What Do We Learn from a Large-Scale Study of Pre-Trained Visual Representations in Sim and Real Environments?

Sneha Silwal[1*], Karmesh Yadav[2*], Tingfan Wu[1*], Jay Vakil[1*], Arjun Majumdar[2*], Sergio Arnaud[1*], Claire Chen[3], Vincent-Pierre Berges[1], Dhruv Batra[1,2], Aravind Rajeswaran[1], Mrinal Kalakrishnan[1], Franziska Meier[1†] and Oleksandr Maksymets[1†]

***Abstract*— We present a large empirical investigation on the use of pre-trained visual representations (PVRs) for training downstream policies that execute real-world tasks. Our study spans five different PVRs, two different policy-learning paradigms (imitation and reinforcement learning), and three different robots for 5 distinct manipulation and indoor navigation tasks. From this effort, we can arrive at three insights: 1) the performance trends of PVRs in the simulation are generally indicative of their trends in the real world, 2) the use of PVRs enables a first-of-its-kind result with indoor ImageNav (zero-shot transfer to a held-out scene in the real world), and 3) the benefits from variations in PVRs, primarily data-augmentation and fine-tuning, also transfer to the real-world performance. See project website[1] for additional details and visuals.**

I. INTRODUCTION

The design of *pre-trained visual representations* (PVRs) for sensorimotor control [1], [2], [3], [4], [5], [6], [7], [8] has been a promising development towards general-purpose visual perception for robotics, overcoming the problem of limited data for any one task and addressing the ability to generalize to new scenes. Today this involves training neural networks (on internet-scale data) that embed camera images into a visual feature space that can support policy learning for a variety of tasks (locomotion, navigation, static and mobile manipulation). To measure the potential of PVRs on being general purpose vision backbones – for a diverse set of Embodied AI and robotics tasks – we have to evaluate them on a wide range of robotics tasks. Yet doing so on hardware is impractical if not infeasible for most researchers in the community. As a result, past studies on the effectiveness of PVRs have either focused on broad systematic analyses in simulation [8] or narrow small-scale experiments on hardware [4], [5], [6].

Simulation lends itself to broad and diverse evaluations, which are needed to evaluate the generalization capabilities of PVRs. Yet it is unclear how much of the results from simulation carries over to the real world where we would hope to deploy PVRs. This gap in our collective scientific knowledge is best captured by the following research question – *Can we use simulation to evaluate and benchmark PVRs and carry over the results to hardware? In other words, is the performance of PVRs in simulation predictive of their performance in the real world?*

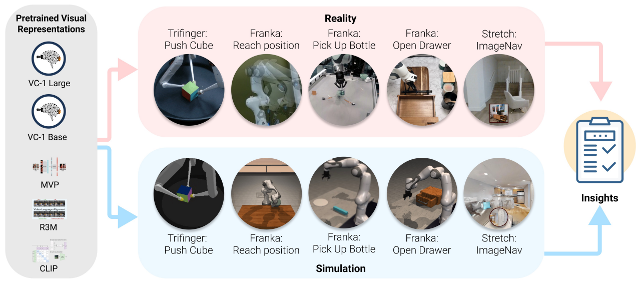


Fig. 1: We conducted 348 experiments with PVRs on five tasks (push cube, pick up bottle, open drawer, reach goal position, and image-goal navigation (ImageNav)), three robots (Trifinger, Franka, and Stretch), two learning paradigms (imitation and reinforcement learning), in simulation and reality.

To answer this question, we conducted the largest empirical study of PVRs in simulation and the real world to date: involving 5 PVRs (R3M [4], CLIP [9], MVP [6], and two variants of VC-1 [8]), 3 different robot platforms (TriFinger, Franka Arm and Stretch), 2 policy-learning paradigms (imitation and reinforcement learning), and 5 distinct tasks (pushing a cube, picking up a bottle, opening a drawer, reaching targets, and image-goal navigation (ImageNav)). For each task, we have a matching simulation and hardware setup - such that we can compare PVRs performance in the simulated setting and the real-world counterpart. Our empirical study comprised a total of 348 experiments and ***over 110 hours of robot experimentation*** on hardware. To ensure statistical significance, all experiments (except ImageNav) were conducted using three random seeds for training policies, enabling us to identify common trends and exceptions.

Our main findings and contributions are:

1. ***Sim Predictivity* of PVR evaluations on hardware.** First, we ask if the performance of PVR-based policies in simulation is indicative of trends in the real world, despite potential variations in absolute performance metrics. In prior work [10], *Sim Predictivity* is presented as a measure of how well performance translates from experiments conducted entirely in simulation to those carried out in real-world settings. Our hope is that if trends in simulation and real-world settings match, we can iterate faster on re-

*Equal Contribution †Equal Contribution

[1]Meta AI {ssilwal,tingfan,jayvakil,sergioarnaud, vincentpierre,aravraj,mrinal,fmeier,maksymets}@meta.com

[2]Georgia Institute of Technology {kyadav32,arjun.majumdar, dbatra}@gatech.edu

[3]Stanford University clairech@stanford.edu

[1]http://pvrs-sim2real.github.io/

search ideas in simulation with greater confidence. We evaluate *'sim-predictivity'* by training a set of policies in simulation and comparing their performance in simulation at evaluation time with similar policies trained with real demonstrations evaluated on hardware. In our experiments, we find a remarkably high degree of *'sim-predictivity'* (a correlation coefficient of 0.929) after basic alignment between the simulation and real-world setups (e.g. camera placement, checkpoint selection schemes, etc.). This suggests that recent progress in training PVRs is indeed materializing into broadly applicable real-world gains and affirms the value of simulation benchmarks for model selection, but practices such as reporting maximum validation split performance should be reconsidered.

2. ***Sim2Real Transfer* of PVR-based policies.** We do the same analysis of *'sim-predictivity'* with policies trained on demonstrations collected in simulation and evaluated on the real robot (addressing *Sim2Real Transfer*). We find that in this setting the performance only transfers well for the ImageNav task. While most tasks do not exhibit *Sim2Real Transfer*, an ImageNav agent is able to achieve a 90% success rate in a zero-shot manner, making it a first result of this kind with regards to ImageNav policies trained on the HM3D dataset and evaluated in the real world.
3. **Impact of Design Choices.** Finally, we study the effects of 1) model size, 2) fine-tuning the visual backbone, and 3) using data-augmentations on PVRs, with a focus on whether these results hold with our set of real-world experiments. Do the improvements we see in simulation also translate to hardware? We find the performance of most variations to be predictive of their real-world success. (Section IV-C).

II. RELATED WORK

Pre-trained Visual Representations. Inspired by the success of pre-trained representations for natural language processing and computer vision tasks, the robotics community has been exploring the use of PVRs to accelerate vision-based robotics, as opposed to the status quo of training models from scratch on in-domain data. [11] trains control policies with PVRs trained on large-scale computer vision datasets and show that in many cases, these policies are competitive or outperform policies trained with ground-truth state. [4], [5], and [6], introduced the R3M, VIP, and MVP models respectively, all of which target manipulation tasks and train representations on egocentric video data. [8] introduced the VC-1 model, which was trained on egocentric data as well as web images. They also introduced a benchmark called CortexBench, consisting of a range of 17 different control tasks including locomotion, navigation, and mobile manipulation, however, all of these tasks were in simulation. Our work does not contribute a new PVR; instead we evaluate 5 previously published PVRs on a range of corresponding simulation and real-world tasks.

Sim2Real Transfer. Compared to other domains, there is a dire lack of large, diverse, real-world robotics datasets. In this context, simulation is a highly appealing source of potentially unlimited data, but the use of this data to improve real-world performance can be challenging due to the domain gap between simulation and reality. There have been a number of approaches which attempt to bridge this domain gap [12], [13], [14], [15], [16], [17], [18]. In particular, for image-based navigation tasks, [16] has shown that modular approaches using semantics significantly outperform end-to-end RL approaches when transferred to the real world. [19] arrives at similar findings, also demonstrating poor performance of an end-to-end network with a prior PVR. In contrast, in this work we show that VC-1 Large [8], which is pre-trained on diverse real-world data (including indoor environments), can be fine-tuned with RL in simulation to achieve a 90% success rate on ImageNav in the real world.

[20] has also explored the role of PVRs in training policies for Sim2Real transfer on robot tasks. However, while they use a PVR trained with visual data from a specific simulated robot setup, and apply it to the same robot in the real world, our study focuses on PVRs trained on out-of-domain real-world data, analyzing their applicability to multiple different robot platforms and settings.

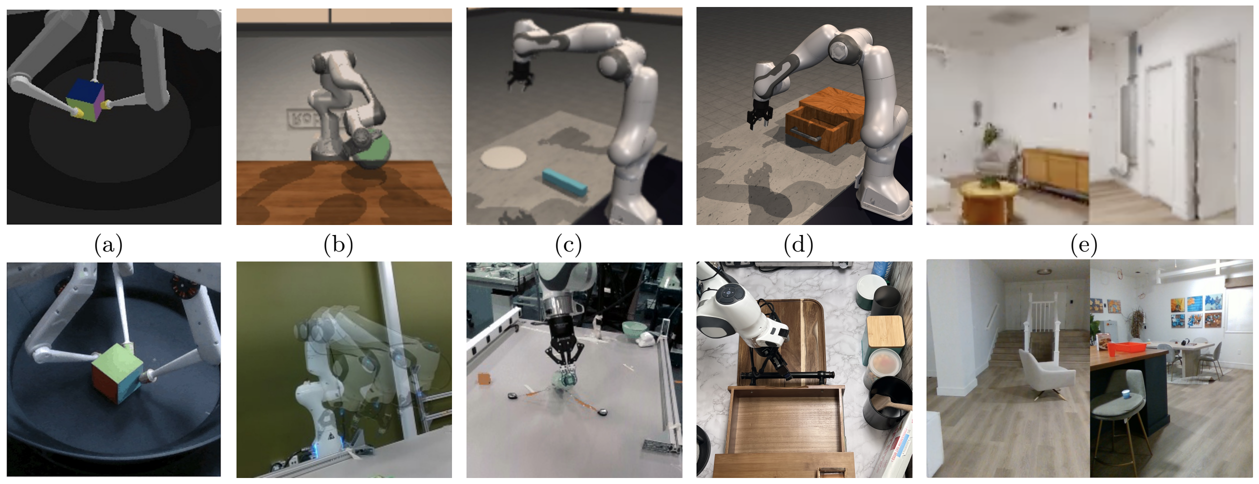
Sim2Real Predictivity. [10] investigates Sim2Real predictivity (what we refer to as *'sim-predictivity'*) in the context of a visual navigation task. They introduce the Sim2Real Correlation Coefficient (SRCC) metric, which we also use in this work. However, while they study Sim2Real predictivity for a single task in a single simulation environment, our work extends this to a broader set of simulation environments and tasks, and studies Sim2Real in the context of PVRs.

III. SIMULATED AND REAL-WORLD MANIPULATION AND NAVIGATION TASKS

In this section, we provide a brief description of the three robot platforms employed in our study and the associated tasks we set up to evaluate the PVRs both in simulation and real-world settings. In the simulation, we configure scenes to visually mimic the real world, including 3D models (rendering and collision meshes) of the robots, grippers, objects, and camera placement, but without rigorous system identification to match dynamics or camera extrinsic parameters.

A. Planar Cube Manipulation with a Trifinger Robot via Behavior Cloning

A TriFinger [21] robot (as seen in Figure 1) consists of three 3-DoF fingers in a shared workspace and is used for fine-grained manipulation of objects. We consider a planar cube re-positioning task designed specifically to require visual perception and use behavior cloning (BC) for policy learning. For both simulation and reality, we collect 30 demonstrations each from an expert policy to train a BC policy for 500 epochs with a learning rate of 10^{-4}. The architecture and training regime closely


(a) (b) (c) (d) (e)

(f) Trifinger (g) Random Reach (h) Bottle Pickup (i) Open Drawer (j) ImageNav

Fig. 2: **Top row:** Our 5 task in the simulation setting. **Bottom row:** Corresponding tasks on hardware.

matches [8].We used 3 seeds for training and evaluation and tested the policies on 12 different start and goal configurations,reporting the average success across seeds for both hardware and simulation experiments.

B. Manipulation Tasks with a Franka Robot via Behavior Cloning

We use a Franka Panda arm with a Robotiq gripper fitted with Festo Adaptive Fingers for 3 different tasks: Reach Pos, Pick Up Bottle, and Open Drawer. We used behavior cloning to train policies on this platform, both in simulation and the real world. For the real-world experiments, we used demonstrations (30 for Reach Pos and Pick Up Bottle, 150 for Open Drawer from [22]), the data was collected via human teleoperation using an Oculus Quest 2 controller [23]. For simulation, we collected the same number of demonstrations using a heuristic policy. For both real-world and simulation, we trained up to the same number of epochs (200 for Reach Pos and Pick Up, and 500 for Open Drawer) with 3 different seeds per task, evaluated the policies for 10 evaluation episodes per task per seed, and reported the average success. The success criteria for the Reach Pos task is determined by whether the final position of the arm is within 5cm of the predetermined pose, which is selected from sampling a random pose. For the Pickup Bottle task, a rollout is considered successful if the arm picked up the bottle at the end of an episode. The Open Drawer task is deemed successful if the drawer is opened more than 50% from its initial closed state.

The policies take as input the proprioception of the arm and gripper (joint angles and velocities), and the PVR-encoded 424×240 RGB image taken from a RealSense D435 camera. The learned policies output desired joint angles which are followed by the default PD control loop.

C. Visual Navigation with a Stretch Robot via Large-Scale Reinforcement Learning

For visual navigation, we use the Stretch robot, a mobile manipulator developed by Hello Robot. We pick the ImageNav task [24], where the agent must navigate to a goal location specified by an image in an unknown 3D environment. In contrast to the other tasks described above, we use reinforcement learning (RL) to train policies for ImageNav. Training is performed only in simulation, and the same policy is evaluated in simulation and the real world. For simulation, we leverage the Habitat [25] simulator and the HM3D scene dataset [26]. We use all 800 HM3D scenes for training, and a simulated replica of an unseen real-world apartment for evaluation, testing the generalization capabilities of the visual encoder and policy. For real-world evaluation, we set up 10 different episodes with different start and goal positions in the unseen apartment. These episodes ranged from easy to hard, incorporating challenges such as multi-room navigation, disambiguation between similar goals using the background, and navigating around a kitchen island to reach the goal viewpoint. The robot's task is to reach the goal location while avoiding obstacles and minimizing collisions.

In Habitat, we represent the Stretch robot as a cylindrical agent with a height of 1.41m and a radius of 0.3m. The Realsense RGBD camera is placed at a height of 1.31m from the ground and aligned vertically, outputting an image of size 640×480 (H×W) with a horizontal field of view of 42°. We create our own training episode dataset using the HM3D scene dataset [26], consisting of 800 scenes and 7.2 million training episodes. We allow the agent to take up to 1,000 steps within each episode. The episode is deemed successful if the agent reaches within 1m of the goal position and calls StopAction.

The policy takes as input PVR encodings of the current camera image and the goal image, both downsampled to a resolution of 160×120 and outputs discrete actions.

We train the agents in the HM3D environments for 600M timesteps (29.8k updates) using 320 environments running in parallel. Each environment collects up to 64 frames of experience, followed by 2 PPO [27] epochs utilizing 2 mini-batches. We use a learning rate of 2.5×10^{-4} and update the parameters using the AdamW optimizer with a weight decay of 10^{-6}. Unless stated

TABLE I: Success rate of policies on CortexBench and three hardware platforms (TriFinger, Franka, and Stretch) with results in reality (real) and simulation (sim).

	CortexBench	TriFinger		Franka		Franka		Franka		Stretch		Average	
	All benchmarks	Push cube		Reach pos.		Pick up bottle		Open drawer		ImageNav		All tasks	
# Method	(sim)	(sim)	(real)	(sim)	(real)	(sim)	(real)	(sim)	(real)	(sim)	(real)	(sim)	(real)
1 R3M [4]	58	31	34	**97**	**100**	**87**	**87**	37	37	25	20	56	56
2 CLIP [9]	57	31	38	80	80	77	63	40	33	39	20	54	47
3 MVP [6]	68	**44**	**46**	90	90	70	50	50	27	**60**	50	63	53
4 VC-1 Base [8]	66	40	37	**97**	97	80	83	**57**	**67**	**61**	60	**67**	**69**
5 VC-1 Large [8]	**69**	41	38	**97**	87	77	43	50	57	**60**	**90**	65	63

otherwise, the PVRs are kept frozen. If finetuned, the PVRs use a separate learning rate of 1.5^{-6}. For augmentation, we use a combination of 30% color jitter and 4 pixel random shifts similar to [3]. Our reward functions are based on those presented in [7], which uses a success weighting $c_s = 5.0$, angle success weighting $c_a = 5.0fi$, goal radius $r_g = 1.0$, angle threshold $\theta_g = 25°$, and slack penalty $\gamma = 0.01$. We add a small collision penalty of 0.03 to reduce the amount of collisions in the real world.

ImageNav performance was assessed using the following criteria: **Success rate:** The proportion of successful episodes, where the robot reached the goal without significant collision and with a remaining distance of less than 1m. **Number of steps:** The total number of steps taken by the robot to reach the goal in each episode. **Distance to goal (m):** The Euclidean distance remaining between the robot and the goal at the conclusion of each episode.

IV. EXPERIMENTAL FINDINGS

In this section, we evaluate policies that use different PVRs on five tasks across three hardware platforms. In total, our real-world experiments required 110 hours of hands-on evaluation. Our experiments address the following questions:

1. How do recently released PVRs, designed specifically for robotics, perform across diverse simulated and real-world robotic tasks? (Section IV-A)
2. How predictive are simulation evaluations of hardware evaluations when policies for real world experiments are trained from real world demonstrations and using frozen PVRs? How can we enhance the correlation between simulation and hardware results to improve simulation predictivity of real-world results? (Section IV-B)
3. How does sim predictivity of PVR results change when we transfer policies from sim to real (Sim2Real transfer)? To what extent do policies trained with PVRs in simulation transfer to real-world scenarios? (Section IV-C)
4. How do model size, fine-tuning, and data augmentations impact simulation predictivity? Can we utilize simulation to benchmark such variations?

A. Evaluating Pre-Trained Visual Representations (PVRs) in Simulation and Reality

In this section, we study how various PVRs perform in both simulation and real-world experiments. We select five PVRs shown in Table I with success rates on the previously introduced simulation benchmark suite, CortexBench [8], ranging from 57% to 69%. We evaluate these PVRs on three platforms (TriFinger, Franka, and Stretch) in 'frozen mode' – i.e., the parameters of the PVR are *not* updated during the policy learning stage. The CortexBench tasks span 17 different tasks, although the set does not include the exact same tasks examined in our study. While the Stretch ImageNav and Trifinger Push Cube tasks are the same, we use similar manipulation tasks to compare with our Franka tasks. We would expect the performance of this subset of CortexBench tasks to correspond with the performance in sim we observe.

In Table I, we show results for both simulation and hardware evaluations. Except for the ImageNav task, all results in reality were achieved by training a policy on real robot demonstrations. We observe that VC-1 Base (row 4) outperforms all other methods, with a 69% success rate when averaged over all five real-world robotics tasks. CLIP (row 2) has the lowest average performance with 47% success in reality. In general, these trends are similar (but not the same) as the trends on CortexBench, where VC-1 Base is the third-best PVR and CLIP is the lowest-ranked method. The correlation between performance on CortexBench and our real-world evaluations is further studied in Section IV-B. Below, we analyze the trends in these real-world results on individual tasks.

The strongest performing model differs by task: MVP for the Trifinger task, R3M and VC-1 Base, the two smallest models, for Franka tasks, and VC-1 Large for the ImageNav task on the Stretch. Unlike other tasks, the policies for ImageNav were trained with large-scale reinforcement learning and evaluated in a Sim2Real manner.

B. Sim Predictivity of hardware results when policies are trained on real demonstrations

In this section, we analyze the correlation between performance on CortexBench and performance on real robots and study how mirroring real-world experimental conditions in evaluations conducted in simulation can further improve *Sim Predictivity* (as measured by SRCC [10]). Specifically, we contrast the simulation benchmark proposed in [8] (CortexBench) with our simulated evaluations from Section III.

For this analysis, we subselected the four tasks that have real-world demonstrations available (from the TriFinger and Franka platforms). We want to understand

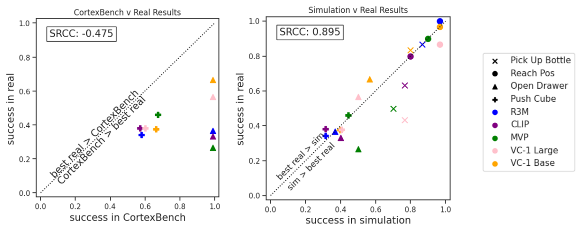


Fig. 3: Comparison of *Sim Predictivity* between CortexBench (left) and our simulation setting (right). Each data point represents a (model, task) tuple. Models and tasks are depicted by colors and symbols respectively, as shown in the legend.

how well simulation performance translates to real-world performance when training on real-world data. The Stretch/ImageNav task was left out of this analysis since it is trained entirely in simulation and thus is a case of *Sim2Real Transfer*, discussed further in Section IV-C.

In Table I, we report the performance in simulations (sim) designed for the tasks studied in this work. Their simulation settings are discussed in Section III and were constructed differently from CortexBench to closely reflect real-world conditions in a few different ways: the number of demonstrations in sim were adjusted to match the number available in the real world, and the last training checkpoint is chosen instead of the best performing on a validation set, since doing that would be prohibitive in the real world.

Figure 3 shows the correlations between simulation and real-world performance, for both Cortexbench and our simulation settings. Analyzing the correlations between CortexBench and real-world performance results in an SRCC value of -0.475, while repeating the calculation with our simulation results in a substantially higher SRCC of 0.895. While this is not a surprising result, these results reinforce the importance of matching real-world settings in simulation in order to improve the predictability of real-world performance from results in simulation.

TABLE II: Zero-shot sim2real evaluations of randomly initialized ViT-Base model with finetuning & augmentations (row 0) and pre-trained visual encoders (rows 1-5) for all tasks.

	TriFinger		Franka		Franka		Franka		Stretch	
	Push cube		Reach pos.		Pick up bottle		Open drawer		ImageNav	
# Model	(real)	(sim)	(real)	(sim)	(real)	(sim)	(real)	(sim)	(real)	(sim)
0 Scratch	8	21	0	27	0	11	30	37	10	35
1 R3M	11	31	0	97	0	87	10	37	20	25
2 CLIP	8	31	0	80	0	77	27	40	20	39
3 MVP	5	44	0	90	0	70	13	50	50	60
4 VC-1 Base	3	40	0	97	0	80	23	57	60	61
5 VC-1 Large	2	41	0	97	0	77	23	50	90	60

C. Effect of Sim2Real Policy Transfer on Simulation Predictivity

Having established that *Sim Predictivity* shows that reproducing setups in simulation and reality will yield similar levels of performance for the frozen PVRs, the natural next question is: how well do PVRs based policies perform when trained in simulation and transferred to the real world? We evaluated our simulation-trained policies obtained using 5 different PVRs on our 5 tasks

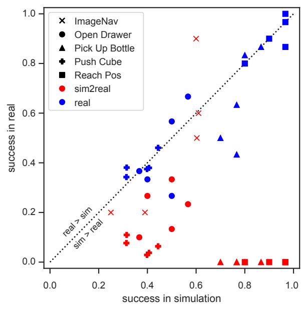


Fig. 4: *Sim Predictivity* chart comparing correlations of sim performance to policies trained in the real world (blue), vs correlations of sim performance to policies trained in sim and evaluated on hardware (Sim2Real transfer). Sim2Real transfer is poor across the board for tasks that use few-shot imitation learning, while transfer performance is substantially better on ImageNav, which is trained using large-scale reinforcement learning on simulated scenes.

and plotted the results in Figure 4. The results suggest that for most tasks, specifically the ones trained using few-shot imitation learning, the performance achieved when running a simulation-trained policy in the real world can not be predicted by that in simulation, with most tasks' success metrics drop to near zero values.

CLIP and **MVP** were both effective at avoiding obstacles (colliding 0.6 and 0.3 times on average), but often stopped short of the goal. We speculate that this might be a result of not observing data similar to the indoor navigation datasets used to train the stronger performing models like VC-1 Base and VC-1 Large.

TABLE III: Zero-shot Sim2Real generalization of policy with randomly initialized (row 0) or pre-trained visual representations (rows 1-5) on the ImageNav task with a Stretch robot.

	Success (%) ↑		Num Steps ↓		Dist-To-Goal ↓		Collisions ↓	
# Model	(sim)	(real)	(sim)	(real)	(sim)	(real)	(sim)	(real)
0 Scratch	34.7	10.0	124.2	119.2	3.3	6.0	11.2	3.0
1 R3M	25.0	20.0	**71.0**	81.7	3.4	4.2	5.1	1.7
2 CLIP	39.0	20.0	74.3	79.5	3.4	2.6	**2.5**	0.6
3 MVP	60.3	50.0	97.0	114.8	2.2	1.8	4.2	**0.3**
4 VC-1 Base	**61.0**	60.0	101.0	109.7	**2.1**	1.6	4.1	1.2
5 VC-1 Large	60.0	**90.0**	107.0	122.5	2.2	**0.8**	3.9	1.0

In contrast, the performance of frozen PVRs on the ImageNav task trained using large-scale RL is comparable to its performance in sim, which we may consider as a successful zero-shot Sim2Real transfer of results. In Table III, we compare the performance of PVRs (rows 1-5) with a randomly initialized model that was fine-tuned with data augmentations for the ImageNav task (row 0). We find that in reality, the best model (VC-1 Large, row 5) far exceeds random-initialization performance by 80% absolute (90% vs. 10%). This result strongly suggests that PVRs can play a crucial role in successful, zero-shot Sim2Real transfer. Achieving this result required modifications to the original simulation settings from CortexBench. Specifically, the performance transfer for this task greatly improved (from values near zero) once the horizontal field of view of the simulated camera was changed to 42°,

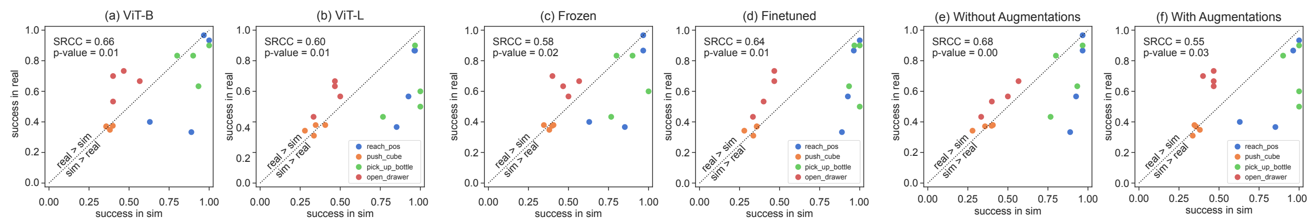


Fig. 5: *Sim Predictivity* correlation plots analyzing the impact of different model variations of VC-1: (a, b) model size; (c, d) fine-tuning; (e, f) data augmentation.

matching the camera angle of the real robot.

Qualitative assessments for Sim2Real transfer on ImageNav: We observed that **R3M** only completed the shorter episodes during real-world evaluation and frequently collided with obstacles (which resulted in episode termination), which reflects the behavior seen in simulation where the average number of collisions (5.1) exceeded other PVRs.

VC-1 Base and **VC-1 Large** exhibited contrasting behavior. The larger model explored the environment effectively (high number of steps) and achieved a high success rate, while the base model appeared to randomly explore, but reached the goal when seen from afar.

D. Impact of Model Size, Fine-Tuning, and Data Augmentation

This section studies the impact of three key design decisions when using PVRs for robotics applications: model size, freezing the PVR vs. fine-tuning, and the use of data augmentations, i.e. random color jitter and translations added to RGB input during policy learning. Specifically, our aim is to analyze the impact of each of these variations on simulation predictivity. We chose the VC-1 PVR as our basis for investigating these design decisions, which are detailed below:

1. **Model Size**: We used VC-1 models trained on the ViT-Base and ViT-Large architectures (called VC-1 Base and VC-1 Large respectively).
2. **Freeze vs Fine-tune**: We trained policies for each downstream task under two settings: keeping the PVR frozen (as in the above sections), as well as allowing the weights of the PVR to be fine-tuned by the task objective.
3. **Data Augmentation**: We trained downstream policies with and without data augmentation. We applied data augmentation in the RGB image space: 20% random color jitter in brightness, contrast and hue, and 8-pixel random translations.

We trained and evaluated policies for each of 4 downstream tasks with the cross-product of the above variations, both in simulation and in the real world. This produced 32 different tuples of (sim, real) performance. To analyze the impact of each variation, we sliced these tuples along the dimension of each variation and computed the SRCC of the 16 data points in each arm. The results are shown in Figure 5.

The SRCC values for all VC-1 variations are statistically significant ($p < .05$) and show a strong positive correlation, greater than 0.5 in all cases. The differences in *Sim Predictivity* are smaller with regards to the backbone size (0.66 for Base v. 0.60 for Large) and whether or not the layers were kept frozen (0.58 for frozen v 0.64 for fine-tuned). Notably, there is a drop in predictivity from 0.68 to 0.55 when we use augmentations. This is perhaps counterintuitive, as we may expect augmentations to help with model robustness, but we observed a negative impact to *Sim Predictivity* with our experiments.

TABLE IV: Success rate of policies using two model sizes, with and without fine-tuning and augmentations on 4 tasks.

				TriFinger		Franka		Franka		Franka		Average	
				Push Cube		Reach Pos		Pick Up Bottle		Open Drawer		All tasks	
#	model size	frozen	aug.	(sim)	(real)	(sim)	(real)	(sim)	(real)	(sim)	(real)	(sim)	(real)
1	VC-1 Base	yes	no	**40**	**37**	97	**97**	80	83	**57**	67	55	57
2	VC-1 Base	yes	yes	38	35	63	40	90	83	40	70	46	46
3	VC-1 Base	no	no	36	**37**	89	33	93	63	40	53	52	37
4	VC-1 Base	no	yes	36	**37**	**100**	93	**100**	**90**	47	**73**	**57**	**59**
5	VC-1 Large	yes	no	**41**	**38**	97	87	77	43	50	57	53	45
6	VC-1 Large	yes	yes	35	**38**	85	37	**100**	60	43	63	53	40
7	VC-1 Large	no	no	28	34	93	57	97	**90**	33	43	50	45
8	VC-1 Large	no	yes	34	31	96	87	**100**	50	47	67	55	47

TABLE V: Sim2Real transfer results. All results are with policies trained in simulation and evaluated on real robots.

				TriFinger		Franka		Franka		Franka		Stretch	
				Push Cube		Reach Pos		Pick Up Bottle		Open Drawer		ImageNav	
#	model size	frozen	aug.	(sim)	(real)	(sim)	(real)	(sim)	(real)	(sim)	(real)	(sim)	(real)
4	VC-1 (ViT-B)	yes	no	40	3	97	0	80	0	57	23	75	60
5	VC-1 (ViT-B)	yes	yes	38	6	63	0	90	0	40	27	75	10
6	VC-1 (ViT-B)	no	no	36	20	89	0	93	0	40	33	61	60
7	VC-1 (ViT-B)	no	yes	36	11	100	0	100	0	47	30	47	90
8	VC-1 (ViT-L)	yes	no	41	2	97	0	77	0	50	23	71	90
9	VC-1 (ViT-L)	yes	yes	35	3	85	0	100	0	43	27	76	80
10	VC-1 (ViT-L)	no	no	28	23	93	0	97	0	33	37	60	60
11	VC-1 (ViT-L)	no	yes	34	15	96	0	100	0	47	40	69	90

V. CONCLUSIONS

Our large-scale empirical study has advanced the understanding of pre-trained visual representations (PVRs) in robot learning. We found a high degree of Sim2Real predictivity of PVR-based policies, suggesting that simulation experiments can inform real-world performance. Notably, we have achieved a landmark result on ImageNav, demonstrating the critical role of PVRs in enabling effective Sim2Real transfer. Finally, our study highlights the impact of key design decisions, such as model size, data augmentation, and fine-tuning when deploying PVRs in real-world robotics tasks. These insights help illuminate the potential of PVRs for robot learning, setting a strong foundation for future research.

References

[1] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi, "Simple but effective: Clip embeddings for embodied ai," in *CVPR*, 2022.

[2] S. Parisi, A. Rajeswaran, S. Purushwalkam, and A. K. Gupta, "The Unsurprising Effectiveness of Pre-Trained Vision Models for Control," *ICML*, 2022.

[3] K. Yadav, R. Ramrakhya, A. Majumdar, V.-P. Berges, S. Kuhar, D. Batra, A. Baevski, and O. Maksymets, "Offline visual representation learning for embodied navigation," in *arXiv preprint arXiv:2204.13226*, 2022.

[4] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3M: A Universal Visual Representation for Robot Manipulation," *CoRL*, 2022.

[5] Y. J. Ma, S. Sodhani, D. Jayaraman, O. Bastani, V. Kumar, and A. Zhang, "Vip: Towards universal visual reward and representation via value-implicit pre-training," *arXiv preprint arXiv:2210.00030*, 2022.

[6] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell, "Real-world robot learning with masked visual pre-training," *CoRL*, 2022.

[7] 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.

[8] A. Majumdar, K. Yadav, S. Arnaud, Y. J. Ma, C. Chen, S. Silwal, A. Jain, V.-P. Berges, P. Abbeel, J. Malik, D. Batra, Y. Lin, O. Maksymets, A. Rajeswaran, and F. Meier, "Where are we in the search for an artificial visual cortex for embodied intelligence?" *arXiv preprint arXiv:2303.18240*, 2023.

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning (ICML)*, 2021.

[10] 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?" *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6670–6677, 2020.

[11] S. Parisi, A. Rajeswaran, S. Purushwalkam, and A. Gupta, "The unsurprising effectiveness of pre-trained vision models for control," *arXiv preprint arXiv:2203.03580*, 2022.

[12] M. Kaspar, J. D. M. Osorio, and J. Bock, "Sim2real transfer for reinforcement learning without dynamics randomization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4383–4388.

[13] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, "Rl-cyclegan: Reinforcement learning aware simulation-to-real," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 157–11 166.

[14] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai, "Retinagan: An object-aware approach to sim-to-real transfer," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 920–10 926.

[15] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 627–12 637.

[16] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot, "Navigating to objects in the real world," *arXiv preprint arXiv:2212.00922*, 2022.

[17] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Robotics: Science and Systems (RSS)*, 2017.

[18] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 438–13 444.

[19] J. Krantz, T. Gervet, K. Yadav, A. Wang, C. Paxton, R. Mottaghi, D. Batra, J. Malik, S. Lee, and D. S. Chaplot, "Navigating to objects specified by images," *arXiv preprint arXiv:2304.01192*, 2023.

[20] A. Dittadi, F. Träuble, M. Wüthrich, F. Widmaier, P. Gehler, O. Winther, F. Locatello, O. Bachem, B. Schölkopf, and S. Bauer, "The role of pretrained representations for the ood generalization of reinforcement learning agents," 2021.

[21] M. Wüthrich, F. Widmaier, F. Grimminger, J. Akpo, S. Joshi, V. Agrawal, B. Hammoud, M. Khadiv, M. Bogdanovic, V. Berenz, *et al.*, "Trifinger: An open-source robot for learning dexterity," *arXiv preprint arXiv:2008.03596*, 2020.

[22] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar, "Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking," 2023.

[23] "Robohive – a unified framework for robot learning," https://sites.google.com/view/robohive, 2020.

[24] 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.

[25] 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.

[26] 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.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv:1707.06347, 2017.

[28] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

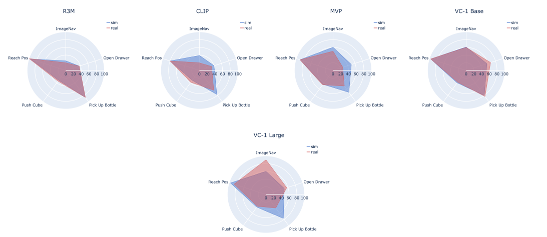


Fig. 6: Comparison of performance in simulation (blue) vs. reality (red) for five PVRs on five tasks. R3M [4] and VC-1 Base [8] performance in sim closely matches reality on *all* tasks. However, for CLIP [9], MVP [6], and VC-1 Large [8] there is mismatched performance on multiple tasks.

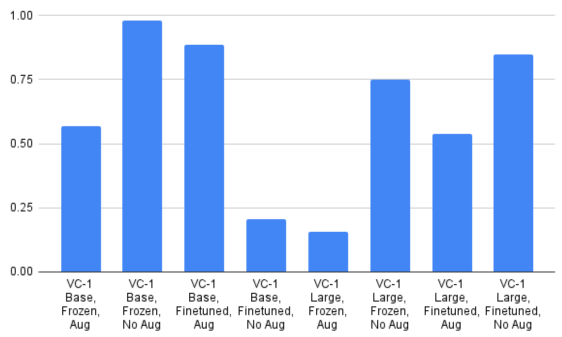


Fig. 7: *Sim Predictivity* correlation for each variation of VC-1 Large.

VI. APPENDIX

A. Details of Manipulation and Navigation tasks

1) Task, Policy and Training details for the TriFinger task: In this task, the robot must move a cube from an arbitrary initial position to an arbitrary goal position specified by a goal image I_g of the cube at the goal position. The state for the BC policy is $[x_t, z_t, z_g]$, where x_t are the proprioceptive states of the fingers, and z_t and z_g are PVR-encoded versions of the current camera image I_t and the goal image I_g, both at a resolution of 270×270. We choose to specify the goal as an image to further underscore the role of visual perception in this task. The initial and goal cube positions are uniformly sampled from within the robot workspace. The success criteria are defined as $1 - \frac{d_f}{d_i}$ where d_f and d_i are the final and initial distance between the cube and the goal (respectively).

We collect demonstrations in simulation (PyBullet [28]) using a hand-designed expert policy that relies on knowing the ground truth initial cube pose, which is easily obtainable in simulation. Given the initial cube pose, the expert policy first computes the contact points on the cube for each finger (or just one finger in the reach task). It then computes trajectories in Cartesian space for each fingertip from their initial positions to their respective contact points on the cube. Once the fingers have reached the contact points and grasped the cube, the expert policy computes trajectories for each fingertip to bring the cube to the goal position. These trajectories are then used to train a policy using behavior cloning, with a learning rate of 10^{-4} for 500 epochs.

We compute the joint torques needed for tracking the desired fingertip trajectories in Cartesian space using the simplified impedance controller from [21] (time index omitted for clarity):

$$\tau = J^T (k_p (x_{\text{ref}} - x) + k_v (\dot{x}_{\text{ref}} - \dot{x})) \qquad (1)$$

where $\tau \in \mathbb{R}^9$ is the vector of joint torques to be applied to each finger, x_{ref} are the desired fingertip positions from the reference trajectory, J is the Jacobian of the 3 fingers, and k_p and k_v are hand-tuned controller gains. We also use this controller to execute policies on the real robot.

See Figure 8 for a comparison between the sim and real-world visuals for the Trifinger task.

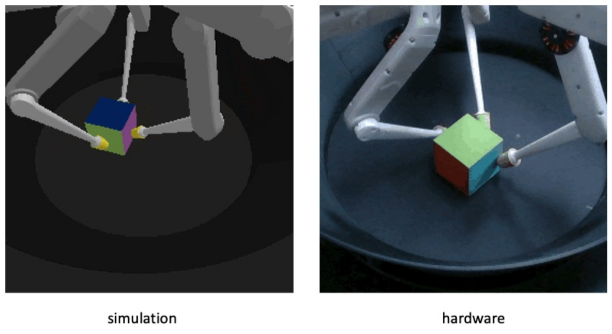


Fig. 8: Trifinger Simulation and Hardware Setup

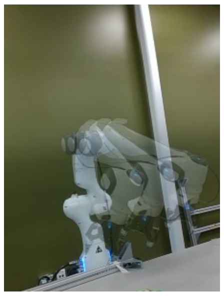
2) Task, Policy and Training details for the Franka tasks: Figure 1 and Figure 9 show the task configurations, which include reaching a target point with the gripper (Reach Pos) within 5cm error, picking up a bottle stably (Pick Up Bottle), and opening a drawer (Open Drawer) more than 50% of the way. The target position or object for each task is randomly reset before each demonstration and evaluation so a naïve kinesthetic replay does not solve the task. For the Reach Pos task, the target is provided to the policy in the form of a PVR-encoded goal image.

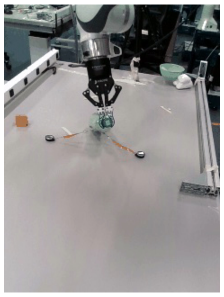
The policies take as input the joint angles and velocities of the arm and gripper, and the PVR-encoded 420×224 RGB image taken from a RealSense D430 camera. The learned policies output desired joint angles which are followed by the default PD control loop.

For the Reach Pos and Pick Up Bottle tasks, we collected 30 demonstrations and trained a behavior cloning policy at a learning rate of 10^{-4} for 200 epochs. For the Open Drawer task, we collected 150 demonstrations and trained a behavior cloning policy at a learning rate 10^{-3} for 500 epochs. For finetuning, we used the same learning rate for the policy and PVR encoder. For augmentation, we added 20% ColorJitter (brightness, contrast, hue) and 8 pixel random shifts to the image.

See Figure 9 for a comparison between the sim and real world visuals for the Franka tasks.

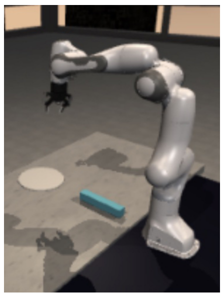
3) Task, Policy, and Training details for the ImageNav task: See Figure 10 for a comparison between the sim


(a) Reaching Random Point on real robot


(b) Bottle Pickup on real robot

(c) Open Drawer task on a kitchen table-top setup on real robot

(d) Reaching Random Point in sim


(e) Bottle Pickup in sim

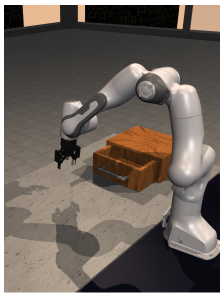

(f) Open Drawer task on a kitchen table-top setup in sim

Fig. 9: Franka real-world manipulation tasks

and real-world visuals for the ImageNav task.

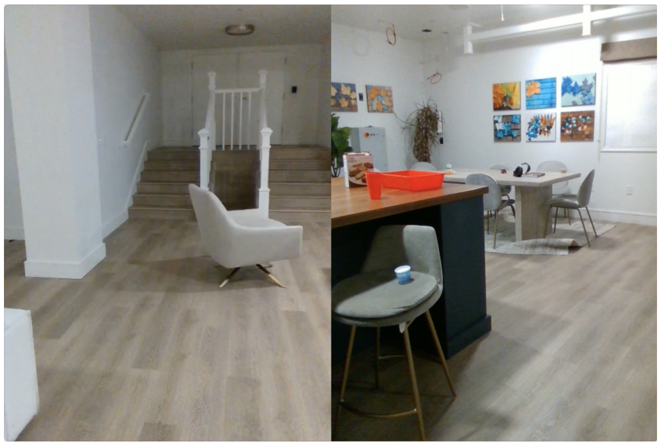
See Figure 11 for a picture of the stretch robot used for our experiments and a top-down map generated by stretch.

B. PVRs details

Table VI is a comparison of the different PVRs studied in this paper.

C. Additional experimental data

Table VII highlights the differences in performance when models are trained and evaluated in a more realistic setting compared to the original CortexBench. The Franka Open Drawer task is compared with the MetaWorld Open Drawer task, which has a similar specification but uses different robots. The results demonstrate that the success rates reported in CortexBench can be overly optimistic.


(a) ImageNav Hardware Setup

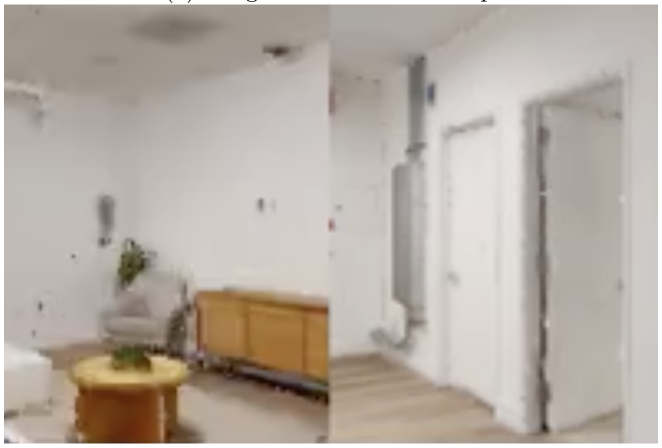

(b) ImageNav Simulation Setup

Fig. 10: Illustration of the ImageNav task setup for both hardware and simulation platforms.

TABLE VI: Details of the various pre-trained visual representations we evaluate in this work.

Model	Type of supervision	Architecture	Datasets	Number of parameters
R3M	Time-contrastive learning, video-text alignment	Resnet50	Ego4D	23M
CLIP	Image-text alignment	ViT-B	WebImageText	86M
MVP	Image only; masked autoencoder	ViT-L	Ego4D, ImageNet, Epic Kitchens, 100DoH, SS	307M
VC-1 Base	Image only; masked autoencoder	ViT-B	Ego4D, ImageNet, Epic Kitchens, 100DoH, SS-V2, RE10k, OpenHouse24	86M
VC-1 Large	Image only; masked autoencoder	ViT-L	Ego4D, ImageNet, Epic Kitchens, 100DoH, SS-V2, RE10k, OpenHouse24	307M

TABLE VII: Comparison of success rates for policies using 5 different frozen PVRs on CortexBench, MetaWorld, and three hardware platforms (TriFinger, Franka, and Stretch) in both simulation and real-world scenarios.

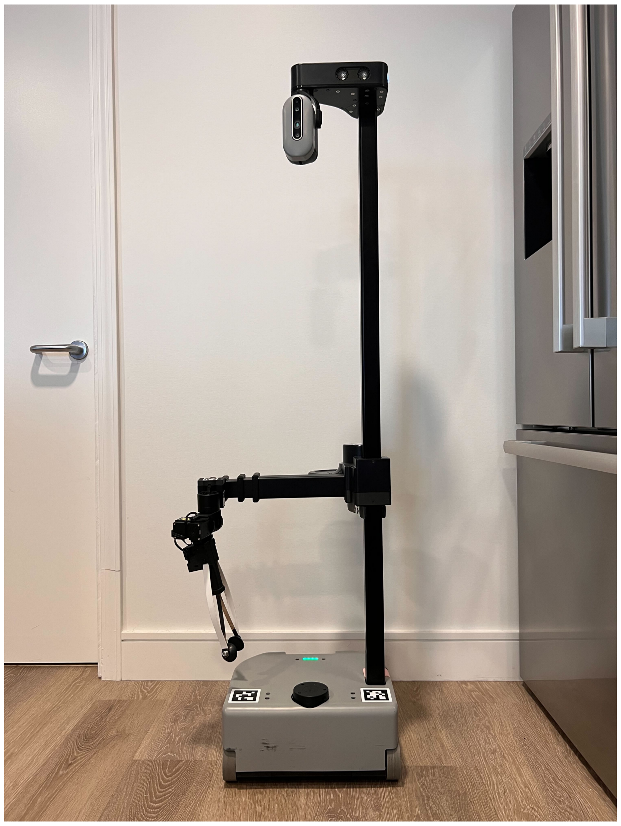
	CortexBench	TriFinger			Stretch			MetaWorld	Franka	
Task	All Tasks	Push cube			ImageNav			Open drawer		
# Method	CortexBench	CortexBench	sim	real	CortexBench	sim	real	CortexBench	(sim)	(real)
1 MVP	67.5	63.4	44.4	30.0	68.1	60.3	50	100.00	33.33	26.67
2 R3M	58.0	51.9	31.4	34.3	30.6	25.0	20	100.00	37.00	36.67
3 CLIP	57.0	40.1	31.5	38.1	52.2	39.0	20	100.00	40.00	33.33
4 VC-1 Base	66.2	60.6	39.9	37.5	67.9	61.0	60	100.00	56.67	66.67
5 VC-1 Large	68.7	60.2	40.7	38.0	70.3	60.0	90	100.00	50.00	56.67

TABLE VIII: All experimental results. Dash (-) means that we did not run this particular configuration due to time constraints.

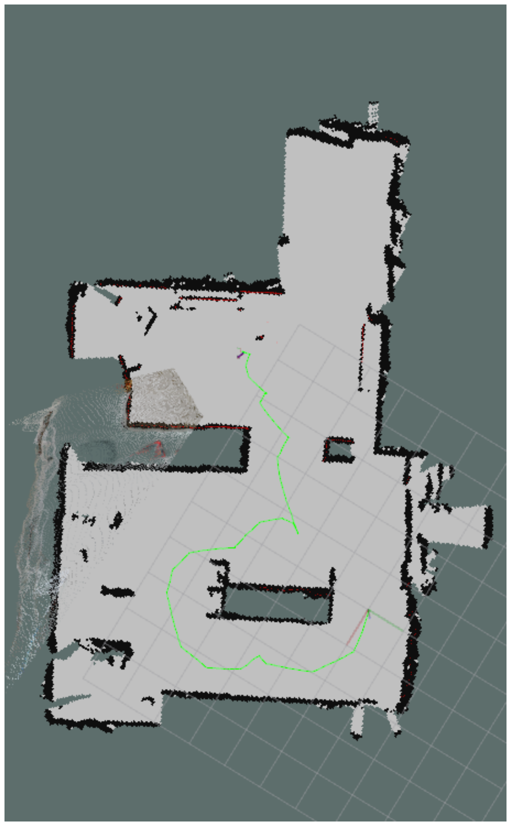
task setting	model	frozen	augmentation	ImageNav real	sim	open_drawer real	sim	pick_up_bottle real	sim	push_cube real	sim	reach_pos real	sim	Average real	sim
0	CLIP	yes	no	20.0±12.65	39.0±1.54	33.33±3.33	40.00±0.00	63.33±13.33	76.67±3.33	38.08±0.38	31.48±6.01	80.0±11.55	80.0±0.0	46.95±8.25	54.1±2.84
1	MVP	yes	no	50.0±15.81	60.3±1.55	26.67±3.33	50.00±0.00	50.0±20.0	70.0±11.55	30.04±4.08	44.4±0.51	90.0±5.77	90.0±0.0	49.34±9.8	59.61±3.39
2	R3M	yes	no	20.0±12.65	25.0±1.37	36.67±3.33	37.0±3.33	86.67±3.33	86.67±8.82	34.25±3.1	31.37±6.08	100.0±0.0	96.67±3.33	55.52±4.48	55.94±5.08
3	VC-1 Base	no	no	-	-	53.33±13.33	57.0±3.33	63.33±3.33	93.33±3.33	37.14±3.7	35.75±3.72	33.33±6.67	88.89±0.0	46.78±6.76	66.99±4.26
4	VC-1 Base	no	yes	-	-	73.33±3.33	73.33±6.67	90.0±10.0	100.0±0.0	36.91±1.41	35.7±2.86	93.33±6.67	100.0±0.0	73.4±5.35	77.26±2.38
5	VC-1 Base	yes	no	60.0±15.49	61.0±1.54	66.67±3.33	70.0±5.77	73.33±10.85	83.33±3.33	37.48±3.43	39.85±1.2	71.67±11.38	70.56±12.12	61.83±8.9	64.95±4.79
6	VC-1 Base	yes	yes	-	-	70.0±5.77	73.33±3.33	83.33±8.82	90.0±5.77	34.83±0.46	38.05±0.35	40.0±10.0	62.96±7.41	57.04±6.26	66.09±4.22
7	VC-1 Large	no	no	60.0±15.49	71.0±1.43	43.33±3.33	50.0±0.00	90.0±5.77	96.67±3.33	34.26±3.27	27.94±4.02	56.67±6.67	92.59±3.7	56.85±6.91	62.97±4.26
8	VC-1 Large	no	yes	90.0±9.49	76.0±1.35	66.67±8.82	76.67±6.67	50.0±25.17	100.0±0.0	31.0±2.98	33.57±0.78	86.67±13.33	96.3±3.7	64.87±11.96	76.51±2.5
9	VC-1 Large	yes	no	90.0±9.49	60.0±1.55	56.67±3.33	66.67±6.67	48.33±8.72	86.67±6.15	37.96±1.15	40.66±4.55	68.33±8.72	68.7±13.33	60.26±6.28	64.54±6.45
10	VC-1 Large	yes	yes	80.0±12.65	69.0±1.46	63.33±6.67	70.0±5.77	60.0±20.82	100.0±0.0	37.93±6.77	34.65±2.05	36.67±8.82	85.19±3.7	55.59±11.14	71.77±2.6

TABLE IX: All experimental results. Dash (-) means that we did not run this particular configuration due to time constraints.

task setting	model	frozen	augmentation	ImageNav sim	Open Drawer sim	Pick Up Bottle sim	Push Cube sim
0	CLIP	no	no	0.20+/-nan(1)	0.27+/-0.03(3)	0.00+/-0.00(3)	0.11+/-0.02(3)
1	MVP	no	no	0.50+/-nan(1)	0.13+/-0.03(3)	0.00+/-0.00(3)	0.06+/-0.02(3)
2	R3M	no	no	0.20+/-nan(1)	0.10+/-0.00(3)	0.00+/-0.00(3)	0.08+/-0.03(3)
3	VC-1 Base	no	no	N/A	0.33+/-0.03(3)	0.00+/-0.00(3)	0.14+/-0.03(3)
3	VC-1 Base	no	yes	N/A	0.30+/-0.00(3)	0.00+/-0.00(3)	0.17+/-0.04(3)
3	VC-1 Base	yes	no	0.60+/-nan(1)	0.23+/-0.03(3)	0.00+/-0.00(3)	0.03+/-0.00(3)
3	VC-1 Base	yes	yes	N/A	0.27+/-0.07(3)	0.00+/-0.00(3)	0.04+/-0.01(3)
4	VC-1 Large	no	no	0.60+/-0.00(2)	0.37+/-0.03(3)	0.00+/-0.00(3)	0.20+/-0.02(3)
4	VC-1 Large	no	yes	0.90+/-0.00(2)	0.33+/-0.03(3)	0.00+/-0.00(3)	0.14+/-0.02(3)
4	VC-1 Large	yes	no	0.90+/-nan(1)	0.33+/-0.03(3)	0.00+/-0.00(3)	0.04+/-0.01(3)
4	VC-1 Large	yes	yes	0.80+/-0.00(2)	0.23+/-0.03(3)	0.00+/-0.00(3)	0.02+/-0.01(3)



(a) The stretch robot used in the ImageNav real-world experiments.



(b) Top-down view of the real-world path (green) the robot took to explore the scene in an episode of the ImageNav task with the point cloud from the robot's head camera and a 2D lidar map.

Fig. 11: Stretch robot and sample of a top-down view of a path from a real-world scene

TABLE X: All hyperparameters and tasks setup comparisons between CortexBench, our simulation and real world settings.

	Trifinger			Stretch		Franka		Franka		MetaWorld	
Task	Push cube			ImageNav		Reach position		Pick up bottle		Open drawer	
Observation Space	RGB + proprio.			RGB		RGB + proprio.		RGB + proprio.		RGB + proprio.	
Action Space	Continuous			Discrete		Continuous		Continuous		Continuous	
Goal Specification	Goal Image			Goal Image		Goal Image		-		-	
Policy Learning	IL			RL		IL		IL		IL	
Context	CortexBench	sim	real	CortexBench	sim2real	sim	real	sim	real	CortexBench	s
Robot	Trifinger	Trifinger	Trifinger	LoCoBot	Stretch	Franka	Franka	Franka	Franka	Sayer	Fra
Epochs trained	100	500	500	500m Steps	600m Step	200		200		100	5
Checkpoint selection	Max eval success	epoch 100	epoch 100	Max eval success	Last	epoch 50	epoch 50	epoch 50	epoch 50	Max eval success	epoc
Num demonstrations	100	31	31	-	-	30	30	30	30	100	1
Number of random seeds	3	3	3	1	1	3	3	3	3	3	
Learning Rate	10^{-4}	10^{-4}	10^{-4}	2.5×10^{-4}	2.5×10^{-4}	10^{-4}	2.5×10^{-4}	10^{-4}	10^{-4}	-	10
Augmentations described	-	Color jitter + translate		-	Color jitter + translate	Color jitter + translate		Color jitter + translate		-	
umber of evaluation episodes	-	12	12	-	10	10	10	10	10	-	1
Sampling of goal position	12 different fixed positions			random		random		random			
Sampling of start position	12 different fixed positions			random		random		random			
Other aspects	demos from RL/heuristic policy			-	-	demos from RL/heuristic policy	demos from tele-operation	demos from RL/heuristic policy	demos from tele-operation	demos from RL/heuristic policy	demos from RL