EMP: Executable Motion Prior for Upper-body Motion Imitation when Humanoid Robot Standing

Abstract— **Achieving stable and accurate imitation of human motions by humanoid robots remains challenging, particularly for tasks requiring firmly standing while executing diverse upper-body movements. While reinforcement learning (RL) offers promise for whole-body motion control, directly tracking upper-body motions often lead to the difficulty in resolving the conflict between stability and similarity. In this paper, we present a RL framework for humanoid robots to stably imitate diverse upper-body motions while standing. Our approach employs a motion retargeting network to translate human motions into humanoid targets. A decoupled RL policy is then trained to control the lower body for standing while tracking upper-body motions. However, large-amplitude motions outside training data may lead to loss of balance. To address this, we introduce an Executable Motion Prior (EMP) network that adjusts potentially unstable actions into more feasible motions while minimizing changes to the intended movement amplitude, enhancing standing robustness without retraining. We evaluate our framework through simulation and real-world tests, demonstrating its practical applicability. Project page.**

I. Introduction

The humanoid form enables humanoid robots to better adapt to human environments, tools, and interactions. Our goal is to empower humanoid robots to execute human-like movements, facilitating a more accurate mapping of human motions onto the robots. This capability allows them to efficiently acquire human motion skills, thereby establishing a foundation for carrying out subsequent tasks.

However, significant challenges persist in the practical implementation of humanoid robots that mimic human motions. The intricate dynamic characteristics of humanoid robots, combined with their high-dimensional state and action spaces, make motion control particularly complex. Although model-based controllers have demonstrated impressive results in whole-body motion imitation [1], [2], [3], the computational demands of complex dynamic models confine these methods to simplified representations, thereby limiting their scalability for dynamic motions.

Recently, reinforcement learning (RL) methods have gained traction in the field of humanoid robotics. Initially, RL was utilized within the graphics community to generate humanoid motions from human motion data for animated characters [4], [5], [6]. Furthermore, RL controllers have been developed for humanoid robot whole-body control [7], [8], [9], [10]. These whole-body controllers integrate imitation learning and reinforcement learning techniques to achieve human-like motion imitation, such as punching and dancing. However, in practical applications, beyond these whole-body tracking tasks, numerous tasks require stable imitation of standing motions. The current mainstream approaches are illustrated

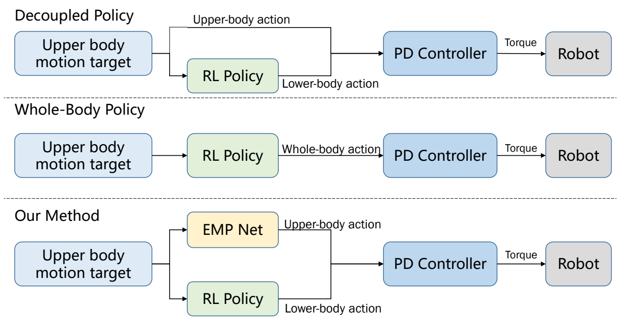

Fig. 1: Different Motion Imitation Framework. (a) Decoupled Policy, such as PMP [12], only generates lower-body actions and executes upper-body target straightly. (b) Whole-Body Policy, like HumanPlus [8] and Exbody [7], controls whole body joints to imitate upper-body motion target. (c) Our method introduces an executable motion prior to optimize upper-body motion target while RL policy provides lower-body actions.

in Figure 1. When we apply whole-body tracking methods, we observe that when joints are entirely controlled by the RL policy, vibrations and deviations can occur in the base and upper-body actions. Additionally, some whole-body tracking algorithms may experience stability issues when confronted with motions outside training data [7], [11]. On the other hand, while implementing a decoupled policy, a conflict between robot stability and motion similarity may occur. Directly executing upper-body actions can cause the robot's limited control capabilities to exceed the RL policy's constraints, leading to a loss of balance.

Our goal is to develop a humanoid robot upper-body motion imitation system capable of maintaining stable standing while executing a variety of upper-body motions and the system can flexibly adapt to motions that lie outside the training data. Our framework is depicted in Figure 2.

Initially, we employ the method described in [13] to construct a graph convolution network that retargets human motions into humanoid movements, generating a comprehensive motion dataset. Subsequently, we train a decoupled RL policy for upper-body motion imitation using these retargeted motions. However, due to the heavy upper limbs, the control capability of the RL policy is restricted, which may result in stability issues when encountering broad movements.

When humans perform upper-body motions, they can recognize potential dangers and make timely adjustments to their movements. Inspired by this, we propose an **E**xecutable **M**otion **P**rior (**EMP**) that modifies the input target upper-body motions based on the robot's current state. This approach

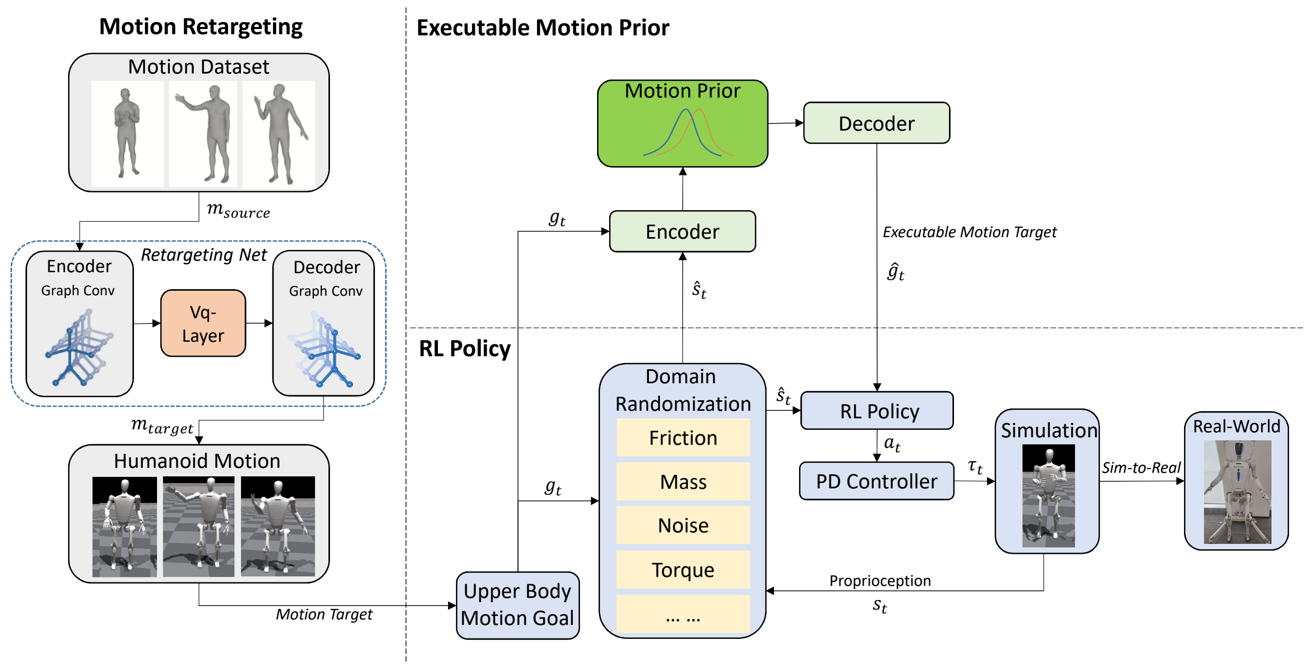

Fig. 2: Overview of our framework. **Motion Retargeting** (section IV): We train a graph convolution retargeting network to convert human motions $\boldsymbol{m}_{source}$ to humanoid joint actions $\boldsymbol{m}_{target}$ as motion goal for imitation. **RL Policy** (section V): We train an upper-body imitation policy for the humanoid to track the upper-body motion goal $\boldsymbol{g}_t$ while keeping balance. **Executable Motion Prior** (section VI): We use a AE-based network to adjust the goal motion based on the current state $\hat{\boldsymbol{s}}_t$, improving stability.

enhances standing stability while minimizing alterations to the motion amplitude. Utilizing the dataset obtained from motion retargeting and the trained RL controller, we train an EMP network. This network transforms unstable upper-body actions into stable ones by simultaneously encoding the robot's current state and action objectives into a latent space and decoding them into new, more reasonable action objectives, functioning as an action optimization module before RL controller. Meanwhile, since variables in the simulator cannot obtain gradients, we train a world model to simulate the state transitions of the simulator, thereby achieving gradient backpropagation during EMP training. Finally we deploy this framework in real-world humanoid robots.

Our contributions are as follows:

1) An RL based framework for humanoid robot imitating upper-body motion, which includes a motion retargeting network to transfer human motions to humanoid motions and an RL policy to control the robot standing stably while tracking any upper-body motions;
2) An executable motion prior for the RL imitation policy system that adjusts target motions based on the humanoid's current state, enhancing stability while minimizing changes in motion amplitude;
3) A world model to simulate the state transition process of the environment to enable efficient learning of motion prior through gradient backpropagation.
4) Sim-to-real transfer of our system that demonstrates its effectiveness in two humanoid robots.

II. Related Works

A. Motion Retargeting

Motion retargeting facilitates the transfer of motion data from a source character to a target character. In the context of animation, both optimization-based methods [14] and learning-based methods [13], [15] are employed for motion transfer between animated characters. Learning-based methods tend to yield more efficient results and can facilitate motion transfer across different skeletal structures [13], [16].

In human-robot motion retargeting, Delhaisse et al. [17] use shared latent variable models to retarget motions between different humanoids. Ayusawa et al. [18] reconstruct human motion within the physical constraints imposed by humanoid dynamics and offer a precise morphing function for different human body dimensions. Zhang et al. [19] utilize latent optimization to train a retargeting network, which achieves similarity and rapid adaptability at the kinematic level. These retargeting methods addressed the issue of human motion imitation at the kinematic level but lack consideration of dynamics.

B. Reinforcement Learning for Humanoid Motion Imitation

To tackle the problems associated with dynamics, researchers propose various control algorithm to enable the humanoid robot to imitate human motions.

Traditional methods, such as model predictive control (MPC), use model-based optimization methods to minimize tracking errors under stability and contact constraints [20]. However, due to the high computational burden, the humanoid model is usually simplified [21], [22], which limits the accuracy of imitation.

Reinforcement Learning (RL) controllers provide an alternative solution. Before RL-based controllers were used in real-world humanoids, they were often used in physics-based animation control [4], [5], [23]. Nevertheless, the humanoid avatars usually have less restrictions on joint positions, torques, and sometimes with additional auxiliary force [24].

On the other hand, realistic humanoids have complex dynamic models, and it is difficult to obtain privileged states, such as base velocity and height, from built-in sensors [7]. This makes it impossible to directly transfer RL models used for animated characters to physical humanoids. Li et al. [25] proposed an end-to-end RL approach and used task randomization to build a robust dynamic locomotion controller for bipedal robots. Siekmann et al. [26] use stair-like terrain randomization to build an RL controller for humanoid traversing stair-like terrain. Cheng et al. [7] train a whole-body humanoid controller with a large-scale motion dataset. He et al. [10] use a privileged policy to select an executable motion dataset, which helps training a robust RL policy for sim-to-real deployment. Fu et al. [8] train a task-agnostic low-level policy to track retargeted humanoid poses. Lu et al. [12] proposed a CVAE-based motion prior to enhance the robustness of controller.

III. Overview

Our method focuses on building a stable standing control system for a humanoid robot capable of adapting to various upper limb movements, which can also accommodate motions outside of the training data. The framework is structured into three segments: Motion Retargeting, RL Policy and Executable Motion Prior (EMP), as illustrated in Figure 2.

The motion retargeting network offers upper-body motion target to the control policy. We decouple the whole-body control policy into π_{lower} and π_{upper}. π_{lower} is an RL-based policy which generates lower-body actions from proprioception state to keep the humanoid robot standing in balance while tracking upper-body motions. The upper-body policy π_{upper} is executable motion prior (EMP) network, which adapts the upper-body goals based on the current status of the robot.

A. Motion Retargeting

The motion retargeting network is responsible for mapping the motions from the human motion dataset to robot actions, which are used for training in reinforcement learning and the EMP network.

Prior work [13] has achieved cross-skeleton motion retargeting between animated characters with a graph network. We develop a network for motion retargeting from human to humanoid. Using networks for motion retargeting offers good real-time performance and generalization capabilities. The details of the retargeting network is elaborated upon in section IV.

B. RL Control Policy

With the robot upper-body motion dataset retargeted from human motions, we incorporate upper-body movements into the RL training process to enable the final policy π_{lower} to adapt to interference from different motions. The specifics of the training are discussed in section V.

C. Executable Motion Prior

When the amplitude of upper-body motion targets exceeds the range that the RL controller can handle, the entire system is at risk of losing balance. In this instance, the EMP network makes adjustments to the upper-body actions based on the present state of the robot. EMP network is a network with an encoder-decoder architecture. Inspired by the framework of ControlVAE [27], Figure 3 displays the framework for the EMP algorithm, categorized into two processes: training and generation. Throughout the training process, we employ a world model to simulate state transitions, allowing us to predict the future states of the robot. Then the EMP network can be trained based on feedback derived from future states. During the generation process, With the trained prior distribution, the EMP network generates the executable target for humanoid from source target and state.

The more details are explained in section VI.

IV. Retargeting Human Motion to Humanoids

A. Retargeting Network Architecture

Figure 2 illustrates the structure of our retargeting network. We regard the upper-body skeleton of the humanoid and the human as a graph. Referring to the framework of VQ-VAE [28], our network consists of a motion encoder, a vq-codebook layer and a motion decoder.

The motion encoder f_e embeds the source motion from human. The source motion is represented as the positions of key nodes $\boldsymbol{Q}_A \in \mathbb{R}^{N_A \times 3}$ and the features of edges $\boldsymbol{E}_A$. After passing through the graph convolutional layers, the source motion features are encoded into the latent space features : $\boldsymbol{z}_A = f_e(\boldsymbol{Q}_A, \boldsymbol{E}_A)$. A transformation net f_{tf} converts the latent features of input skeleton A into the latent features of output skeleton B: $\boldsymbol{z}_B = f_{tf}(\boldsymbol{z}_A)$. Then the codebook layer chooses the nearest element of the latent embedding vectors:

$$\boldsymbol{z}_e = \boldsymbol{e}_k \text{ where } k = \arg\min_j \|\boldsymbol{z}_B - \boldsymbol{e}_j\|_2 \tag{1}$$

The motion decoder f_d generates the target motion $\boldsymbol{Q}_B \in \mathbb{R}^{N_B}$ (represented by joint angles) with latent embedding vector $\boldsymbol{z}_e$ and edge features $\boldsymbol{E}_B$: $\boldsymbol{Q}_B = f_d(\boldsymbol{z}_e, \boldsymbol{E}_B)$.

The key nodes are waist, torso, shoulder, elbow and wrist.

B. Training Loss

Combined with the method in [29], the training loss of our retargeting network is composed of five terms: end effector loss L_{ee}, orientation loss L_{ori}, elbow loss L_{elb}, embedding loss L_{emb} and commitment loss L_{com}. We list the losses in Tab I, where $\boldsymbol{p}$ and $\hat{\boldsymbol{p}}$ mean the node position of human and humanoid respectively, $\boldsymbol{R}$ and $\hat{\boldsymbol{R}}$ mean the end effector (namely wrist) rotation matrix, sg() means stop gradient.

TABLE I
TRAINING LOSS FOR RETARGETING NETWORK

Term	Expression	Weight
L_{ee}	$\Vert\frac{\boldsymbol{p}^{ee}-\boldsymbol{p}^{elb}}{\Vert\boldsymbol{p}^{ee}-\boldsymbol{p}^{elb}\Vert_2}-\frac{\hat{\boldsymbol{p}}^{ee}-\hat{\boldsymbol{p}}^{elb}}{\Vert\hat{\boldsymbol{p}}^{ee}-\hat{\boldsymbol{p}}^{elb}\Vert_2}\Vert_2^2$	100
L_{ori}	$\Vert\boldsymbol{R}-\hat{\boldsymbol{R}}\Vert_2^2$	100
L_{elb}	$\Vert\frac{\boldsymbol{p}^{elb}-\boldsymbol{p}^{sho}}{\Vert\boldsymbol{p}^{elb}-\boldsymbol{p}^{sho}\Vert_2}-\frac{\hat{\boldsymbol{p}}^{elb}-\hat{\boldsymbol{p}}^{sho}}{\Vert\hat{\boldsymbol{p}}^{elb}-\hat{\boldsymbol{p}}^{sho}\Vert_2}\Vert_2^2$	100
L_{emb}	$\Vert\mathrm{sg}(\boldsymbol{z}_e)-\boldsymbol{e}\Vert_2^2$	10000
L_{com}	$0.25\Vert\boldsymbol{z}_e-\mathrm{sg}(\boldsymbol{e})\Vert_2^2$	10000

TABLE II
REWARDS EXPRESSIONS AND WEIGHTS

Term	Expression	Weight
	Regularization	
Base orientation	$\exp(-10\Vert\boldsymbol{rpy}_t^{xy}\Vert_1)$	3.0
Projected gravity	$\exp(-20\Vert\boldsymbol{pg}_t^{xy}\Vert_2)$	3.0
Base height	$\exp(-100\vert h_t-h^{\text{ref}}\vert)$	0.2
Base linear velocity	$\exp(-10\Vert\boldsymbol{v}_t\Vert_2^2)$	0.75
Base angular velocity	$\exp(-20\Vert\boldsymbol{\omega_t}\Vert_2)$	0.75
Base acceleration	$\exp(-3\Vert\boldsymbol{v}_t-\boldsymbol{v}_{t-1}\Vert_2)$	0.2
Leg DoF position	$\exp(-100\Vert\boldsymbol{q}_t^{\text{leg}}-\boldsymbol{q}^{\text{leg,ref}}\Vert_2)$	1.0
Feet contact	$\mathbb{1}(F_{\text{feet}}^z\geqslant 5)$	0.5
Feet slip	$\mathbb{1}(F_{\text{feet}}^z\geqslant 5)\times\sqrt{\Vert\boldsymbol{v}_t^{\text{feet}}\Vert_2}$	0.2
	Energy	
Action range	$\Vert\boldsymbol{a}_t\Vert_1$	-0.075
Action rate	$\Vert\boldsymbol{a}_t-\boldsymbol{a}_{t-1}\Vert_2^2$	-1.5
Action acceleration	$\Vert\boldsymbol{a}_t+\boldsymbol{a}_{t-2}-2\boldsymbol{a}_{t-1}\Vert_2^2$	-1.5
Torques	$\Vert\boldsymbol{\tau}_t\Vert_2^2$	-1e-5
Dof velocity	$\Vert\dot{\boldsymbol{q}}_t\Vert_2^2$	-1e-4
Dof acceleration	$\Vert\ddot{\boldsymbol{q}}_t\Vert_2^2$	-1e-7

V. RL CONTROL POLICY TRAINING FOR HUMANOID UPPER-BODY IMITATION

A. State Space

We consider our RL control policy as a goal-conditioned policy $\pi:\mathbf{G}\times\mathbf{S}\longrightarrow\mathbf{A}$, where $\mathbf{G}$ is goal space that indicates the upper-body motion target, $\mathbf{S}$ is the observation space and $\mathbf{A}$ is the action space for lower-body joints.

We define goal state as $\boldsymbol{g}_t\triangleq\boldsymbol{q}_{\text{target}}\in\mathbb{R}^{15}$, where $\boldsymbol{q}_{\text{target}}$ represents the target joint position of upper-body joints, including two 7-dof arms and one 1-dof waist. The action is denoted as $\boldsymbol{a}_t\in\mathbb{R}^{12}$. We define our observation state as $\boldsymbol{s}_t\triangleq[\boldsymbol{q}_t,\boldsymbol{a}_{t-1},\boldsymbol{rpy}_t,\boldsymbol{g}_t]$, where $\boldsymbol{q}_t\in\mathbb{R}^{27}$ indicates the joint position and $\boldsymbol{rpy}_t\in\mathbb{R}^3$ is the euler angle of robot base. We combine states of last T frames together as $\boldsymbol{S}_t=\{\boldsymbol{s}_{t-T:t}\}\in\mathbb{R}^{T\times 65}$ to utilize history message. We set $T=15$ in experiments. The action space consists of 12-dim joint position targets (two 6-dof legs). The joint actions will be converted to joint torque by a PD controller.

B. Reward Design

The rewards are detailed in Tab II, where h^{ref} is reference height of base, $\boldsymbol{q}^{\text{leg,ref}}$ is reference joint positions of legs.

Our policy focuses on upper-body motion imitation while standing, so we just set $\boldsymbol{v}_t^{\text{ref}}=0$ and $\boldsymbol{\omega}_t^{\text{ref}}=0$.

To improve training efficiency, we reset training process when the projected gravity on x or y axis exceeds 0.7.

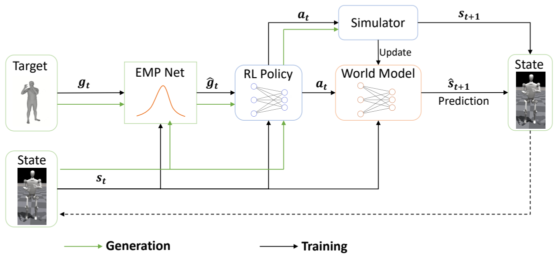

Fig. 3: Framework of our EMP System. EMP network generates optimized upper-body motion targets conditioned on the state of the robot. The world model learns the state transition model from the simulator for gradient backpropagation.

VI. EXECUTABLE MOTION PRIOR

A. Network Architecture

The EMP network consists of an encoder and a decoder, showed in Figure 2. The encoder is composed of 3 sub-networks: a state encoder f_s, a target encoder f_t and a fusion network f_ψ. The state encoder and target encoder encode the state and the target into the latent space variable $\boldsymbol{z}_1,\boldsymbol{z}_2$, respectively.

$$\boldsymbol{z}_1,\boldsymbol{z}_2=f_s(\boldsymbol{s}_t),f_t(\boldsymbol{g}_t) \tag{2}$$

Then the fusion network encodes two variables into a single latent space vector $\boldsymbol{z}=f_\psi(\boldsymbol{z}_1,\boldsymbol{z}_2)$, which follows a standard normal distribution $\boldsymbol{z}\sim\mathcal{N}(0,1)$ and then the decoder generates a new target for the humanoid. Then the EMP can be described as:

$$\hat{\boldsymbol{g}}_t=f_\theta(\boldsymbol{s}_t,\boldsymbol{g}_t) \tag{3}$$

where θ is the learnable variable of EMP net. The encoder and decoder nets are both MLP networks.

B. Training Process

The training process consists of two parts: world model f_w training and EMP training.

World Model Training. Due to the inability to obtain gradients from the robot state information in the simulation, we use a world model to simulate the state transition process of the humanoid robot environment. The world model predicts the next state of the humanoid robot depending on the current state and action:

$$\hat{\boldsymbol{s}}_{t+1}=f_w(\boldsymbol{s}_t,\boldsymbol{a}_t) \tag{4}$$

where w is the learnable variable of world model. Then we have world model prediction loss:

$$L_{pre}=\Vert\boldsymbol{s}_{t+1}-\hat{\boldsymbol{s}}_{t+1}\Vert_2^2 \tag{5}$$

where $\boldsymbol{s}_{t+1}$ is the state given by the simulator, namely isaacgym here and $\hat{\boldsymbol{s}}_{t+1}$ is the prediction of the world model. The state here is defined the same as section V-A.

EMP Training. While the robot is losing its balance, the following situations usually occur: (1) The center of gravity is projected away from the support surface; (2) The robot's torso is no longer oriented vertically upwards. Therefore we train the network to avoid these situations. Meanwhile, the self-collision and smoothness of the motion can also influence the balance.

The training process of EMP is illustrated in Figure 3. We have the following losses:

i) *Reconstruction Loss*. The reconstruction loss L_{rec} encourages the generated motion $\hat{\boldsymbol{g}}_t$ to be as identical to the source target $\boldsymbol{g}_t$. We define

$$L_{rec} = \|\boldsymbol{g}_t - \hat{\boldsymbol{g}}_t\|_2^2 \tag{6}$$

ii) *Centroid Loss*. The centroid loss L_{cen} prompts the centroid of humanoid to stay in the range of support surface under foot. L_{cen} is defined as

$$L_{cen} = \min\{\exp(-7(0.03 - d)), 10\} - 1 \tag{7}$$

where d is the distance between the center of the foot support surface and the projection of the centroid onto the ground.

iii) *Orientation Loss*. The orientation loss L_{ori} promotes the humanoid's base to stay upright, which can improve the stability of the humanoid. Then L_{ori} is defined as

$$L_{ori} = \exp(-\|\widehat{\boldsymbol{pg}}_{t+1}^{xy}\|_2^2) - 1 \tag{8}$$

where $\widehat{\boldsymbol{pg}}_{t+1}^{xy}$ is the projected gravity vector, which is calculated from $\widehat{\boldsymbol{rpy}}_{t+1}^{xy}$ predicted by the world model.

iv) *Collision Loss*. The collision loss L_{col} encourages the motion to reduce self-collision of the humanoid. We simplify the links that may collide into a spherical model, and calculate the distance between the links. We define

$$L_{col} = \sum_{i,j \in \mathbb{J}} \exp[-2(0.08 - \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2)] \tag{9}$$

where $\mathbb{J}$ is the set of the links that may collide with each other, we define $\mathbb{J} = \{\text{torso, hand, sacrum, thigh}\}$ here. $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ mean the coordinate of the link centers, which can be calculated with forward kinematics (FK).

v) *Smoothness Loss*. The smoothness loss L_{smo} promotes the motion to be smooth and reduces the occurrence of motion mutations. L_{smo} is defined as

$$L_{smo} = \|\hat{\boldsymbol{g}}_t - \hat{\boldsymbol{g}}_{t-1}\|_2^2 + 0.2\|\hat{\boldsymbol{g}}_t + \hat{\boldsymbol{g}}_{t-2} - 2\hat{\boldsymbol{g}}_{t-1}\|_2^2 \tag{10}$$

vi) *Regularization Loss*. The regularization loss L_{reg} encourages the latent variable to conform to standard Gaussian distribution, similar to method in [19]. L_{reg} is defined as

$$L_{reg} = \|\boldsymbol{z}\|_2^2 \tag{11}$$

where $\boldsymbol{z}$ is the latent variable.

Finally we get overall loss for EMP training:

$$\begin{aligned} L =& \lambda_{rec} L_{rec} + \lambda_{ori} L_{ori} + \lambda_{col} L_{col} \\ &+ \lambda_{cen} L_{cen} + \lambda_{smo} L_{smo} + \lambda_{reg} L_{reg} \end{aligned} \tag{12}$$

We set $\lambda_{rec} = 20, \lambda_{ori} = 10, \lambda_{col} = 1, \lambda_{cen} = 10, \lambda_{smo} = 100, \lambda_{reg} = 1$ here. The overall training process is shown in Algorithm 1.

Algorithm 1 Training process of EMP

```
1: for number of training epochs do:
2:   for batch of motions in training set do:
3:     Reset simulation environment;
4:     for $t \leftarrow 0$ to $T-1$ do:
5:       Sample $\boldsymbol{a}_t = \pi(\boldsymbol{s}_t, \boldsymbol{g}_t)$;
6:       Sample $\boldsymbol{s}_{t+1}$ and $\hat{\boldsymbol{s}}_{t+1} = f_w(\boldsymbol{s}_t, \boldsymbol{a}_t)$;
7:       Update world model $f_w$ with $\nabla_w L_{pre}$;
8:     end for
9:     Reset simulation environment;
10:    for $t \leftarrow 0$ to $T-1$ do:
11:      Sample $\hat{\boldsymbol{g}}_t = f_\theta(\boldsymbol{s}_t, \boldsymbol{g}_t)$;
12:      Sample $\boldsymbol{a}_t = \pi(\boldsymbol{s}_t, \hat{\boldsymbol{g}}_t)$;
13:      Sample $\hat{\boldsymbol{s}}_{t+1} = f_w(\boldsymbol{s}_t, \boldsymbol{a}_t)$;
14:      Update EMP $f_\theta$ with $\nabla_\theta L$;
15:    end for
16:  end for
17: end for
```

VII. Experiments

A. Experiment Setup

Hardware Platform. The main humanoid platform we use is a full-sized robot (1.65m, 60kg) which feature 27 degrees of freedom, including two 7-dof arms (about 6kg for one arm, which brings higher load capacity and control difficulty), two 6-dof legs and one 1-dof in waist.

Implementation Details. The encoder and decoder of retargeting network are both graph convolutional neural networks with three graph convolutional layers, and the hidden sizes are [16,32,64] and [66,32,16], respectively. The codebook of retargeting network has 2048 latent space vectors, each with a dimensionality of 64. The world model is implemented as a multi-layer perceptrons (MLP) with hidden size of [1024,512]. The state encoder and target encoder of EMP network are MLPs with hidden sizes of [1024,1024], and the fusion network and decoder are MLPs with hidden sizes of [2048,2048]. The RL training is conducted on an NVIDIA A800 (80GB) GPU and takes about 6 hours with a learning rate of 1e-3 in Isaac Gym [30] with domain randomization [31]. The EMP network is trained on an NVIDIA RTX4060 GPU for 5 hours.

Motion Dataset. We utilize our retargeting network to construct our humanoid motion dataset, selecting the GRAB dataset [32] from the AMASS dataset [33] as our source. Our network is trained on this dataset, and we use the retargeting results to train our reinforcement learning policy. To address the issue of varying motion lengths, we segment these motions into smaller clips, each consisting of 60 frames, and then reconnect them. For EMP training, we further divide these motions into segments of 200 frames each, which facilitates our batch collection process.

EMP Training. We train our Executable motion prior (EMP) on retargeted GRAB dataset, which we randomly divided into

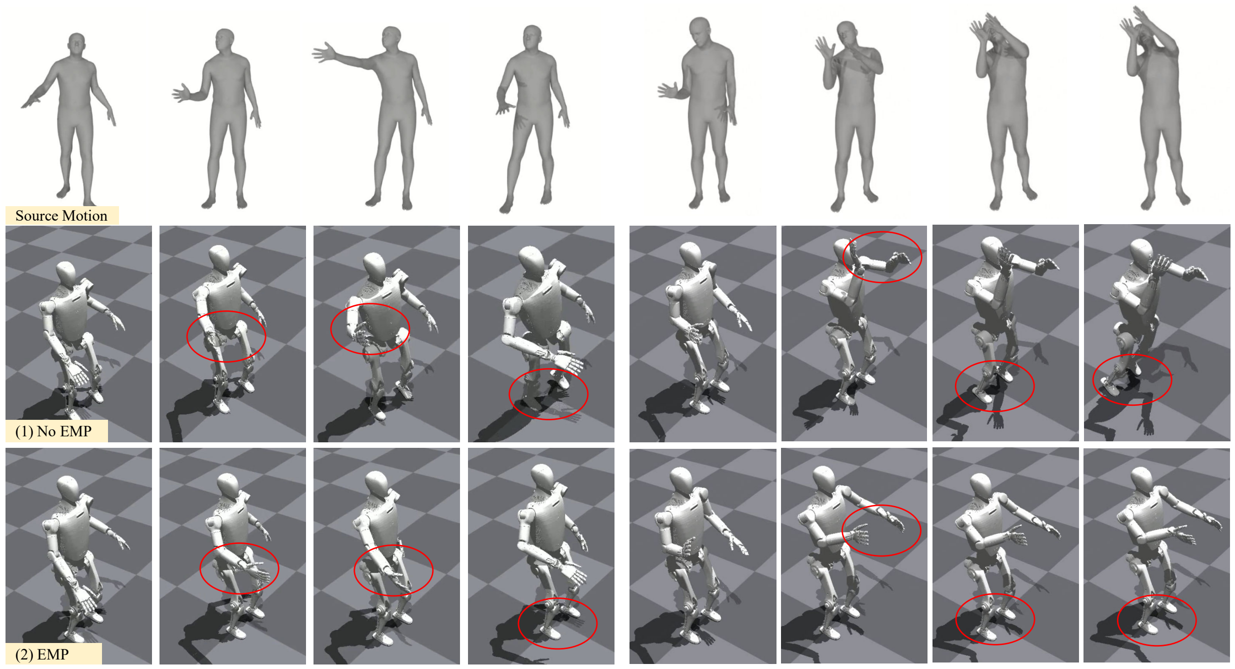

Fig. 4: Simulation experiments (left motion: spheremedium pass, right motion: lightbulb screw). The results show that while executing dangerous motions, EMP network will optimize the unexecutable motion and keep the robot standing stably.

a training set (1,070 motions) and a test set (270 motions). We train the world model and EMP with Adam [34] optimizer with an initial learning rate of 1e-3.

Baselines. We consider the following baselines:

i) **Privileged Policy**. Referring to the settings in [10], the observation space for the privileged policy input includes all first-hand robot state, and no noise or domain randomization is added during training. The privileged policy demonstrates the upper limit of the robot's mobility.

ii) **Whole-Body Policy**. Instead of only controlling lower-body joints, the whole-body policy controls all 27 joints. We train this policy based on the rewards and methods in Exbody [7] and HumanPlus [8]. We use this policy to track upper-body motions and keep lower-body joints in default angles.

iii) **Decoupled Imitation Policy**. Our main RL policy, which controls lower-body joints to keep balance while tracking upper-body motions.

iv) **Decoupled Imitation Policy with Predictive Motion Prior (PMP)** [12]. We add PMP features into the observation state of decoupled policy.

v) **Decoupled Imitation Policy with EMP**. Our full system, RL policy with executable motion prior.

vi) **EMP when Danger**. Enable EMP only when regloss of the latent space exceeds 0.04. The regloss reflects the degree of the motion deviation from prior distribution.

Metrics. The metrics are as follows:

- **Success Rate** (SUC). We define imitation failed when termination conditions in section **??** are triggered.
- **Mean upper-joint position reward** (MJP). We define upper-body joint position reward as $r_{jp} = \exp\left(-\|\boldsymbol{q}_t - \boldsymbol{g}_t\|_2\right)$.
- **Mean self-collision reward** (MSC). Self-collision often happens while tracking motions, which will seriously disturb the balance control of the robots. We use link contact force to evaluate this metric: $r_{\text{col}} = -\|\boldsymbol{f}_t\|_2$. We only consider the contact between these links: torso, thigh, hand and sacrum.
- **Mean Base Velocity reward** (MBV).
- **Mean Base Acceleration reward** (MBA).
- **Mean Base Orientation reward** (MBO).
- **Mean upper-body action Smoothness** (MUS). We calculate the velocity rate of upper-body joints to evaluate smoothness: $r_{smo} = \|\dot{q}_t - \dot{q}_{t-1}\|_2^2$

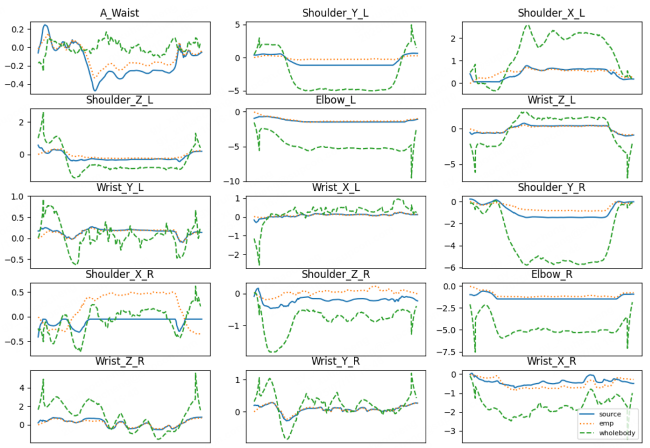

Fig. 5: The upper motion (lightbulb screw) of whole-body policy and EMP. The whole-body policy brings significant vibration to upper-body motions.

B. Simulation Experiments

The results of simulation experiments are summarized in Tab III. The results reveal that our EMP methods outperform other baselines. Compared with Decoupled Policy, the Whole-Body Policy has a much higher success rate. However, it

TABLE III
EXPERIMENT RESULTS

Baselines	Metrics						
	SUC ↑	MJP ↑	MSC ↓	MBV ↑	MBA ↑	MBO ↑	MUS ↓
Privileged Policy	97.0%	0.8192	0.3816	0.7727	0.7158	0.7702	2.4420
Whole-Body Policy	**100%**	0.7915	0.3915	0.7153	0.6801	0.5204	7.6708
Decoupled Policy	89.6%	**0.8296**	0.3602	0.7813	0.7513	0.6571	**2.2486**
PMP	96.7%	0.8192	0.3505	0.7423	0.7150	0.6871	2.2708
EMP when Danger	98.1%	0.8233	0.1617	0.8029	0.7571	0.6868	2.3700
EMP (Ours)	98.5%	0.8230	**0.1423**	**0.7997**	**0.7588**	**0.6892**	2.3678

performs poor in other metrics, especially upper-body motion smoothness. The PMP baseline has achieved a certain improvement in base decoupled policy, but its effectiveness is weaker than that of EMP.

The EMP network optimizes upper-body motion while minimizing deviations as much as possible, thereby improving control stability. The acceleration, velocity and orientation stability of the base are improved remarkably and the collision is reduced while the joint position error is slightly increased.

Figure 4 shows some simulation results. While the upper-body motions are executable, our framework maintains consistency with the initial motions. Once the amplitude of the motion exceeds the control capability of the controller, the EMP will optimize the motion to keep the overall robot stable and avoid falling situations. We analyze the upper-body motion variation curve in Figure 5. We can see that while performing upper-body motions, whole-body policy exhibits noticeable oscillations, especially in wrist joints.

TABLE IV
ABLATION STUDY

Methods	SUC ↑	MJP ↑	MSC ↓	MBV ↑	MBA ↑	MBO ↑	MUS ↓
EMP w/o smoothness	27.0%	0.637	0.211	0.702	0.591	0.555	5.434
EMP w/o orientation	2.6%	0.327	3.982	0.470	0.283	0.375	12.82
EMP w/o centroid	10.7%	0.396	2.850	0.531	0.232	0.422	11.00
Full EMP	**98.5%**	**0.823**	**0.142**	**0.800**	**0.759**	**0.689**	**2.368**

TABLE V
CROSS EMBODIMENT VALIDATION

Baselines	SUC ↑	MJP ↑	MSC ↓	MBV ↑	MBA ↑	MBO ↑	MUS ↓
Privileged Policy	99.3%	0.840	1.317	0.787	0.284	0.702	3.642
Whole-body Policy	**99.3%**	0.773	3.354	0.678	**0.634**	0.168	4.253
Decoupled Policy	90.0%	0.841	1.748	0.792	0.381	0.727	1.604
EMP when Danger	95.9%	0.835	0.799	0.797	0.371	0.742	1.691
EMP (Ours)	97.8%	**0.861**	**0.129**	**0.807**	0.394	**0.754**	**1.435**

C. Ablation Study

To validate the impact of different losses on the effectiveness of EMP, we conducted ablation experiments on smoothness loss, orientation loss and centroid loss. As illustrated in Tab IV, the results of the ablation experiments indicate that all three loss functions play an important role in the training of the EMP network. The absence of these loss functions not only affects the directly related metrics but also impacts the overall stability of the system. In contrast, the impact of the smoothness loss on the system is smaller than that of the other two losses.

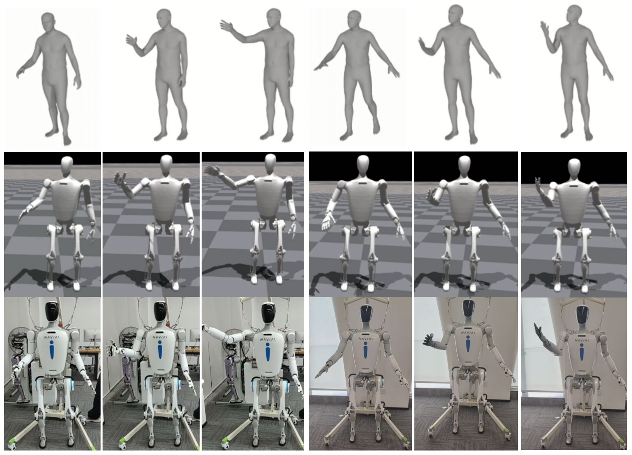
Fig. 6: Real-world experiments. The robot is imitating motions from dataset while standing.

D. Real-world Experiments

We test our system on real-world humanoid robot platform. All the proprioception of the robot comes from built-in sensors. Our RL policy and EMP run at 50Hz. The PD controller runs at 1kHz. We test several human motions from AMASS dataset in Figure 6.

E. Cross Embodiment Validation

We retrained and deployed our system on another humanoid platform, which also features two 7-dof arms, two 6-dof legs and one 1-dof in waist.

The results are shown in Figure 7, and the metrics of partial baselines are shown in Table V. We find that EMP has advantages in the vast majority of indicators, but it lags behind Whole-body Policy significantly in terms of MBA. The main reason is that Whole-body Policy makes sacrifices in tracking the waist joint, resulting in lower angular acceleration in the yaw direction. Experiments across multiple platforms demonstrate the adaptability and portability of our framework.

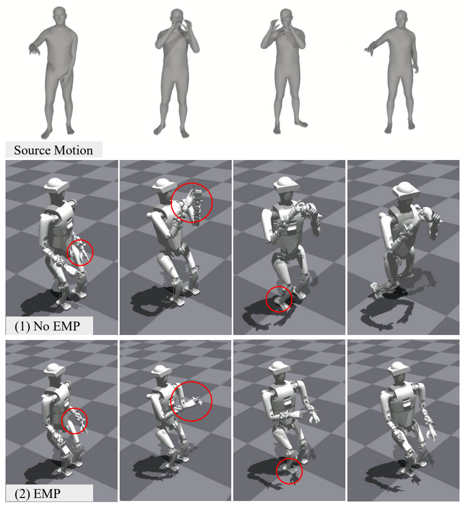

Fig. 7: Simulation experiment of cross embodiment validation

VIII. Conclusions and Future Work

In this work, we introduce a framework that enables the humanoid to imitate upper-body motions retargeted from human motions. We train a retargeting network from a humanoid motion dataset and an upper-body imitation RL policy to control the humanoid to keep balance while tracking motions. Then our approach utilizes executable motion prior before RL controller to transform difficult motions into executable targets that fit the humanoid control ability. Through simulations and real-world tests, we validated the effectiveness of our framework. However, we have not realized whole-body motion imitation due to high DoF and complex dynamics of the full-sized humanoid robot. Meanwhile, joint limitations of the robot result in a significant disparity between the retargeted motions and the source movements. We hope to address these limitations in the future to build a whole-body motion imitation system.

References

[1] Y. Ishiguro, T. Makabe, Y. Nagamatsu, Y. Kojio, K. Kojima, F. Sugai, Y. Kakiuchi, K. Okada, and M. Inaba, "Bilateral humanoid teleoperation system using whole-body exoskeleton cockpit tablis," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6419–6426, 2020.

[2] Y. Ishiguro, K. Kojima, F. Sugai, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba, "High speed whole body dynamic motion experiment with real time master-slave humanoid robot system," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5835–5841.

[3] J. Ramos and S. Kim, "Humanoid dynamic synchronization through whole-body bilateral feedback teleoperation," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 953–965, 2018.

[4] 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.

[5] 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.

[6] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 143:1–143:14, July 2018.

[7] X. Cheng, Y. Ji, J. Chen, R. Yang, G. Yang, and X. Wang, "Expressive whole-body control for humanoid robots," 2024.

[8] Z. Fu, Q. Zhao, Q. Wu, G. Wetzstein, and C. Finn, "Humanplus: Humanoid shadowing and imitation from humans," in *arXiv*, 2024.

[9] M. Ji, X. Peng, F. Liu, J. Li, G. Yang, X. Cheng, and X. Wang, "Exbody2: Advanced expressive humanoid whole-body control," *arXiv preprint arXiv:2412.13196*, 2024.

[10] T. He, Z. Luo, W. Xiao, C. Zhang, K. Kitani, C. Liu, and G. Shi, "Learning human-to-humanoid real-time whole-body teleoperation," 2024.

[11] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbab, C. Pan, Z. Yi, G. Qu, K. Kitani, J. Hodgins, L. J. Fan, Y. Zhu, C. Liu, and G. Shi, "Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills," 2025.

[12] C. Lu, X. Cheng, J. Li, S. Yang, M. Ji, C. Yuan, G. Yang, S. Yi, and X. Wang, "Mobile-television: Predictive motion priors for humanoid whole-body control," 2025.

[13] H. Zhang, Z. Chen, H. Xu, L. Hao, X. Wu, S. Xu, R. Xiong, and Y. Wang, "Unified cross-structural motion retargeting for humanoid characters," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–14, 2024.

[14] S. Tak and H.-S. Ko, "A physically-based motion retargeting filter," *ACM Trans. Graph.*, vol. 24, no. 1, p. 98–117, jan 2005.

[15] J. Zhang, J. Weng, D. Kang, F. Zhao, S. Huang, X. Zhe, L. Bao, Y. Shan, J. Wang, and Z. Tu, "Skinned motion retargeting with residual perception of motion semantics & geometry," 2023.

[16] K. Aberman, P. Li, D. Lischinski, O. Sorkine-Hornung, D. Cohen-Or, and B. Chen, "Skeleton-aware networks for deep motion retargeting," *ACM Transactions on Graphics*, vol. 39, no. 4, Aug. 2020.

[17] B. Delhaisse, D. Esteban, L. Rozo, and D. Caldwell, "Transfer learning of shared latent spaces between robots with similar kinematic structure," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 4142–4149.

[18] K. Ayusawa and E. Yoshida, "Motion retargeting for humanoid robots based on simultaneous morphing parameter identification and motion optimization," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1343–1357, 2017.

[19] H. Zhang, W. Li, J. Liu, Z. Chen, Y. Cui, Y. Wang, and R. Xiong, "Kinematic motion retargeting via neural latent optimization for learning sign language," *IEEE Robotics and Automation Letters*, 2022.

[20] K. Darvish, L. Penco, J. Ramos, R. Cisneros, J. Pratt, E. Yoshida, S. Ivaldi, and D. Pucci, "Teleoperation of humanoid robots: A survey," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1706–1727, 2023.

[21] J. Z. Zhang, S. Yang, G. Yang, A. L. Bishop, D. Ramanan, and Z. Manchester, "Slomo: A general system for legged robot motion imitation from casual videos," 2023.

[22] J. Ramos and S. Kim, "Dynamic locomotion synchronization of bipedal robot and human operator via bilateral feedback teleoperation," *Science Robotics*, vol. 4, no. 35, p. eaav4282, 2019.

[23] Z. Luo, J. Cao, A. Winkler, K. Kitani, and W. Xu, "Perpetual humanoid control for real-time simulated avatars," 2023.

[24] Y. Yuan and K. M. Kitani, "Residual force control for agile human behavior imitation and extended motion synthesis," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[25] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control," 2024.

[26] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," 2021.

[27] H. Shao, S. Yao, D. Sun, A. Zhang, S. Liu, D. Liu, J. Wang, and T. Abdelzaher, "Controlvae: Controllable variational autoencoder," 2020.

[28] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," 2018.

[29] H. Zhang, W. Li, J. Liu, Z. Chen, Y. Cui, Y. Wang, and R. Xiong, "Kinematic motion retargeting via neural latent optimization for learning sign language," 2022.

[30] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021.

[31] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2018.

[32] O. Taheri, N. Ghorbani, M. J. Black, and D. Tzionas, *GRAB: A Dataset of Whole-Body Human Grasping of Objects*. Springer International Publishing, 2020, pp. 581–600.

[33] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, "AMASS: Archive of motion capture as surface shapes," in *International Conference on Computer Vision*, Oct. 2019, pp. 5442–5451.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.