization includes four elements. Mass variation is used to address the difference in mass between the real world and the simulation. Friction coefficient randomization ensures the robot can adapt to various surfaces in the real world. Motor strength variation helps the robot handle fluctuations in motor performance, such as those caused by low battery levels. Lastly, sensor noise is introduced to enable the robot to cope with real-world sensor inaccuracies.

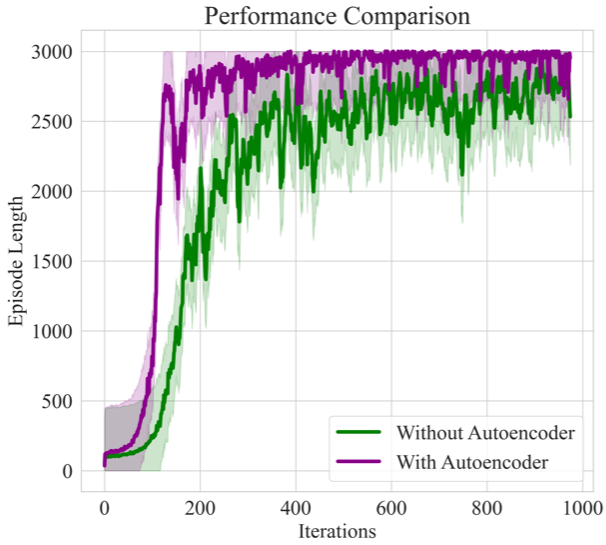

Fig. 7: Visualization of the episode length between the our model and that without Autoencoder.

B. Evaluation and Result

In this section, we evaluate the effectiveness and robustness of our proposed framework, with the real-world results shown in Figure 5. Our robot demonstrates robust, human-like walking, running, and jumping abilities. It can effectively adapt to external disturbances as well. In Figure 5.1, external forces are applied to the robot's leg, showing its ability to maintain balance and recover to a stable state when pushed. Figures 5.2 shows the robot's performance on diverse surfaces, such as soft sponge, highlighting its adaptability to various terrains. Figure 5.3 shows that except jumping on the plane, our robot can also jump onto a 15cm stair. We also compare the performance of our Autoencoder-integrated framework with a model that does not incorporate Autoencoder. Figures 5.4 and 5.5 demonstrate that the non-Autoencoder model struggles with large external forces and challenging surfaces. The training curves in Figure 7 indicate that the Autoencoder-integrated model converges to the maximum episode length more quickly than the model without Autoencoder. Finally, we also conduct experiments where the humanoid robot is pushed on its base in the Mujoco simulation. The curve in Figure 6 demonstrates that our framework enables the robot to recover quickly from disturbances, whereas the framework without the Autoencoder fails to recover after being pushed. These results collectively underscore the superiority of our Autoencoder-enhanced framework.

TABLE II: Hyperparameters for Training

Hyperparameter	Value	Description
Disc learning rate	5×10^{-5}	Learning rate for discriminator
Policy learning rate	5×10^{-5}	Learning rate for policy
Amp replay buffer	100000	Number of amp motions
Batch Size	4	Number of samples per batch
Discount Factor (γ)	0.994	Discount factor for future rewards
PPO Clip Ratio	0.2	Clipping parameter for PPO algorithm
GAE Lambda (λ)	0.9	Lambda for Generalized Advantage Estimation
Amp task reward lerp (λ)	0.8	Blending the task reward and the AMP reward

TABLE III: Domain Randomization Parameters

Parameter	Range	Description
Mass Variation	[-5, 5]	Random variation in the mass of the robot
Friction Coefficient	[0.1, 2.0]	Variation in the friction of the contact surfaces
Motor Strength	[0.9, 1.1]	Random scaling of motor torque output
Sensor Noise	[0.95, 1.05]	Random noise added to sensor readings

C. Limitation

Although our framework enables the robot to walk, run, and jump in a human-like manner, it is limited by the inability to perform all three motions within a single policy. Additionally, smooth transitions between different motions are not yet possible, which impacts the robot's versatility in practical applications. Moreover, our current framework only supports target velocity as the input command, further restricting its usability. Ideally, the user should be able to input a desired motion state, either through natural language or 3D motion parameters, allowing the framework to achieve the specified target state.

VI. CONCLUSION

In this paper, we retarget human motions from both the CMU dataset and those accurately recorded using our own mocap equipment. Initially, we use Poselib in ASE [8] for retargeting, followed by applying Pink for inverse kinematics to ensure that the motion data stays within the robot's physical limits. We successfully deploy the AMP framework [18] on a real humanoid robot, enabling it to walk, run, and jump in a human-like manner. Additionally, we employ a Autoencoder [11] to compress both historical and future observations into a latent space, incorporating this information into the privileged observation, making the policy aware of both past and future states. Our experiments demonstrate that the proposed Autoencoder-integrated framework significantly improves the stability of the humanoid robot.

REFERENCES

[1] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005109889900022

[2] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109801001741

[3] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, "Sim-to-real learning of all common bipedal gaits via periodic reward composition," 2021. [Online]. Available: https://arxiv.org/abs/2011.01387

[4] H. Duan, A. Malik, M. S. Gadde, J. Dao, A. Fern, and J. Hurst, "Learning dynamic bipedal walking across stepping stones," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 6746–6752.

[5] H. Duan, A. Malik, J. Dao, A. Saxena, K. Green, J. Siekmann, A. Fern, and J. Hurst, "Sim-to-real learning of footstep-constrained bipedal dynamic walking," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 428–10 434.

[6] Q. Zhang, P. Cui, D. Yan, J. Sun, Y. Duan, G. Han, W. Zhao, W. Zhang, Y. Guo, A. Zhang, and R. Xu, "Whole-body humanoid robot locomotion with human reference," 2024. [Online]. Available: https://arxiv.org/abs/2402.18294

[7] X. Cheng, Y. Ji, J. Chen, R. Yang, G. Yang, and X. Wang, "Expressive whole-body control for humanoid robots," 2024. [Online]. Available: https://arxiv.org/abs/2402.16796

[8] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler, "Ase: large-scale reusable adversarial skill embeddings for physically simulated characters," *ACM Transactions on Graphics*, vol. 41, no. 4, p. 1–17, July 2022. [Online]. Available: http://dx.doi.org/10.1145/3528223.3530110

[9] S. Caron, Y. De Mont-Marin, R. Budhiraja, S. H. Bang, I. Domrachev, and S. Nedelchev, "Pink: Python inverse kinematics based on Pinocchio," 2024. [Online]. Available: https://github.com/stephane-caron/pink

[10] J. Siekmann, Y. Godse, A. Fern, and 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)*, May 2021. [Online]. Available: http://dx.doi.org/10.1109/icra48506.2021.9561814

[11] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *CoRR*, vol. abs/2003.05991, 2020. [Online]. Available: https://arxiv.org/abs/2003.05991

[12] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, Oct. 2020. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abc5986

[13] H. Lai, W. Zhang, X. He, C. Yu, Z. Tian, Y. Yu, and J. Wang, "Sim-to-real transfer for quadrupedal locomotion via terrain transformer," 2023. [Online]. Available: https://arxiv.org/abs/2212.07740

[14] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024.

[15] I. Radosavovic, B. Zhang, B. Shi, J. Rajasegaran, S. Kamat, T. Darrell, K. Sreenath, and J. Malik, "Humanoid locomotion as next token prediction," 2024. [Online]. Available: https://arxiv.org/abs/2402.19469

[16] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics*, vol. 37, no. 4, p. 1–14, July 2018. [Online]. Available: http://dx.doi.org/10.1145/3197517.3201311

[17] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Robust and versatile bipedal jumping control through reinforcement learning," 2023. [Online]. Available: https://arxiv.org/abs/2302.09450

[18] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics*, vol. 40, no. 4, p. 1–20, July 2021. [Online]. Available: http://dx.doi.org/10.1145/3450626.3459670

[19] Y. Mu, S. Zou, K. Yin, Z. Tian, L. Cheng, W. Zhang, and J. Wang, "Racon: Retrieval-augmented simulated character locomotion control," 2024. [Online]. Available: https://arxiv.org/abs/2406.17795

[20] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

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