

# INTERIORSIM: A PHOTOREALISTIC SIMULATOR FOR EMBODIED AI

## SUPPLEMENTARY MATERIAL

**Anonymous authors**

Paper under double-blind review

### 1 INTERIORSIM IMPLEMENTATION DETAILS

**Environments and agents** We show additional environments in Figures 1. We show our *Fetch* and *LoCoBot* agents in Figure 2, and our *OpenBot* agent in Figure 4 of the main paper.

**C++ plugin architecture** At a high level, our C++ plugin has a remote procedure call server running over a TCP/IP connection in a background thread, which our Python code uses to send actions, receive observations, and to coordinate the behavior of the plugin. At a low level, our plugin hooks into specific entry points at the beginning and end of the engine’s game-loop, and inside these entry points, our plugin can optionally pause and unpause the game, block and unblock the game-loop, and execute tasks in a work queue. By making a sequence of remote procedure calls to our plugin, our Python code can implement any function in the Gym interface.

For example, to implement `observation = environment.step(action)`, our Python code makes the following remote procedure calls: *BeginStep*, *ApplyAction*, *Step*, *GetObservation*, *EndStep*. We briefly describe the behavior of each remote procedure call from the perspective of our C++ plugin below, and in parentheses we describe the call’s return behavior from the perspective of our Python code. For this discussion, we assume that the game is initially paused, the game-loop is initially unblocked, our plugin’s work queue is initially empty, and the reward and other similar metadata is encoded directly in the observation. *BeginStep* requests to block at the beginning of the game-loop, and execute any tasks that arrive in the work queue (returns immediately). *ApplyAction* adds a task to the work queue to apply a user-specified action (returns when the task is completed). *Step* requests to unpause the game, unblock the beginning of the game-loop, immediately block at the end of the game-loop, and execute any tasks that arrive in the work queue when the end of the game-loop is reached (returns when the end of the game-loop is reached). *GetObservation* adds a task to the work queue to retrieve the latest rendered image observation from GPU memory and return it (returns an image when the task is completed). *EndStep* requests to pause the game, and unblock the end of the game-loop (returns immediately). At this point, our user-specified action has been applied, the state of the world has been updated for exactly one time-step, an up-to-date image observation has been returned to our Python code, and our Python code is ready to return. All other functions in the Gym interface (e.g., `observation = environment.reset()`) can be implemented similarly.

**Extending InteriorSim** We designed our C++ plugin to make it as easy as possible for researchers and practitioners to implement new agents, sensors, and tasks. Implementing an agent or task involves implementing the desired agent behavior or task logic in a C++ class (e.g., using Unreal Engine primitives to apply control forces, or to compute a reward), and implementing a small C++ interface for sending and receiving data in the format expected by our framework. These implementations tend to be lightweight, e.g., our *Camera Agent* and *Point-Goal Navigation Task* can each be implemented in a few hundred lines of code. Additionally, we provide tools to import any robot whose geometry and kinematic structure has been described in URDF format (URDF, 2022).

**Personally identifiable information and offensive content** Our environments are populated exclusively with common household objects, and therefore do not contain personally identifiable information or offensive content.



Figure 1: Additional InteriorSim environments in the following architectural styles (top to bottom, row-major): American, Chinese, European, European (simple), Japanese, Modern.



Figure 2: Our *Fetch* (left) and *LoCoBot* (right) agents interacting with InteriorSim environments.

## 2 ZERO-SHOT NAVIGATION POLICY TRANSFER DETAILS

**Training details** We show egocentric views seen during training in Figure 3. We train all of our policies on a cluster with Intel Xeon CPUs and NVIDIA 3090 GPUs. The policy that sees 100% of our simulated data takes approximately 26 hours to train on our cluster, and the training times of our other policies are proportional to the amount of data seen during training.

**Evaluation details** We show egocentric views seen in our real-world evaluation in Figure 3. The success metric we report is stricter than what is typically reported in the point-goal-navigation literature (Anderson et al., 2018), because we count any collision between the agent and its environment as a failed episode (see Table 2 in the main paper). We opt for this stricter success metric because



Figure 3: Simulated egocentric views seen during training (top row), and real-world egocentric views seen during evaluation (bottom row) in our navigation experiments. We show images at the native resolution ( $160 \times 120$ ) seen by our navigation policies.

Simulator	Humans prefer InteriorSim (%)	Images		Patches	
		FID ( $\downarrow$ )	KID ( $\times 10^2 \downarrow$ )	FID ( $\downarrow$ )	KID ( $\times 10^2 \downarrow$ )
Hypersim	$59.5 \pm 1.4$	48.5	$2.4 \pm 0.1$	32.8	$1.5 \pm 0.1$

Table 1: Quantitative photorealism metrics for Hypersim (Roberts et al., 2021).

OpenBots are more sensitive to collisions than the robots typically used in sim-to-real navigation experiments (Deitke et al., 2020; Kadian et al., 2020; Xia et al., 2018; 2020). On an OpenBot, even minor collisions with the environment can lead to significant state estimation errors, so for simplicity we count any collision as a failed episode. Therefore, the strict success percentages we report are not directly comparable to the more relaxed success percentages found in the existing literature.

**Real-world data and crowd workers** We recruited crowd workers to collect real-world data in their own homes, and we provided each worker with their own OpenBot. The crowd workers were able to review all collected data before uploading it to our servers, to guarantee that they did not accidentally upload any personal information. Each crowd worker was scheduled to collect data for approximately 5 hours over 5 weeks, at a rate of approximately \$10 USD per hour of collected data. We anonymized all real-world data before using it for training.

### 3 EVALUATING PHOTOREALISM METHODOLOGICAL DETAILS

We show images from our photorealism experiment in Figure 4. For additional context, we show photorealism metrics for Hypersim (Roberts et al., 2021) in Table 1. Hypersim is a synthetic dataset that has been carefully rendered offline, and it achieves high photorealism scores across all metrics. Nonetheless, humans consider InteriorSim to be more realistic than Hypersim 59% of the time.

**Generating views in simulation** RealEstate10K is biased towards upright views captured at human eye-level. We want to generate similarly biased views in each simulator, because we do not want the metrics we use to be confounded by drastic differences in viewpoint sampling. With this motivation in mind, we generate views for each simulator using a simple two-phase heuristic. In our first phase, for each available scene, we randomly sample camera positions that are in the scene’s reachable space and are roughly at human eye-level, and we randomly sample camera orientations that are roughly upright. In our second phase, we automatically select a subset of the views from our first phase. When performing subset selection, we prefer views where no single semantic class (e.g., wall or floor) is dominant. So for each scene, we select views where the entropy of the view’s per-pixel semantic class histogram is highest. For each simulator, we generate 54K images in our first phase, and we automatically select 10K images in our second phase. We generate images at  $852 \times 480$  resolution with a horizontal field-of-view of  $90^\circ$ , which roughly matches the field-of-view of a typical RealEstate10K image.



Figure 4: Images from the scanned datasets, simulators, and real-world reference dataset used in our photorealism experiments (top to bottom, row major): Matterport3D (Chang et al., 2017), Replica (Straub et al., 2019), HM3D (Habitat-Matterport, 2022; Ramakrishnan et al., 2021), InteriorSim (ours), ThreeDWorld (Gan et al., 2021a;b), AI2-THOR (Deitke et al., 2020; Ehsani et al., 2021; Kolve et al., 2017), ReplicaCAD (Szot et al., 2021), and RealEstate10K (Zhou et al., 2018).

#### 4 LIMITATIONS AND SOCIETAL IMPACT

Although our environments cover a variety of architectural styles (see Figure 1 and Figure 1 in the main paper), there are many styles that are not well-represented. Moreover, even within architectural styles that are well-represented, our environments do not necessarily match the statistics of real-world homes. The limited diversity of our environments can have an adverse effect on embodied agents that are trained in InteriorSim, and subsequently deployed in homes that are not well-represented in our data. The metrics we use for evaluating photorealism attempt to quantify this issue (e.g., FID (Heusel et al., 2017) and KID (Binkowski et al., 2018) explicitly measure how well our environments match a large dataset of real-world homes (Zhou et al., 2018)), and we generally outperform existing simulators on these metrics, but the limited diversity of our environments is nonetheless an issue that researchers and practitioners should be aware of.

The speed of our physics and rendering is another important limitation. Although our speed is comparable to existing simulators that are built on top of video game engines (Deitke et al., 2020; Dosovitskiy et al., 2017; Ehsani et al., 2021; Gan et al., 2021a;b; Lee et al., 2021; Puig et al., 2018; Shah et al., 2017; Urakami et al., 2019; Yan et al., 2018), we are 2–3 orders of magnitude slower than the fastest 3D simulators that are optimized for embodied AI workloads (Savva et al., 2019; Szot et al., 2021). However, our simulator is quantitatively more photorealistic than existing simulators (see Table 3 in the main paper), and we believe that highly photorealistic simulators will play a complimentary role to very fast simulators for the foreseeable future. Indeed, embodied agents with the best real-world performance will likely be trained in both types of simulator (e.g., basic skills and behaviors can be learned in very fast simulators, and can be subsequently fine-tuned in highly photorealistic simulators to improve sim-to-real transfer performance). Therefore, we do not consider this limitation to be overly burdensome.

#### 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.
- Mikolaj Binkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *ICLR*, 2018.
- 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.
- Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Motlaghi, 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.
- Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwalder, Nick Haber, Megumi Sano, Kuno Kim, Elias Wang, Michael Lingelbach, Aidan Curtis, Kevin Feigels, 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 Bhandwalder, 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.
- Habitat-Matterport. Habitat-Matterport 3D Semantics Dataset. <http://aihabitat.org/datasets/hm3d-semantics>, 2022.
- 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.
- 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.
- 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.
- Youngwoon Lee, Edward S. Hu, and Joseph J. Lim. IKEA furniture assembly environment for long-horizon complex manipulation tasks. In *ICRA*, 2021.
- Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. VirtualHome: Simulating household activities via programs. In *CVPR*, 2018.
- 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.
- 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.
- 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.
- 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 Von-drus, 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.
- 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.
- URDF. URDF. <http://github.com/ros/urdf>, 2022.
- 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 Tchaptmi, 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.
- Claudia Yan, Dipendra Misra, Andrew Bennett, Aaron Walsman, Yonatan Bisk, and Yoav Artzi. CHALET: Cornell house agent learning environment. arXiv, 2018.
- Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018.