2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Embodiment Randomization for Cross Embodiment Navigation

Pranav Putta[1] Gunjan Aggarwal[1] Roozbeh Mottaghi[2] Dhruv Batra[1]
Naoki Yokoyama[1] Joanne Truong[1] Arjun Majumdar[1]
[1]Georgia Institute of Technology [2]FAIR, Meta AI

***Abstract*— We present Embodiment Randomization, a simple, inexpensive, and intuitive technique for training robust behavior policies that can be transferred to multiple robot embodiments. While prior works require real-world data from multiple robots, or complex algorithmic adjustments to address the challenge of embodiment generalization, our approach leverages the power of simulation and large-scale reinforcement learning and can be easily integrated within existing policy learning methods. We show that policies trained with embodiment randomization implicitly perform system identification, enabling them to adapt to new embodiments during deployment. Our approach not only shows significant improvements in adapting to novel robot configurations, but also in generalizing from simulation to reality and contending with real-world perturbations, highlighting the potential of embodiment randomization in creating versatile and adaptable robotic navigation policies.**

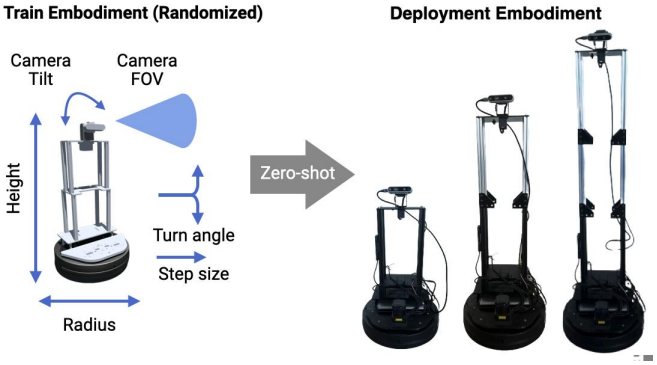

Fig. 1: Embodiment randomization enables zero-shot policy transfer to novel embodiments in new, unseen environments.

I. INTRODUCTION

Modern robots come in various shapes and sizes, and use different sensor suites to observe their environment (*e.g.*, Fetch [1], LoCoBot [2], Stretch [3], etc.). By contrast, when training behavior policies for robots, these differences are typically ignored and a single robot embodiment (height, radius, camera parameters, *etc.*) is used. As a result, such policies only work for the fixed embodiment used in training, and generalize poorly when even minor changes to the robot's embodiment are introduced, such as changing the robot's height or camera field-of-view. However, embodiment changes are inevitable in the real world as new generations of robots are developed or existing robots are updated and modified for new applications. This raises the question: *how do we train behavior policies that can adapt, as needed, to new embodiments during deployment?*

Recent work has proposed a number of techniques for addressing this embodiment generalization challenge [4]–[7]. One prominent method is to leverage large offline datasets of real-world navigation trajectories gathered from robots operating across a large range of platforms and environments [4], [8]. Other works leverage simulation to learn robust low-level controllers for multiple robots [5], or condition the navigation policy using a learned robot embedding vector to adapt to robots with different dynamics [7]. While these methods show promise, they often require large amounts of real-world data and/or introduce complexity in the form of additional (often explicit) conditioning inputs, action space normalization, or requiring data augmentation.

In this work, we remove the need for costly real-world training data and additional algorithmic complexity by instead leveraging simulation and large-scale reinforcement learning with *embodiment randomization* to train embodiment agnostic visual navigation policies that directly transfer from simulation to reality. In embodiment randomization a robot configuration is randomly sampled at the beginning of each training episode (in a similar manner to domain randomization [9]). Since modifying a robot's embodiment in simulation is trivial, embodiment randomization is a simple, inexpensive, and intuitive technique for addressing the embodiment generalization challenge. Embodiment randomization does not require any modifications to the policy architecture or learning algorithm, allowing it to be easily integrated within existing methods for learning behavior policies.

We empirically study embodiment randomization using the image-goal navigation (ImageNav) task [10]. First, we observe that on this task, policies trained using a fixed embodiment catastrophically fail to generalize to new robot configurations. However, with embodiment randomization, agents recover a substantial portion of this lost performance (+53.8% success rate). Furthermore, we study the robustness of embodiment randomized policies in an out-of-distribution "robot bump" scenario in which the robot is perturbed and its embodiment is altered during the course of a navigation episode in deployment. Unsurprisingly, we find that traditional, single embodiment policies fail to adapt in this scenario. However, embodiment randomized policies adapt successfully despite never being trained for dynamic conditions.

We additionally conduct experiments in several real-world environments using multiple robot embodiments and demonstrate successful zero-shot sim2real transfer of ImageNav policies trained with embodiment randomization, for the first time. Specifically, an embodiment randomized policy

succeeds in 65 of 81 trials (80.2%) in the real world across 3 different robot embodiments, closely matching navigation performance in simulation and outperforming prior works.

Through our analyses, we discover that policies trained with embodiment randomization implicitly perform system identification. Specifically, we find that many (but not all) of the robot's embodiment parameters can be reliably predicted from intermediate representations within policies trained with embodiment randomization, allowing these agents to adapt to new embodiments during deployment.

Overall, this work primarily focuses on providing (a) extensive empirical evidence to demonstrate that embodiment randomization is a simple, but highly effective technique for addressing the embodiment generalization challenge in both simulation and reality, and (b) an explanation for *why* embodiment randomization works: agents learn to implicitly perform system identification.

II. Related Work

A. Randomization in Simulation

Prior works have studied the effect of domain randomization in simulation. Peng *et al*. [11] introduce a methodology of randomization to the dynamics of the simulator during the training of a manipulator arm in order to improve transfer of the policy to the real world setting. Feng *et al*. [5] study the development of generalized locomotion controllers for quadrupedal robots. They present a simple methodology of applying morphology randomization by procedurally generating a diverse set of simulated robots during training, as well as randomizing environment dynamics to improve robustness to domain shifts during transfer to the real world. Finally, Truong *et al*. [7] present a method to train a shared navigation policy for the PointGoal Navigation task across multiple legged robots by combining the experience of 3 different robots in simulation. In this work, we take inspiration from Peng *et al*. [11]'s method of randomization in low-level control systems and adapt it to a complex, high-level navigation task. Lastly, in contrast to prior works such as Truong *et al*. [7], our work does not require any embodiment-specific inputs to the policy; rather, the policy learns to identify embodiment capabilities from context.

B. Heterogeneous Offline Data

Prior work has also studied the effectiveness of training navigation policies via offline data collected from real robots. Shah *et al*. [4], [12] train a general navigation policy by combining offline trajectories from heterogenous robot datasets, conditioning the policy on past observations (used to infer a robot's embodiment) and using a normalized action space (to generalize actions across embodiments). They use the learned policy in conjunction with a topological graph of the environment collected prior to inference to navigate between subgoals in the graph. Moreover, Hirose *et al*. [8] extend this work with ExAug which introduces geometric augmentations of offline trajectory data collected from robots on a diverse set of platforms and environments. They do this by extracting 3D information in the form of a point cloud to generate synthetic images to train robot-conditioned policies that can accept different robot parameters. Finally, recent work by Padalkar *et al*. [13] studies the effectiveness of improving robot learning models by training over large heterogeneous offline datasets of robot demonstrations. They train a single monolith policy, RT-2, that aims to efficiently adapt to new robots, tasks, and environments and test time. These works aim to remove the need for training separate navigation policies for every robot platform a practioner expects to operate, but often require a large amount of offline real-world data.

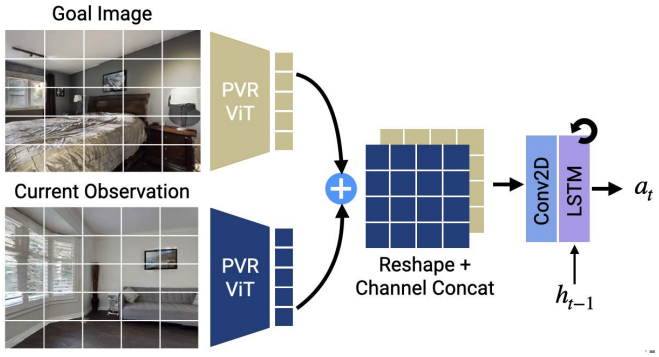

Fig. 2: Our navigation policy uses a pre-trained ViT to process the goal and current observation images (adapted from Yadav *et al*. [14]). The resulting goal and observation tokens from the ViT are reshaped and concatenated channel-wise. Next, the concatenated representation is passed through a convolutional layer, which is followed by a 2-layer LSTM that predicts the next action.

III. Methodology

We study embodiment randomization using the image-goal navigation task (Section III-A). Specifically, we train visual navigation agents using large-scale reinforcement learning (RL) (Section III-B). The key idea is varying the agent's embodiment in training (Section III-C), which allows the policy to adapt to different embodiments during deployment without *any* prior knowledge of the deployment configuration.

A. Task

In Image-Goal Navigation (ImageNav) [10], an agent receives RGB observations, and is tasked with navigating from a randomly sampled initial position in the environment to a goal location. The goal is provided as an RGB image captured from the goal location at a specified viewing angle. We consider navigation agents that use four discrete actions: `STEP_FORWARD`, `TURN_LEFT`, `TURN_RIGHT`, and `STOP`. Agents are successful if they call `STOP` within 1m of the goal. We report two navigation metrics: Success Rate (SR ↑) and Success weighted by Path Length (SPL ↑) [15].

B. Navigation Agent Architecture and RL Reward

Agent Architecture. We use the navigation agent architecture illustrated in Figure 2, which is adapted from Yadav *et al*. [14]. Specifically, the RGB goal and observation images are processed with a pre-trained visual image transformer (ViT) [16]. We use pre-trained weights from [14], which were obtained via self-supervised learning using the masked autoencoding (MAE) [17] training objective on images from

TABLE I: Training and Evaluation Embodiments

Agent Property	Single Embodiment	Multi-Embodiment					
		Small		Medium		Large	
		In-Distribution	Out-of-Distribution	In-Distribution	Out-of-Distribution	In-Distribution	Out-of-Distribution
Height (m)	0.61	[0.60, 0.91]	[0.15, 0.60] ∪ [0.91, 2.0]	[0.56, 1.19]	[0.15, 0.56] ∪ [1.19, 2.0]	[0.25, 1.5]	[0.15, 0.25] ∪ [1.5, 2.0]
Radius (m)	0.18	[0.18, 0.23]	[0.05, 0.18] ∪ [0.23,0.4]	[0.15, 0.25]	[0.05, 0.15] ∪ [0.25, 0.4]	[0.1, 0.3]	[0.05, 0.1] ∪ [0.3, 0.4]
Step Size (m)	0.2	[0.14, 0.21]	[0.05, 0.14] ∪ [0.21, 0.4]	[0.11, 0.24]	[0.05, 0.11] ∪ [0.24, 0.4]	[0.05, 0.3]	[0.3, 0.4]
Turn Angle (deg.)	30	[28.75, 41.25]	[5, 28.75] ∪ [41.25, 70]	[22.5, 47.5]	[5, 22.5] ∪ [47.5, 70]	[10, 60]	[5, 10] ∪ [60, 70]
Camera Tilt (deg.)	0	[-5, 5]	[-35, -5] ∪ [5, 35]	[-10, 10]	[-35, -10] ∪ [10, 35]	[-20, 20]	[-35, -20] ∪ [20, 35]
Camera FoV (deg.)	55	[52, 70]	[30, 52] ∪ [70, 150]	[48.5, 85.5]	[30, 48.5] ∪ [85.5, 150]	[45, 120]	[30, 45] ∪ [120, 150]

the HM3D [18] and Gibson [19] datasets. We use color jitter and random horizontal/vertical shift as data augmentations during training (following [14]) to improve the policies robustness. Next, the encoded RGB and goal images are fused channel-wise and fed through a convolutional layer (inspired by Sun *et al*. [20]). Then, the visual representation is flattened and projected into a latent state vector using a linear layer. Finally, the latent state vector is consumed by a 2-layer LSTM [21] which predicts a distribution over the discrete action space. We train agents with reinforcement learning using DD-PPO [22] for up to 1 billion frames of experience in the Habitat simulator [23], [24] using 800 training scenes from HM3D [18].

Reward Specifications. We use the following reward function for RL training (adapted from [14]):

$$\begin{aligned} R(s_t, a_t, s_{t+1}) = c_s \times ([d_t < r_g] \;\&\; [a_t = \texttt{STOP}]) \\ + c_a \times ([\theta_t < \theta_g] \;\&\; [a_t = \texttt{STOP}]) \\ + (d_{t-1} - d_t) + (\theta_{t-1} - \theta_t) \\ - \gamma - \beta \times [\text{collide}_{t+1}] \end{aligned} \tag{1}$$

c_s is a sparse reward (set to 5.0) for successfully stopping at the goal, which happens when the agent's distance d_t is within the goal radius r_g (set to 0.8m during train time) and the agent calls $a_t = \texttt{STOP}$. c_a is an additional sparse reward term (set to 5.0) to incentive the agent to stop in the correct goal heading θ_g within a threshold θ_t (set to 25°). At each step, the agent also receives a dense reward for getting closer to the goal, calculated as $(d_{t-1} - d_t)$, and for turning toward the correct heading angle, calculated as $(\theta_{t-1} - \theta_t)$. The latter term is only active when the agent is within the goal radius ($d_t \leq r_g$). Specifically, when the agent is not in the goal radius ($d_t > r_g$), the angle to the goal is fixed to π as a constant penalty, but when the agent is inside the goal radius, the actual angle to the goal is used. The agent also receives a slack penalty γ at every step (set to 0.01) and a collision penalty scaled by β (set to 0.03). We train without sliding in simulation. We find that turning sliding off and a collision penalty are necessary for successful transfer of the policy into a real-world agent, consistently with the findings from [25], [26].

C. Embodiment Randomization

In training we randomly sample the agent's height, radius, step size, turn angle, camera tilt, and camera field-of-view (FoV) from a continuous range of values at the start of every episode. We experiment with three different embodiment parameter ranges, referred to as "small", "medium", and "large" ranges in Table I. These ranges are used to study the navigation behaviors that emerge with different levels of randomization. We select these ranges such that the LoCoBot Single Embodiment (SE) parameters are included within each range for comparison, and aimed to randomize over all possible realistic ranges of robot sizes that one might encounter in real-world environments. We compare our policies trained with embodiment randomization to policies trained on a single, fixed embodiment (using the LoCoBot [2] configuration).

IV. Experimental Findings

We conduct a series of experiments to study the effectiveness of embodiment randomization in training navigation policies that adapt to many robot embodiments. We evaluate in 3 real-world environments and 100 simulation environments from the HM3D [18] validation split, resulting in over 300k unique evaluation episodes. We report the average Success Rate (SR) and Success weighted by Path Length (SPL) over 3 seeds.

We study the following research questions (RQs):

1) Can embodiment randomized policies match single embodiment policy performance?
2) Can embodiment randomized policies successfully operate over the full range of embodiments seen in training?
3) Can embodiment randomized policies generalize to new, out-of-distribution embodiments that were not seen in training?
4) Can embodiment randomized policies generalize to scenarios in which the robot's embodiment changes during navigation?
5) Can embodiment randomized policies generalize, zero-shot, from simulation to reality and is performance competitive with state-of-the-art baselines?
6) Do embodiment randomized policies implicitly perform system identification?

To answer these questions, we compare the performance of three different multi-embodiment (ME) policies against a single embodiment (SE) policy that was trained solely with the LoCoBot [2] specifications. The multi-embodiment agents sample new embodiment parameters at the beginning of every episode in training. Specifically, we use the small, medium, and large parameter ranges specified in Table I column 3. Note that the LoCoBot embodiment is within all of the multi-embodiment parameter ranges. We evaluate all

of the agents in three evaluation settings: (1) single embodiment using LoCoBot parameters (Figure 3), (2) multiple embodiments in-distribution to the ME training (Figure 4), and (3) multiple embodiments out-of-distribution to the ME training (Figure 5).

A. *In-Distribution Embodiment Results*

RQ 1: **Can embodiment randomized policies match single embodiment policy performance?**

First, we compare the single embodiment (SE) and multi-embodiment (ME) policy performance using the LoCoBot [2] embodiment at test-time. The results in Figure 3 show that all of the policies achieve similar navigation performance in this setting. Specifically, we observe a -1.9% drop in SR and +2.0% gain in SPL for the ME (Large) policy over the SE policy. In general, SR slowly degrades as we increase the embodiment randomization parameter range in training (-0.3%, -0.9%, -1.9%), while SPL improves (+1.2%, +0.5%, +2.0%). *These results demonstrate that embodiment randomized policies can approximately match the performance of a single embodiment policy even when the evaluation is restricted to the configuration targeted by single embodiment training.*

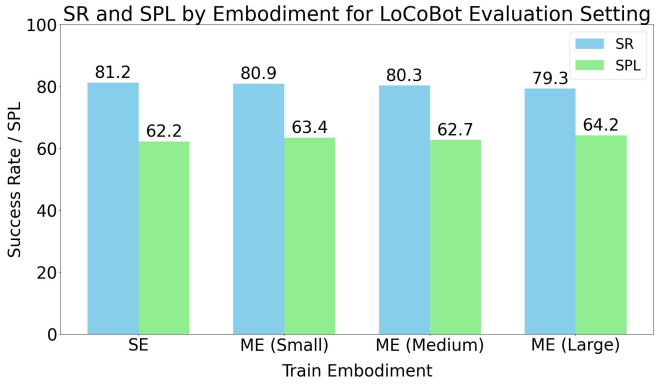

Fig. 3: We evaluate both the single embodiment (SE) and multi-embodiment (ME) policies using LoCoBot embodiment. We find that all the ME policies are able to outperform the SPL of the SE policy and closely match the success rate of the SE policy.

RQ 2: **Can embodiment randomized policies successfully operate over the full range of embodiments seen in training?**

Next, we evaluate the policies using embodiments within the small, medium, and large embodiment parameter ranges at test-time. In Figure 4, we find that the SE policy performance substantially drops as the embodiment parameter range grows (from 81.2% SR in the LoCoBot setting to 32.7% SR in the large in-distribution setting). These results demonstrate that single embodiment training is insufficient for adapting to new test-time embodiments, and suggests that separate SE policies need to be trained for every embodiment expected at test time.

By contrast, the ME policies show strong performance over the parameter ranges used in their respective training setups. Specifically, we observe minimal reductions in success rate with ME training (-0.7% SR for small, -1.3% for medium, and -1.7% for the large range). Finally, in Figure 4, we observe that the ME (Large) policy substantially improves over the SE policy in the Large In-Distribution parameter range (+44.9% gain in SR). *These results indicate that expanding the set of robot configurations used in training via embodiment randomization allows a single policy to successfully operate over a wide range of test-time embodiments.*

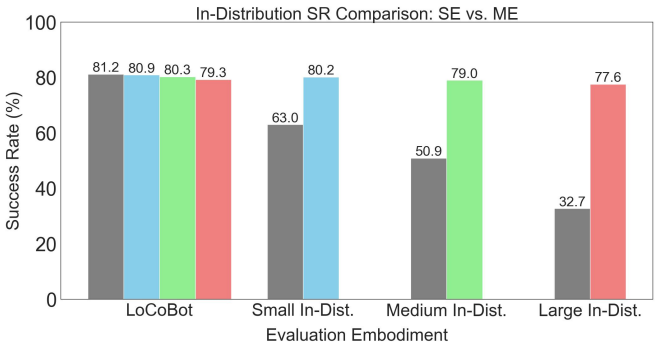

Fig. 4: We evaluate the single embodiment (SE) and multi-embodiment (ME) policies using the small, medium and large embodiment parameter ranges. We find that the SE policy is not able to adapt to the varying evaluation embodiments, while the ME policies remain robust.

B. *Out-of-Distribution Embodiment Generalization Results*

RQ 3: **Can embodiment randomized policies generalize to new, out-of-distribution embodiments that were not seen in training?**

In Figure 4, we study the performance of policies on embodiments outside of the training distribution. As seen in the previous section, we find that the SE policy struggles – e.g., dropping to 16.4% SR (Large Out-Distribution), a decrease of -64.8% from the 81.2% SR in the LoCoBot settings. By contrast, the ME policies consistently outperform the SE policy, with the ME (Large) agent achieving a 70.2% SR, which is a decrease of only -9.1% from the LoCoBot setting, despite the out-of-distribution evaluation conditions.

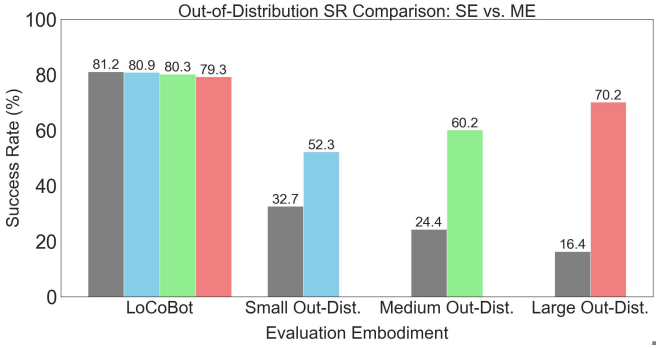

Fig. 5: We evaluate the single embodiment (SE) and multi-embodiment (ME) policies using the small, medium and large embodiment parameter ranges outside of the training distribution. We find that the SE policy catastrophically fails to generalize. Training over the largest range of parameters (ME-Large) leads to the best performance.

Apartment Lounge Workspace

Fig. 6: We test our policy in 3 real-world environments.

In general, we find that ME policies still degrade when evaluated further out of distribution. Furthermore, the ME (Small) and ME (Medium) policies suffer more than ME (Large) (dropping -28.6% (Small) and -20.1% (Medium) vs. -9.1% (Large) from their LoCoBot setting performance). *In summary, the evidence from the last two sections suggests that it is best to train with embodiment randomization over the full range of parameters expected during deployment.*

Lastly, Figure 7 presents a fine-grained analysis of navigation performance over smaller embodiment parameter ranges. We observe that smaller FoV and smaller step-sizes are more difficult for all policies. Qualitatively, we observe that in lower FoV settings, the ME policy often spins in place before making movement decisions as reducing FoV can require accumulating more information about the surrounding environment in order to make an informed decision. We speculate that this is a cause of these parameters' substantial impact on evaluation performance. We see that our ME policies learn this optimal behavior to adapt to robots with limited FoV. A future line of work from this analysis would aim to study a deeper understanding of which parameters play the most important role in agent performance.

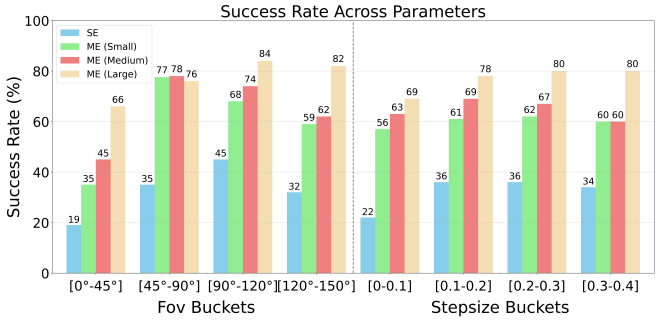

Fig. 7: Success policies conditioned on FoV and step size.

RQ 4: **Can embodiment randomized policies generalize to scenarios in which the robot's embodiment changes during navigation?**

We further evaluate the ability of the policies to adapt to changes in embodiment parameters during an episode. There are many instances in which a robot in the real-world may encounter perturbations that affect things such as camera tilt and height (a "bump" that causes misalignment of the sensor), camera FOV (occlusion from some perturbation), step size changes (changes in speed or sampling rate from the camera). We test the robustness of the ME and SE policies against this "bump" perturbation in simulation. First, we start both agents at the same start location and identical goal conditions with fixed embodiment parameters. Next, the agents both encounter a "camera bump" 20 steps into the episode, in which random embodiment change is sampled and applied to the camera. For example, the camera tilt or height might be moved up/down, or the FOV may get smaller (Figure Section IV-B). From Table II, we see that the ME policy is resilient to changes during an episode, with the largest drop being -13.2% SR (columns 1-2 for ME-Large). In contrast, the perturbation causes significant detriment to the performance of the SE policy, with the largest drop being -57.4% SR (columns 1 and 3).

TABLE II: We report the success rate for ImageNav evaluation performance during perturbation experiments. The ME policies are robust to perturbations, while the SE policies experience a significant performance drop to FoV occlusions.

Train Embodiment	No Perturbations	Camera Bump	FoV Occlusion	Step Size	Turn Increment
SE	81.2	36.2	23.8	68.0	62.3
ME (Small)	80.9	64.2	45.1	79.1	80.1
ME (Medium)	80.3	63.9	65.2	80.1	79.8
ME (Large)	79.3	66.1	75.3	78.1	79.0

C. Real-World Performance

RQ 5: **Can embodiment randomized policies generalize, zero-shot, from simulation to reality and is performance competitive with state-of-the-art baselines?**

Next, we study the performance of policies trained entirely in simulation and then zero-shot transferred onto a LoCoBot operating in the real-world.

Real-world Setup. In order to test different embodiment characterizations, we create 3 different embodiment settings of LoCoBot of by modifying the height, radius, and camera parameters (Table III, and Figure 1). To emulate taller and shorter agents, we move the camera height using extruded aluminum beams, as shown in Figure 1. We also modify the camera tilt angle to emulate a camera mounted on the top of the robot. We choose 3 indoor environments for testing: an apartment, lounge, and office workspace (Figure 6). For

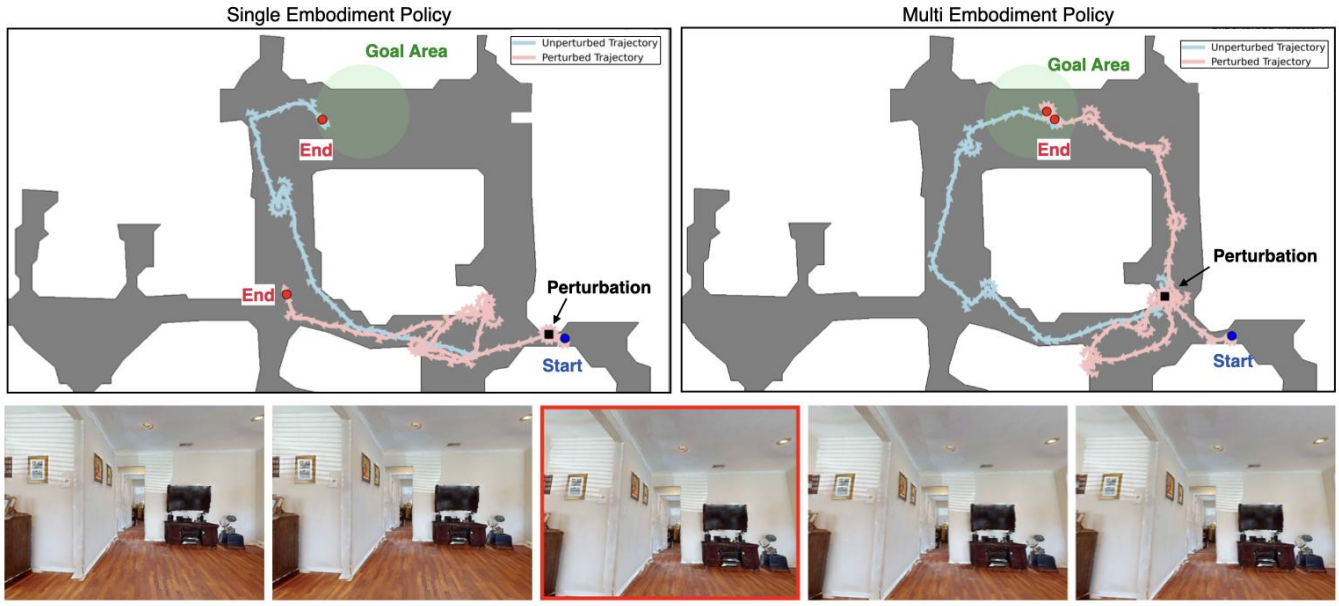

Fig. 8: Perturbation experiments: Agent trajectories from the single embodiment policy (left) and multi embodiment policy (right) when a camera "bump" occurs during the 20th environment step (at the black square marker, and image with red border). The single embodiment policy is not able to adapt to these changes and struggles to recover. Notice that the multi embodiment policy initially struggles, but is able to adapt and choose the proper route. We notice that in many situations, the single embodiment policy often chooses to stop early, suggesting that its goal detection is severely degraded in the perturbation experiments.

each, we select 3 different episodes by selecting a start and goal location. Each episode is of varying difficulty (easy, medium, hard) where easy episodes are $d \leq 3m$ in path length, medium episodes are $3m \leq d \leq 5m$, and hard episodes are $5m \leq d \leq 10m$. For each episode, we run 3 trials of each policy.

TABLE III: Real World Embodiment Parameters

Agent Property	Small (LoCoBot)	Medium	Tall
Height (m)	0.61	0.91	1.21
Radius (m)	0.18	0.18	0.18
Step Size (m)	0.2	0.25	0.25
Turn Angle (deg.)	30	30	30
Camera Tilt (deg.)	10	0	-15
Camera FOV (deg.)	55	55	55

Baseline Methods. Our real-world evaluations compare the embodiment randomized ME (Large) policy with three baselines: the single embodiment (SE) policy and two cross-embodiment policies from prior work, ViNT [12] and ExAug [8]. The ViNT and ExAug policies are trained with supervised learning on a large, diverse offline set of visual navigation data. ExAug additionally trains with 3D visual augmentations to improve policy performance. Both methods require a pre-exploration stage in which a topological graph of environment is constructed and subsequently used for navigation. Specifically, the topological graph is used to help localize the agent and for path planning. We report results with two forms of pre-exploration. In the first, termed "Path Follower," the topological graph only consists of ground-truth waypoints from the start to the goal. The "Path Follower" results are provided as an upper bound on performance, since the agent is provided with oracle information about the path to the goal. In the second, termed "Dijkstra," the graph contains additional nodes from a full pre-exploration of the environment, and Dijkstra's [27] algorithm is used for planning.

Results. In Table IV, we find that the multi-embodiment (ME) policy consistently outperforms each of the (non-oracle) baselines in all three environments and across all three embodiments. *Overall, the ME agent succeeds in 65 of 81 trials (80.2%), closely matching policy performance in simulation.* We find that the single embodiment (LoCoBot) policy performs well in short (LoCoBot) setting, but fails to generalize to the medium and tall embodiments. The ViNT (Path Following) and ExAug (Path Following) agents are generally successful at navigating the ground-truth path. However, we see a large drop in performance for the full versions ViNT (Dijkstra) and ExAug (Dijkstra) variations. Next, we provide a qualitative assessment of the behaviors observed in the real-world trials.

1) Qualitative Assessment: Single vs. Multi-Embodiment

The multi-embodiment (ME) policy is consistently on par or better than the single embodiment (SE) agent. We speculate that three factors contribute to this gap. First, ME policies train over a range of radii, which requires maintaining a large distance from objects to avoid collisions (especially in the maximum radius setting). On the other hand, the SE policy was trained with a single radius, and thus narrowly misses objects in training to optimize the navigation path. As a result, in the real-world, we observe the SE agent is more likely to bump into objects. Second, we observe that the ME policy more often "spins" in place, which we hypothesize was learned to compensate for narrower fields-of-view (FoV) seen in training. As a result, the ME agent

TABLE IV: Sim2Real Evaluation Results. Note that the greyed out rows use a path following agent where the topological map is given in the order of the trajectory from the start to goal point. We include these results for reference, but are not directly comparable to the ImageNav [10] task we study.

Eval Environment	Policy	Uses Map	Eval Embodiment		
			Short SR ↑	Medium SR ↑	Tall SR ↑
Apartment	ViNT (Path Follower)	✓	9/9	7/9	7/9
	ExAug (Path Follower)	✓	9/9	7/9	7/9
	ViNT (Dijkstra)	✓	1/9	0/9	0/9
	ExAug (Dijkstra)	✓	1/9	1/9	0/9
	Single Embodiment	-	7/9	0/9	0/9
	Multi Embodiment	-	**8/9**	**8/9**	**7/9**
Lounge	ViNT (Path Follower)	✓	9/9	8/9	8/9
	ExAug (Path Follower)	✓	9/9	9/9	9/9
	ViNT (Dijkstra)	✓	2/9	3/9	2/9
	ExAug (Dijkstra)	✓	3/9	3/9	3/9
	Single Embodiment	-	7/9	2/9	0/9
	Multi Embodiment	-	**8/9**	**8/9**	**6/9**
Office	ViNT (Path Follower)	✓	6/9	4/9	4/9
	ExAug (Path Follower)	✓	6/9	6/9	6/9
	ViNT (Dijkstra)	✓	2/9	2/9	0/9
	ExAug (Dijkstra)	✓	3/9	3/9	3/9
	Single Embodiment	-	6/9	1/9	0/9
	Multi Embodiment	-	**8/9**	**7/9**	**5/9**

often spots the goal before choosing a direction to explore, leading to successful navigation. Third, the SE policy, when deployed on the tall agent, often ran into objects below its camera vertical FoV. We speculate that the ME agent learns to better track objects and avoid such situations. Overall, the SE policy failed nearly all tests on different embodiment parameters due to collisions with the environment or simply unexplainable exploration behavior likely due to the distribution shift from its training data.

2) Qualitative Assessment: ViNT and ExAug vs. ME

We observed that the ViNT and ExAug baselines primarily suffered from poor error recovery. Specifically, the agents failed when small deviations from the path demonstration led the robot to explore areas of the environment far from the goal and not in the topological graph. Similarly, if the robot accidentally bump into an object, such as the bottom of a chair, the policy was unable to circumnavigate after the collision. By contrast, we speculate that the embodiment randomized policies learned error recovery due to the large-scale of RL training (up to 1B steps of experience); we observed that when the ME agent got stuck and was unable to move forward it would turn and find a different path.

D. System Identification

RQ 6: **Do embodiment randomized policies implicitly perform system identification?**

Next, we attempt to better understand how ME policies generalize to new embodiments. We hypothesize that embodiment randomized agents may implicitly learn to perform system identification (i.e., they infer the embodiment parameters during navigation). To test this, we train a non-linear probe to predict embodiment parameters, such as camera FOV, step size, and camera tilt from the internal representations of the learned policy. Specifically, each probe is a 2-layer MLP that takes as input the last hidden state from the LSTM at the end of the episode and outputs a predicted embodiment parameter. We divide each embodiment parameter range into 4 classes, which are predicted by the MLP probe.

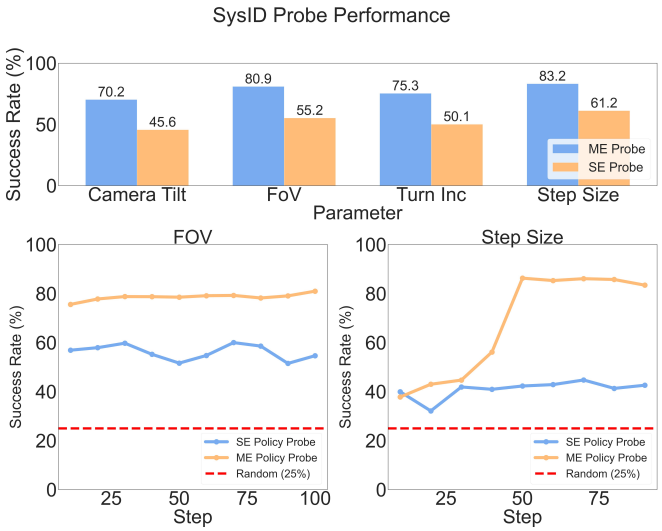

Fig. 9: System identification probes are used to understand the effect of embodiment changes on the policy hidden states. We train several probes at different steps of episode rollouts to understand how the policy's system identification performance evolves over time.

In Figure 9, we see that the embodiment parameters are predicted with high accuracy (well above chance; 25%), suggesting that agents are performing implicit system identification. In particular, parameters such as step size, turn increment, and camera FOV are detected with high accuracy (83.2%, 75.3%, 80.9% respectively). On the other hand, we observe that MLP probes trained on the hidden states of the single embodiment policy do not have such easily separable information. *These results suggest that the ME policies implicitly learn to be embodiment-aware, which may explain how these agents adapt to novel embodiments during deployment.*

V. Conclusion

In this work, we present Embodiment Randomization for training visual navigation policies that demonstrate robust performance across a variety of embodiments during deployment in simulation and the real-world. In contrast to prior works that rely on extensive offline datasets from multiple real-world robots or complicate the process with conditional inputs and data augmentation, our approach leverages the idea that new robot embodiment can be easily simulated, making embodiment randomization a simple, inexpensive and intuitive technique for addressing the embodiment generalization challenge. We find that policies trained with embodiment randomization implicitly perform system identification. Through probe experiments using the intermediate representation within the policy, we observe that the policy

is able to to reliably predict many (but not all) of the robot's embodiment parameters, allowing the policy to adapt to new embodiments during deployment. We additionally show that embodiment randomization also enables policies to dynamically adapt to unexpected changes as the robot is navigating. We hope our work will enable others to develop methods that are readily deployable across diverse robotic embodiments.

ACKNOWLEDGMENT

The Georgia Tech effort was supported in part by ONR YIP and ARO PECASE. JT was supported by an Apple Scholars in AI/ML PhD Fellowship. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government, or any sponsor.

REFERENCES

[1] "Fetch robotics." http://fetchrobotics.com/.

[2] A. Murali, T. Chen, K. V. Alwala, D. Gandhi, L. Pinto, S. Gupta, and A. Gupta, "Pyrobot: An open-source robotics framework for research and benchmarking," *arXiv preprint arXiv:1906.08236*, 2019.

[3] C. C. Kemp, A. Edsinger, H. M. Clever, and B. Matulevich, "The design of stretch: A compact, lightweight mobile manipulator for indoor human environments," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3150–3157.

[4] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "GNM: A General Navigation Model to Drive Any Robot," in *arXiV*, 2022. [Online]. Available: https://arxiv.org/abs/2210.03370

[5] G. Feng, H. Zhang, Z. Li, X. B. Peng, B. Basireddy, L. Yue, Z. SONG, L. Yang, Y. Liu, K. Sreenath, and S. Levine, "Genloco: Generalized locomotion controllers for quadrupedal robots," in *Proceedings of The 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 14–18 Dec 2023, pp. 1893–1903.

[6] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," in *Robotics: Science and Systems (RSS)*, 2017.

[7] J. Truong, D. Yarats, T. Li, F. Meier, S. Chernova, D. Batra, and A. Rai, "Learning Navigation Skills for Legged Robots with Learned Robot Embeddings," in *International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[8] N. Hirose, D. Shah, A. Sridhar, and S. Levine, "Exaug: Robot-conditioned navigation policies via geometric experience augmentation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 4077–4084.

[9] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[10] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. K. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning," *ICRA*, 2017.

[11] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[12] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine, "ViNT: A foundation model for visual navigation," in *7th Annual Conference on Robot Learning*, 2023. [Online]. Available: https://arxiv.org/abs/2306.14846

[13] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models," *arXiv preprint arXiv:2310.08864*, 2023.

[14] K. Yadav, A. Majumdar, R. Ramrakhya, N. Yokoyama, A. Baevski, Z. Kira, O. Maksymets, and D. Batra, "Ovrl-v2: A simple state-of-art baseline for imagenav and objectnav," *arXiv preprint arXiv:2303.07798*, 2023.

[15] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[17] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 000–16 009.

[18] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. M. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang, *et al.*, "Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[19] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson Env: Real-World Perception for Embodied Agents," in *CVPR*, 2018.

[20] X. Sun, P. Chen, J. Fan, J. Chen, T. Li, and M. Tan, "Fgprompt: Fine-grained goal prompting for image-goal navigation," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] E. Wijmans, A. Kadian, A. S. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames," in *International Conference on Learning Representations (ICLR)*, 2020.

[23] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *International Conference on Computer Vision (ICCV)*, 2019.

[24] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra, "Habitat 2.0: Training home assistants to rearrange their habitat," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[25] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, "Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance?" in *RA-L*, 2020.

[26] J. Truong, M. Rudolph, N. H. Yokoyama, S. Chernova, D. Batra, and A. Rai, "Rethinking sim2real: Lower fidelity simulation leads to higher sim2real transfer in navigation," in *Conference on Robot Learning*. PMLR, 2023, pp. 859–870.

[27] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 2022, pp. 287–290.