INTERIORSIM: A PHOTOREALISTIC SIMULATOR FOR EMBODIED AI

Anonymous authors
Paper under double-blind review

ABSTRACT

Interactive simulators are becoming powerful tools for training embodied agents, but existing simulators suffer from limited content diversity, physical interactivity, and visual fidelity. We address these limitations by introducing InteriorSim, a photorealistic simulator for embodied AI in the home. To create our simulator, we worked closely with a team of professional artists for over a year to construct 300 unique virtual indoor environments with 2,566 unique rooms and 17,234 unique objects that can be manipulated individually. Each of our environments features detailed geometry, photorealistic materials, and a unique floor plan and object layout designed by a professional artist, i.e., we do not rely on remixing existing layouts to create additional content. Our environments are implemented as Unreal Engine assets, and we provide an OpenAI Gym interface for interacting with the environments via Python. We demonstrate the utility of our simulator by using it in a zero-shot sim-to-real transfer scenario, i.e., we train a point-goal navigation policy entirely in simulation that can successfully navigate through cluttered real-world environments when deployed on a real robot. We also demonstrate that our simulator is quantitatively more photorealistic than existing simulators measured by human comparisons and standard metrics for evaluating generative models. Finally, we demonstrate that our simulator achieves better sim-to-real performance than existing simulators on a real-world semantic segmentation task. All of our assets and code will be made available online.

1 INTRODUCTION

Training embodied agents to act safely and intelligently in the physical world is slow, labor-intensive, and potentially dangerous. In response to this challenge, the embodied AI community has developed a variety of interactive simulators, where agents can be trained and validated in simulation prior to being deployed in the physical world (Anderson et al., 2018; Armeni et al., 2016; 2017; Chang et al., 2017; Deitke et al., 2020; Dosovitskiy et al., 2017; Ehsani et al., 2021; Gan et al., 2021a;b; Kadian et al., 2020; Kolve et al., 2017; Li et al., 2021; Puig et al., 2018; Ramakrishnan et al., 2021; Savva et al., 2019; Shah et al., 2017; Shen et al., 2021; Straub et al., 2019; Szot et al., 2021; Xia et al., 2018; 2020; Xiang et al., 2020; Yan et al., 2018). These simulators have enabled rapid progress on increasingly complex and open-ended real-world tasks (e.g., point-goal navigation (Kadian et al., 2020; Xia et al., 2018; 2020), object navigation (Deitke et al., 2020), object manipulation (OpenAI et al., 2019), and autonomous driving (Codevilla et al., 2018)).

However, existing simulators have important limitations (see Table 1). Simulators that use artist-created environments typically provide a limited selection of scenes (e.g., a few dozen homes, or a few hundred isolated rooms), which can lead to severe over-fitting and poor sim-to-real transfer performance. To work around this limitation, simulators often provide *remixed* variants of each scene (e.g., where objects have been arranged in different configurations), but this approach is limited by the diversity of scenes prior to remixing. On the other hand, simulators that use scanned 3D environments provide larger collections of scenes, but offer little or no interactivity with objects, and are therefore not suitable for tasks that involve rearranging the environment to achieve a particular goal (e.g., loading a dishwasher) (Batra et al., 2020).

Additionally, both types of simulator offer limited visual fidelity, but for different reasons. Simulators that use artist-created environments are often forced to provide simplified assets, because it is too labor-intensive to author high-resolution geometry and complex materials at scale. Simulators

Figure 1: Example scenes in InteriorSim. We show one scene from each of the following styles (top to bottom, row-major): American, Chinese, European, European (simple), Japanese, Modern.

that use scanned environments avoid this issue, but exhibit pronounced 3D mesh reconstruction artifacts, and bake view-dependent lighting effects (e.g., glossy surfaces and specular highlights) onto meshes in a way that is incorrect at most viewing angles.

In this work, we introduce InteriorSim, a photorealistic simulator for embodied AI that addresses all of the limitations described above (see Figure 1). To create our simulator, we worked closely with a team of professional artists for over a year to construct 300 unique virtual indoor environments with 2,566 unique rooms and 17,234 unique objects that can be manipulated individually. Each of our environments features detailed geometry, photorealistic materials, and a unique floor plan and object layout designed by a professional artist, i.e., we do not rely on remixing existing layouts to create additional content. Our environments are implemented as Unreal Engine assets, and we provide an OpenAI Gym interface for interacting with the environments via Python. Together, these features make our simulator well-suited for embodied AI tasks that involve rearranging the environment (e.g., (Batra et al., 2020)) and deploying agents in the physical world (e.g., (Codevilla et al., 2018; Deitke et al., 2020; Kadian et al., 2020; Xia et al., 2018; 2020)).

We demonstrate the utility of our simulator by using it in a zero-shot sim-to-real transfer scenario, i.e., we train a point-goal navigation policy entirely in simulation that can successfully navigate through cluttered real-world environments when deployed on a real robot. We train our navigation policy with imitation learning, and we compare *simulated demonstrations* obtained automatically via global path planning in our simulator, to *real-world demonstrations* obtained via human pilots controlling a physical robot. Remarkably, we find that our simulated demonstrations outperform real-world demonstrations, suggesting that the sim-to-real gap can be overcome by using simulation-only privileged information, alongside high-fidelity rendering and physics. We demonstrate that our simulator is quantitatively more photorealistic than existing simulators measured by human comparisons and standard metrics for evaluating generative models. Finally, we demonstrate that our simulator achieves better sim-to-real transfer performance than existing simulators on a real-world semantic segmentation task. InteriorSim's code will be made available for the community under an open-source MIT license, and all assets will be free to use for academic purposes.

Simulator	Num. scenes		Num. rooms	Num. objects	Phys. realism	Photo-realism
	Before remixing	After remixing				
CHALET	10	–	58	330	⋆	⋆
VirtualHome	6	–	–	308	⋆	⋆⋆
SAPIEN	–	–	–	2,346	⋆⋆⋆	⋆⋆⋆
ThreeDWorld	15	–	~120	~2,500	⋆⋆⋆	⋆⋆⋆
AI2-THOR	120	–	–	~2,000	⋆⋆	⋆⋆
iGibson 2.0	15	**12,015**	108	570	⋆⋆⋆	⋆⋆
Habitat 2.0	6	105	–	92	⋆⋆	⋆⋆
InteriorSim (ours)	**300**	–	**2,566**	**17,234**	⋆⋆⋆	⋆⋆⋆

Table 1: Comparison to existing interactive simulators. We limit our comparison to indoor simulators providing collections of scenes and objects that can be manipulated individually, and we sort chronologically by latest release. For each simulator, we show the number of scenes *before remixing* and *after remixing* (e.g., rearranging objects, combining with previous datasets), as well as the *number of rooms* prior to remixing (– indicates quantities that are not specified in a simulator's publications or documentation). We show the level of *physical realism* available when agents are interacting with objects (⋆ indicates that only high-level interactions are possible; ⋆⋆ indicates that an articulated robot arm can be controlled; ⋆⋆⋆ indicates that both a robot arm and a gripper can be controlled). We also show the level of *photorealism* (⋆ indicates flat shading; ⋆⋆ indicates physically-based shading; ⋆⋆⋆ indicates more advanced rendering effects, e.g., path tracing).

2 Related work

Learning with synthetic data Synthetic datasets and simulation environments play a critical role in computer vision, machine learning, and robotics. See the recent survey by Nikolenko (2019).

Realistic simulators for embodied AI We discuss realistic indoor simulators for mobile agents in Section 1. Realistic simulators have also been developed for outdoor tasks (e.g., drone navigation (Shah et al., 2017), autonomous driving (Dosovitskiy et al., 2017)), and indoor tasks that involve a robot arm on a fixed base (e.g., opening doors (Urakami et al., 2019), object manipulation (James et al., 2020; Yu et al., 2019; Zhu et al., 2020), assistive care (Erickson et al., 2020), furniture assembly (Lee et al., 2021)). Additionally, standalone physics engines (Bullet, 2022; Lee et al., 2018; Todorov et al., 2012) and game engines (Epic Games, 2022; NVIDIA, 2022; Makoviychuk et al., 2021; Unity, 2022) can be used as simulators to train embodied agents, when combined with an appropriate learning objective, art assets, and a model of the agent dynamics. However, these simulators do not include ready-made collections of indoor environments, and are therefore not directly applicable to household tasks. In contrast, our simulator includes a large collection of photorealistic indoor environments, and is applicable to a wide range of household navigation and manipulation tasks.

Sim-to-real transfer for robotics Training perception models and control policies entirely in simulation, and subsequently deploying them on real-world robots, has been successfully demonstrated for wide range of tasks (e.g., grasping (Tobin et al., 2017), obstacle avoidance (Sadeghi & Levine, 2017) and aerobatic maneuvering (Kaufmann et al., 2020) on drones, following high-level user commands on ground vehicles (Codevilla et al., 2018), quadruped locomotion (Tan et al., 2018)). In a similar spirit to (Kadian et al., 2020; Xia et al., 2018; 2020), we demonstrate sim-to-real transfer on a point-goal navigation task, but we use a significantly less expensive robot platform with fewer on-board sensors, and therefore our sim-to-real experiments are easier to reproduce.

Metrics for evaluating generative models Quantitative metrics for evaluating generative models (Binkowski et al., 2018; Heusel et al., 2017) have been used to assess the photorealism of scanned indoor scenes (Ramakrishnan et al., 2021) and data-driven rendering methods (Richter et al., 2021). We use these same metrics to evaluate the photorealism of our simulator.

3 InteriorSim

We designed InteriorSim according to three main desiderata. First, we wanted to support a collection of environments that is as large, diverse, and high-quality as possible. Second, we wanted sufficient

Figure 2: Highlighted functionality in InteriorSim. InteriorSim supports a high-quality path-tracing rendering mode (left), a real-time rendering mode (middle-top), and multiple lighting configurations for each scene (middle-bottom). To illustrate that our scenes are cluttered with dynamic objects, we show this scene after a simulated earthquake (right).

physical realism to support realistic interactions with a wide range of household objects. Third, we wanted as much photorealism as possible, while still maintaining enough rendering speed to support training complex embodied agent behaviors.

Motivated by these desiderata, we chose to implement InteriorSim on top of the Unreal Engine (Epic Games, 2022), which is an industrial-strength open-source game engine. Unreal Engine has a vibrant community of developers and artists, a rich ecosystem of plugins, a large marketplace of photorealistic assets, and it is free for non-commercial use. Additionally, Unreal Engine features state-of-the-art physical simulation and real-time rendering functionality, and it is constantly improving. By building on top of the Unreal Engine, we can expect to benefit as new engine features are introduced and optimized to take advantage of evolving hardware (e.g., (Epic Games, 2021; Karis et al., 2021)). We highlight the functionality of InteriorSim in Figure 2, and we provide additional implementation details in the supplementary material.

Environments In this subsection, we describe our pipeline for generating interactive Unreal Engine environments. We provide summary statistics for our environments in Figure 3.

Objects: Our artists modeled, textured, and semantically labeled 26,019 unique household objects from scratch using commercial and in-house geometric modeling tools. Our artists marked 17,234 of these objects as movable, and the remaining 8,785 as static. The semantic label applied to each object comes from a set of 100 common household semantic categories.

Physical properties: We assigned physical properties (e.g., mass, density, friction) to each object using a supervised learning approach that takes the object's rendering data as input. Once an object has been modeled by an artist, it consists of one or more *rendering meshes*, each *rendering mesh* has exactly one *rendering material*, and each *rendering material* has a set of *parameters* (e.g., roughness, glossiness, etc). Using this data as input, our supervised learning approach consists of the following steps. First, for each rendering mesh, we compute a simplified closed *collision mesh* using the V-HACD library (V-HACD, 2022). Second, we define a set of 78 *physical materials* with known friction and density coefficients. Third, we construct a training set of 1,000 rendering materials, and we manually assign a physical material to each rendering material in the training set. Fourth, we train a random forest classifier (Breiman, 2001) to automatically assign a physical material to each of our remaining rendering materials, using the values of the rendering material's parameters as features. Fifth, we assign a mass to each collision mesh based on its volume and the density of its assigned physical material. Using this approach, we automatically assigned physical materials to 42,723 unique rendering materials, achieving a top-1 classification accuracy of 80% on a small held-out validation set.

Scenes: Using our collection of unique objects as a starting point, our artists designed and assembled 300 unique indoor scenes (2,566 unique rooms) using in-house modeling tools, and exported them as Unreal Engine assets using an in-house conversion pipeline. Each scene features a unique floor plan and object layout, i.e., we do not rely on remixing existing layouts to create additional content. In total, our scenes contain 100,165 object instances (56,971 are movable, 43,194 are static).

Data cleaning: As a final data cleaning step, we loaded each scene into the Unreal Engine editor, and manually corrected any content problems that were introduced earlier in our pipeline (e.g., objects that are grouped together incorrectly, objects that are missing semantic information, collision meshes that are slightly interpenetrating, etc).

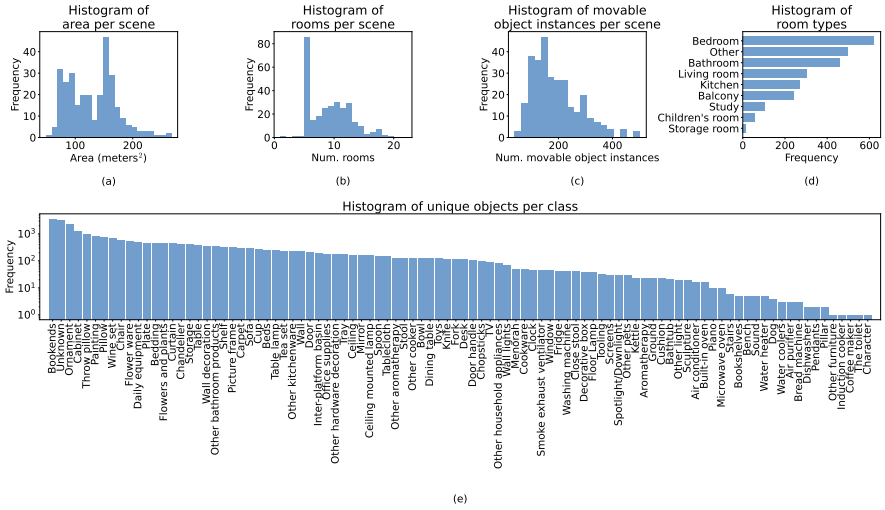

Figure 3: Environment statistics at the granularity of scenes (a,b,c), rooms (d), and objects (e).

Gym interface We provide a complete OpenAI Gym (Brockman et al., 2016) interface for interacting with our environments via Python. InteriorSim is therefore compatible with any high-level reinforcement learning framework that expects a Gym interface (e.g., (Liang et al., 2018; Plappert, 2016; Raffin et al., 2021; Weng et al., 2021)). We verified the correctness of our interface by training a variety of point-goal navigation policies in our environments using RLLib (Liang et al., 2018).

In order to provide a well-behaved Gym interface (i.e., fixed time-steps, synchronous execution) on top of the Unreal Engine, we must carefully coordinate the execution of the engine's game-loop via Python, and access internal engine data at specific points within the game-loop's control flow. We perform this coordination via a custom Unreal Engine plugin written in C++. Our plugin also makes it easy for other researchers to extend InteriorSim by implementing new agents, sensors, and tasks.

Agents InteriorSim currently supports four distinct embodied agents. Our *OpenBot Agent* provides identical image observations to a real-world OpenBot (Müller & Koltun, 2021), implements an identical control interface, and has been modeled with accurate geometry and physical parameters. It is therefore well-suited for sim-to-real experiments (see Section 4.1). Our *Fetch Agent* and *LoCoBot Agent* have also been modeled using accurate geometry and physical parameters (LoCoBot, 2022; Wise et al., 2016), and each features a physically realistic gripper. These agents are therefore well-suited for rearrangement tasks (Batra et al., 2020). Our *Camera Agent* can be teleported anywhere, and is useful for collecting static datasets (see Section 4.2).

Sensors By default, our agents return photorealistic egocentric observations from camera sensors, as well as wheel encoder states and joint encoder states. Additionally, our agents can optionally return several types of privileged information. First, our agents can return a sequence of waypoints representing the shortest path to a goal location, as well as perfect GPS and compass observations that point directly to the goal, both of which can be useful when defining navigation tasks (see Section 4.1). Second, our agents can return pixel-perfect semantic segmentation and depth images, which can be useful when controlling for the effects of imperfect perception in downstream embodied tasks, and collecting static datasets (see Section 4.2).

Tasks InteriorSim currently supports two distinct tasks. Our *Point-Goal Navigation Task* randomly selects a goal position in the scene's reachable space, computes a reward based on the agent's distance to the goal, and triggers the end of an episode when the agent hits an obstacle or the goal. Our *Freeform Task* is an empty placeholder task that is useful for collecting static datasets.

Rendering quality and performance InteriorSim supports a high-quality path-tracing rendering mode that is suitable for generating static datasets, and a real-time mode that is suitable for interactive training (see Figure 2 for a quality comparison). Our high-quality mode takes multiple minutes per frame, and our real-time mode runs at roughly 30–60 frames per second on a typical gaming PC. We use our real-time rendering mode in all of the experiments in this paper.

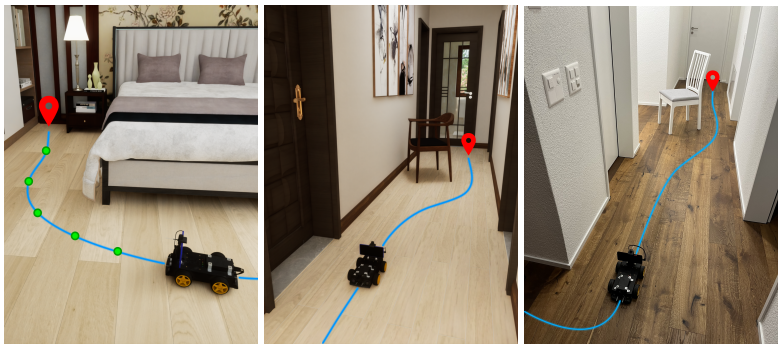

Figure 4: Our pipeline for zero-shot navigation policy transfer. Left: to generate training data in simulation, we compute a sequence of collision-free waypoints (green dots) to a goal (red marker) via global path planning, and a virtual OpenBot agent tracks these waypoints with a PID controller (forming a trajectory to the goal shown in blue). Center: we evaluate trained policies in simulation without using any privileged waypoint information, but using ground-truth localization information. Right: we evaluate trained policies on a real-world OpenBot without using any privileged information.

4 EXPERIMENTS

In order to evaluate the usefulness of InteriorSim for downstream tasks, we perform three experiments. First, we train navigation policies in InteriorSim and deploy them in the real world. Second, we compare the photorealism of InteriorSim to existing simulators. Third, we evaluate the sim-to-real transfer performance of InteriorSim on a real-world semantic segmentation task.

4.1 ZERO-SHOT NAVIGATION POLICY TRANSFER

In this subsection, we demonstrate that it is possible to train a point-goal navigation policy with simulated observations from InteriorSim, and successfully transfer it to the real world. The trained navigation policy runs in real-time on a smartphone, enabling anyone with a smartphone and a $50 robot (Müller & Koltun, 2021) to reproduce our results. We also investigate the influence of data quantity, diversity, and realism on downstream task performance by training policies with varying amounts of simulated data, and comparing the real-world performance of each policy to a baseline policy that has been trained with real data. We summarize our policy transfer pipeline in Figure 4, we summarize our results in Table 2, and we provide additional details in the supplementary material.

Point-goal navigation using the OpenBot framework To conduct our sim-to-real experiments, we use the OpenBot framework (Müller & Koltun, 2021), which leverages the capabilities of modern smartphones to deploy navigation policies on small ground vehicles. The OpenBot framework includes an imitation learning pipeline that takes driving demonstrations as input (typically provided by a user manually piloting a vehicle), and produces a learned control policy as output. This learning pipeline ingests egocentric visual observations captured by the smartphone camera and high-level navigation commands (e.g., turn signal, goal vector) as input data; and corresponding low-level control outputs as target data. Once this data has been used to train a control policy, the policy can be deployed on a smartphone as a TensorFlow Lite model (Ignatov et al., 2019), and used to autonomously control an OpenBot vehicle.

The policy network architecture used in the OpenBot framework is a straightforward variant of PilotNet (Bojarski et al., 2016), which maps egocentric RGB observations (160×90 resolution) and high-level navigation inputs to low-level control outputs. In our case, the navigation input vector is three-dimensional, and consists of the distance between the agent and its goal, concatenated with a two-dimensional heading vector that points towards the goal. When deployed on a real-world OpenBot vehicle, we use odometry data from ARCore (Google, 2022) running on the smartphone to estimate the navigation input vector. The control output vector is two-dimensional, and consists of normalized motor voltages to be applied to the left and right wheels of the vehicle. Differences between these two voltage values cause the vehicle to turn, and positive voltage values cause the vehicle to move forwards.

	Evaluated in simulation		Evaluated in the real world	
Policy	Success (% ↑)	Collision (% ↓)	Success (% ↑)	Collision (% ↓)
Trained with 2% of sim data	10.9	87.3	2.5	86.6
Trained with 10% of sim data	44.9	54.9	45.7	43.3
Trained with 50% of sim data	53.0	46.5	58.5	34.7
Trained with 100% of sim data	**54.3**	**43.1**	**63.7**	**29.4**
Trained with real data	35.5	63.9	44.8	38.9

Table 2: Navigation policy transfer results. Policies trained with 50% or more of our simulated data outperform a policy trained with real data across all metrics. Even a policy trained with 10% of our simulated data achieves comparable performance to a policy trained with real data, despite having access to roughly half as many data samples (~260K samples versus ~424K samples).

Data collection We use InteriorSim to automatically collect demonstrations for the OpenBot imitation learning pipeline. More specifically, we use the virtual OpenBot agent included in InteriorSim, which can access ground-truth localization information, and can obtain an obstacle-free path to any reachable point in an environment. Building on this functionality, our data collection procedure consists of randomly sampling a start and goal position in an InteriorSim environment, obtaining a collision-free path to the goal, and tracking the path to the goal with a PID controller. We repeat this procedure in different environments to automatically collect a large dataset of demonstrations. In total, we collect data along 5,000 trajectories in 50 InteriorSim environments, resulting in ~2.6M data samples consisting of {image, high-level navigation input, low-level control output} triplets. We also construct 3 subsets containing 2%, 10%, and 50% of the data respectively.

Additionally, we collect a real-world dataset using crowd workers, each of whom is tasked with piloting a real-world OpenBot. We filter the raw data in our real-world dataset to discard very short (<30 frames) and very long (>300 frames) trajectories. Our final real-world dataset consists of 10,475 trajectories (~424K data samples) collected in 70 unique environments by novice pilots.

Training details We train policies with the same hyperparameters on each of the aforementioned datasets. We train all polices to regress from an image and navigation input vector to a control output vector, using the training recipe and loss from Müller & Koltun (2021), with a batch size of 512 and a learning rate of 0.001 for 200 epochs.

Evaluation in simulation When evaluating in simulation, we deploy our policies on a virtual OpenBot agent in 3 distinct InteriorSim environments that have not been seen during training. For each policy, and for a total of 100 runs per test environment, the policy is tasked with navigating from a predefined set of initial locations to a predefined set of goal locations while avoiding obstacles. The same initial and goal locations are used for all policies. For this evaluation, the policies do not have access to any privileged waypoint information, but do have access to perfect localization information.

Evaluation in the real world When evaluating in the real world, we deploy our policies on a real-world OpenBot in 5 distinct real-world environments that have not been seen during training. For each policy and environment, the policy is tasked with navigating to a goal location in 4 distinct configurations: a diagonal trajectory with no obstacles, a U-turn trajectory, an L-turn trajectory, and a straight trajectory with an obstacle. We conduct 10 repeated trials for each configuration. This protocol results in 20 distinct trajectories (5 environments $\times$ 4 trajectories per environment), with 10 repeated trials per trajectory, for each of our 5 policies. For this evaluation, the policies do not have access to any privileged information.

Results and discussion We find that policies trained with automatically generated InteriorSim demonstrations outperform those trained with real-world human demonstrations. For example, the policy trained with 10% of our simulated data achieves a better success rate than our policy trained with real data when evaluated in the real world (46% versus 45%), despite having access to roughly half as many training samples (~260K samples versus ~424K samples). This finding suggests that the sim-to-real gap can be overcome by using simulation-only privileged information, alongside high-fidelity rendering and physics. We attribute this finding to the improved quality and consistency of demonstrations that can be obtained from InteriorSim, i.e., InteriorSim automatically generates close-to-optimal trajectories to use as demonstrations, whereas our crowd workers were not able to produce high-quality trajectories as consistently).

Simulator	Humans prefer InteriorSim (%)	Images		Patches	
		FID (↓)	KID ($\times 10^2$ ↓)	FID (↓)	KID ($\times 10^2$ ↓)
Matterport3D	72.5 ± 1.3	59.6	4.0 ± 0.1	54.5	4.0 ± 0.1
Replica	63.5 ± 1.4	77.4	4.1 ± 0.1	59.7	$\underline{3.1 \pm 0.1}$
HM3D	65.9 ± 1.3	$\underline{51.7}$	$\underline{3.7 \pm 0.2}$	$\underline{49.6}$	3.9 ± 0.1
ThreeDWorld	67.3 ± 1.3	107.1	9.2 ± 0.2	97.2	7.2 ± 0.1
AI2-THOR	60.7 ± 1.4	64.0	4.3 ± 0.2	53.2	3.3 ± 0.1
ReplicaCAD	60.3 ± 1.4	106.1	9.0 ± 0.2	90.0	4.8 ± 0.1
InteriorSim (ours)	–	**52.4**	$\mathbf{4.2 \pm 0.2}$	**47.3**	$\mathbf{3.2 \pm 0.1}$

Table 3: Photorealism results. We show how often humans consider InteriorSim images to be more realistic than images from existing simulators in pairwise comparisons, as well as other quantitative metrics for evaluating generative models. Humans consider InteriorSim to be more realistic than existing interactive simulators between 60% and 67% of the time, and InteriorSim is quantitatively more photorealistic across all metrics. Bold values indicate the best-performing interactive simulator, underlined values indicate the best-performing non-interactive scanned dataset.

4.2 Evaluating photorealism

In this subsection, we quantitatively evaluate the photorealism of InteriorSim. At a high level, our methodology is to generate images using InteriorSim and other baseline simulators. We compare InteriorSim images to the images from other simulators in a set of pairwise human comparisons. We also compare simulator images to real-world images using standard quantitative metrics for evaluating generative models: Frèchet Inception Distance (FID) (Heusel et al., 2017) and Kernel Inception Distance (KID) (Binkowski et al., 2018). These metrics are well-suited to compare simulators, because they measure a simulator's ability to generate images that are high-quality (i.e., near the manifold of real-world images) and diverse (i.e., sufficient to cover the manifold of real-world images). We summarize our results in Table 3, and we provide additional methodological details in the supplementary material.

Real-world dataset We use RealEstate10K (Zhou et al., 2018) as our real-world dataset, which contains roughly 10M images of indoor homes gathered from roughly 10K YouTube videos. We chose RealEstate10K because it depicts a wide variety of indoor scenes that are semantically similar to InteriorSim and the baseline simulators we use.

Baseline simulators We use ThreeDWorld (Gan et al., 2021a;b), AI2-THOR (Deitke et al., 2020; Ehsani et al., 2021; Kolve et al., 2017), and ReplicaCAD (Szot et al., 2021) as our baseline simulators, because they provide a level of photorealism that is representative of existing interactive indoor simulators (see Table 1). For additional context, we also include several non-interactive scanned datasets in our experiments: Matterport3D (Chang et al., 2017), Replica (Straub et al., 2019), and HM3D (Habitat-Matterport, 2022; Ramakrishnan et al., 2021). We chose these scanned datasets because they include semantic segmentation information (which we use to automatically select appropriate rendered images, see the supplementary material for details), and they are the most semantically similar to RealEstate10K. To facilitate fair comparisons, we set the rendering quality parameters in each simulator to be as high as possible, and we set InteriorSim to use its real-time rendering mode.

Comparing simulator images using pairwise human comparisons When performing our pairwise human comparisons, we follow the same experimental protocol as Chen & Koltun (2017). Specifically, we randomly sample 500 images from each simulator, and we perform untimed pairwise comparisons between InteriorSim and each simulator using crowd workers. We ask each worker, *"which image is more realistic, A or B?"* We repeat each image comparison 10 times, leading to a total of 30,000 pairwise comparisons.

Comparing simulator images to real-world images using metrics for generative models As noted in (Richter et al., 2021), the metrics we use are sensitive to the spatial layouts of simulator images and real-world images (i.e., the spatial layout of a simulator image must roughly match a real-world image to be considered photorealistic). In order to isolate this potential confounding effect, we randomly sample 6 patches per image at 256×256 resolution (we perform comparisons of entire images at 852×480 resolution), and we include comparisons of these patches in Table 3 alongside comparisons of entire images.

4.3 EVALUATING SIM-TO-REAL SEMANTIC SEGMENTATION PERFORMANCE

In this subsection, we evaluate the sim-to-real transfer performance of InteriorSim on a real-world semantic segmentation task. At a high level, our methodology is to pre-train several semantic segmentation models using images from InteriorSim and other baseline simulators. We then fine-tune each model on an appropriate real-world training set, and evaluate the task performance of each model on a real-world validation set.

We evaluate 40-class RGB-only segmentation performance on the NYUv2 dataset (Gupta et al., 2013; Silberman et al., 2012), and we report mean intersection-over-union (mIoU) as the evaluation metric. We chose NYUv2 as our real-world dataset in order to facilitate comparisons with existing synthetic datasets (Li et al., 2020; McCormac et al., 2017; Roberts et al., 2021; Song et al., 2017; Zhang et al., 2017).

Pre-training	mIoU (↑)
None	39.8
Matterport3D	41.1
Replica	42.4
HM3D	42.4
AI2-THOR	43.2
ReplicaCAD	42.0
InteriorSim (ours)	**43.7**

Table 4: Sim-to-real transfer results on 40-class RGB-only NYUv2 semantic segmentation. Pre-training on InteriorSim achieves better sim-to-real transfer performance than existing interactive simulators and non-interactive scanned datasets.

For this experiment, we use the same images that we collected from each simulator in Section 4.2, and we manually map the category labels available in each simulator to NYU40 labels. We use DeepLabv3 (Chen et al., 2017) as our model architecture with a ResNet-101 encoder (He et al., 2016) initialized using ImageNet weights (Russakovsky et al., 2015). We follow the training recipe provided in the PyTorch documentation exactly (PyTorch, 2022) during our pre-training and fine-tuning phases, and we train for 100 epochs during the pre-training phase, and 40 epochs during the fine-tuning phase. We encountered numerical instabilities when training on ThreeDWorld images, and therefore we do not report results for ThreeDWorld.

5 CONCLUSIONS

We introduced InteriorSim, a photorealistic simulator for embodied AI in the home. InteriorSim provides a wider variety of fully interactive scenes (300) and objects (17,000+) than existing interactive simulators, and achieves an unprecedented level of visual fidelity. We leveraged these features to train a point-goal navigation policy entirely in simulation that can successfully navigate through cluttered real-world environments, and we found that automatically generated demonstrations from our simulator outperform real-world human demonstrations. We also found that InteriorSim improves sim-to-real transfer performance on a real-world semantic segmentation task.

We believe InteriorSim can enable progress on a wide range of embodied AI tasks, especially those that involve traversing and rearranging densely cluttered indoor environments. InteriorSim also has the potential to enable new grasping and manipulation behaviors that generalize across objects and scenes, and ultimately across the sim-to-real gap. Additionally, InteriorSim could be used to actively render photorealistic synthetic datasets, generating the right data at the right time for a perception model as it is being trained. Moving beyond InteriorSim, we see abundant opportunities to co-design game engines and learning algorithms to amortize rendering costs, extend simulation capabilities in ways that assist training, and ultimately increase the spatial intelligence of embodied agents.

REFERENCES

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018.

Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D semantic parsing of large-scale indoor spaces. In *CVPR*, 2016.

Iro Armeni, Alexander Sax, Amir R. Zamir, and Silvio Savarese. Joint 2D-3D-semantic data for indoor scene understanding. arXiv, 2017.

Dhruv Batra, Angel X. Chang, Sonia Chernova, Andrew J. Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, Manolis Savva, and Hao Su. Rearrangement: A challenge for embodied AI. arXiv, 2020.

Mikolaj Binkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *ICLR*, 2018.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. arXiv, 2016.

Leo Breiman. Random forests. *Machine Language*, 45(1), 2001.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. arXiv, 2016.

Bullet. Bullet Real-Time Physics Simulation. `http://pybullet.org`, 2022.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *3DV*, 2017.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. arXiv, 2017.

Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017.

Felipe Codevilla, Matthias Muller, Antonio Lopez, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *ICRA*, 2018.

Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. RoboTHOR: An open simulation-to-real embodied AI platform. In *CVPR*, 2020.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *CoRL*, 2017.

Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ManipulaTHOR: A framework for visual object manipulation. In *CVPR*, 2021.

Epic Games. Lumin in UE5: Let there be light! `http://www.youtube.com/watch?v=Dc1PPYl2uxA`, 2021.

Epic Games. Unreal Engine. `http://www.unrealengine.com`, 2022.

Zackory Erickson, Vamsee Gangaram, Ariel Kapusta, Karen C. Liu, and Charles C. Kemp. Assistive Gym: A physics simulation framework for assistive robotics. In *ICRA*, 2020.

Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, Megumi Sano, Kuno Kim, Elias Wang, Michael Lingelbach, Aidan Curtis, Kevin Feigelis, Daniel M. Bear, Dan Gutfreund, David Cox, Antonio Torralba, James J. DiCarlo, Joshua B. Tenenbaum, Josh H. McDermott, and Daniel L.K. Yamins. ThreeDWorld: A platform for interactive multi-modal physical simulation. In *NeurIPS Datasets and Benchmarks Track*, 2021a.

Chuang Gan, Siyuan Zhou, Jeremy Schwartz, Seth Alter, Abhishek Bhandwaldar, Dan Gutfreund, Daniel L.K. Yamins, James J. DiCarlo, Josh McDermott, Antonio Torralba, and Joshua B. Tenenbaum. The ThreeDWorld Transport Challenge: A visually guided task-and-motion planning benchmark for physically realistic embodied AI. arXiv, 2021b.

Google. ARCore. `http://developers.google.com/ar`, 2022.

Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *CVPR*, 2013.

Habitat-Matterport. Habitat-Matterport 3D Semantics Dataset. `http://aihabitat.org/datasets/hm3d-semantics`, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017.

Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. AI Benchmark: All about deep learning on smartphones in 2019. In *ICCV Workshops*, 2019.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The robot learning benchmark & learning environment. *R-AL*, 2020.

Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *R-AL*, 2020.

Brian Karis, Rune Stubbe, and Graham Wihlidal. Nanite: A Deep Dive. SIGGRAPH Course on Advances in Real-Time Rendering and Games, 2021.

Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Matthias Muller, Vladlen Koltun, and Davide Scaramuzza. Deep drone acrobatics. In *RSS*, 2020.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An interactive 3D environment for visual AI. arXiv, 2017.

Jeongseok Lee, Michael X. Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S. Srinivasa, Mike Stilman, and C. Karen Liu. DART: Dynamic animation and robotics toolkit. *The Journal of Open Source Software*, 3(22), 2018.

Youngwoon Lee, Edward S. Hu, and Joseph J. Lim. IKEA furniture assembly environment for long-horizon complex manipulation tasks. In *ICRA*, 2021.

Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. iGibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *CoRL*, 2021.

Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, Sai Bi, Zexiang Xu, Hong-Xing Yu, Kalyan Sunkavalli, Miloš Hašan, Ravi Ramamoorthi, and Manmohan Chandraker. OpenRooms: An end-to-end open framework for photorealistic indoor scene datasets. arXiv, 2020.

Eric Liang, Richard Liaw, Philipp Moritz, Robert Nishihara, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *ICML*, 2018.

LoCoBot. LoCoBot: An Open Source Low Cost Robot. `http://www.locobot.org`, 2022.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. arXiv, 2021.

John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J. Davison. SceneNet RGB-D: Can 5M synthetic images beat generic ImageNet pre-training on indoor segmentation? In *ICCV*, 2017.

Matthias Müller and Vladlen Koltun. OpenBot: Turning smartphones into robots. In *ICRA*, 2021.

Sergey I. Nikolenko. Synthetic data for deep learning. arXiv, 2019.

NVIDIA. NVIDIA Isaac Sim. http://developer.nvidia.com/isaac-sim, 2022.

OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. arXiv, 2019.

Matthias Plappert. keras-rl. http://github.com/keras-rl/keras-rl, 2016.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. VirtualHome: Simulating household activities via programs. In *CVPR*, 2018.

PyTorch. Models and pre-trained weights. http://pytorch.org/vision/stable/models.html#semantic-segmentation, 2022.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268), 2021.

Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M 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. In *NeurIPS Datasets and Benchmarks Track*, 2021.

Stephan R. Richter, Hassan Abu AlHaija, and Vladlen Koltun. Enhancing photorealism enhancement. arXiv, 2021.

Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 2015.

Fereshteh Sadeghi and Sergey Levine. CAD^{2}RL: Real single-image flight without a single real image. In *RSS*, 2017.

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. In *ICCV*, 2019.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017.

Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Perez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne Tchapmi, Micael Tchapmi, Kent Vainio, Josiah Wong, Li Fei-Fei, and Silvio Savarese. iGibson 1.0: A simulation environment for interactive tasks in large realistic scenes. In *IROS*, 2021.

Nathan Silberman, Pushmeet Kohli, Derek Hoiem, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012.

Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017.

Julian Straub, Thomas Whelan Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica Dataset: A digital replica of indoor spaces. arXiv, 2019.

Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *NeurIPS*, 2021.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *RSS*, 2018.

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 *IROS*, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IROS*, 2012.

Unity. Unity Real-Time Development Platform. `http://www.unity.com`, 2022.

Yusuke Urakami, Alec Hodgkinson, Casey Carlin, Randall Leu, Luca Rigazio, and Pieter Abbeel. DoorGym: A scalable door opening environment and baseline agent. In *NeurIPS Workshop on Deep Reinforcement Learning*, 2019.

V-HACD. V-HACD. `http://github.com/kmammou/v-hacd`, 2022.

Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Hang Su, and Jun Zhu. Tianshou: A highly modularized deep reinforcement learning library. arXiv, 2021.

Melonee Wise, Michael Ferguson, Derek King, Eric Diehr, and David Dymesich. Fetch & Freight: Standard platforms for service robot applications. In *IJCAI Workshop on Autonomous Mobile Service Robots*, 2016.

Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: Real-world perception for embodied agents. In *CVPR*, 2018.

Fei Xia, William B. Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive Gibson Benchmark (iGibson 0.5): A benchmark for interactive navigation in cluttered environments. *R-AL*, 5(2), 2020.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A Simulated Part-based Interactive ENvironment. In *CVPR*, 2020.

Claudia Yan, Dipendra Misra, Andrew Bennett, Aaron Walsman, Yonatan Bisk, and Yoav Artzi. CHALET: Cornell house agent learning environment. arXiv, 2018.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2019.

Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *CVPR*, 2017.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. arXiv, 2020.