X-Nav: Learning Cross-Embodiment Navigation for Wheeled and Quadrupedal Robots

Haitong Wang, Aaron Hao Tan, *Student Member, IEEE*, Angus Fung, *Student Member, IEEE*, and Goldie Nejat, *Member, IEEE*

***Abstract*—Existing navigation methods are primarily designed for specific robot embodiments, limiting their generalizability across diverse robot platforms. In this paper, we introduce X-Nav, a novel framework for cross-embodiment navigation where a single unified policy can be deployed across various embodiments for both wheeled and quadrupedal robots. X-Nav consists of two learning stages: 1) multiple expert policies are trained using reinforcement learning with privileged observations using a wide range of randomly generated embodiments; 2) a single general policy is distilled from the expert policies via navigation action chunking with transformer (Nav-ACT). The unified policy directly maps visual and proprioceptive observations to low-level control commands, enabling generalization to novel robot embodiments. Simulated experiments demonstrated that X-Nav can effectively achieve zero-shot transfer to unseen embodiments and unseen photorealistic environments. An ablation study is conducted to evaluate the design choices of X-Nav. Furthermore, a sim-to-real study was conducted to validate the generalizability of X-Nav to real-world environments.**

I. Introduction

Robot navigation in diverse and challenging environments is crucial for mobile robots to be able to perform tasks such as detection and search [1], [2], [3], exploration in unknown environments [4], [5], and assistive services in healthcare settings [6]. However, existing robot navigation methods are typically designed for very specific robot embodiments that consider only their mobilities, degrees-of-freedom (DOFs) and sensory configurations [7]. This embodiment-specific design limits generalization to robot deployment across multiple robot embodiments. In this paper, we address the problem of cross-embodiment navigation where a single generalized navigation policy can be deployed on a wide range of robot embodiments. In particular, we consider the visual point-goal navigation problem where a mobile robot is required to navigate to a target goal position using visual observations (i.e., depth images) obtained from an onboard camera [8].

Cross-embodiment navigation methods have mainly used either imitation learning (IL) [9], [10], [11], [12], [13], [14], [15], or reinforcement learning (RL) [16], [17], [18], [19], [20], [21]. In particular, IL methods learn a navigation policy from datasets collected from heterogeneous robot platforms (e.g., TurtleBot2, Jackal, Spot) and use visual observations to extract spatial and embodiment features to predict robot waypoints or velocities. On-the-other-hand, RL methods learn to navigate by training on diverse robot embodiments, including randomly generated robots or off-the-shelf robots. They use proprioceptive (e.g., joint angles, joint velocities) and/or visual observations as input to implicitly infer embodiments for robot velocities or joint position generation. However, these aforementioned methods are limited by 1) their reliance on embodiment-specific modules to track the predicted waypoints or velocities [9], [10], [11], [12], [13], [14], [15], which is particularly challenging to achieve especially for quadrupeds due to the high DOFs [22]; 2) their focus on locomotion where the methods only learn to track given velocity commands but lack the capability to plan velocities for visual navigation [18], [19], [20], [21].

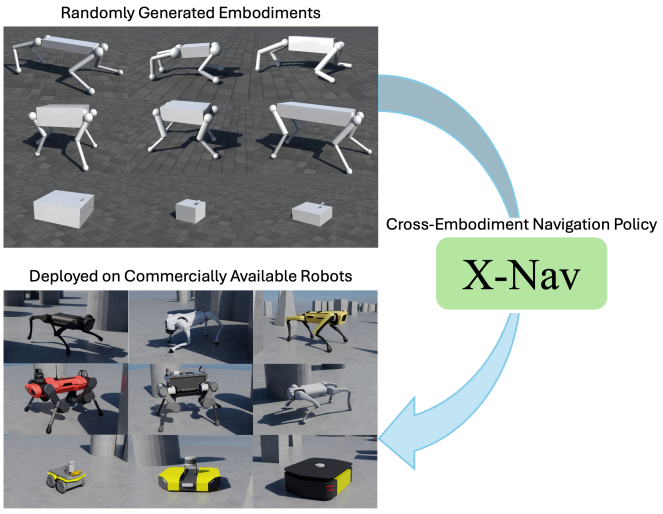


Fig. 1. X-Nav trains a single general navigation policy using randomly generated robot embodiments, which can be deployed across a variety of commercially available robots. X-Nav is the first end-to-end cross-embodiment navigation model for both wheeled and quadrupedal robots.

To address the above limitations, in this paper, we propose X-Nav, a novel two-stage learning framework for cross-embodiment navigation. Our approach is end-to-end that it directly maps from visual and proprioceptive observations to low-level control commands (i.e., velocities for wheeled robots, joint positions for quadrupeds), without relying on embodiment-specific modules to plan velocities/waypoints or to track them. In the first stage, we use RL with privileged observations to train multiple expert policies on randomly generated robot embodiments, enabling each policy to acquire navigation skills optimized for a group of similar robot

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and in part by the Canada Research Chairs program (CRC). *(Corresponding author: Haitong Wang.)*

The authors are with the Autonomous Systems and Biomechatronics Laboratory (ASBLab), Department of Mechanical and Engineering, University of Toronto, Toronto, ON M5S 3G8, Canada (e-mail: haitong.wang@mail.utoronto.ca; aaronhao.tan@utoronto.ca; angus.fung@mail.utoronto.ca; nejat@mie.utoronto.ca.

embodiments (e.g., wheeled robots, quadrupeds). In the second stage, the expert policies are distilled into a single general policy using a transformer model with a unified observation and action space. The key contributions of this work include:

1) the development of the first end-to-end cross-embodiment navigation approach for both wheeled and quadrupedal robots.

2) the introduction of a novel two-stage learning framework that integrates expert policy learning with general policy distillation, enabling zero-shot transfer to unseen robots and real-world environments.

II. Related Works

We discuss the existing works on cross-embodiment navigation that have used: 1) imitation learning (IL) [9], [10], [11], [12], [13], [14], [15] and 2) reinforcement learning [16], [17], [18], [19], [20], [21].

A. Imitation Learning-based Methods

Imitation learning-based methods learn a cross-embodiment navigation policy from training on datasets collected from heterogeneous robot platforms. These methods take as input robot current visual observations (i.e., RGB images) and goal images [9], [10], [11], [12], [13], [14], [15] and use visual encoders such as custom ConvNets [10], MobileNetv2 [11], [23], EfficientNet [9], [12], [13], [24], ResNet [14], [25], DINOv2 [15], [26] to extract spatial features from the input images. Fully connected layers (FCLs) [10], [11], [15], transformer blocks, [9], [12], [13], [14], [27], diffusion action heads [9], [13], [28] were used to generate relative waypoints [9], [11], [12], [13] or velocities [10], [15] to guide navigation.

IL methods are trained on datasets such as KITTI [10], GNM [9], [10], [11], [12], [14], SACSoN [9], [13] and are evaluated in indoor [9], [10], [11], [12], [13], [14], [15] and outdoor environments [9], [10], [11], [12], [13] using wheeled robots such as Vizbot [10], [11], [12], LoCoBot [9], [10], [11], [12], [13], [14], Clearpath Jackal [9], [11], [12], and quadrupeds such as Unitree Go1 [9], [12], [14] and Go2 [15].

B. Reinforcement Learning-based Methods

Reinforcement learning-based methods consist of 1) hierarchical methods [16], [17] and 2) end-to-end methods [18], [19], [20], [21].

Hierarchical methods integrate separate modules for robot navigation. For example, in [16], the system consists of a shared high-level policy that generates robot trunk velocities and a low-level whole-body model predictive control (MPC) policy to track the velocities using inverse kinematics [16]. The high-level policy uses a ConvNet to extract visual features from robot observations (RGB images), a robot embedding network to generate robot-specific embedding and an MLP to generate trunk velocities. In [17], a dynamics module is used to predict future robot poses given current image observation and a sequence of future actions, a general perception module was used to predict future rewards given current image observation and a sequence of future robot poses. The two modules were used in an MPC framework to plan robot velocities.

End-to-end methods directly map the raw observations such as proprioception [18], [19], [20], [21], joint descriptions (e.g., torque limit, velocity limit) [19] to quadruped joint motor positions. They use randomly generated robots such as quadrupeds [18], [20], [21] or actual robot embodiments [16], [19].

These methods are trained using model-free RL methods such as SAC+AE [16], [29] and Proximal Policy Optimization (PPO) [18], [19], [20], [21], [30] or model-based RL using MPC and cross-entropy method [17], [31].

They were deployed on wheeled robots such as Yujin Kobuki [17], Clearpath Jackal [17] and quadrupeds such as Unitree Aliengo [16], [20], [21], A1 [16], [18], [19], Go1 [20], [21], Go2 [21], MIT Mini Cheetah [18], CUHK Sirius [18], MAB Honey Badger [19] and tested in real-world indoor [16], [18], [20], [21] and outdoor environments [17], [19], [20].

C. Summary of Limitations

The IL-based methods have been able to learn navigation skills from heterogeneous datasets. However, they still rely on robot-specific controllers to track the generated waypoints [9], [11] or velocities [10], [15]. This is particularly challenging for quadrupeds due to the complexity introduced by the high DOFs [22]. The RL-based methods have been able to learn navigation from interacting with the environments. However, they either lack the ability to plan trunk velocities for autonomous navigation [18], [19], [20], [21], still requires robot-specific dynamics module [17], or exhaustive search of embedding space to generalize to new robots [16].

To address the limitations, we propose X-Nav, the first end-to-end method for cross-embodiment navigation for both wheeled and quadrupedal robots.

III. Problem Formulation

The cross-embodiment navigation problem consists of a mobile robot $r \in \mathcal{R}_{wheel} \cup \mathcal{R}_{quad}$ that needs to navigate from a known start location $l_s \in \mathbb{R}^2$ to a given goal location $l_g \in \mathbb{R}^2$ in an unknown cluttered environment. $\mathcal{R}_{wheel}$ and $\mathcal{R}_{quad}$ represent the set of wheeled robots and quadrupeds respectively. The robot navigation is guided by visual observations which are represented by depth images $o_{depth} \in \mathbb{R}^{H \times W}$ from an onboard depth camera, goal location l_g, and proprioceptive observations o_{prio} from IMU and motor encoders.

For $r \in \mathcal{R}_{wheel}$, proprioception $o_{prio}^{wheel} \in \mathbb{R}^2$ represents its current linear and angular velocity, action $a^{wheel} \in \mathbb{R}^2$ represents the target linear and angular velocity. For $r \in \mathcal{R}_{quad}$, proprioception $o_{prio}^{quad} \in \mathbb{R}^{30}$ represents the concatenation of robot trunk velocity $o_{prio,vel}^{quad} \in \mathbb{R}^3$, projected gravity in the base frame $o_{prio,g}^{quad} \in \mathbb{R}^3$, current joint position $o_{prio,jp}^{quad} \in \mathbb{R}^{12}$ and joint velocity $o_{prio,jv}^{quad} \in \mathbb{R}^{12}$. The action

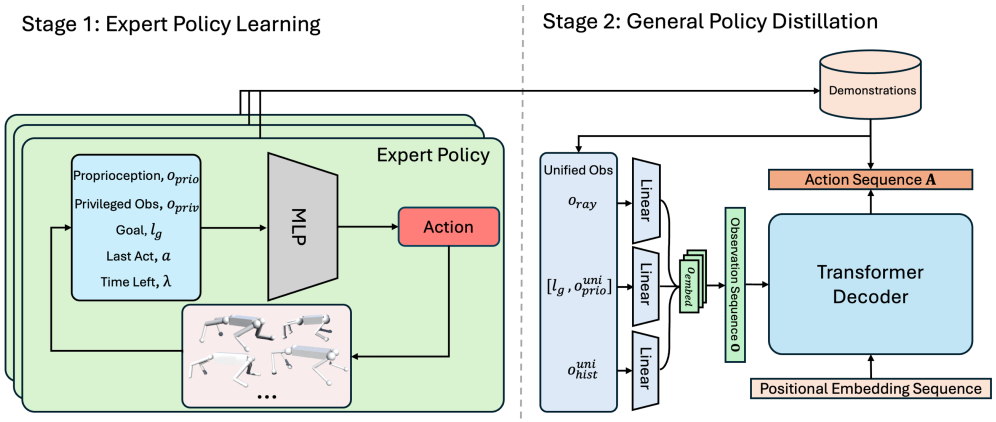


Fig. 2. The proposed X-Nav framework consists of two stages: 1) Expert Policy Learning, and 2) General Policy Distillation. In the first stage, multiple expert policies are trained on randomly generated robot embodiments using reinforcement learning with privileged observations. In the second stage, the knowledge of expert policies is distilled into a single general policy using imitation learning.

$a^{quad} \in \mathbb{R}^{12}$ represent the 12 target joint angles corresponding to the hip, thigh and knee joints.

The objective of the robot r is to minimize the travel distance between l_s and l_g:

$$\min \mathbb{E}\left[d_{nav}(l_s, l_g)\right], \tag{1}$$

where $\mathrm{d_{nav}}(\cdot,\cdot)$ is a function measuring travel distance.

IV. X-Nav Architecture

The proposed architecture consists of two stages: 1) Expert Policy Learning, and 2) General Policy Distillation, Fig. 2. In Stage 1, we randomize robot embodiments and train multiple expert navigation policies using privileged observations with reinforcement learning. In Stage 2, we first collect demonstrations using the trained expert policies from Stage 1, and then train a single general navigation policy on the demonstrations using imitation learning.

A. Expert Policy Learning

In this stage, we train $N_p = 3$ expert policies π_{ϕ_i} $(i = 1, \ldots, N_p)$ in simulation using RL with privileged observations. Each expert policy is trained on 4096 robots that share similar embodiments. Namely, the expert policies are separately trained for small-sized quadrupeds that are within 30 kg of total mass, large-sized quadrupeds and wheeled robots. The expert policies have access to privileged observations obtained from the simulator, including embodiment parameters and terrain height scans that cannot be obtained by the distilled general policy during deployment [32].

TABLE I
Parameters and Value Ranges for Embodiment Randomization

Type	Parameters	Range	Parameters	Range
Small-Sized Quadrupeds	Trunk length	[0.24, 0.91] m	Thigh mass	[0.56, 1.69] kg
	Trunk width	[0.16, 0.39] m	Calf radius	[0.02, 0.05] m
	Trunk height	[0.06, 0.21] m	Calf length	[0.12, 0.39] m
	Trunk mass	[4.8, 19.5] kg	Calf mass	[0.12, 0.39] kg
	Thigh radius	[0.02, 0.05] m	Motor P gain	[0.7, 1.3]
	Thigh length	[0.16, 0.46] m	Motor D gain	[0.7, 1.3]
Large-Sized Quadrupeds	Trunk length	[0.56, 1.04] m	Thigh mass	[2, 5.2] kg
	Trunk width	[0.28, 0.52] m	Calf radius	[0.02, 0.04] m
	Trunk height	[0.14, 0.26] m	Calf length	[0.24, 0.36] m
	Trunk mass	[24, 39] kg	Calf mass	[0.4, 0.6] kg
	Thigh radius	[0.03, 0.05] m	Motor P gain	[0.5, 1.3]
	Thigh length	[0.24, 0.39] m	Motor D gain	[0.5, 1.3]
Wheeled Robots	Base length	[0.3, 0.8] m	Base height	[0.15, 0.3] m
	Base width	[0.2, 0.65] m	Base mass	[5, 20] kg

1) Observations

At each timestep t $(0 \le t < T)$ of the navigation episode, the policy observation is $o = \left[a, o_{prio}, l_g, \lambda, o_{priv}\right]$, where a denotes the last robot action, $\lambda = 1 - \frac{t}{T}$ denotes the time left in the current episode normalized to the range between 0 and 1, T is the maximum timesteps of the episode. $o_{priv} = [o_{embod}, o_{scan}]$ denotes the privileged observations which include embodiment parameters o_{embod} and terrain height scans o_{scan} obtained from a virtual sensor in the simulator for ground-truth terrain height measurement. For wheeled robots, embodiment parameters include mass and robot size. For quadrupeds, embodiment parameters include trunk size, trunk mass, thigh size, thigh mass, calf size, and calf mass.

2) Embodiment Randomization

In order to train an expert policy that can control robots of different but share similar embodiments, we implement embodiment randomization, where the RL policy is trained on randomly generated robot embodiments [18], [20]. This method ensures that real-world robot embodiments fall within the distribution of the randomly generated robots. Three types of robots are considered: small-sized quadrupeds, large-sized quadrupeds and wheeled robots. For each robot type, we design a corresponding template robot with predefined parameters to serve as a baseline. Using this template, we generate random robots by sampling their embodiment parameters, as listed in Table I. The motor PD gains are computed as [18]:

$$v_{temp} \times \frac{m_{new}}{m_{temp}} \times v, \tag{2}$$

where v denotes the random value sampled based on the ranges in Table I, v_{temp} denotes the PD gains of the template robot, m_{new} denotes the total mass of the generated robot, and m_{temp} denotes the total mass of the template robot.

3) Rewards

The reward function is designed to encourage the robots to navigate to the goal location and avoid obstacles. It includes two components:

$$\mathrm{r} = r_{task} + r_{reg}, \quad (3)$$

where r_{task} denotes the task rewards, and r_{reg} denotes the regularization rewards. Namely, r_{task} are designed to encourage the robot to navigate to the goal location while avoiding collisions with obstacles. r_{reg} are designed to ensure smooth and efficient movement by penalizing undesired behaviors (e.g. excessive oscillations, abrupt velocity changes) that may lead to unstable or inefficient navigation. Wheeled and quadrupedal robots share the same task rewards but different regularization rewards.

The task rewards are defined as:

$$r_{task} = r_{pos,soft} + r_{pos,hard} + r_{fwd} + r_{stand} + r_{collide}, \quad (4)$$

where $r_{pos,soft}$ and $r_{pos,hard}$ are two goal position tracking terms to encourage the robot to move to the goal location, r_{fwd} is the term that encourages the robot to move forward, r_{stand} is the term to encourage the robot to stand still at the goal location, $r_{collide}$ denotes the collision penalty term to penalize any undesired collisions between the robot and obstacles. For wheeled robots, undesired collisions refer to contacts between robot base and obstacles; for quadrupeds, they refer to contacts between robot trunk and obstacles.

$r_{pos,soft}$ and $r_{pos,hard}$ are defined to reward the robot for moving close to the goal location as [33]. They are activated only near the end of the episode which allows the robot to explore in the beginning of the episode to avoid local minima:

$$r_{pos,(soft/hard)} = \frac{c_{1,(soft/hard)}}{1 + \left\| \frac{d_{goal}}{c_{2,(soft/hard)}} \right\|^2} \cdot \mathbb{1}(t > T - T_r), \quad (5)$$

where T_r denotes the duration of timesteps that these reward terms are activated, $\mathbb{1}$ is an indicator function that outputs 1 if $t > T - T_r$ and 0 otherwise. d_{goal} denotes the distance between the robot and the goal location. $c_{1,(soft/hard)}$, $c_{2,(soft/hard)}$ are positive constants. $c_{2,soft}$ is set to be larger than $c_{2,hard}$ so that $r_{pos,soft}$ provides dense rewards and $r_{pos,hard}$ reinforces the robot to accurately track the goal location.

The forward term r_{fwd} is defined as [34] to encourage forward movement:

$$r_{fwd} = c_{fwd} \cdot \max\left\{ \mathrm{ReLU}\left(\frac{v_x}{v_{max}}\right) \cdot \mathbb{1}(\delta_{goal} < \sigma_{direct}), \mathbb{1}(d_{goal} < \sigma_{hard}) \right\}, \quad (6)$$

where v_x is the linear velocity of the robot trunk along the x axis, v_{max} is the maximum velocity of the robot, c_{fwd} is a positive constant. δ_{goal} is the absolute angle difference between the robot's heading direction and the goal location, and σ_{direct} is an angle threshold, within which the robot heading is considered in the correct navigation direction. The Rectified Linear Unit (ReLU) function is used to reward forward velocity and to normalize this reward value to be less than 1. σ_{hard} is a distance threshold, within which the robot is considered as reaching the goal and will receive a reward of 1.

The standing term r_{stand} is used to enforce the robot to stand still at the goal location after reaching the goal:

$$r_{stand} = \frac{c_{1,stand}}{1 + \left\| \frac{o_{prio,vel}}{c_{2,stand}} \right\|^2} \cdot \mathbb{1}(t > T - T_r) \cdot \mathbb{1}(d_{goal} < \sigma_{hard}), \quad (7)$$

where $c_{1,stand}$, $c_{2,stand}$ are positive constants.

The collision penalty term is $r_{collide} = \mathrm{c}_{collide} \cdot \mathbb{1}_{collide}$, where $c_{collide}$ is a negative constant, $\mathbb{1}_{collide}$ indicates whether undesired collision happens between the robot and obstacles.

The regularization rewards of quadrupeds r_{reg}^{quad} are defined as [35]:

$$r_{reg}^{quad} = c_{v_z} v_z^2 + c_{\omega}(\omega_{xy}^2) + c_{\tau}\|\tau\|^2 + c_{\dot{a}}\|\dot{a}\|^2 + c_{\ddot{q}}\|\ddot{q}\|^2 + c_{air} \sum_{f=1}^{4} \left(t_{air,f} - 0.5\right) + c_{flat}\left\|o_{prio,g,xy}\right\|^2, \quad (8)$$

where v_z is the linear velocity of the robot trunk along z axis, ω_{xy} represents the angular velocity of the robot trunk around x and y axes, τ is the vector of 12 joint torques, $\dot{a}$ is the action rate, $\ddot{q}$ is the joint acceleration, $t_{air,f}$ is the duration of time when foot f is in the air (i.e., not in contact with the ground), $o_{prio,g,xy}$ is the x, y component of the projected gravity vector in the base frame. c_{v_z}, c_{ω}, c_{τ}, $c_{\dot{a}}$, $c_{\ddot{q}}$, c_{flat} are negative constants, c_{air} is a positive constant.

The regularization reward of wheeled robots r_{reg}^{wheel} is defined as $r_{reg}^{wheel} = c_{\dot{a}}\|\dot{a}\|^2$.

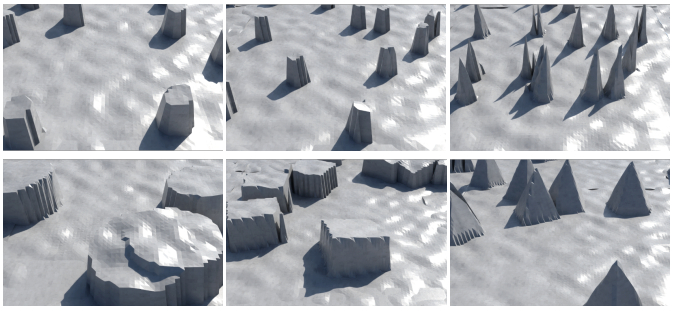
4) Curriculum Learning

We use a game-inspired curriculum [35] to progressively train the policy to navigate from simple to complex environments, Fig. 3. The overall training environment consists of 384 subfields arranged in 6 rows and 64 columns, each measuring 10 × 10 m. Each subfield contains randomly generated obstacles with varying levels of difficulty. We use three types of obstacles: box, pyramid, and cylinder. Each obstacle type has two aspects of difficulty progression, with 6 levels for each aspect: one progressively increases with obstacle density, while the other increases the obstacle size. In addition to obstacle difficulty, the terrain roughness (i.e., the maximum vertical deviation of the surface from a flat plane) within the subfields are also progressively increased from 0 to 8 cm across the difficulty levels.

At the beginning of training, all robots are randomly spawned in the lowest level. The progression of robots through difficulty levels is governed by two distance thresholds σ_{close} and σ_{far} ($\sigma_{close} < \sigma_{far}$). When the robot reaches the vicinity of the goal location at the end of an episode (i.e., $d_{goal} < \sigma_{close}$), the robot is promoted to a higher level. Conversely, if the robot fails to achieve the goal and the goal distance exceeds σ_{far} (i.e., $d_{goal} > \sigma_{far}$), the robot is demoted to the lower level. Otherwise, the robot would stay in the current level. The robot's subfield level is updates as:

$$L = \begin{cases} min(L+1, L_{max}) & \text{if } d_{\text{goal}} < \sigma_{\text{close}} \\ \max(L-1, L_{min}) & \text{if } d_{\text{gool}} > \sigma_{\text{far}} \\ L & \text{otherwise} \end{cases}, \quad (9)$$

where L denotes the current level.



(a) (b) (c)

Fig. 3. The subfields used for expert policy learning. (a) cylinders, (b) boxes, (c) pyramids.

5) Reinforcement Learning

We use a model-free RL algorithm PPO [30] to train the three expert policies. PPO is used due to its stability and efficiency in training. Each policy network π_{ϕ_i} is implemented as an MLP with (1024, 512, 256) units. The NVIDIA Isaac Sim simulator and Isaac Lab framework [36] are used for expert policy training. Each policy is trained using 4096 randomly generated embodiments (Section IV.A.2) and curriculum learning (Section IV.A.4). Rough terrain is only applied for quadrupeds, the wheeled robot policy is trained on flat terrain.

B. General Policy Distillation

In this stage, we distill the trained expert policies $\pi_{\phi_i}, (\mathrm{i} = 1, \ldots, N_p)$ into a single general policy π_θ using imitation learning. First, we define a unified observation and action space for wheeled and quadrupedal robots. Then, we collect demonstrations from the expert policies trained in Stage 1, and preprocess them to fit the unified observation and action space. Finally, a single general policy π_θ is trained to learn from the collected demonstrations.

1) Unified Actions and Observations

To create a unified action and observation space, we apply zero padding to ensure that the observation and action vectors have the same dimensionality across all robots. This padding ensures that the policy network can work on a uniform input and output size, regardless of the robot's embodiment, enabling effective cross-embodiment learning.

The unified action a^{uni}, $a^{uni} \in \mathbb{R}^{14}$ is a 14-dimensional vector where the first two dimensions represent the linear and angular velocity of wheeled robots, while the last 12 dimensions represent the target joint positions for quadrupeds.

The unified observation o^{uni} is:

$$o^{uni} = \left[l_g, o^{uni}_{prio}, o^{uni}_{hist}, o_{ray}\right], \tag{10}$$

where $o^{uni}_{prio} \in \mathbb{R}^{30}$ denotes the unified proprioception. For wheeled robots, where joint position and velocity data are absent from proprioception o^{wheel}_{prio}, we pad the dimensions with zeros to match o^{quad}_{prio} to obtain o^{uni}_{prio}. $o^{uni}_{hist} \in \mathbb{R}^{220}$ denotes the concatenation of the last 5 frames of actions a^{uni} and proprioception o^{uni}_{prio}. $o_{ray} \in \mathbb{R}^{128}$ denotes a unified set of laser rays for distance measurement derived from the raw depth image o_{depth}. We convert o_{depth} to a set of 2D laser rays by projecting depth pixels onto a horizontal plane and extracting distance at evenly spaced angles within the camera field of view (FOV). The resulting laser scan is then interpolated into a unified format with a FOV of 90°, minimum and maximum measurement distance of 0.2 m and 8 m, and number of laser rays of 128. This ensures consistency across robots with different camera intrinsic parameters.

2) Demonstration Collection

We collect 4096 demonstrations for each of the expert policies π_{ϕ_i} trained in Stage 1, totaling 12288 demonstrations. Each demonstration consists of a sequence of unified robot observations o^{uni} and actions a^{uni}, collected from a successful navigation episode within a randomly generated environment of the most difficult level used in Section IV.A.4).

3) Navigation Action Chunking with Transformer

We introduce Navigation Action Chunking with Transformer (Nav-ACT), a transformer model to learn cross-embodiment navigation from demonstrations based on ACT [37]. Nav-ACT is used to generate the unified actions a^{uni} from the unified observations o^{uni}.

Nav-ACT consists of 1) a set of encoders to convert the unified observations o^{uni} into observation embeddings, and 2) a transformer decoder [27] to generate a sequence of unified actions conditioned on the observation embedding sequence.

At each timestep t, the unified observation o^{uni} is converted into an embedding o_{embed} using three linear encoders. The laser ray observation o_{ray}, historical observation o^{uni}_{hist} proprioception o^{uni}_{prio} are each input to a linear layer to generate o^{ray}_{embed}, o^{hist}_{embed} and o^{prio}_{embed}. These three embeddings are then concatenated into o_{embed}. The latest L_o steps of observation embeddings o_{embed} are stacked to construct an observation sequence $\mathrm{O_t}$, which is used by the transformer decoder to generate an action sequence $\mathrm{A_t}$ of L_a actions.

The transformer decoder takes as input a sequence of L_a learnable positional embeddings P and uses the observation embedding sequence $\mathrm{O_t}$ for cross-attention computation to generate a sequence of L_a actions $\mathrm{A_t}$. The action sequence is generated in a single forward pass instead of auto-regression as the original transformer [27], which is more computationally efficient.

Nav-ACT is trained using the mean squared error (MSE) loss:

$$\mathcal{L}_{Nav-ACT} = \frac{1}{L_a}\sum_{i=1}^{L_a} \left\|\mathbf{A}^{(i)}_{pred} - \mathbf{A}^{(i)}_{gt}\right\|^2, \tag{11}$$

where $\mathbf{A}_{pred}$ denotes the predicted action sequence generated by Nav-ACT, and $\mathbf{A}_{gt}$ denotes the ground truth actions from the collected demonstrations.

4) Inference

At inference time, we employ different strategies for wheeled and quadrupedal robots: for wheeled robots, we use temporal ensemble [37] to improve smoothness and avoid jerky movements. For quadrupeds, we disable temporal ensemble and only take the first action from the action

TABLE II
HYPERPARAMETERS AND VALUES

Hyperparameter	Value	Hyperparameter	Value
$c_{1,soft}$	20	$c_{collide}$	-40
$c_{2,soft}$	5	c_{v_z}	-2
$c_{1,hard}$	15	c_{ω}	-0.05
$c_{2,hard}$	0.5	c_{τ}	-0.0002
T_r	1.5 s	$c_{\dot{a}}$	-0.01
T	8 s	$c_{\ddot{q}}$	-2.5e-7
v_{max}	0.3 m/s	c_{flat}	-5
σ_{direct}	1.75 rad	c_{air}	0.5
σ_{hard}	0.5 m	σ_{close}	0.5 m
$c_{1,stand}$	10	σ_{far}	3.0 m
$c_{2,stand}$	0.2	c_{fwd}	2

sequence to ensure fast adaptation to variations in the robot's dynamics during navigation.

For wheeled robots, at each timestep t, Nav-ACT generates a sequence of action predictions $\mathbf{A}_t$, then we take the last L_a action sequence predictions $\mathbf{A}_i$, $i = (t - L_a + 1, \dots, t)$ to retrieve the action predictions for the current timestep t from each $\mathbf{A}_i$. Weighted average is applied to generate the action to execute as [37]:

$$a_t = \frac{\sum_{i=t-L_a+1}^{t} w_i A_i[t-i]}{\sum_{i=t-L_a+1}^{t} w_i}, \quad (12)$$

where $w_i = exp(-k * (i - t + L_a - 1))$ and k is a positive constant set to be 0.0001.

For quadrupeds, at each timestep, we take the action from the latest action sequence, $a_t = \mathbf{A}_t[0]$.

V. TRAINING

Training was done on a workstation with an NVIDIA RTX 4090 GPU, an Intel Core i9-13900KF CPU and 32GB RAM.

A. Expert Policy Learning

Each of the expert policies was trained a batch size of 24576 for 4000 epochs. Adam optimizer [38] was used with a learning rate of 0.001. Each expert policy took 2 hours to train. The hyperparameters used in reward function and curriculum learning are defined as Table II.

B. General Policy Distillation

The general policy, Nav-ACT was trained with a batch size of 256, learning rate of 0.0001 for 100 epochs. Adam optimizer [38] was used with weight decay of 0.001. We used an action sequence length L_a of 6, and an observation sequence length L_o of 4. Nav-ACT has 4 transformer layers and 4 heads with an embedding size of 256, totaling 5M parameters. Nav-ACT took 19 hours to train.

VI. EXPERIMENTS

We conducted four sets of experiments to evaluate the performance of X-Nav: 1) a simulated comparison study to evaluate the generalizability of X-Nav on unseen robot embodiments, 2) a simulated comparison study in unseen photorealistic environments, 3) a simulated ablation study to investigate the design choices of X-Nav, 4) a sim-to-real study to evaluate the generalizability of X-Nav in real-world environments.

TABLE III
COMPARISON STUDY WITH UNSEEN ROBOT EMBODIMENTS

	Go2		**A1**		**ANYmal B**		**Jackal**		**Dingo**		**Create3**	
	SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR	SPL
BC	63.6	0.55	70.2	0.60	45.1	0.40	57.5	0.51	31.2	0.24	46.7	0.36
BCT	66.8	0.58	70.2	0.59	45.6	0.43	41.6	0.34	16.0	0.11	16.8	0.12
DP	64.7	0.56	71.1	0.59	36.2	0.32	66.6	0.60	18.6	0.14	20.2	0.16
CP	27.1	0.24	40.2	0.36	33.0	0.30	51.2	0.45	15.4	0.11	20.9	0.16
X-Nav	69.1	0.60	71.6	0.61	59.6	0.52	90.4	0.84	83.5	0.76	93.1	0.85

A. Comparison Study on Unseen Robot Embodiments

The objective of this study is to evaluate the generalizability of X-Nav on unseen robot embodiments. We used commercially available robots that are unseen during the training of X-Nav, including 3 wheeled robots: Clearpath Jackal, Clearpath Dingo, iRobot Create3, and 3 quadrupeds: Unitree A1, Unitree Go2, ANYmal B. Each robot is equipped with a front-facing depth camera.

We used two performance metrics for these experiments: 1) success rate (SR) of robots reaching goal locations, and 2) success weighted by normalized inverse path length (SPL) which measures the efficiency of the navigation path [39]: $\frac{1}{N_\tau}\sum_{i=1}^{N_\tau} S_i \frac{\ell_i}{max(\mathcal{p}_i,\ell_i)}$, where N_τ denotes the number of trials, ℓ_i denotes the shortest path length from the start to goal location, $\mathcal{p}_i$ denotes the actual robot path length, and S_i is a binary indicator of success in trial i.

We randomly generated 100 environments that are unseen during training. Each environment is of 10 m × 10 m. They consist of the same types of obstacles as those used during training Section IV.A.4). Quadrupeds were tested on rough terrain and wheeled robots were tested on flat terrain. The start and goal locations were randomly sampled from the boundary of the environment.

1) Comparison Methods: We compared with the following four imitation learning methods. They were all trained on the same dataset collected in Section IV. B. 2.

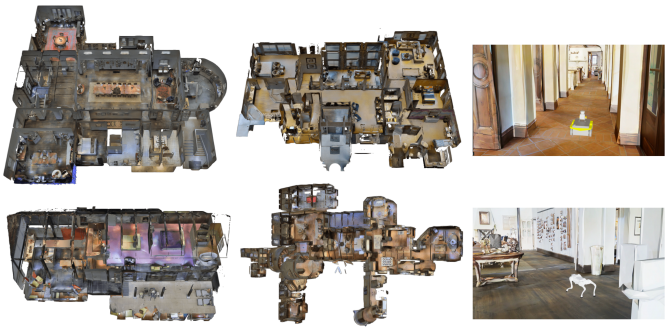
Behavior Cloning (BC) [40]: The BC method uses the same encoders as our approach and an MLP to generate the next action.

BC Transformer (BCT) [27]: The BCT method uses the same encoders as our approach and a transformer decoder to generate the next action. The transformer decoder takes the last 6 observations and actions as input to generate the next action.

Diffusion Policy (DP) [28]: The DP method uses the EDM scheduler [41] and a transformer decoder to generate action sequence as [28]. DP used 80 steps for training and 10 steps for inference.

Consistency Policy (CP) [42]: The CP method uses consistency trajectory model (CTM) [43] and transformer to generate actions. CP used DP as the teacher model and used one-step inference.

2) Procedure: At the beginning of each trial, both robot start and goal locations were randomly sampled. A robot completed the task when the distance between the robot and goal location was within 0.5 m after the trial ended. A trial terminated when the total timesteps exceeded 750. Each timestep is 0.02s. We ran 3000 trials for each method and robot embodiment.



(a) (b)

Fig. 4. (a) Top view of photorealistic indoor environments from MP3D dataset, (b) Jackal and Go2 deployed inside the environments.

3) Results: The SR, SPL of X-Nav and comparison methods are shown in Table III. In general, X-Nav was able to control different unseen robot embodiments and consistently outperformed BC, BCT, DP, CP in terms of SR and SPL. BC achieved lower SR and SPL than X-Nav across all embodiments because BC generates the next action solely based on the latest observation. It did not account for history of past observations, which are essential for spatial understanding for efficient navigation. BCT performed worse than X-Nav in terms of both SR and SPL because X-Nav implicitly predict future states by generating an action sequence while BCT only predicts the next action which can lead to suboptimal trajectory for navigation. DP achieved lower SR and SPL than X-Nav because DP's iterative denoising is not suitable for high-frequency tasks like navigation [28]. Additionally, the inference of DP is computationally intensive. Though it has been used for quadruped locomotion task [44], specialized hardware and software acceleration are required to deploy the model in real-time, making it less practical for general navigation applications. CP, on the other hand, has faster inference speed than DP by directly generating the denoised action through one single forward pass. However, CP achieved the lowest SR and SPL for quadrupeds and worse SR and SPL for wheeled robots when compared with X-Nav. This can be attributed to CP's inability to handle the high-frequency control demands of quadrupeds, which require rapid action updates to maintain stability.

B. Comparison Study in Unseen Photorealistic Environments

The objective of this study is to evaluate the generalizability of X-Nav in unseen photorealistic environments. We used Matterport3D (MP3D) dataset [45] that contain 3D mesh of real-world indoor environments. Specifically, four houses from the MP3D dataset were loaded into Isaac Sim for the experiments, Fig. 4. The four houses have an average size of 22 × 25 m. In each house, we randomly generated 20 pairs of start and goal locations. We performed 200 trials in each environment for all methods and all robots.

1) Results: The results are shown in Table IV. X-Nav achieved zero-shot transfer to unseen photorealistic environments, outperforming BC, BCT, DP, CP across all embodiments. This generalization to environments that are out of the training distribution can be primarily attributed to the use of depth images as input, which are robust to variations in lighting and texture, allowing the model to focus on spatial geometry and obstacles.

TABLE IV
COMPARISON STUDY IN UNSEEN PHOTOREALISTIC ENVIRONMENTS

	Go2		**A1**		**ANYmal B**		**Jackal**		**Dingo**		**Create3**	
	SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR	SPL
BC	54.1	0.44	63.7	0.53	43.9	0.41	57.3	0.46	45.1	0.41	55.9	0.46
BCT	55.8	0.49	67.5	0.61	41.8	0.39	54.6	0.45	47.	0.43	54.2	0.43
DP	56.3	0.48	68.1	0.58	48.1	0.41	52.2	0.44	42.8	0.37	51.0	0.42
CP	31.5	0.34	52.4	0.47	22.1	0.18	53.7	0.44	47.6	0.43	57.3	0.48
X-Nav	63.5	0.54	70.8	0.65	51.2	0.47	80.3	0.72	78.7	0.69	83.8	0.72

TABLE V
ABLATION STUDY

	Go2		**Jackal**	
	SR	SPL	SR	SPL
X-Nav w L1	55.3	0.44	85.4	0.78
X-Nav w EC	41.0	0.36	89.1	0.82
X-Nav w/o TE	69.1	0.60	74.9	0.65
X-Nav w FTE	42.7	0.38	90.4	0.84
X-Nav	69.1	0.60	90.4	0.84

C. Ablation Study

We conducted an ablation study to evaluate the impact of the design choices of X-Nav.

1) Variants: We considered the following variants of X-Nav:

X-Nav with L1 Loss: The Nav-ACT was trained using L1 loss as the original ACT [37].

X-Nav with Executing Chunk (EC): It executed the entire predicted action sequence before predicting new action sequences.

X-Nav without Temporal Ensemble: It planed an action sequence at each timestep and always executed the first action from the current predicted sequence without temporal ensemble.

X-Nav with Full Temporal Ensemble (FTE): It applied TE to both wheeled and quadrupedal robots at inference time.

We conducted 3000 trials for each method using the Unitree Go2 and Clearpath Jackal.

2) Results: The results are shown in Table V. X-Nav achieved overall the highest SR and SPL among all variants. X-Nav w L1 performed worse than X-Nav for both Go2 and Jackal because L2 loss penalizes larger errors more heavily, encouraging smoother navigation action predictions. For quadrupeds, X-Nav achieved higher SR and SPL compared to X-Nav w EC and X-Nav w FTE, indicating that that unlike manipulation tasks that executing an action chunk helps

Fig. 5. Visualization of X-Nav deployed on the Jackal (top) and the TurtleBot2 (bottom) in indoor environments. The Jackal robot is equipped with a ZED 2 camera and TurtleBot2 is equipped with a Kinect camera.

mitigate compound errors [37], quadruped navigation requires more frequent planning to maintain stability due to the high DOFs. On the other hand, for wheeled robots, X-Nav achieved higher SR and SPL than X-Nav w EC and X-Nav w/o TE, showing that action chunking and temporal ensemble improved wheeled robot navigation. This is because wheeled robots have simpler motion dynamics, which could benefit from smoother and more consistent motion.

D. Sim-to-Real Study

We conducted a sim-to-real study to evaluate the generalizability of X-Nav in real-world indoor environments, including hallways with obstacles, corners, and doorways, Fig. 5. We used two wheeled robots: Clearpath Jackal and TurtleBot2. The Jackal robot was equipped with a ZED 2 camera and TurtleBot2 was equipped with a Kinect camera to obtain depth images for navigation. Both robots operated on the Robot Operating System (ROS) Noetic. X-Nav was deployed on the two robots without any additional training, demonstrating its strong zero-shot transfer capabilities by successfully adapting to distinct robot embodiments, camera configurations, and environments. A video of X-Nav deployed on various robots in both the simulated and real-world environments is provided on our YouTube channel at xxxxx.

VII. Conclusion

In this paper, we present a novel two-stage learning framework, X-Nav, to address the problem of cross-embodiment navigation for both wheeled and quadrupedal robots. The first stage utilized reinforcement learning with privileged observations to train multiple expert policies tailored to specific groups of embodiments. The second stage distilled these expert policies into a single general policy using a transformer model, Nav-ACT, which operates on a unified observation and action space. Through extensive simulated experiments, we demonstrated the effectiveness of X-Nav in zero-shot transfer to unseen robot embodiments and photorealistic environments, outperforming SOTA learning methods. An Ablation study validated our design choices and inference strategy. The real-world experiments showed the generalizability of X-Nav in complex indoor environments. Future work will extend X-Nav to more robot types such as humanoid robots, and object-goal navigation to expand its applicability.

References

[1] A. Fung, B. Benhabib, and G. Nejat, "Robots Autonomously Detecting People: A Multimodal Deep Contrastive Learning Method Robust to Intraclass Variations," *IEEE Robot. Autom. Lett.*, vol. 8, no. 6, pp. 3550–3557, Jun. 2023, doi: 10.1109/LRA.2023.3269306.

[2] A. Fung, B. Benhabib, and G. Nejat, "LDTrack: Dynamic People Tracking by Service Robots Using Diffusion Models," *Int. J. Comput. Vis.*, Jan. 2025, doi: 10.1007/s11263-024-02336-9.

[3] H. Wang, A. H. Tan, and G. Nejat, "NavFormer: A Transformer Architecture for Robot Target-Driven Navigation in Unknown and Dynamic Environments," *IEEE Robot. Autom. Lett.*, vol. 9, no. 8, pp. 6808–6815, Aug. 2024, doi: 10.1109/LRA.2024.3412638.

[4] A. H. Tan, F. P. Bejarano, Y. Zhu, R. Ren, and G. Nejat, "Deep Reinforcement Learning for Decentralized Multi-Robot Exploration With Macro Actions," *IEEE Robot. Autom. Lett.*, vol. 8, no. 1, pp. 272–279, Jan. 2023, doi: 10.1109/LRA.2022.3224667.

[5] A. H. Tan, S. Narasimhan, and G. Nejat, "4CNet: A Confidence-Aware, Contrastive, Conditional, Consistency Model for Robot Map Prediction in Multi-Robot Environments," Feb. 27, 2024, *arXiv*: arXiv:2402.17904. Accessed: Mar. 04, 2024. [Online]. Available: http://arxiv.org/abs/2402.17904

[6] N. Tyagi, D. Sharma, J. Singh, B. Sharma, and S. Narang, "Assistive Navigation System for Visually Impaired and Blind People: A Review," in *2021 International Conference on Artificial Intelligence and Machine Vision (AIMV)*, Sep. 2021, pp. 1–5. doi: 10.1109/AIMV53313.2021.9670951.

[7] J. Lee, M. Bjelonic, A. Reske, L. Wellhausen, T. Miki, and M. Hutter, "Learning robust autonomous navigation and locomotion for wheeled-legged robots," *Sci. Robot.*, vol. 9, no. 89, p. eadi9641, Apr. 2024, doi: 10.1126/scirobotics.adi9641.

[8] E. Wijmans *et al.*, "DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames," Nov. 2019, doi: 10.48550/arxiv.1911.00357.

[9] J. Yang *et al.*, "Pushing the Limits of Cross-Embodiment Learning for Manipulation and Navigation," Feb. 29, 2024, *arXiv*: arXiv:2402.19432. Accessed: May 16, 2024. [Online]. Available: http://arxiv.org/abs/2402.19432

[10] N. Hirose, D. Shah, A. Sridhar, and S. Levine, "ExAug: Robot-Conditioned Navigation Policies via Geometric Experience Augmentation," Oct. 13, 2022, *arXiv*: arXiv:2210.07450. Accessed: May 17, 2024. [Online]. Available: http://arxiv.org/abs/2210.07450

[11] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "GNM: A General Navigation Model to Drive Any Robot," May 22, 2023, *arXiv*: arXiv:2210.03370. Accessed: May 16, 2024. [Online]. Available: http://arxiv.org/abs/2210.03370

[12] D. Shah *et al.*, "ViNT: A Foundation Model for Visual Navigation".

[13] A. Sridhar, D. Shah, C. Glossop, and S. Levine, "NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration," Oct. 11, 2023, *arXiv*: arXiv:2310.07896. Accessed: May 16, 2024. [Online]. Available: http://arxiv.org/abs/2310.07896

[14] R. Doshi, H. Walke, O. Mees, S. Dasari, and S. Levine, "Scaling Cross-Embodied Learning: One Policy for Manipulation, Navigation, Locomotion and Aviation," Aug. 21, 2024, *arXiv*: arXiv:2408.11812. Accessed: Aug. 22, 2024. [Online]. Available: http://arxiv.org/abs/2408.11812

[15] W. Liu *et al.*, "X-MOBILITY: End-To-End Generalizable Navigation via World Modeling," Oct. 23, 2024, *arXiv*: arXiv:2410.17491. Accessed: Oct. 24, 2024. [Online]. Available: http://arxiv.org/abs/2410.17491

[16] J. Truong *et al.*, "Learning Navigation Skills for Legged Robots with Learned Robot Embeddings," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Prague, Czech Republic: IEEE, Sep. 2021, pp. 484–491. doi: 10.1109/IROS51168.2021.9635911.

[17] K. Kang, G. Kahn, and S. Levine, "Hierarchically Integrated Models: Learning to Navigate from Heterogeneous Robots," Nov. 04, 2021, *arXiv*: arXiv:2106.13280. Accessed: May 17, 2024. [Online]. Available: http://arxiv.org/abs/2106.13280

[18] G. Feng *et al.*, "GenLoco: Generalized Locomotion Controllers for Quadrupedal Robots".

[19] N. Bohlinger *et al.*, "One Policy to Run Them All: an End-to-end Learning Approach to Multi-Embodiment Locomotion".

[20] Z. Luo *et al.*, "MorAL: Learning Morphologically Adaptive Locomotion Controller for Quadrupedal Robots on Challenging Terrains," *IEEE Robot. Autom. Lett.*, vol. 9, no. 5, pp. 4019–4026, May 2024, doi: 10.1109/LRA.2024.3375086.

[21] F. Di Giuro, F. Zargarbashi, J. Cheng, D. Kang, B. Sukhija, and S. Coros, "Meta-Reinforcement Learning for Universal Quadrupedal Locomotion Control," arXiv.org. Accessed: Aug. 24, 2024. [Online]. Available: https://arxiv.org/abs/2407.17502v1

[22] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Sci. Robot.*, vol. 5, no. 47, p. eabc5986, Oct. 2020, doi: 10.1126/scirobotics.abc5986.

[23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Mar. 21, 2019, *arXiv*: arXiv:1801.04381. Accessed: Oct. 30, 2024. [Online]. Available: http://arxiv.org/abs/1801.04381

[24] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," Sep. 11, 2020, *arXiv*: arXiv:1905.11946. Accessed: Oct. 30, 2024. [Online]. Available: http://arxiv.org/abs/1905.11946

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[26] M. Oquab *et al.*, "DINOv2: Learning Robust Visual Features without Supervision," Feb. 02, 2024, *arXiv*: arXiv:2304.07193. Accessed: Oct. 30, 2024. [Online]. Available: http://arxiv.org/abs/2304.07193

[27] A. Vaswani *et al.*, "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, pp. 5999–6009, 2017.

[28] C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," Mar. 14, 2024, *arXiv*: arXiv:2303.04137. Accessed: May 15, 2024. [Online]. Available: http://arxiv.org/abs/2303.04137

[29] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving Sample Efficiency in Model-Free Reinforcement Learning from Images," Jul. 09, 2020, *arXiv*: arXiv:1910.01741. Accessed: Mar. 20, 2023. [Online]. Available: http://arxiv.org/abs/1910.01741

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 28, 2017, *arXiv*: arXiv:1707.06347. Accessed: Jul. 27, 2024. [Online]. Available: http://arxiv.org/abs/1707.06347

[31] R. Rubinstein, "The Cross-Entropy Method for Combinatorial and Continuous Optimization," *Methodol. Comput. Appl. Probab.*, vol. 1, no. 2, pp. 127–190, Sep. 1999, doi: 10.1023/A:1010091220143.

[32] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by Cheating," in *Proceedings of the Conference on Robot Learning*, PMLR, May 2020, pp. 66–75. Accessed: May 30, 2023. [Online]. Available: https://proceedings.mlr.press/v100/chen20a.html

[33] J. Jin *et al.*, "Resilient Legged Local Navigation: Learning to Traverse with Compromised Perception End-to-End," Oct. 05, 2023, *arXiv*: arXiv:2310.03581. Accessed: May 21, 2024. [Online]. Available: http://arxiv.org/abs/2310.03581

[34] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile But Safe: Learning Collision-Free High-Speed Legged Locomotion," May 17, 2024, *arXiv*: arXiv:2401.17583. Accessed: May 21, 2024. [Online]. Available: http://arxiv.org/abs/2401.17583

[35] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning," Aug. 19, 2022, *arXiv*: arXiv:2109.11978. Accessed: May 20, 2024. [Online]. Available: http://arxiv.org/abs/2109.11978

[36] M. Mittal *et al.*, "Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments," Feb. 16, 2024, *arXiv*: arXiv:2301.04195. Accessed: Nov. 01, 2024. [Online]. Available: http://arxiv.org/abs/2301.04195

[37] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware," Apr. 23, 2023, *arXiv*: arXiv:2304.13705. Accessed: Aug. 23, 2024. [Online]. Available: http://arxiv.org/abs/2304.13705

[38] D. P. Kingma and J. Lei, "Adam: A Method for Stochastic Optimization," 2015.

[39] P. Anderson *et al.*, "On Evaluation of Embodied Navigation Agents," Jul. 17, 2018, *arXiv*: arXiv:1807.06757. Accessed: Oct. 19, 2023. [Online]. Available: http://arxiv.org/abs/1807.06757

[40] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Adv. Neural Inf. Process. Syst.*, vol. 1, 1988, Accessed: Nov. 19, 2024. [Online]. Available: https://proceedings.neurips.cc/paper/1988/hash/812b4ba287f5ee0bc9d43bbf5bbe87fb-Abstract.html

[41] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the Design Space of Diffusion-Based Generative Models," Oct. 11, 2022, *arXiv*: arXiv:2206.00364. Accessed: Sep. 02, 2024. [Online]. Available: http://arxiv.org/abs/2206.00364

[42] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg, "Consistency Policy: Accelerated Visuomotor Policies via Consistency Distillation," May 13, 2024, *arXiv*: arXiv:2405.07503. Accessed: May 15, 2024. [Online]. Available: http://arxiv.org/abs/2405.07503

[43] D. Kim *et al.*, "Consistency Trajectory Models: Learning Probability Flow ODE Trajectory of Diffusion," Mar. 30, 2024, *arXiv*: arXiv:2310.02279. Accessed: Sep. 02, 2024. [Online]. Available: http://arxiv.org/abs/2310.02279

[44] X. Huang *et al.*, "DiffuseLoco: Real-Time Legged Locomotion Control with Diffusion from Offline Datasets," Apr. 30, 2024, *arXiv*: arXiv:2404.19264. Accessed: May 18, 2024. [Online]. Available: http://arxiv.org/abs/2404.19264

[45] A. Chang *et al.*, "Matterport3D: Learning from RGB-D Data in Indoor Environments," Sep. 18, 2017, *arXiv*: arXiv:1709.06158. Accessed: Nov. 19, 2024. [Online]. Available: http://arxiv.org/abs/1709.06158