Articulated-Body Dynamics Network: Dynamics-Grounded Prior for Robot Learning

***Abstract*—Recent work in reinforcement learning has shown that incorporating structural priors for articulated robots, such as link connectivity, into policy networks improves learning efficiency. However, dynamics properties, despite their fundamental role in determining how forces and motion propagate through the body, remain largely underexplored as an inductive bias for policy learning. To address this gap, we present the Articulated-Body Dynamics Network (ABD-NET), a novel graph neural network architecture grounded in the computational structure of forward dynamics. Specifically, we adapt the inertia propagation mechanism from the Articulated Body Algorithm, systematically aggregating inertial quantities from child to parent links in a tree-structured manner, while replacing physical quantities with learnable parameters. Embedding ABD-NET into the policy actor enables dynamics-informed representations that capture how actions propagate through the body, leading to efficient and robust policy learning. Through experiments with simulated humanoid, quadruped, and hopper robots, our approach demonstrates increased sample efficiency and generalization to dynamics shifts compared to transformer-based and GNN baselines. We further validate the learned policy on real Unitree G1 and Go2 robots, state-of-the-art humanoid and quadruped platforms, generating dynamic, versatile and robust locomotion behaviors through sim-to-real transfer with real-time inference.**

I. INTRODUCTION

Across machine learning domains, incorporating domain-specific architectural priors has been a key driver of success. For example, convolutional neural networks exploit translation equivariance in images, while transformers' attention mechanisms capture long-range dependencies in sequential data. In reinforcement learning (RL) for robots, prior work has explored incorporating the robot's spatial structure into policy networks. Graph neural networks (GNNs) were a natural choice [1], [2], and more recently, transformer-based architectures have emerged as a competitive alternative by combining attention mechanisms with structural information [3]–[5]. These approaches leverage the geometric structure of the robot to impose message-passing order in GNNs, or define masking schemes and biases in attention mechanisms.

Despite their success, existing methods primarily use spatial structure to specify which components of the robot should exchange information (e.g., via adjacency or attention masks), but how information is aggregated and transformed between connected components carries no physical semantics and must be learned solely from data. However, articulated systems possess a richer structure than mere connectivity: inertial quantities propagate from child to parent links along the kinematic tree, determining how each link's mass affects the motion of the entire chain. This propagation structure, which fundamentally determines how rigid bodies move, is precisely what forward dynamics algorithms compute [6]–[9]. This raises a natural question:

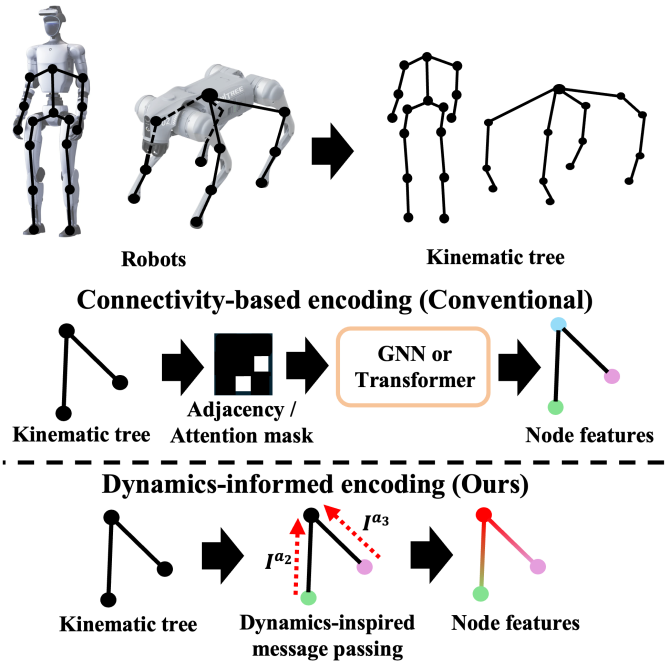

Fig. 1. Different approaches to computing node features for articulated robots. Conventional methods use link connectivity to define information flow via adjacency or attention mask, leaving the network to learn how to form node features from scratch. Our approach encodes the computational structure of forward dynamics by performing dynamics-inspired message passing, where learnable inertia-related quantities (denoted as I^a) are propagated and aggregated from children to parents to form node features.

Can the computational structure of forward dynamics, embedded directly into the policy architecture, serve as an effective inductive bias for learning control policies?

In this work, we investigate this question and answer it affirmatively by proposing **A**rticulated **B**ody **D**ynamics **Net**work ABD-NET, a graph neural network architecture that embeds the computational structure of forward dynamics into a policy actor network while replacing exact physical quantities with learnable representations (Fig. 1). Specifically, ABD-NET propagates features from child to parent links, mirroring how inertial effects accumulate in the Articulated Body Algorithm [9]. The resulting architecture not only provides an inductive bias rooted in dynamics, but also retains the flexibility to go beyond exact physical models. By structuring representations according to how physical quantities accumu-

late throughout the body, the actor can more directly infer how joint torques influence global motion, enabling effective and coordinated control actions. This design distinguishes ABD-NET from prior policy architectures [1]–[4], which leverage the geometric connectivity of links but do not encode how features propagate along the kinematic tree.

While embedding forward dynamics into a neural network as a next-state predictor for model-based RL is natural, our objective differs: we investigate whether the forward-dynamics structure in the actor itself can improve policy learning efficiency, rather than building an accurate dynamics model. We therefore consider model-free learning and embed the proposed structure directly into the actor network. This approach is compatible with the dominant paradigm in learning-based control, in which on-policy, model-free methods leverage massively parallel simulation. We evaluate ABD-NET on humanoid, quadruped, and hopper robots, demonstrating that our approach outperforms baselines in sample efficiency, generalization, and computational efficiency.

The main contributions of our work are as follows. ***(i)*** We propose ABD-NET, a novel graph neural network architecture that embeds the computational structure of forward dynamics into a policy actor. ***(ii)*** We provide reformulations of forward dynamics inertia propagation that avoid expensive matrix operations, enabling computationally efficient training. ***(iii)*** We empirically show that ABD-NET achieves superior sample efficiency, robustness under dynamics shifts, and computational efficiency across diverse morphologies and tasks. We further validate the learned policy through sim-to-real transfer on real robots, including Unitree G1 and Go2.

II. RELATED WORKS

A. Exploiting Body Structure for Policy Learning

GNNs have been adopted as policy architectures for robot control [1], [2], [10], [11]. A common formulation represents each link and joint as graph nodes connected by edges that follow the robot's kinematic tree and performs learned message passing between physically adjacent components, leaving the semantics of feature aggregation to be inferred from data. Building on this framework, NerveNet [1] applies GNNs to design general RL actors, DGN [10] extends the idea to modular RL by allowing dynamically assembled actors, and SMP [2] employs GNNs for morphology-agnostic policy learning. More recently, transformer-based actors have emerged as a strong alternative. With variable context length, transformers naturally support fully connected graphs and have been shown to outperform GNNs even without explicit morphological priors [5], [12]. Subsequent works inject morphological knowledge through a traversal-order-based attention bias [4] or connectivity-based attention masking [3]. More recently, MS-PPO [13] encodes morphological symmetries, such as bilateral reflection, directly into a GNN policy to enforce equivariant behavior across symmetric gaits of quadrupeds.

While also leveraging body structure, ABD-NET goes beyond connectivity by encoding the computational structure of forward dynamics, providing an explicit inductive bias for how features are transformed between connected links, applicable to arbitrary articulated morphologies.

B. Physics-Informed Reinforcement Learning

Physics-informed RL aims to incorporate physical structures or priors into the policy learning process [14]. Such inductive biases can improve the sample efficiency of RL algorithms and enhance the safety of learned policies [15]–[17]. For example, Ramesh and Ravindran [18] employ Lagrangian Neural Networks [19] to learn the inertia tensor, which is then used with a numerical integrator to predict the next state, given analytically computed Coriolis and gravitational forces. Rodrigues Network [20] is conceptually closest to our work in that it builds a neural operator from a classical robotics computation, by parameterizing the Rodrigues rotation formula within a transformer-style architecture. While it demonstrates effectiveness in motion prediction and imitation learning, its extension to reinforcement learning remains unexplored.

ABD-NET similarly incorporates physics-based inductive bias into the network architecture, but focuses on forward dynamics and applies this structure directly to policy learning in model-free RL, allowing the actor to implicitly learn how its actions propagate through the body.

III. PRELIMINARIES

A. Reinforcement Learning

We consider Markov decision processes defined by $(\mathcal{S}, \mathcal{A}, p, r, d_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action space, $p(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$ the transition density, $r(\mathbf{s}, \mathbf{a})$ the reward, d_0 the initial state distribution, and $\gamma \in [0, 1)$ the discount factor. The RL objective is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected sum of discounted rewards $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$.

B. Robot Morphology and Graph Neural Networks

We represent an articulated robot with K links as a tree graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node $v_i \in \mathcal{V}$ for $i \in \{0, \dots, K-1\}$ represents a link of the robot, and there is an edge $(v_i, v_j) \in \mathcal{E}$ if link i and link j are connected by a joint. Designating node v_0 as the root (e.g., the torso or base link) induces a parent-child hierarchy: for each node v_i, let $\text{CH}(i)$ denote the set of its children, and for each non-root node i, let $\text{PA}(i)$ denote its unique parent.

In GNNs, each node v_i is associated with a representation vector $\mathbf{v}_i$, which we refer to as the *link representation* to reflect its association with a specific link in the robot's morphology. GNNs update these representations through message aggregation:

$$\mathbf{m}_i \leftarrow \sigma\left(\{\mathbf{v}_j : j \in \mathcal{N}_i\}\right), \quad \forall i \in \{0, \cdots, K-1\} \tag{1}$$

$$\mathbf{v}_i \leftarrow f_\theta(\mathbf{v}_i, \mathbf{m}_i), \quad \forall i \in \{0, \cdots, K-1\} \tag{2}$$

where $\mathcal{N}_i = \{\text{PA}(i)\} \cup \text{CH}(i)$ denotes the neighborhood of node v_i, σ is an aggregation function (e.g., sum or mean), and f_θ is learned functions that define how neighboring nodes communicate and update their representations.

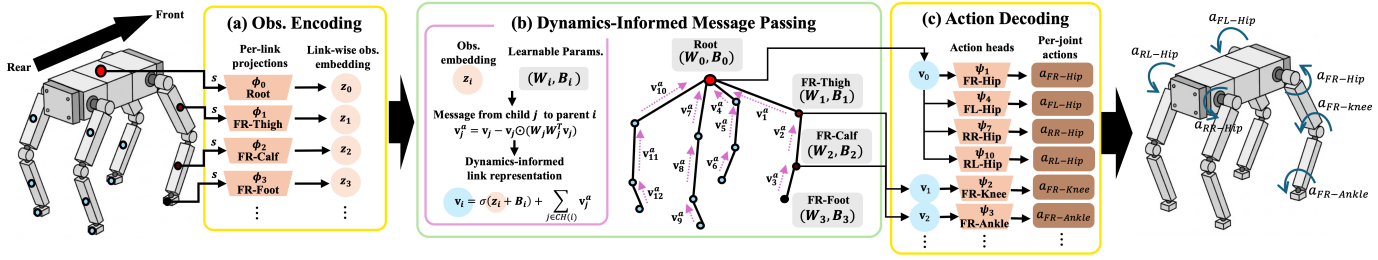

Fig. 2. Overview of ABD-NET on a quadruped robot. **(a) Observation Encoding:** Each link i has its own projection ϕ_i that transforms the observation $\mathbf{s}$ into a link-wise observation embedding $\mathbf{z}_i$ (e.g., FR-Thigh, FR-Calf, FR-Foot for the front-right leg). **(b) Dynamics-Informed Message Passing:** Each link i is associated with learnable parameters $(\mathbf{W}_i, \mathbf{B}_i)$. Link j computes a contribution $\mathbf{v}_j^a$ using $\mathbf{W}_j$ and sends it to its parent, which aggregates contributions from all children to form its link representation $\mathbf{v}_i$ (σ denotes softplus). **(c) Action Decoding:** For joint j connecting $\text{PA}(j)$ and link j, the action $\mathbf{a}_j$ is computed from the parent's representation $\mathbf{v}_{\text{PA}(j)}$.

C. Forward Dynamics

The forward dynamics problem is to compute link accelerations given applied forces. The Articulated Body Algorithm (ABA) [9] models the dynamics of each link as $f_i = I_i^A \alpha_i + b_i^A$, where f_i is the spatial force acting on link i and α_i is its spatial acceleration. Here, I_i^A is the articulated-body inertia, which represents the effective inertia of link i accounting for all dynamic effects from its subtree. The term b_i^A is the bias force, which captures velocity-dependent effects, such as Coriolis and centrifugal forces accumulated from the subtree. Notably, once I_i^A and b_i^A are known, one can compute the acceleration of link i without requiring explicit knowledge of each descendant link. In this sense, I_i^A and b_i^A serve as a compact representation that encapsulates the dynamic effects of the entire subtree rooted at link i.

ABA computes I_i^A and b_i^A recursively from leaves to root. Specifically, for each link i, the articulated-body inertia I_i^A is obtained by combining its own rigid-body inertia I_i with the contributions from its children:

$$I_i^A = I_i + \textstyle\sum_{j \in \text{CH}(i)} I_j^a. \tag{3}$$

Here, I_j^a is the contribution from child j:

$$I_j^a = I_j^A - I_j^A S_j (S_j^\top I_j^A S_j)^{-1} S_j^\top I_j^A, \tag{4}$$

where S_j is the motion subspace matrix of the joint connecting link j to its parent, whose columns span the directions of motion permitted by the joint. Eq. (4) subtracts inertia along the joint-permitted directions from the child's articulated-body inertia, so that only the constrained portion propagates to the parent. The bias force b_i^A is computed similarly.

IV. Our Approach

Architecture overview. In ABD-NET, the parameterized actor network $\pi_\theta : \mathcal{S} \to \mathcal{A}$ consists of three modules:

$$\pi_\theta = \Psi \circ \mathcal{M} \circ \Phi, \tag{5}$$

where $\Phi : \mathcal{S} \to \mathbb{R}^{K \times d}$ encodes the observation into link-wise embeddings $\{\mathbf{z}_i\}_{i=0}^{K-1}$, $\mathcal{M} : \mathbb{R}^{K \times d} \to \mathbb{R}^{K \times d}$ performs dynamics-informed message passing to produce link representations $\{\mathbf{v}_i\}_{i=0}^{K-1}$, and $\Psi : \mathbb{R}^{K \times d} \to \mathcal{A}$ decodes joint actions from link representations. We assume each non-root link is connected to its parent via an actuated joint, and the action specifies the control signal (e.g., target position or torque) for each joint. Fig. 2 illustrates the overall architecture of ABD-NET.

A. Link-wise Observation Encoding

Given observation $\mathbf{s}$, the observation encoder Φ transforms $\mathbf{s}$ into link-wise embeddings $\{\mathbf{z}_i\}_{i=0}^{K-1}$. Φ consists of per-link projections $\{\phi_i\}_{i=0}^{K-1}$, in which each $\mathbf{z}_i = \phi_i(\mathbf{s})$ is computed independently for link i.

B. Dynamics-Informed Message Passing

Given observation embeddings $\{\mathbf{z}_i\}$, the message passing module $\mathcal{M}$ transforms them into link representations $\{\mathbf{v}_i\}$ by exploiting the computational structure of ABA. To this end, we associate each link i with two learnable parameters: $\mathbf{B}_i \in \mathbb{R}^d$, a base feature analogous to the rigid-body inertia I_i in Eq. (3); $\mathbf{W}_i \in \mathbb{R}^{d \times d}$, a motion basis analogous to the motion subspace S_j in Eq. (4).

Bottom-Up message passing. $\mathcal{M}$ aggregates messages only from children $\text{CH}(i)$, mirroring the leaf-to-root computation of ABA in Eq. (3). Specifically, the message $\mathbf{m}_i$ is defined as:

$$\mathbf{m}_i = \textstyle\sum_{j \in \text{CH}(i)} \mathbf{v}_j^a, \tag{6}$$

where $\mathbf{v}_j^a$ is the contribution from child j, analogous to I_j^a in Eq. (4). The link representation is then computed as:

$$\mathbf{v}_i = \text{softplus}(\mathbf{z}_i + \mathbf{B}_i) + \mathbf{m}_i. \tag{7}$$

We apply softplus to ensure positivity, reflecting the positive-definiteness of rigid-body inertia.

Child contribution. We translate Eq. (4) into a learnable form by replacing the articulated-body inertia I_j^A with $\text{diag}(\mathbf{v}_j)$ and the motion subspace S_j with a learnable motion basis $\mathbf{W}_j$. To avoid the numerically unstable inverse, we assume $(\mathbf{W}^\top \text{diag}(\mathbf{v}) \mathbf{W})^{-1} \approx \mathbf{I}$, which holds when $\mathbf{W}$ has orthonormal columns weighted by $\mathbf{v}$. Retaining the projection structure $\mathbf{W}_j \mathbf{W}_j^\top$, this yields:

$$\mathbf{v}_j^a = \mathbf{v}_j - \mathbf{v}_j \odot (\mathbf{W}_j \mathbf{W}_j^\top \mathbf{v}_j), \tag{8}$$

where $\odot$ is element-wise multiplication.

Algorithm 1: Forward pass of ABD-NET

Require: Observation $\mathbf{s}$
Ensure: Action $\mathbf{a}$

/* Link-wise Observation Encoding (Sec. IV-A) */
$\mathbf{z}_i \leftarrow \phi_i(\mathbf{s})$ for all $i \in \{0, \dots, K-1\}$
/* Dynamics-Informed Message Passing (Sec. IV-B) */
$\mathbf{m}_i \leftarrow \mathbf{0}$ for all $i \in \{0, \dots, K-1\}$
for i in leaf-to-root traversal **do**
  $\mathbf{v}_i \leftarrow \text{softplus}(\mathbf{z}_i + \mathbf{B}_i) + \mathbf{m}_i$
  if $i \neq 0$ **then**
    $\mathbf{v}_i^a = \mathbf{v}_i - \mathbf{v}_i \odot (\mathbf{W}_i \mathbf{W}_i^\top \mathbf{v}_i)$
    $\mathbf{m}_{\text{PA}(i)} \leftarrow \mathbf{m}_{\text{PA}(i)} + \mathbf{v}_i^a$
  end if
end for
/* Action Decoding (Sec. IV-C) */
return $\mathbf{a} = \{\psi_i(\mathbf{v}_{\text{PA}(i)})\}_{i=1}^{K-1}$

Overall, our formulation in Eq. (6)–(8) explicitly encodes the computational structure through which physical quantities propagate in ABA. The term $\mathbf{W}_j \mathbf{W}_j^\top$ in Eq. (8) serves as a learned approximation of the constraint elimination mechanism in Eq. (4), identifying feature directions to attenuate during child-to-parent propagation, analogous to how ABA removes inertia along joint-permitted directions. To maintain this projection structure while allowing adaptability, we introduce an auxiliary loss.

Orthogonality constraint. As described above, we approximate the inverse term $(\mathbf{W}^\top \text{diag}(\mathbf{v}) \mathbf{W})^{-1}$ as identity to avoid numerical instability. To encourage this approximation to hold, we regularize $\mathbf{W}_i$ via an auxiliary loss:

$$\mathcal{L}_{\text{orth}} = \frac{1}{K} \sum_{i=0}^{K-1} \|\mathbf{W}_i^\top \text{diag}(\mathbf{v}_i) \mathbf{W}_i - \mathbf{I}\|_F^2. \tag{9}$$

As a soft constraint, this loss guides the network toward structural correspondence to ABA while allowing $\mathbf{W}_j \mathbf{W}_j^\top$ to deviate when doing so benefits policy learning, balancing structural fidelity with task-specific adaptation. $\mathcal{L}_{\text{orth}}$ is added to the PPO objective during training.

C. Action Decoding

Given link representations $\{\mathbf{v}_i\}_{i=0}^{K-1}$, the action decoder Ψ outputs the action for each joint. Ψ consists of per-joint action heads $\{\psi_j\}_{j=1}^{K-1}$, one for each actuated joint. We define joint j as the joint connecting link j to its parent $\text{PA}(j)$. Since each link representation $\mathbf{v}_i$ encapsulates the dynamic effects aggregated from its descendants, we use $\mathbf{v}_{\text{PA}(j)}$ to generate the action for joint j. Specifically, the action for joint j is computed as:

$$\mathbf{a}_j = \psi_j(\mathbf{v}_{\text{PA}(j)}) \in \mathbb{R}^{n_j}, \tag{10}$$

where n_j is the degrees of freedom of joint j. The complete forward pass of ABD-NET is outlined in Algorithm 1.

V. EXPERIMENTS

We evaluate ABD-NET in both simulation and real-world settings. In simulation, we assess sample efficiency and generalization to dynamics shifts, and computational efficiency across diverse morphologies and tasks using two physics backends (Sec V-A). We then validate the learned policy on real Unitree G1 and Go2 robots, demonstrating robust sim-to-real transfer with real-time onboard inference (Sec V-B).

TABLE I
REWARD FUNCTIONS AND WEIGHTS FOR GENESIS ENVIRONMENTS. (↓) DENOTES PENALTY TERMS.

Category	Reward Term	Go1	Go2	G1	T1
Vel. Track.	Lin. vel. tracking	1.5	1.0	1.0	1.0
	Ang. vel. tracking	0.75	0.2	0.5	0.5
Base Reg.	Base height	12.5	50.0	10.0	20.0
	Vertical lin. vel. (↓)	2.0	1.0	2.0	1.0
	Ang. vel. xy (↓)	0.025	–	0.05	0.025
Joint Reg.	Action rate (↓)	0.02	0.005	0.01	0.1
	Default pose (↓)	0.1	0.1	–	0.1
	Torques (↓)	1e-5	1e-5	1e-5	5e-5
	Joint vel. (↓)	–	–	1e-3	1e-4
	Joint pos. limits (↓)	–	–	5.0	–
Orientation	Orientation (↓)	0.2	0.2	0.1	0.1
	Hip deviation (↓)	0.2	–	1.0	–
Foot Reg.	Feet height	0.05	–	–	–
	Feet air time	1.5	–	–	–
	Feet slip (↓)	–	–	0.2	0.1
	Feet roll (↓)	–	–	–	0.1
	Feet yaw diff. (↓)	–	–	1.0	1.0
	Feet yaw mean dev. (↓)	–	–	1.0	1.0
	Feet distance (↓)	–	–	1.0	1.0
Other	Invalid contact (↓)	1.0	–	–	1.0
	Gait pattern	–	–	0.18	6.0
	Alive bonus	–	–	0.15	–
	Base acc. (↓)	–	–	1e-4	1e-4

A. Simulation Experiments

Environments. To evaluate ABD-NET across different physics backends, we use two simulators: Genesis [21] and SAPIEN [22]. In Genesis, we consider Booster T1, Unitree G1 (humanoid), Go1, and Go2 (quadruped) for velocity tracking tasks. In SAPIEN, we use the MuJoCo Humanoid and Hopper from ManiSkill [23]. Humanoid tasks include Walk (moving along a target direction), Stand (rising from a random initial pose), and Run (moving forward at high speed). Hopper tasks include Hop (moving forward by hopping while staying upright) and Stand (remaining stationary and balanced).

Implementation. Observations in Genesis tasks include proprioceptive information (joint positions and velocities, IMU angular velocities, projected gravities), velocity command signals, previous actions, and base linear velocities. For G1 and T1, we additionally include gait phase encoding and foot positions relative to the base. For SAPIEN tasks, we use the default observation spaces from ManiSkill: Humanoid observations include joint positions and velocities, actuator forces, and external forces, while Hopper observations consist of joint positions and velocities. For rewards, T1 and G1 use reward functions adapted from Booster Robotics and Unitree Gym, respectively. Go1 and Go2 use reward functions adapted from IsaacLab and Genesis, respectively. SAPIEN tasks use the default ManiSkill reward functions. Detailed reward functions and weights for Genesis tasks are summarized in Table I.

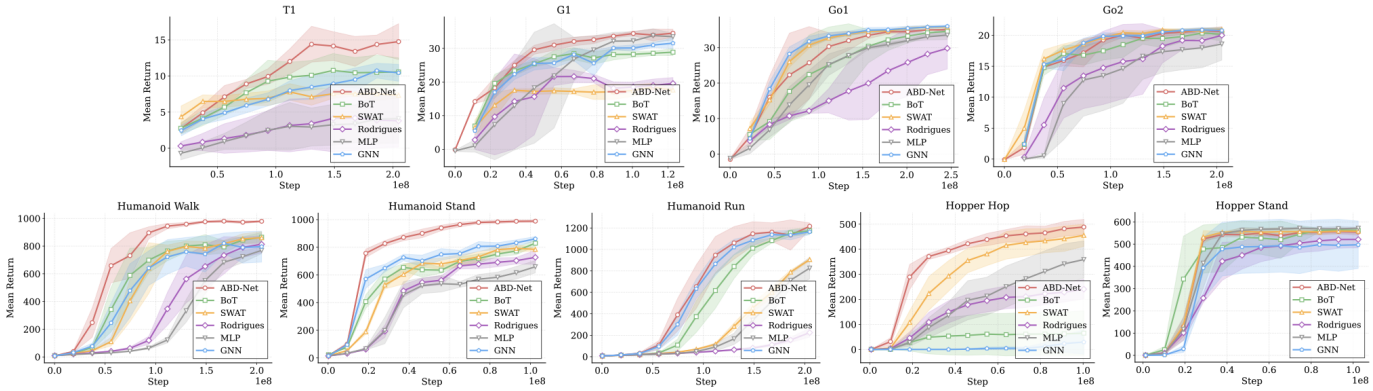

Fig. 3. Learning curves comparing ABD-Net (ours) and baselines: mean return versus the number of environment steps. All methods are trained with 5 seeds. Shaded regions indicate 95% standard confidence intervals.

TABLE II
NORMALIZED FINAL PERFORMANCE ON GENESIS AND SAPIEN.

Genesis (T1 + G1 + Go1 + Go2)			
Method	IQM	Median	Mean
ABD-NET	**0.85** (0.73, 0.95)	**0.86** (0.70, 0.97)	**0.83** (0.75, 0.91)
BOT	0.69 (0.56, 0.85)	0.67 (0.54, 0.93)	0.70 (0.58, 0.82)
SWAT	0.79 (0.50, 0.94)	0.83 (0.39, 0.95)	0.75 (0.57, 0.89)
RODRIGUES	0.45 (0.17, 0.71)	0.45 (0.04, 0.81)	0.45 (0.25, 0.65)
GNN	0.76 (0.65, 0.85)	0.76 (0.65, 0.87)	0.74 (0.67, 0.82)
MLP	0.34 (0.07, 0.72)	0.32 (0.05, 0.78)	0.40 (0.19, 0.63)
SAPIEN (Humanoid + Hopper)			
Method	IQM	Median	Mean
ABD-NET	**0.97** (0.94, 0.98)	**0.97** (0.96, 0.98)	**0.94** (0.91, 0.97)
BOT	0.66 (0.47, 0.80)	0.67 (0.50, 0.88)	0.59 (0.47, 0.70)
SWAT	0.71 (0.62, 0.80)	0.71 (0.67, 0.81)	0.70 (0.64, 0.77)
RODRIGUES	0.44 (0.31, 0.51)	0.45 (0.35, 0.54)	0.41 (0.32, 0.49)
GNN	0.73 (0.55, 0.87)	0.76 (0.58, 0.92)	0.64 (0.51, 0.76)
MLP	0.56 (0.41, 0.70)	0.57 (0.39, 0.68)	0.55 (0.44, 0.65)

TABLE III
MASS GENERALIZATION PERFORMANCE (RETENTION, %).
N/C INDICATES THAT THE TRAINING POLICY DID NOT CONVERGE.

Algorithm	Humanoid	Hopper	Go2	T1
ABD-NET	**91.1 ± 1.4**	**62.4 ± 0.5**	**82.4 ± 2.1**	**81.1 ± 3.5**
BOT	17.0 ± 0.6	N/C	69.7 ± 2.3	57.0 ± 2.1
SWAT	88.9 ± 2.0	29.7 ± 1.0	81.4 ± 2.2	53.4 ± 1.6
RODRIGUES	28.0 ± 1.7	9.1 ± 0.4	53.7 ± 3.1	N/C
GNN	54.9 ± 0.8	N/C	82.9 ± 2.2	69.9 ± 2.4
MLP	78.4 ± 2.9	55.4 ± 2.1	66.5 ± 2.7	N/C

To implement ABD-NET, we represent each robot as a kinematic tree extracted from its URDF or MJCF model file, parsing parent-child relationships between links while excluding sensor-related components (e.g., camera, IMU) to focus on the physical structure relevant to control.

Baselines. To investigate how architectural inductive biases affect policy learning, we compare ABD-NET with the following actor designs. **BOT** [3] is a transformer that uses link connectivity as an attention mask, alternating between masked and unmasked attention; **SWAT** [4] is a transformer that uses tree-traversal-based positional embeddings and graph-based attention biases; **RODRIGUES** [20] encodes the parent-to-child transformation in forward kinematics by extending the Rodrigues rotation formula into a learnable operator; **GNN** is a standard graph neural network that aggregates messages from all neighbors including both parent and children; **MLP** is a standard multi-layer perceptron. All methods, including ABD-NET, are trained with PPO [24] using an identical MLP value network, with comparable parameters.

Overall performance. Fig. 3 compares the learning curves of ABD-NET against baselines (BOT, SWAT, RODRIGUES, GNN, MLP) in terms of average training return. ABD-NET consistently outperforms baselines in both sample efficiency and final performance, except on Go1/Go2 and Hopper Stand, where it achieves results comparable to other methods. Importantly, the performance gap increases for more complex morphologies and tasks requiring dynamic behavior. In contrast, simpler morphologies with quasi-static tasks (Go1/Go2 velocity tracking, Hopper Stand) show smaller gains, suggesting that dynamics-aware inductive biases become more critical as task complexity increases.

Table II summarizes the normalized final performance across both simulators. ABD-NET achieves the highest IQM in both Genesis and SAPIEN, outperforming the strongest baseline SWAT by 7.6% and 36.6%, respectively. We attribute this performance gain to the dynamics-informed message passing, which explicitly encodes how inertial effects propagate throughout the body, enabling more efficient learning of coordinated control.

Mass generalization. Table III evaluates the robustness of learned policies to dynamics shifts. Specifically, we deploy trained policies on modified morphologies with increased base mass: 1.5–2.0× for MuJoCo Humanoid, Go2, and Hopper, and 1.1–1.5× for T1. We report the retention rate (percentage of final performance) with 95% confidence interval computed from 500 evaluation episodes. ABD-NET achieves the highest retention rate across all morphologies, with an average improvement of 23.9% over the strongest baseline, SWAT.

Fig. 4 illustrates this difference qualitatively on the Hopper Hop task with 2× base mass. When the increased mass causes the body to tilt downward, ABD-NET successfully applies a stronger corrective torque to complete the forward rotation and recover balance, while SWAT fails to generate sufficient force and falls backward. This robustness arises from the dynamics-

Fig. 4. Recovery behavior under 2× mass on Hopper Hop. Top: ABD-NET applies stronger torque to recover from the downward tilt. Bottom: SWAT fails to compensate and falls.

informed architecture of ABD-NET, which explicitly constrains feature propagation to follow inertial accumulation in forward dynamics. As a result, the policy leverages relative propagation structure rather than specific parameter values, yielding robustness to dynamics mismatch.

Learned link representations. To examine whether the learned representations capture meaningful structure, we visualize the link representations of the hip joints (FL, FR, RL, RR) from a trained Go2 policy during locomotion in Fig. 5.

The learned policy exhibits a trot gait, where diagonal leg pairs (FL-RR, FR-RL) move in synchrony. The left plot shows that feature norms of diagonal pairs oscillate together over time. The right plot confirms this pattern through correlation analysis: each hip joint shows the highest correlation (excluding itself) with its diagonal counterpart (e.g., FL with RR, FR with RL), with RR_hip being a minor exception. This correspondence indicates that the dynamics-informed message passing in ABD-NET learns link representations that capture physically meaningful relationships between body parts while remaining effective for task performance.

Extension to dynamics modeling. While ABD-NET is designed for model-free policy learning, its architecture naturally extends to model-based settings. To examine this, we train a dynamics model based on ABD-NET and an MLP baseline to predict next observations given current observations and actions, i.e., $(\mathbf{s}, \mathbf{a}) \mapsto \mathbf{s}'$. As shown in Fig. 6, the ABD-NET-based model achieves 10× lower validation loss than the MLP baseline in multi-step rollout prediction on both Double Pendulum (contact-free) and Hopper (contact-involved) environments. This result demonstrates that the ABA-inspired architecture learns representations that capture physically meaningful state transitions, explaining its effectiveness in policy learning while also suggesting potential for model-based extensions such as model-based RL.

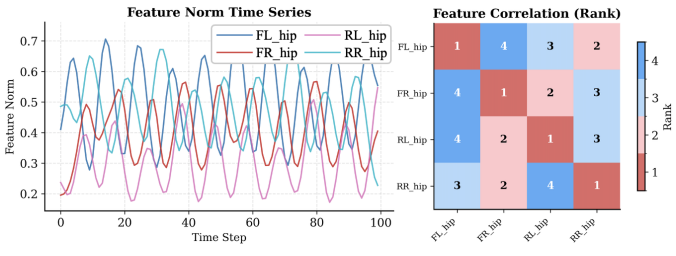

Fig. 5. Learned link representations on Go2 during trot gait. Left: Feature norm time series of hip joints (FL: front-left, RR: rear-right, etc.). Right: Correlation rank matrix between hip joint features, where lower rank indicates higher correlation.

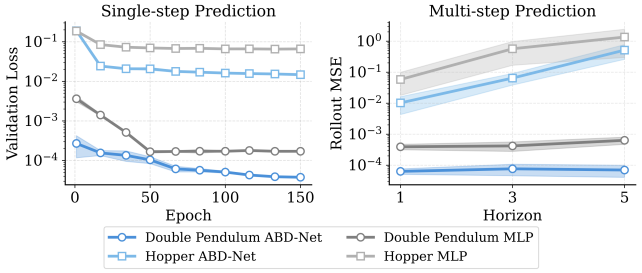

Fig. 6. Comparison of ABD-NET and MLP as dynamics models. Left: Single-step validation loss on Double Pendulum and Hopper. Right: 1, 3, 5-step rollout prediction error.

Ablation studies. We conduct ablation studies to isolate the effect of each architectural component.

Effect of dynamics-grounded constraints. To verify the contributions of the two key components in ABD-NET—the structured projection $\mathbf{W}\mathbf{W}^\top$ with orthogonality constraint and the bottom-up message passing—we consider two ablations: (1) *w/o orth.*, which replaces $\mathbf{W}_j\mathbf{W}_j^\top$ in Eq. (8) with an unconstrained matrix and removes $\mathcal{L}_{\text{orth}}$ in Eq. (9); (2) GNN, a graph neural network. Fig. 7 compares learning curves, and Table IV reports mass generalization performance.

As shown, while *w/o orth.* achieves comparable training performance to ABD-NET on most tasks, the orthogonality constraint stabilizes learning in some cases, such as Hopper Hop. Moreover, as shown in Table IV, the orthogonality constraint significantly improves generalization under dynamics shifts. These results suggest that constraining policy structure to follow the propagation principles of forward dynamics leads to more reliable behavior under dynamics shifts. GNN fails

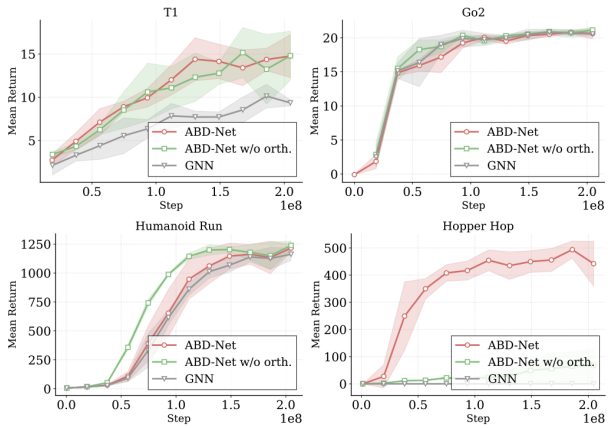

Fig. 7. Learning curves comparing ABD-NET, ABD-NET w/o orth., and GNN across four tasks

TABLE IV
MASS GENERALIZATION PERFORMANCE

Algorithm	Humanoid	Hopper	Go2	T1
ABD-NET	**91.1 ± 1.4**	**62.4 ± 0.5**	**96.2 ± 0.6**	**81.1 ± 3.5**
W/O ORTH.	59.8 ± 0.4	56.3 ± 0.8	77.4 ± 2.2	51.6 ± 2.2
GNN	54.9 ± 0.8	N/C	82.9 ± 2.2	69.9 ± 2.4

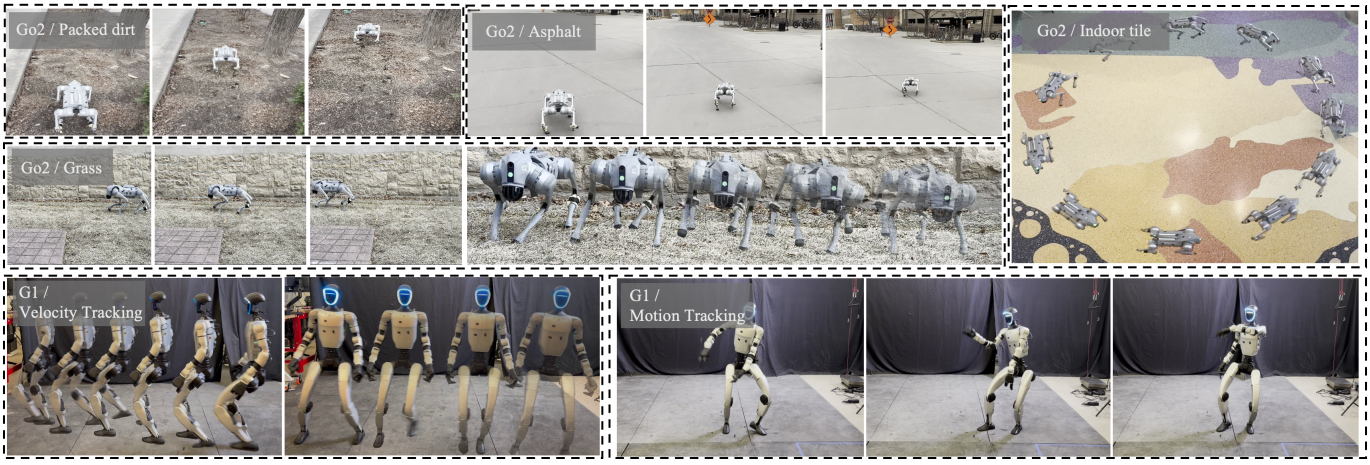

Fig. 8. Sim-to-real transfer on Unitree Go2 and G1 robots. Top two rows: Go2 velocity tracking across diverse terrains (packed dirt, asphalt, grass, indoor tile), with composite motion sequences showing lateral walking on grass and yaw tracking on indoor tile. Bottom: G1 velocity tracking with composite motion sequences showing forward and lateral walking (left) and motion tracking of a dance sequence (right).

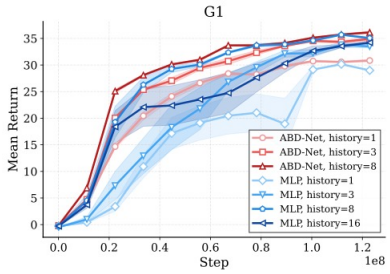

Fig. 9. Effect of observation history length on G1. ABD-NET and MLP are compared across history lengths $\in \{1, 3, 8, 16\}$.

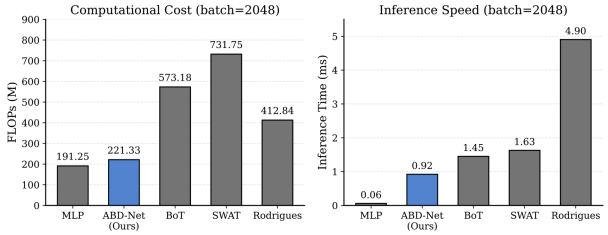

Fig. 10. Left: FLOPs per forward pass. Right: Inference time. All measured on an RTX 4090 with batch size 2048.

to converge on T1 and Hopper Hop, indicating that bottom-up message passing aligned with ABA provides a stronger inductive bias than bidirectional propagation.

Observation history length. To assess whether the dynamics-informed architecture provides a more efficient prior than simply extending the observation history, we vary the history length $\in \{1, 3, 8, 16\}$ for both ABD-NET and MLP on the G1 task (Fig. 9). MLP improves with longer history ($1 \rightarrow 3 \rightarrow 8$), but degrades at 16 due to the increased input dimensionality, indicating sensitivity to history length that may require task-specific tuning. By contrast, ABD-NET consistently outperforms MLP at each history length and exhibits smaller performance degradation at higher history lengths despite the increased input dimensionality. These results suggest that the dynamics-informed architecture provides a structural prior that efficiently captures dynamics information with short histories, reducing the need for history length tuning.

Computational efficiency. Fig. 10 compares computational cost across methods with similar parameter counts ($\sim$91–95K) on an NVIDIA RTX 4090. To reflect the demands of large-scale parallel simulation training, we report forward throughput with batch size 2048. ABD-NET achieves $3\times$ lower FLOPs than transformer-based methods while maintaining faster inference time. Although ABD-NET exhibits higher latency compared to the MLP due to its sequential leaf-to-root computation, this overhead is marginal when considering the substantial gains in sample efficiency and robustness. These results demonstrate that ABD-NET strikes an effective balance between computational efficiency and task performance, making it suitable for modern parallelized RL frameworks.

B. Hardware Experiments

To verify that ABD-NET is suitable for real-world deployment, e.g., capable of real-time inference and producing stable behavior on hardware, we perform hardware experiments on Unitree G1 (humanoid) and Go2 (quadruped) robots.

Training and deployment. We train velocity tracking and motion tracking policies using MJLab [25], an RL framework built on MuJoCo [26]. For G1 and Go2, we train velocity tracking policies on flat terrain; for G1, we additionally train a motion tracking policy to evaluate more dynamic behaviors, using a dance motion sequence provided by Unitree [27]. The simulation runs at 200 Hz with a policy control frequency of 50 Hz (decimation of 4). The policy outputs desired joint positions, offset from the default standing pose. In hardware deployment, both policy inference and PD control run on the robot's onboard computer (NVIDIA Jetson Orin NX), with the policy running at 50 Hz and the PD controller at 200 Hz. During deployment, velocity commands of up to ± 1.0 m/s for forward and lateral directions and up to 0.5 rad/s for yaw rate are used. To bridge the sim-to-real gap, we apply domain randomization over foot friction, encoder bias, base center-of-mass offset, and external force perturbations.

Results. As shown in Fig. 8, the Go2 achieves robust locomotion across diverse terrains including asphalt, packed dirt, grass, and indoor tile, and successfully follows forward, lateral, and yaw commands. The G1 also demonstrates forward and lateral walking as well as dynamic dancing behavior. These results confirm that ABD-NET is compatible with standard sim-to-real pipelines and capable of generating robust and dynamic motions with real-time onboard inference across different morphologies. Furthermore, we verify that while ABD-NET incurs higher inference latency than an MLP due to its sequential computation, the worst-case onboard inference time remains under 5 ms in the G1 motion tracking tasks, well within the 20 ms budget required for 50 Hz control.

VI. CONCLUSION

In this work, we presented ABD-NET, a graph neural network architecture that embeds the computational structure of forward dynamics into the policy actor for articulated robot control. By propagating learnable features from child to parent links, mirroring how inertial quantities accumulate along the kinematic tree in the Articulated Body Algorithm, ABD-NET provides a dynamics-grounded architectural inductive bias for policy learning. Empirically, we demonstrated that ABD-NET achieves increased sample efficiency, robustness under dynamics shifts, and computational efficiency across diverse morphologies and tasks, outperforming transformer-based and GNN baselines. We further validated the learned policy through sim-to-real transfer on real Unitree Go2 and G1 robots, confirming compatibility with standard deployment pipelines and real-time onboard inference.

Limitations and future work. While the worst-case inference latency remains well within real-time control budgets (Sec. V-B), the sequential leaf-to-root computation in ABD-NET incurs higher wall-clock training time compared to an MLP (4–6$\times$ in our PyTorch implementation), though recent results with a JAX implementation have reduced this gap to approximately 2$\times$. Additionally, the current formulation operates on proprioceptive observations and does not directly handle high-dimensional sensory inputs such as images.

We plan to explore integration with model-based RL, where the dynamics-informed architecture can serve as both policy and dynamics model, potentially improving data efficiency further. We also aim to extend ABD-NET to image-based observation settings and investigate its applicability to manipulation tasks involving contact-rich interactions.

REFERENCES

[1] T. Wang, R. Liao, J. Ba, and S. Fidler, "Nervenet: Learning structured policy with graph neural networks," in *6th International Conference on Learning Representations, ICLR 2018*.

[2] W. Huang, I. Mordatch, and D. Pathak, "One policy to control them all: Shared modular policies for agent-agnostic control," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*.

[3] C. Sferrazza, D. Huang, F. Liu, J. Lee, and P. Abbeel, "Body transformer: Leveraging robot embodiment for policy learning," in *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*.

[4] S. Hong, D. Yoon, and K. Kim, "Structure-aware transformer policy for inhomogeneous multi-task reinforcement learning," in *The Tenth International Conference on Learning Representations, ICLR 2022*.

[5] A. Gupta, L. Fan, S. Ganguli, and L. Fei-Fei, "Metamorph: Learning universal controllers with transformers," in *The Tenth International Conference on Learning Representations, ICLR 2022*.

[6] D.-S. Bae and E. J. Haug, "A recursive formulation for constrained mechanical system dynamics: Part i. open loop systems," *Journal of Structural Mechanics*, vol. 15, no. 3, pp. 359–382, 1987.

[7] M. W. Walker and D. E. Orin, "Efficient dynamic computer simulation of robotic mechanisms," *Journal of Dynamic Systems, Measurement, and Control*, vol. 104, no. 3, pp. 205–211, 09 1982.

[8] K. W. Lilly and D. E. Orin, "Alternate formulations for the manipulator inertia matrix," *The International Journal of Robotics Research*, vol. 10.

[9] R. Featherstone and D. Orin, "Robot dynamics: equations and algorithms," in *Proceedings 2000 ICRA. IEEE International Conference on Robotics and Automation. Symposia Proceedings*.

[10] D. Pathak, C. Lu, T. Darrell, P. Isola, and A. A. Efros, "Learning to control self-assembling morphologies: A study of generalization via modularity," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*.

[11] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. A. Riedmiller, R. Hadsell, and P. W. Battaglia, "Graph networks as learnable physics engines for inference and control," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*.

[12] V. Kurin, M. Igl, T. Rocktäschel, W. Boehmer, and S. Whiteson, "My body is a cage: the role of morphology in graph-based incompatible control," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

[13] S. Wei, X. Chen, F. Xie, G. E. Katz, Z. Gan, and L. Gan, "MS-PPO: morphological-symmetry-equivariant policy for legged robot locomotion," *CoRR*, vol. abs/2512.00727.

[14] C. Banerjee, K. N. Thanh, C. Fookes, and M. Raissi, "A survey on physics informed reinforcement learning: Review and open problems," *Expert Syst. Appl.*, vol. 287.

[15] S. L. Jurj, D. Grundt, T. Werner, P. Borchers, K. Rothemann, and E. Möhlmann, "Increasing the safety of adaptive cruise control using physics-guided reinforcement learning," *Energies*.

[16] H. Cao, Y. Mao, L. Sha, and M. Caccamo, "Physics-regulated deep reinforcement learning: Invariant embeddings," in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria*.

[17] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*.

[18] A. Ramesh and B. Ravindran, "Physics-informed model-based reinforcement learning," in *Learning for Dynamics and Control Conference, L4DC 2023, 15-16 June 2023, Philadelphia, PA, USA*.

[19] M. D. Cranmer, S. Greydanus, S. Hoyer, P. W. Battaglia, D. N. Spergel, and S. Ho, "Lagrangian neural networks," *CoRR*, vol. abs/2003.04630.

[20] J. Zhang, H. Geng, Y. You, C. Deng, P. Abbeel, J. Malik, and L. J. Guibas, "Rodrigues network for learning robot actions," *CoRR*, vol. abs/2506.02618.

[21] G. Authors, "Genesis: A generative and universal physics engine for robotics and beyond," December 2024. [Online]. Available: https://github.com/Genesis-Embodied-AI/Genesis

[22] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su, "SAPIEN: A simulated part-based interactive environment," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[23] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T. kai Chan, Y. Gao, X. Li, T. Mu, N. Xiao, A. Gurha, V. N. Rajesh, Y. W. Choi, Y.-R. Chen, Z. Huang, R. Calandra, R. Chen, S. Luo, and H. Su, "Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai," 2025.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347.

[25] K. Zakka, Q. Liao, B. Yi, L. L. Lay, K. Sreenath, and P. Abbeel, "mjlab: A lightweight framework for gpu-accelerated robot learning," 2026. [Online]. Available: https://arxiv.org/abs/2601.22074

[26] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[27] U. Robotics, "Unitree rl lab," 2025. [Online]. Available: https://github.com/unitreerobotics/unitree_rl_lab