

Phone2Proc: Bringing Robust Robots Into Our Chaotic World

Matt Deitke^{*† ψ} , Rose Hendrix^{*†}, Ali Farhadi ^{ψ} ,
 Kiana Ehsani[†], Aniruddha Kembhavi^{† ψ}

[†]PRIOR @ Allen Institute for AI, ^{ψ} University of Washington, Seattle
<https://phone2proc.allenai.org/>

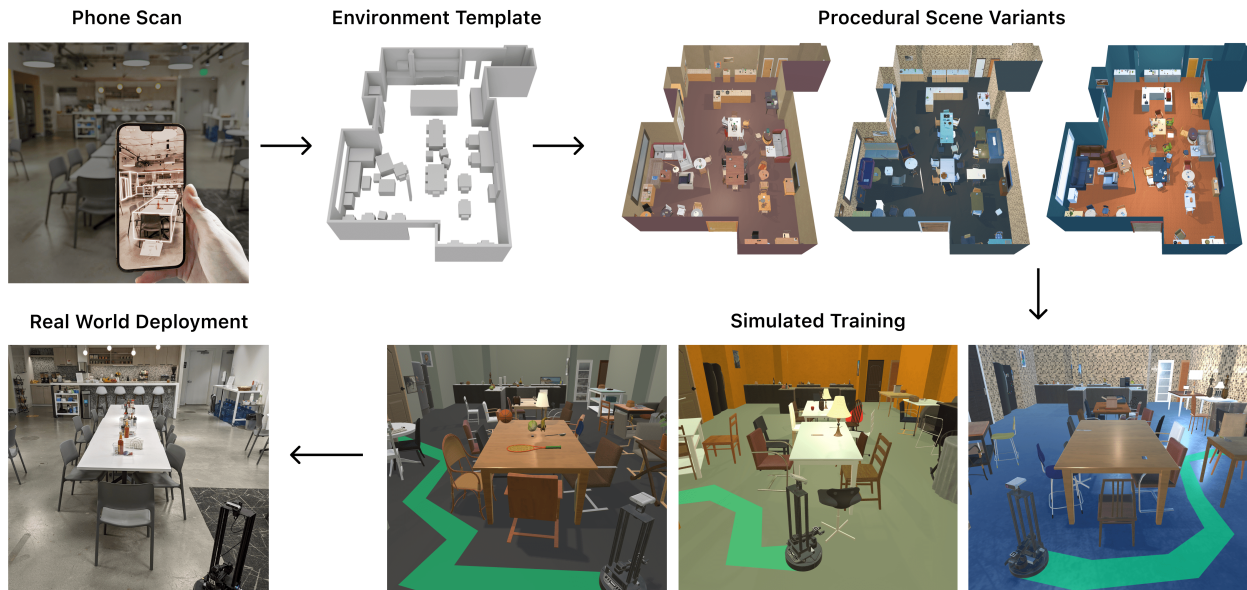


Figure 1. Successfully deploying agents trained in simulation to the real world has generally proved fraught - we present PHONE2PROC, a simple approach that uses a cellphone to scan an environment and procedurally generate targeted training scene variations of that location, whose usage results in successful and robust agents in the real environment.

Abstract

Training embodied agents in simulation has become mainstream for the embodied AI community. However, these agents often struggle when deployed in the physical world due to their inability to generalize to real-world environments. In this paper, we present Phone2Proc, a method that uses a 10-minute phone scan and conditional procedural generation to create a distribution of training scenes that are semantically similar to the target environment. The generated scenes are conditioned on the wall layout and arrangement of large objects from the scan, while also sampling lighting, clutter, surface textures, and instances of smaller objects with randomized placement and materials. Leveraging just a simple RGB camera, training with Phone2Proc shows massive improvements from 34.7% to 70.7% success rate in sim-to-real ObjectNav performance

* Equal contribution.

across a test suite of over 200 trials in diverse real-world environments, including homes, offices, and RoboTHOR. Furthermore, Phone2Proc’s diverse distribution of generated scenes makes agents remarkably robust to changes in the real world, such as human movement, object rearrangement, lighting changes, or clutter.

1. Introduction

The embodied AI research community has increasingly relied on visual simulators [30, 49, 61] to train embodied agents, with the expectation that the resulting policies can be transferred onto robots in the physical world. While agents trained within simulated environments have shown increased capabilities, progress in successfully deploying these policies onto physical robots has been limited.

Robots trained in simulation must overcome daunting challenges if they are to work effectively in a real space such as our home. First, they must overcome the generalization

gap between the limited set of simulated environments they are trained on and the test scene of interest. In practice, policies trained to perform complex visual tasks with reinforcement learning struggle to perform well in novel scenes with novel layouts and object instances. Second, they must work in realistic environments where we live and work, which are often full of clutter, with objects that keep being moved around, with people in and out of the scene and with lighting changes. In short, we expect our agents to learn from a small set of training data points and generalize not just to a single test data point, but to a distribution of test data that is often semantically distant from the training data. Today’s methods are a ways away from delivering such performant, robust, and resilient robots [9, 12].

In this work, we present PHONE2PROC, which represents a significant advancement towards the goal of creating performant, robust, and resilient robots. Instead of training policies in simulated environments that may be semantically distant from the target physical scene, PHONE2PROC efficiently generates a distribution of training environments that are semantically similar to the target environment. This significantly reduces the generalization gap between the training and target distributions, resulting in more capable robots.

PHONE2PROC utilizes a freely available mobile application to quickly scan a target environment and create a template of the surroundings, including the scene layout and 3D placements of large furniture. This template is then used to conditionally generate a fully interactive simulated world using ProcTHOR [13], closely mirroring the real-world space. Importantly, this single simulated environment is then transformed into a distribution of simulated worlds by randomizing objects, their placements, materials, textures, scene lighting, and clutter. This allows for the creation of arbitrary large training datasets that are semantically similar to the desired real-world scene.

We produce policies for object goal navigation using PHONE2PROC and deploy them onto a LoCoBot robot in the physical world. We conduct extensive evaluations with 234 episodes in five diverse physical environments: a 3-room and 6-room apartment, a test scene from RoboTHOR-real, a conference room, and a cafeteria. This represents one of the largest and most diverse studies of sim-to-real indoor navigation agents to date. Across all environments, PHONE2PROC significantly outperforms the state-of-the-art embodied AI model built with ProcTHOR, with an average improvement in success rate from 34.7% to 70.7%. Our robot is able to explore the scene efficiently and effectively navigate to objects of interest, even in the presence of clutter, lighting changes, shifts in furniture, and human movement. These strong navigation results are achieved using an **RGB-only camera**, **no depth** sensors, **no localization** sensors, and **no explicit mapping** components.

In summary, we present: (1) PHONE2PROC, a simple and highly effective method for reducing the generalization gap between datasets of simulated environments and a target environment in the real world, (2) large-scale real-world robotics experiments with 234 trials showing significant improvements for PHONE2PROC compared to state-of-the-art models, and (3) experiments demonstrating the robustness of PHONE2PROC in the face of variations such as changes in lighting, clutter, and human presence.

2. Related Works

Navigation in Simulated Environments. Visual navigation [1, 4] is a popular task in the embodied AI community with benchmarks in several simulators [30, 44, 49, 61]. An effective approach is to use semantic mapping to explore environments efficiently [6–8, 20]. Kumar *et al.* [32] adapts mapping methods to condition the policy on the target. These works utilize agent pose and depth sensors to build their maps and localize the agent. In contrast, our method only relies on RGB information without any additional sensor. Other methods for navigation use scene priors [64], meta-learning [60], paired grid world environments [25], scene memory transformers [18], passive videos of roaming in a scene as a training cue [23] and expert human trajectories for imitation learning [45].

The community has also made progress in training embodied agents for navigation exclusively using RGB observations and barebones neural architectures. These include using frozen ImageNet trained visual encoders [65], learning visual encoders from scratch [12], and using CLIP [33] based encoders [29]. Gadre *et al.* [19] and [35] use an off-the-shelf exploration method and clip-based object localization to accomplish 0-shot object navigation. Deitke *et al.* [13] show the benefits of procedural generation for navigation and manipulation.

While the above works show promising results in simulation, most are not deployed and tested on real robots. We provide comparisons to the current state-of-the-art method, ProcTHOR [13], via large-scale real-world evaluations.

Sim-to-Real Transfer. While most models are evaluated only in simulation, for practical applications, policies learned in simulation must function in real life. Often, policies trained only in simulation can prove brittle or nonfunctional in transfer [26]. Chatopadhyay *et al.* [9] find that standard embodied agents significantly underperform (or fail) in the presence of realistic noise in the system. Truong *et al.* [55] compare the correlation of the performance of 4-legged robots navigating in simulation against the real world. They find that adding fidelity to the simulation does not help with the performance in the real world.

An alternate approach to higher fidelity is to add randomization to sensing or dynamics in simulation. This does help [46, 53], but too much randomization can degrade train-



Figure 2. **Examples of environment templates for our five target test environments.** These are produced by an iPhone scanning each environment using our iOS app that leverages Apple’s RoomPlan API. These environment templates contain the room layouts and some 3D locations for large furniture objects. They do not contain small objects, textures, lighting, etc.

ing efficacy [36], and hand-tuning appropriate randomization requires expert knowledge and does not scale. Some address this pitfall by leveraging real-world rollouts or inputs at train time to tune simulation randomization [10, 14]. However, these works are randomizing a subset of a well-parameterized dynamical system for a narrow task (swing-peg-in-hole or cabinet opening), as opposed to a more open-ended task or randomizing the entire visual appearance and object instances of the environment.

Recent works have deployed and measured policies on real robots [5, 15, 28, 47, 54] for the task of point goal navigation. In contrast to most, we use no mapping, explicit localization, or depth, as well as targeting a more complex task. We also test more extensively and in a wider variety of environments. Anderson *et al.* [2] study the sim-to-real gap for vision and language navigation and discover a crucial need for an oracle occupancy map and navigation graph. Our method does not require a manually annotated map and is robust to moving obstacles without the necessity for an additional dataset.

Real-to-Sim Transfer. Transferring observations from the real world to simulation can open up further capacities for training agents. This has been studied in the domain of object manipulation [34, 57]. [17] replicate an observed manipulation trajectory by predicting contact points and the forces. [27, 40] generate a 3D mesh of an object with articulation from observed interactions. [3, 52] infer simulation parameters for deformable object manipulation. [56, 63] learn cloth material recovery from videos. Our focus is on conditioning our procedural generation on the real scene rather than perfectly replicating the observation. [42] use a self-supervised technique to utilize unlabeled images for acquiring data for training scene graph models. We focus on generating 3D interactable environments for training embodied agents.

Navigation in Robotics. The robotics community has made progress in navigating robots with different embodiments in a diverse set of environments. Gupta *et al.* [22] combine a differentiable planner module with mapping to train a visual navigation agent end-to-end. In contrast to our work, their method assumes perfect odometry. Different methods have been used to build agents that follow demonstrated paths

and trajectories or navigate [24, 31, 37, 38, 48, 62]. Shah *et al.* [50, 51] build models for open world navigation. The main focus of these works is on the low-level control systems of the robots, whereas our focus is on building end-to-end models for embodied visual reasoning.

3. Approach

We now present PHONE2PROC which generates a distribution of training environments that closely match the real world physical space we are interested in. We begin with a phone scan of a target scene (Sec 3.1), then condition on this scan to procedurally generate variations of the scene for training agents (Sec 3.2), and finally transfer onto a Lo-Cobot robot that navigates in the physical world (Sec 3.3).

3.1. Scanning

PHONE2PROC is designed to optimize a robot’s performance within a desired real world environment. The first step in this process is to scan the target environment. This is accomplished using an iOS app that we built and will release using Apple’s freely available RoomPlan API [11]. Scanning a large apartment with several rooms only takes a few minutes, can be done using an iPhone or iPad and it outputs the environment template as a USDZ file.

The RoomPlan API provides us with a high-level bounding box template of the environment, which contains the room layouts and 3D placement of large objects that are visible to the camera. While scanning an environment, the app provides detailed real-time feedback about the construction of the scene to help the user capture a more accurate scan.

The resulting environment template includes the 3D locations and poses of walls, large objects, windows, and doors. Each object in the scan is assigned to one of 16 object types, including storage, sofa, table, chair, bed, refrigerator, oven, stove, dishwasher, washer or dryer, fireplace, sink, bathtub, toilet, stairs, and TV. Smaller objects, such as those typically on surfaces, are ignored. The metadata produced for each object includes the size, position, and rotation of its 3D bounding box, along with the forward-facing orientation. Doors and windows are provided as cutouts in the walls. Figure 2 presents examples of scanned environments

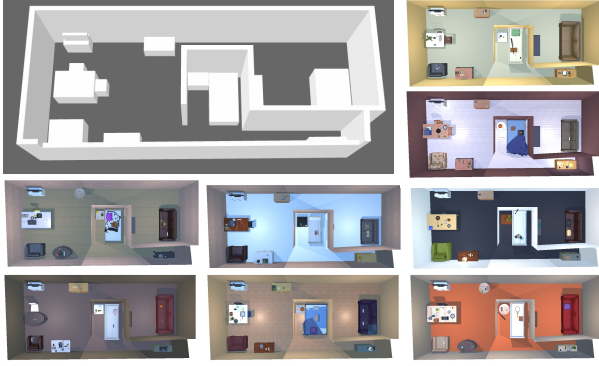


Figure 3. **Examples of Procedurally Generated Houses.** The procedural generation of the houses is conditioned on the target environment scanned using a phone. We are able to sample a rich and diverse set of scenes from this distribution with varying lighting, textures, objects and placements.

showcasing the diversity of the layouts in our test set.

3.2. Environment-Conditioned Procedural Generation

Procedural generation of simulation environments allows for a vast diversity of scenes for agents to train on. Deitke *et al.* [13] begin with a high-level room specification (*e.g.* 2 bedroom house with a kitchen and living area) and create an environment that matches it. In contrast, our approach uses a scan of the target real-world environment to condition the generation and create variations of that scene. This process involves (a) parsing the environment template, (b) generating the scene layout, (c) sampling objects from the asset library to match scanned semantic categories, (d) accounting for object collisions in Unity, (e) populating the scene with small objects not captured by the scan, and (f) assigning materials and lighting elements.

PHONE2PROC parses the USDZ environment template produced by the iOS app and extracts wall, door and window positions and 3D large object bounding boxes. It then leverages ProcTHOR to generate a fully rendered scene in Unity and finally populates this scene using ProcTHOR’s asset database of 1,633 assets across 108 object types. The generation process is very fast and can generate 1000 procedural scene variants in around an hour with an 8 Quadro RTX 8000 GPU machine. We now provide further details on each of these steps.

Layout. The environment specification file contains the placement of walls in each room. Unlike walls generated in ProcTHOR-10K [12], which are only aligned to orthogonal axes, PHONE2PROC allows for a more diverse wall generation that can accommodate any scanned layout. Each wall’s specification comes from its 3D bounding box, width, height and a constant depth (*e.g.* 16cm for all walls) – which

are used to produce a wall asset within the simulated environment. Placing walls produces the external boundary of the environment as well as its internal layout.

Rooms. We partition the space located within the external boundary walls into distinct rooms. A room is formed if the walls formed from the top-down 2D plane fully enclose a polygon. This is followed by floor and ceiling generation.

Windows and doors. The environment template specifies if each wall has cutouts for windows and doors. Our USDZ parser extracts the size and position of the holes along the wall. If the hole includes a cutout at the bottom of the floor, we place a door there; otherwise, we place a window. Here, we uniformly sample a door or window asset from the asset database and scale it appropriately.

For doors between connecting rooms, the sampled door may either include just a frame or both a frame with an openable door and a degree of openness sampled uniformly between 0.8 and 1.0. The room that the door opens into is randomly sampled. If the door is connected to the outside of the environment, we sample a door frame with an openable door component and fully close the door (to prevent agents from getting out of the scene).

Semantic objects. For each object in the template, we wish to sample an appropriate asset matching its semantic category. For each semantically similar ProcTHOR object candidate, we compute its 3D bounding box IoU with the object it may represent in the environment template and reject candidates with an IoU less than 75%. We then uniformly sample from the rest. The sampled asset’s position on the floor and its forward-facing direction come from the environment template’s corresponding object. We compute the vertical position of the object based on if it is on a surface (*e.g.* a couch on the floor or a television on top of a table) or attached to a wall (*e.g.* a wall television).

This procedure to find a matching asset can sometimes lead to large variations. For example, a table object may match a ProcTHOR coffee table, side table, or dining table and a TV may match a flat-screen TV or a vintage box television. Randomly sampling different asset instances in the library of a particular semantic object type makes agents more robust as they must learn to generalize to many visually distinct instances for each type.

Object collisions. We check to make sure that none of the placed objects collide with one another or the walls in the scene to avoid unrealistic configurations.

Small objects. After placing the large objects that match the scan, we generate smaller objects to be placed on top of them. Here, for instance, we might populate a bed with pillows or place fruits and plates on the counter. Unlike what happens in a 3D reconstruction, where all the objects are static, we are able to randomize the placement of small and target objects to produce many scene variations and prevent overfitting (See experiments in Sec 4).

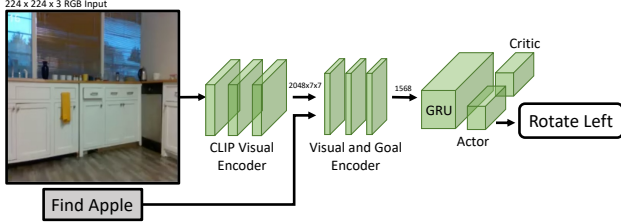


Figure 4. Our architecture is a simple GRU operating on the CLIP encoding of solely RGB input.

Lighting. Scene lighting is randomized such that each room is guaranteed one light, and then additional lights are uniformly sampled throughout the scene scaled to the number of rooms. Each light is then randomized in its intensity, RGB values, shadow bias, and strength.

Materials. We randomize materials following ProcTHOR’s material randomization by sampling from sets of structure materials (*i.e.* wall, floor, and ceiling) and object materials.

Clutter. After all the semantic objects from the scan have been sampled and smaller objects have been sampled to be placed on top of them, we also sample additional clutter objects (such as boxes, dumbbells, pillows, and plants) that are placed on the floor of each room – similar to how a real house may have objects like kids toys thrown around. These objects prevent overfitting to particular paths in the environment and helps teach agents to avoid obstacles.

3.3. Transfer to Real World

We first detail our model architecture and training regime then discuss our robot and physical scenes.

Model and Training Details. Our goals are to: a) design models that do not depend on unrealistic sensory data in real indoor environments like agent or target localization, b) use only RGB observations since real world depth cameras offer few choices, generally come with small FOV and are fairly noisy, and c) create agents that are robust to clutter and changes in the environment.

PHONE2PROC provides a distribution of simulated worlds that are sampled to produce a large training set. These scenes differ in the placement of small objects, materials, lighting, clutter, etc. This allows us to train policies that do not overfit to a single scene configuration, but instead generalize to realistic scene variations.

In terms of the model design, we adopt a simple architecture introduced in [29] and also used in [13]. The model uses a CLIP encoder to embed the visual observation (egocentric RGB frame) followed by a GRU to capture temporal information (Fig 4). We pre-train our model on the ProcTHOR-10k dataset using the same training regime presented in [13] and then finetune on the Phone2Proc environments for the task of object navigation on 16 object categories. We use AllenAct [59] to train our models. More

details on the training pipeline are provided in the appendix.

In principle, it is fairly straightforward to adopt a more complex model architecture, but we found this simple design to be highly performant not just in simulation but also in our real world experiments. Similarly, it is also easy to train agents for other tasks, including one involving object manipulation using an arm, since PHONE2PROC produces scenes that are fully interactive with support for all agents in AI2-THOR [30] including the arm-based agent [16].

During fine-tuning, we lower the learning rate to 0.00003 to avoid catastrophic forgetting of the skills learned in pre-training. We add a failed action penalty (0.05) in the reward shaping to encourage the agent to avoid hitting the obstacles in the environment. This is especially important as we deploy these models in the real world and would like to avoid damage to the environment or the robot. Instead of hand-tuning the camera parameters to match perfectly with the real world, the FOV of the camera in simulation is randomly sampled from a distribution approximating the real world.

Real-World Experiments. Models trained on procedurally-generated variants of the scene scans are then directly evaluated in real environments. We use 5 environments: a 3-room apartment, a large 6-room apartment, a real world test scene from RoboTHOR [12], a large reconfigurable office conference room, and a cafeteria. Models are evaluated against 5 different target objects from 3 different starting locations in the environment. No training or calibration is performed in the real world.

No particular effort was made to arrange for ease of robotic experimentation. The goal was to use real environments in their most natural setting. The lighting, object instances, textures, and window views are not recognized by the PHONE2PROC scan and are thus unseen by the agent at training time. As there are many objects in the real-world scenes that are not present in ProcTHOR’s asset library (*e.g.* whiteboards, bicycle), there are several object categories in each environment that are novel to the agent. No additional information is used in the preparation/scanning step besides the output of the RoomPlan API.

Experiments are run on LoCoBot [21], a low-cost, platform about 60cm tall using the PyRobot API [39]. The agent’s discrete action space is look up/down, turn right/left (each 30°), move ahead 25cm, and a “done” action to indicate reaching the target. Actions are sampled using PyTorch’s categorical distribution. FPS in real is ≈ 0.25 , and for practical reasons, physical trajectories were limited to 250 or 500 steps depending on the size of the environment.

4. Experiments

We provide extensive real-world evaluations of PHONE2PROC. In Sec. 4.1 we compare PHONE2PROC to ProcTHOR in 5 diverse real environments. In Sec. 4.2 we show that PHONE2PROC performs as well as a privileged

upper bound setting that utilizes a simulated counterpart of the real-world environment, painstakingly modeled by a digital artist. Sec. 4.3 illustrates the robustness of PHONE2PROC to various realistic changes in the environment, showing how PHONE2PROC hugely improves over using static reconstructions. Finally, we statistically analyze the significance of our results (Sec 4.4).

Scale of real-world evaluations. In aggregate we conduct 234 episodic evaluations in 5 diverse real-world environments. Our environments are large and challenging and each episode takes between 5 and 20 minutes to run. This represents one of the largest and most diverse real-world evaluation studies of sim-to-real indoor navigation agents. We put this number in the context of related works that provide 20 trials (1 scene) [6], 1 qualitative example [7], 36 trials (1 scene) [12] and 9 episodes (1 scene) [41]. A recent study for PointNav for studying Sim-vs-Real correlation [28] conducts 405 real trials but only uses a single laboratory setting.

Training models and baselines. All models use the same architecture and begin from a checkpoint trained on ProcTHOR-10k train set for the task of object navigation. This checkpoint is state of the art on 6 benchmark Embodied AI tasks [13]. This checkpoint is then fine-tuned for 5M steps on ProcTHOR-10K train with modifications detailed in section 3.3, and is henceforth referred to as PROCTHOR or “baseline”. The PHONE2PROC models are environment-specific and are fine-tuned on 1K procedurally generated variants of scans of the relevant environment. Results for these are presented in Figure 5.

4.1. How Well Does Phone2Proc Work?

We evaluate PHONE2PROC in 5 diverse real-world environments: a 3-room apartment, a 6-room apartment, 1 RoboTHOR-Real apartment, a conference room, and a cafeteria. In each space, our model and the baseline are each evaluated for 15 trials (5 object categories with 3 agent initial locations per category). The 5 categories (apple, bed, sofa, television, and vase) were chosen to showcase both fixed objects whose locations can be learned (e.g. television) and small objects that must be searched for (e.g. apple). Where necessary (e.g. conference rooms do not usually contain beds), the bed and sofa are substituted for environment-appropriate objects such as chairs or garbage cans. Starting locations are geographically distributed and we avoid ones that would achieve trivial success.

Fig. 5 shows that in every real-world scene, PHONE2PROC performs remarkably well and significantly outperforms the PROCTHOR baseline. In aggregate, PHONE2PROC achieves a Success Rate of 70.68% compared to 34.68% for PROCTHOR. Overall, we find that the bigger environments with multiple rooms (RoboThor, 3 room apartment and 6 room apartment) are

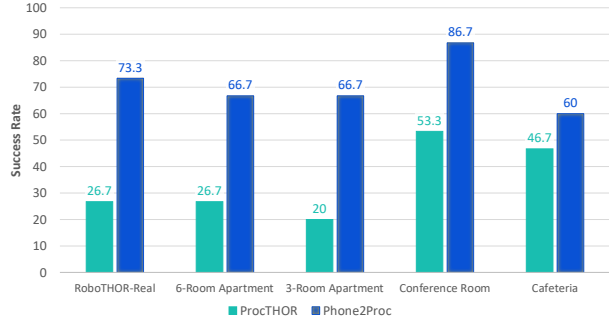


Figure 5. Results for PHONE2PROC vs ProcTHOR baseline in a variety of real environments. Each number represents fifteen trajectories - five objects from three starting locations.

quite challenging for the baseline. PHONE2PROC on the other hand, performs very effectively in these scenes, that require it to perform long range exploration.

Model	Success Rate	Episode Length
Habitat [44]	33.3	204.8
PROCTHOR [13]	33.3	92.5
PHONE2PROC (ours)	77.8	82.6

Table 1. Results of 9 trajectories evaluated in RoboTHOR-Real.

Table 1 compares PHONE2PROC with the same model architecture trained on Habitat [44] (implementation details on baseline training in the appendix). These results are presented on RoboTHOR-Real for 9 (instead of 15) episodes since Habitat only covers 3 of the 5 objects in our target set (bed, sofa, and television).


4.2. How Does Phone2Proc Compare To A Privileged Upper Bound?

RoboTHOR test scenes come with a carefully and manually reconstructed simulation counterpart. This allows us to train a privileged model on this perfect replica. Producing this replica for a real scene took 5 days and is intractable practically but represents a theoretical upper bound, in terms of quality and correctness, for static 3D reconstruction methods that may employ sensors such as LiDAR. The model, referred to as *Reconstructed Simulation* or RECON, is only finetuned in this 1 scene. In simulation, RECON achieves 100% success since it overfits that scene easily.

We evaluate PHONE2PROC and RECON for 15 episodes in the RoboTHOR-real apartment (Fig. 6). PHONE2PROC is able to match the performance of this privileged baseline both in terms of Success and Episode length, which shows the effectiveness of our proposed approach to scan the target environment and train within its variations. In Sec 4.3, we show the pitfalls of this privileged model.

Model	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14	Task 15	Mean Len	Mean Success
Reconstructed Simulation (Upper bound)	250	239	19	92	217	250	58	76	89	34	250	7	67	51	250	86.3	73.3
Phone2Proc (Ours)	24	117	6	122	116	167	110	43	13	83	217	9	50	223	114	85.2	73.3

Figure 6. **Comparison with the reconstructed simulation.** We compare PHONE2PROC with the privileged reconstruction baseline among 15 different tasks (each task represents a different pair of agent’s initial location and target object). Each square shows the episode length of the corresponding model for each task. The red squares represent failed episodes and the blue ones indicate successful ones. Despite the baseline’s privileges, PHONE2PROC achieves a similar success rate and episode lengths.



Model	No Variance	Change in Lighting	Change in Target Location	Furniture Rearrangement	Clutter	People Moving Around	Change in Camera Parameters
ProcTHOR	✗	✗	✗	✗	✗	✗	20.0%
Recon	✓	✓	✗	✗	✗	✗	26.7%
Phone2Proc	✓	✓	✓	✓	✓	✓	66.7%

Figure 7. Illustration of the scene disturbances used to comparatively evaluate models, along with their performances over each episode. The third column showcases the performance of models navigating to a vase when the object’s location is changed. For the last column, a full set of 15 trajectories is evaluated for each model. For other disturbances, we evaluated models for the target of Television.

4.3. How Robust is Phone2Proc to Chaos?

In reality, our homes and offices aren’t static and picture-perfect. Objects move around, furniture gets shifted, kids leave their toys on the floor, people keep moving around in the scene, lighting keeps changing throughout the day, and more! We evaluate the baseline model PROCTHOR, the privileged model RECON and our model PHONE2PROC in these settings (Fig. 7).

First, PROCTHOR does poorly in all settings, unsurprising given that it also fails on the episode with no variation. RECON performs well with no variations (consistent with Fig. 6). However, it performs very poorly when variations are introduced in the scene. When objects are moved around by just 1.5m, RECON fails, as it has memorized the location of every target object. Clutter and chair position adjustment confuses it, and the agent is simply unable to move around the scene and explore effectively. Moving the dining room furniture closer to the wall, as one might in a real house, produces interesting behavior. RECON calls the *Done* action for the television target when it sees the lamp. This is because in the original scan, the lamp was next to the television, and this is what the model likely memorized. RECON also fails when people move around during an episode. In stark contrast, PHONE2PROC is robust to every variation we tested, showing that procedurally generating variations of the scan helps train robust agents.

Finally, we tested all three models for robustness towards a change in camera parameters between simulation and the real robot. A full 15 trajectories were evaluated for each model trained with a wide vertical FOV and evaluated with a narrow one. PHONE2PROC is robust to this change, while RECON’s performance drops drastically.

4.4. Statistical Analysis

As described above, we have jointly evaluated PHONE2PROC and PROCTHOR models across 3 starting positions in 5 real environments with 5 target objects per environment (chosen from 7 unique types). Together this amounts to $(2 \text{ model types}) \times (3 \text{ positions}) \times (5 \text{ environments}) \times (5 \text{ targets}) = 150$ datapoints. In order to validate the statistical significance of our results, we follow a similar analysis as in [58] and model agent success using a logistic regression model in \mathbb{R} [43]. In particular, here we model all exogenous variables as fixed effects and, as starting positions are inherently nested within environments, we include all environment and starting position interactions. When fitting this model, we obtain the coefficient estimates

	(Intercept)	Phone2Proc	Bed	Chair	GarbageCan	Sofa	TV	Vase
Coef.	0.17	0.33	0.00	0.41	-0.12	0.2	-0.03	0.13
p-value	0.31	<0.0001	0.97	0.02	0.59	0.13	0.78	0.27

where, for space, we have excluded coefficients corresponding to environments and locations, as none of these coefficients were statistically significant at a 0.05 level. As the

above shows, the coefficient of interest (PHONE2PROC) is statistically significant even at a 0.0001 level. Interpreting these results, we see that when holding other factors constant, the use of a PHONE2PROC model is associated with $\exp(0.33) = 1.39$ times greater odds of success (95% confidence interval: $[1.2, 1.62]$) than when using a PROC-THOR model. Of all object categories, only the coefficient associated with chair was found to be statistically significant at a 0.05 level suggesting that chairs were associated with higher levels of success (*i.e.* may be generally easier to find). Altogether, we find strong statistical evidence suggesting that, across tested environments and object categories, PHONE2PROC was associated with higher success rates and that the estimated effect size was substantial.



Figure 8. Qualitative results. These demonstrate the ability of PHONE2PROC models to navigate to their desired object. The top-down map is for visualization purposes only and is an approximation of the path taken by the agent.

4.5. Qualitative Analysis

Fig. 8 illustrates exemplary trajectories from each test environment with a few ego-centric RGB images that is the agent’s only input. The trajectories show meaningfully different behavior for large vs. small objects (bed, sofa, and TV vs. apple and vase).

For large objects that don’t change location drastically (*e.g.* row 1), the agent seems to initially localize itself using known landmarks that appear in the scan (similar object categories, for instance) and then demonstrate efficient motion towards the room which contains the target object. We observed during our trials that often, when an agent navigating toward a large target loses its way, it would double back to a familiar large object and then restart direct progress. Note that the agent has no ground truth localization and must rely

Environment	Area (m ²)	Longest Path (m)	# Rooms	# Objects	# Scanned Objects
RoboTHOR-Real	34.5	8.1	4	51	14
6-Room Apartment	104.4	21.8	6	189	57
3-Room Apartment	65.4	8.2	3	105	26
Conference Room	98.3	10.0	1	48	32
Cafeteria	133.2	18.8	1	252	67

Table 2. The test real environments have a wide variety of layouts, usages, space, and object density. For visual layouts, see Fig. 2.

on its observations and its recollection of environment layout and object presence.

In contrast to big objects, for smaller items, the agent needs to explore efficiently to find the target. For instance, in the fourth row of Fig. 8, the agent searches for a small object that does not appear in the scan but is placed randomly in training rooms. Though again, it has no map, ground truth localization, or additional memory support, it demonstrates true exploration and high coverage of the possible area, ultimately achieving success (more qualitative results are provided in the supplementary videos).

4.6. Quantitative Analysis of The Environments

Results presented in Fig. 5 span a wide range of space usage, layout, and complexity as quantified in Table 2 to better demonstrate the power of PHONE2PROC. The conference room and cafeteria are large open spaces. The three living spaces require moving from room to room to locate objects. The 6-room apartment for instance, while most comparable in floor area to the conference room, is a long and narrow layout that requires hallway traversal for nearly every room transition. PHONE2PROC is most helpful in these environments over the baseline but makes a significant improvement in all layouts and semantic types of space.

5. Conclusion

In this paper, we introduced PHONE2PROC, a simple yet effective approach for training performant agents that are robust to the unpredictable nature of the real world. We demonstrated the capabilities of PHONE2PROC in five diverse environments and showed significant improvements in sim-to-real performance. Our environment-conditioned procedurally generated scenes are fully interactable, and we believe that future work will continue to explore its capabilities.

Acknowledgements The authors would particularly like to thank Luca Weihs for his invaluable assistance with project direction, baseline training, and the statistical analysis in Section 4.4. We would also like to thank Klemen Kotar, Winson Han, Eli VanderBilt, Ian Grunfeld, and Alvaro Her-rasti for assistance and discussions.

References

- [1] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On evaluation of embodied navigation agents. *ArXiv*, abs/1807.06757, 2018. 2
- [2] Peter Anderson, Ayush Shrivastava, Joanne Truong, Arjun Majumdar, Devi Parikh, Dhruv Batra, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. In *CoRL*, 2020. 3
- [3] Rika Antonova, Jingyun Yang, Priya Sundareshan, Dieter Fox, Fabio Ramos, and Jeannette Bohg. A bayesian treatment of real-to-sim for deformable object manipulation. *IEEE Robotics and Automation Letters*, 7:5819–5826, 2022. 3
- [4] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *ArXiv*, abs/2006.13171, 2020. 2
- [5] Roberto Bigazzi, Federico Landi, Marcella Cornia, Silvia Cascianelli, Lorenzo Baraldi, and Rita Cucchiara. Out of the box: Embodied navigation in the real world. In *CAIP*, 2021. 3
- [6] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Kumar Gupta, and Ruslan Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Conference on Neural Information Processing Systems*, abs/2007.00643, 2020. 2, 6
- [7] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. *arXiv preprint arXiv:2004.05155*, 2020. 2, 6
- [8] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Kumar Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12872–12881, 2020. 2
- [9] Prithvijit Chattopadhyay, Judy Hoffman, Roozbeh Mottaghi, and Aniruddha Kembhavi. Robustnav: Towards benchmarking robustness in embodied navigation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15671–15680, 2021. 2
- [10] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019. 3
- [11] Afshin Dehghan, Yikang Liao, Hongyu Xu, Fengfu Li, Yang Yang, Alex Ke, Max Maung, Kota Hara, Peter Fu, Louis Gong, Yihao Qian, Zhuoyuan Chen, Guangyu Zhao, Tianxin Deng, Shaowei Liu, Chunyuan Cao, Rongqi Chen, Zhenlei Yan, Peiyin Heng, Jimmy Pan, Yinbo Li, Haiming Gang, Praveen Sharma, Antoine Tarault, Daniel Ulbricht, Haris BaigKai Kang, Joerg Liebelt, Peng Wu, and Feng Tang. 3d parametric room representation with roomplan, 2022. 3
- [12] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, et al. Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3164–3174, 2020. 2, 4, 5, 6
- [13] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, et al. Procthor: Large-scale embodied ai using procedural generation. *Conference on Neural Information Processing Systems*, 2022. 2, 4, 5, 6
- [14] Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1290–1296. IEEE, 2021. 3
- [15] Daniel Dugas, Olov Andersson, Roland Y. Siegwart, and Jen Jen Chung. Navdreams: Towards camera-only rl navigation among humans. *ArXiv*, abs/2203.12299, 2022. 3
- [16] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Manipulathor: A framework for visual object manipulation. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4495–4504, 2021. 5
- [17] Kiana Ehsani, Shubham Tulsiani, Saurabh Gupta, Ali Farhadi, and Abhinav Kumar Gupta. Use the force, luke! learning to predict physical forces by simulating effects. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 221–230, 2020. 3
- [18] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 538–547, 2019. 2
- [19] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. Clip on wheels: Zero-shot object navigation as object localization and exploration. *ArXiv*, abs/2203.10421, 2022. 2
- [20] Georgios Georgakis, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, and Kostas Daniilidis. Learning to map for active semantic goal navigation. *ArXiv*, abs/2106.15648, 2022. 2
- [21] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *Advances in neural information processing systems*, 31, 2018. 5
- [22] Saurabh Gupta, Varun Tolani, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *International Journal of Computer Vision*, 128:1311–1330, 2017. 3
- [23] Meera Hahn, Devendra Singh Chaplot, and Shubham Tulsiani. No rl, no simulation: Learning to navigate without navigating. In *NeurIPS*, 2021. 2
- [24] Noriaki Hirose, F. Xia, Roberto Martín-Martín, Amir Sadeghian, and Silvio Savarese. Deep visual mpc-policy

- learning for navigation. *IEEE Robotics and Automation Letters*, 4:3184–3191, 2019. 3
- [25] Unnat Jain, Iou-Jen Liu, Svetlana Lazebnik, Aniruddha Kembhavi, Luca Weihs, and Alexander G. Schwing. Gridtopix: Training embodied agents with minimal supervision. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 15121–15131. IEEE, 2021. 2
- [26] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *European Conference on Artificial Life*, pages 704–720. Springer, 1995. 2
- [27] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5606–5616, 2022. 3
- [28] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, S. Chernova, and Dhruv Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5:6670–6677, 2020. 3, 6
- [29] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: Clip embeddings for embodied ai. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14809–14818, 2022. 2, 5
- [30] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Kumar Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *ArXiv*, abs/1712.05474, 2017. 1, 2, 5
- [31] Ashish Kumar, Saurabh Gupta, David F. Fouhey, Sergey Levine, and Jitendra Malik. Visual memory for robust path following. In *NeurIPS*, 2018. 3
- [32] Gulshan Kumar, N Sai Shankar, Himansu Didwania, Brojeshwar Bhowmick, and K Madhava Krishna. Gcexp: Goal-conditioned exploration for object goal navigation. In *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, pages 123–130. IEEE, 2021. 2
- [33] Guohao Li, Feng He, and Zhifan Feng. A clip-enhanced method for video-language understanding. *ArXiv*, abs/2110.07137, 2021. 2
- [34] Vincent Lim, Huang Huang, Lawrence Yunliang Chen, Jonathan Wang, Jeffrey Ichnowski, Daniel Seita, Michael Laskey, and Ken Goldberg. Real2sim2real: Self-supervised learning of physical single-step dynamic actions for planar robot casting. *2022 International Conference on Robotics and Automation (ICRA)*, pages 8282–8289, 2022. 3
- [35] Arjun Majumdar, Gunjan Aggarwal, Bhavika Devnani, Judy Hoffman, and Dhruv Batra. Zson: Zero-shot object-goal navigation using multimodal goal embeddings. *ArXiv*, abs/2206.12403, 2022. 2
- [36] Jan Matas, Stephen James, and Andrew J Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 734–743. PMLR, 2018. 3
- [37] Xiangyun Meng, Nathan D. Ratliff, Yu Xiang, and Dieter Fox. Scaling local control to large-scale topological navigation. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 672–678, 2020. 3
- [38] Xiangyun Meng, Yu Xiang, and Dieter Fox. Learning composable behavior embeddings for long-horizon visual navigation. *IEEE Robotics and Automation Letters*, 6:3128–3135, 2021. 3
- [39] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*, 2019. 5
- [40] Neil Nie, Samir Yitzhak Gadre, Kiana Ehsani, and Shuran Song. Structure from action: Learning interactions for articulated object 3d structure discovery. *ArXiv*, abs/2207.08997, 2022. 3
- [41] Ruslan Partsey, Erik Wijmans, Naoki Yokoyama, Oles Dobosevych, Dhruv Batra, and Oleksandr Maksymets. Is mapping necessary for realistic pointgoal navigation? *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17211–17220, 2022. 6
- [42] Aayush Prakash, Shoubhik Debnath, Jean-Francois Lafleche, Eric Cameracci, Gavriel State, Stan Birchfield, and Marc Teva Law. Self-supervised real-to-sim scene generation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16024–16034, 2021. 3
- [43] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022. 7
- [44] Santhosh K. Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X. Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *ArXiv*, abs/2109.08238, 2021. 2, 6
- [45] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. *CVPR*, 2022. 2
- [46] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016. 2
- [47] Assem Sadek, Guillaume Bono, Boris Chidlovskii, and Christian Wolf. An in-depth experimental study of sensor usage and visual reasoning of robots navigating in real environments. *2022 International Conference on Robotics and Automation (ICRA)*, pages 9425–9431, 2022. 3
- [48] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *ArXiv*, abs/1803.00653, 2018. 3
- [49] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv

- Batra. Habitat: A platform for embodied ai research. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9338–9346, 2019. [1](#), [2](#)
- [50] Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. Recon: Rapid exploration for open-world navigation with latent goal models. In *CoRL*, 2021. [3](#)
- [51] Dhruv Shah, Ajay Kumar Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. Gnm: A general navigation model to drive any robot. *ArXiv*, abs/2210.03370, 2022. [3](#)
- [52] Priya Sundaesan, Rika Antonova, and Jeannette Bohg. Diffcloud: Real-to-sim from point clouds with differentiable simulation and rendering of deformable objects. *ArXiv*, abs/2204.03139, 2022. [3](#)
- [53] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter 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)*, pages 23–30. IEEE, 2017. [2](#)
- [54] Joanne Truong, S. Chernova, and Dhruv Batra. Bi-directional domain adaptation for sim2real transfer of embodied navigation agents. *IEEE Robotics and Automation Letters*, 6:2634–2641, 2021. [3](#)
- [55] Joanne Truong, Max Rudolph, Naoki Yokoyama, S. Chernova, Dhruv Batra, and Akshara Rai. Rethinking sim2real: Lower fidelity simulation leads to higher sim2real transfer in navigation. *ArXiv*, abs/2207.10821, 2022. [2](#)
- [56] Huamin Wang, James F. O’Brien, and Ravi Ramamoorthi. Data-driven elastic models for cloth: modeling and measurement. *ACM SIGGRAPH 2011 papers*, 2011. [3](#)
- [57] Luobin Wang, Runlin Guo, Quan Ho Vuong, Yuzhe Qin, Hao Su, and Henrik I. Christensen. A real2sim2real method for robust object grasping with neural surface reconstruction. *ArXiv*, abs/2210.02685, 2022. [3](#)
- [58] Luca Weihs, Aniruddha Kembhavi, Kiana Ehsani, Sarah M. Pratt, Winson Han, Alvaro Herrasti, Eric Kolve, Dustin Schwenk, Roozbeh Mottaghi, and Ali Farhadi. Learning generalizable visual representations via interactive gameplay. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. [7](#)
- [59] Luca Weihs, Jordi Salvador, Klemen Kotar, Unnat Jain, Kuo-Hao Zeng, Roozbeh Mottaghi, and Aniruddha Kembhavi. Allenact: A framework for embodied ai research. *arXiv preprint arXiv:2008.12760*, 2020. [5](#)
- [60] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6743–6752, 2019. [2](#)
- [61] F. Xia, Amir Roshan Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9068–9079, 2018. [1](#), [2](#)
- [62] Linhai Xie, A. Markham, and Niki Trigoni. Snapnav: Learning mapless visual navigation with sparse directional guidance and visual reference. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1682–1688, 2020. [3](#)
- [63] Shan Yang, Junbang Liang, and Ming C. Lin. Learning-based cloth material recovery from video. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4393–4403, 2017. [3](#)
- [64] Wei Yang, X. Wang, Ali Farhadi, Abhinav Kumar Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *ArXiv*, abs/1810.06543, 2019. [2](#)
- [65] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Kumar Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3357–3364, 2017. [2](#)