In this supplement, we provide more details about our datasets (Appendix A), episode generation procedure (Appendix B), analysis & experiment details (Appendix C), training plots and finetuning results (Appendix C), and agent failure case analysis (Appendix E).

A. Dataset details

A.1. HSSD dataset construction details

We show the process for constructing HSSD in Figure 1. Starting with the glTF format assets representing the 211 scenes from Floorplanner, we use node information of the underlying asset IDs to decompose and extract over 18K unique 3D assets representing furniture and other objects. The architectural layout for each of the 211 scenes is what remains after this extraction from each scene. The object assets and the architecture are then compressed as described in the main paper.

The 3D assets are used to create a 3D model database which we clean and annotate with semantic information. We use a UI that allows us to select multiple 3D models and tag them with semantic attributes such as WordNet synset, room that the object is typically found in, and on what side the object typically attaches to other objects (bottom vs vertical vs top). For the semantic categorization step, we start with tags that are provided by Floorplanner and refine and correct the categories. The interface allows annotators to pull up a 3D view of each object and closely examine it. The 3D view also provides an interface for semantically annotating the up and front orientation of each object so we have semantically aligned objects. We find most objects already have consistent alignment, and only re-align objects that are not consistently aligned (typically wall objects).

In addition, we also mark whether the 3D asset represents multiple objects. For 3D assets that are marked as being composed of multiple objects, we follow the process depicted in Figure 2 to decompose the 3D asset into multiple objects. We first obtain an automatic segmentation using connected component analysis, and then have users manually "paint" and "label" the objects. Due to the clean geometry, we can obtain clean segmentations. Our interface allows for easy marking of object parts, displaying of the bounding box of annotated objects, copying of labels, undo/redo operations etc. The annotated objects are algorithmically extracted, deduplicated and aligned.

At the scene level, we also identify floater objects and exterior doors. Floater objects are objects that are outside of the scenes, and should not be part of the scene. We remove such objects. For ObjectNav experiments, we also remove interior doors but keep exterior doors (to prevent the agents from wandering outside). In addition, we also remove animate objects (animals and humans) from our scenes for all ObjectNav experiments.

A.2. HSSD dataset statistics

Figure 3 shows a word cloud visualization of categories in HSSD, with the text font size representing the total count of unique object instances in each category. We see that our dataset contains a diverse set of object categories. As described in the main paper, we annotate the objects in HSSD using a taxonomy based on WordNet, but extended to include additional common object categories. Figure 4 shows a subtree of this WordNetCO category hierarchy, focusing on lamp objects. The breadth and fine-grained nature of the taxonomy allows for future experiments with embodied AI agents tackling scenarios requiring perception of objects closer to an open vocabulary setting.

The object co-occurrence analysis in the main paper is on the basis of a set of 28 common object types that are shared between ProcTHOR [2], HSSD, and HM3DSem [9]. The complete list of these categories is: alarm_clock, bed, book, bottle, bowl, chair, chest_of_drawers, couch, cushion, drinkware, fridge, laptop, microwave, picture, plate, potted_plant, shelves, shoes, sink, stool, table, table_lamp, toaster, toilet, trashcan, tv, vase, washer_dryer.

We plot the size distribution (measured by the diagonal length of the bounding box in meters) of the 28 object categories in Figure 5. We see that the HSSD objects exhibit realistic sizes, with some categories having fairly narrow size distributions (e.g., shoes) and some having fairly broad distributions (e.g., pictures, shelves and beds).

A.3. HSSD qualitative visualizations

In Figure 6 we show example object instances for a number of categories from HSSD. We see that HSSD exhibits a rich diversity of object geometry, apperance, and physical sizes across many categories. This diversity is beneficial for experiments studying generalization of perception capabilities for embodied AI agents. These objects also populate the scenes in HSSD in a way that produces more realistic environments. We show first-person views in Figure 7 and top-down views in Figure 8. Overall, we see that HSSD scenes exhibit more realistic architectural layouts than ProcTHOR [2], and come closer to real-world scans from HM3DSem [9] in terms of the richness and density of objects populating each room.

A.4. AI2-THOR datasets in Habitat

To construct rigorous experiments comparing between the HSSD and ProcTHOR [2] datasets, we port and optimize ProcTHOR assets in the same fashion as HSSD so that they can be efficiently used in the Habitat simulator platform [7]. We built on top of the AI2-THOR Unity interface [5] to export all Unity prefab objects and scene assets to glTF format using UnityGLTF[1]. We also export a corresponding JSON-format metadata file with each asset to record information

[1] github.com/KhronosGroup/UnityGLTF

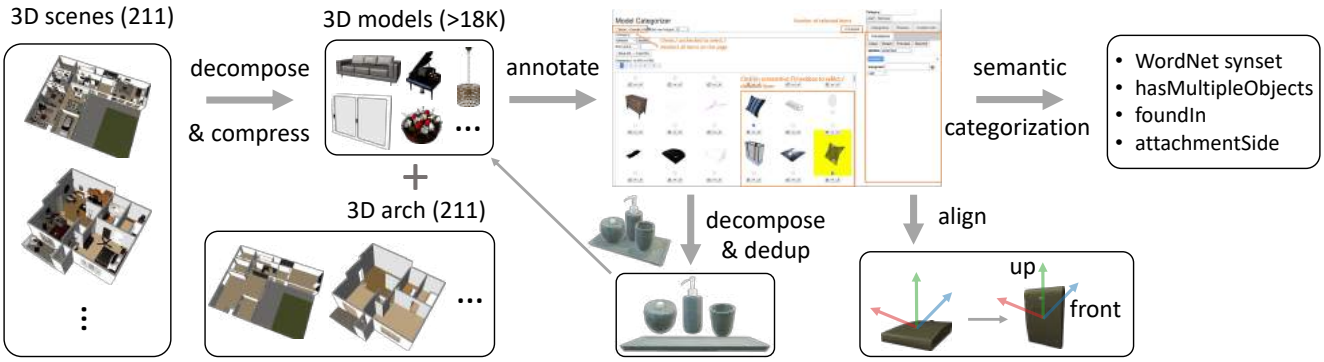

Figure 1. **Overview of HSSD construction pipeline.** We first decompose the original 211 3D scenes into more than 18K individual per-object 3D models, and architectural geometry for each scene. The per-object models are then annotated with a variety of semantics including WordNet synsets. The objects are also decomposed and semantically aligned so that they have a consistent up and front orientation.

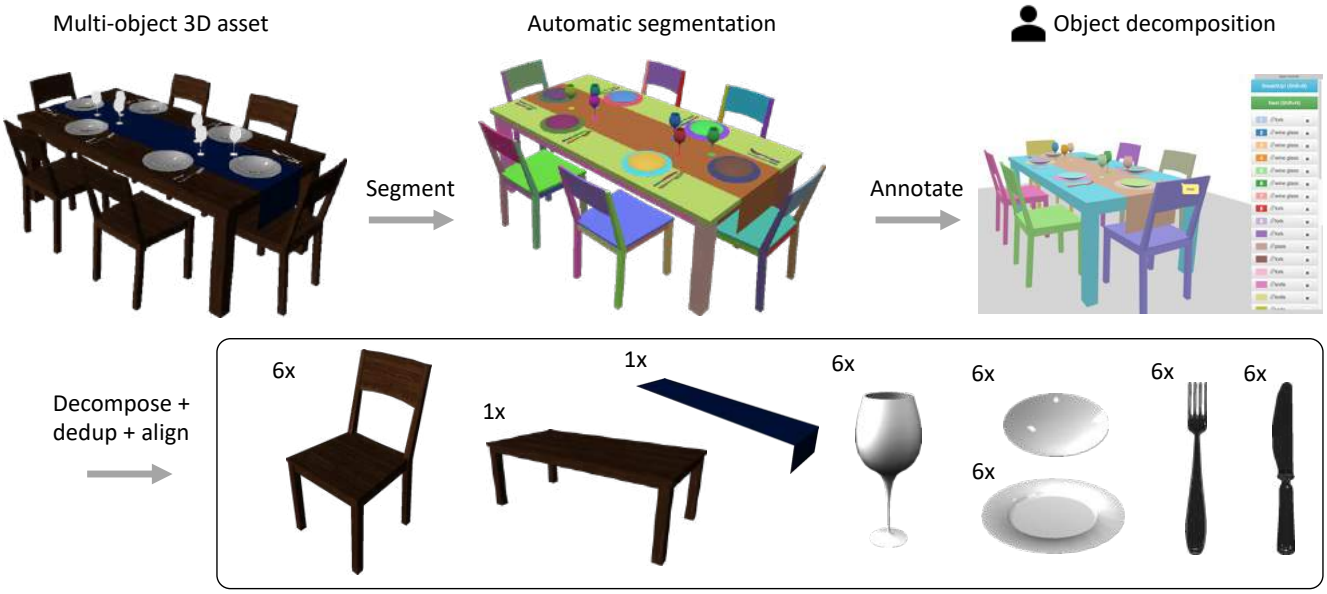

Figure 2. **Illustration of decomposition process for multi-object assets.** We create independent object assets for arrangements such as the dining table seen at the left. The process involves an automatic segmentation, manual annotation to group segments into individual objects, and finally an algorithmic decomposition, deduplication, and alignment of all extracted object instances.

such as the semantic category label, position and orientation of the object. For iTHOR [5], RoboTHOR [1] and ArchitecTHOR, we filter the structural objects in the scene so that we leave only architectural objects (walls, floors, ceilings). We then zero-center all exported objects, and re-orient the objects to standardized object-centric coordinates. Subsequently, we can use the position and orientation information to correctly place the objects wherever they are observed in the original scene. Note that we re-use assets across scenes to reduce on-disk and in-memory size. Since ProcTHOR [2] has procedurally generated architectures, we construct the geometry of the architecture with the specified textures from the ProcTHOR scene layout specification JSON format, and create a glTF asset for each scene architecture. All doors are exported using the AI2-THOR Unity interface in opened state to allow for navigation between rooms. This porting of the AI-2THOR assets to Habitat format enables us to take advantage of the faster simulation speeds provided by Habitat [8] and run experiments with any combination of iTHOR, RobotTHOR, ArchitectTHOR, and ProcTHOR scenes.

A.5. Why not use 3D-FRONT?

3D-FRONT [3] is a popular dataset for 3D scene generation research. However, the scenes are sparsely populated. Due to limited rights in releasing the original 3D assets for the scenes, 3D models in the scenes for the 3D-FRONT dataset are replaced with 3D models from the 3D-FUTURE [4] dataset. This limited the presence of object in rooms to a

Figure 3. **Word cloud of object categories in HSSD.** Font sizes indicate unique instance count per category. The HSSD contains a rich variety of categories, with many instances especially for categories that exhibit diversity in the real world (wall art, rugs etc.)

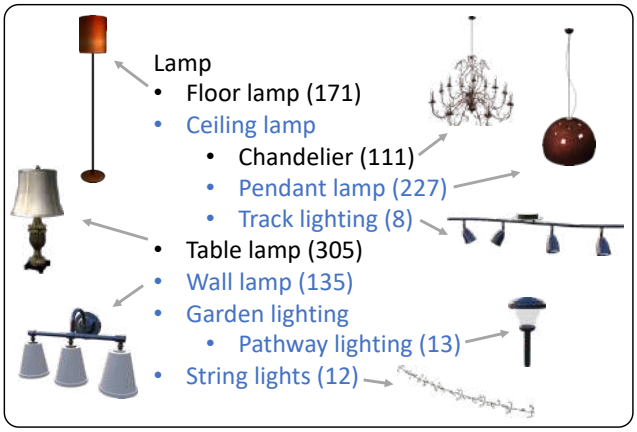

Figure 4. **Lamp object category hierarchy in HSSD.** Our category hierarchy leverages an augmented taxonomy of WordNet synsets that we call WordNetCO (for WordNet Common Objects). The figure shows synsets that we introduced in blue. The count of object instances within each synset is indicated by the number, and an example instance is shown for select synsets.

subset of room categories (e.g., living rooms, dining rooms, bedrooms) and left other rooms (e.g., kitchens, bathrooms, and closets) empty. There is also a limited number of placements of smaller objects on top of larger furniture objects (other than un-decomposed multi-object combinations), and no wall decoration objects. Another impact of using replaced assets is that there are scenes with objects that are interpenetrating each other. All of these factors make the 3D-FUTURE dataset much less well-suited for semantic indoor navigation where we expect the scenes to be well-populated, and objects to be realistically placed.

B. Episode generation details

Viewpoint sampling. In each scene, for each indoor goal object instance (e.g., bed, chair, couch), we first sample a grid of prospective viewpoints around the object and then

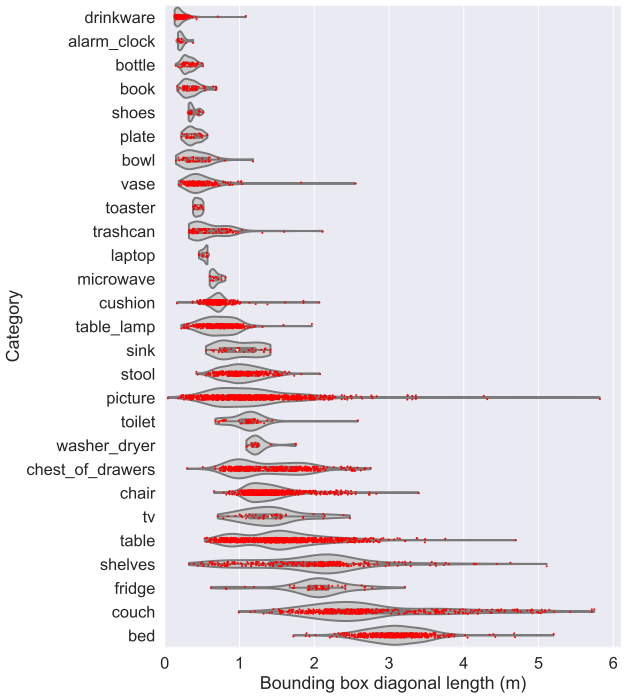

Figure 5. **Distributions of physical sizes for the 28 category set from HSSD.** Distributions are on the diagonal length of the bounding box for each instance within the category. Categories are sorted by average diagonal length from top (smallest) to bottom (largest). The HSSD objects are scaled to have realistic physical sizes. Shelves, tables, pictures, couches have a wide range of physical sizes. In contrast, categories such as alarm clocks, bottles, plates, and shows have much narrower physical size distributions.

reject viewpoints that do not have a navigable position below them, those that are too far (distance > 1m) from the object's bounding box, and those that are outdoors or outside the house (in HSSD scenes). After snapping the valid viewpoints from the previous step to a nearby navigable position, we remove all the viewpoints snapped to small (radius < 1.5m) navigable islands (e.g., found on tabletops, counters, beds). Next, we spawn the agents on these valid viewing positions (facing the object) and compute the object's visibility at these positions by measuring the frame coverage, i.e. the fraction of image pixels belonging to object. We reject positions from where frame coverage is less than 0.1%. After getting valid viewpoints, we uniformly sample random starting positions (one per episode) across the scene such that: 1) the geodesic distance between each starting position and the goal instance is at least 1 meter but less than 30 meters; and 2) a fraction of the nearly straight-line episodes (ratio of geodesic distance to Euclidean distance < 1.05) are rejected.

We present sample episode visualizations for two goal TV instances in HSSD and ProcTHOR through a top-down map in Figure 11. The bounding box of the goal TV instance

Figure 6. **Example objects from several categories.** The HSSD-200 dataset contains a broad variety of object categories, each with a diverse set of object instances within each category. Note the variety of lamp categories (table lamps, floor lamps, wall lamps), each exhibiting diversity of object instances with different physical sizes, geometry, and fine-grained appearance.

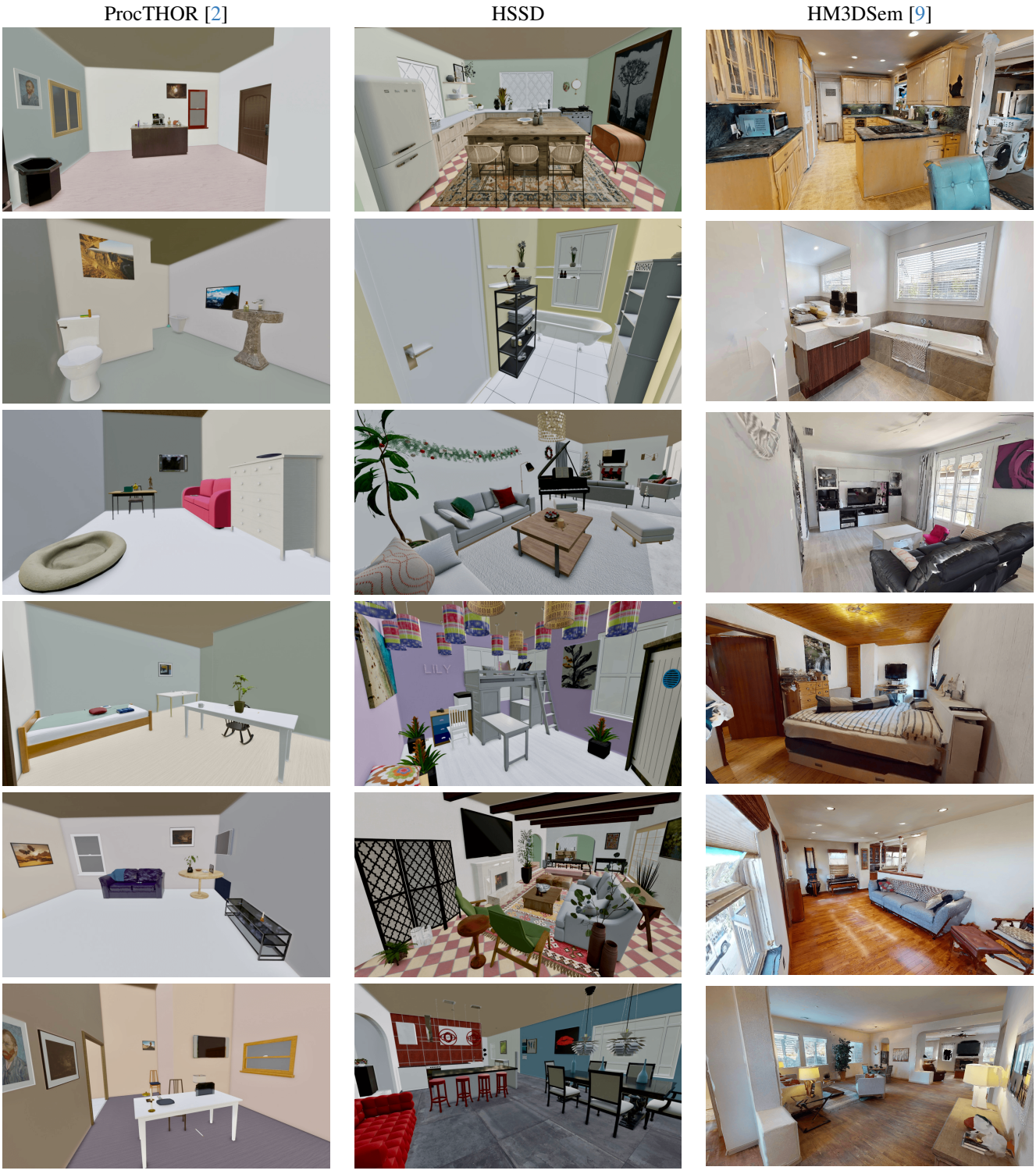

Figure 7. **Qualitative comparison of first-person views from ProcTHOR [2], HSSD and HM3DSem [9].** The HSSD scenes exhibit a richer diversity of objects and are more realistically populated than the ProcTHOR scenes. Images are rendered using Blender's Eevee renderer with all parameters at default settings. The ProcTHOR scenes and HSSD scenes are shaded with ambient occlusion and screen-space reflections, while the HM3DSem scenes are rendered without shading.

Figure 8. **Top-down views of scenes from ProcTHOR [2], HSSD and HM3DSem [9].** Compared to ProcTHOR, the HSSD scenes exhibit more realistic architectural layouts with corridors between rooms, non-rectilinear wall outlines and densely populated rooms. These characteristics bring HSSD closer to real-world environments as captured in the HM3DSem dataset.

is outlined with a black box and the viewpoints are shown in green (valid), blue (invalid due to being far from the object), and yellow (invalid due to being unnavigable) pixels. The orange pixels denote the episode starting positions.

Note that HSSD also has scene regions outside the house (e.g., backyards, gardens, balconies). We restrict all episodes to indoor regions as we focus on indoor-only navigation. Both the goal object and episode start position are required to be inside the house, and doors leading to the exterior are closed. A handful of scenes do not have a clear distinction

Figure 9. **3D-FRONT [3] scenes are sparsely populated.** Algorithmic object replacement was used to place object instances, and some room types are unfortunately left empty (e.g., kitchens, bathrooms, closets). This is due to limited rights to release the original 3D assets used in the scenes.

Figure 10. **Object inter-penetrations in the 3D-FRONT dataset.** The algorithmic object replacement unfortunately produces cases such as the ones shown here (see red circles). We show the scene with original object textures (left) and semantically colored by object category (right).

between indoors and outdoors and are therefore excluded from episode generation. For this reason, we generate training episodes for 122 scenes out of the 125 scenes in the training set.

The inherent scene size distribution differences between ProcTHOR, HSSD, and HM3D are also reflected in the distributions of episode geodesic distance that emerge in episodes generated from each of these scene datasets. See Figure 12 for a comparison. ProcTHOR has a high number of (easier) low geodesic distance episodes (due to a good number of small 1-3 room houses), with an exponential decay in the number of episodes as the distance increases (in bigger houses with more rooms). On the other hand, HSSD and HM3D have more similar distributions for both train and validation episode datasets.

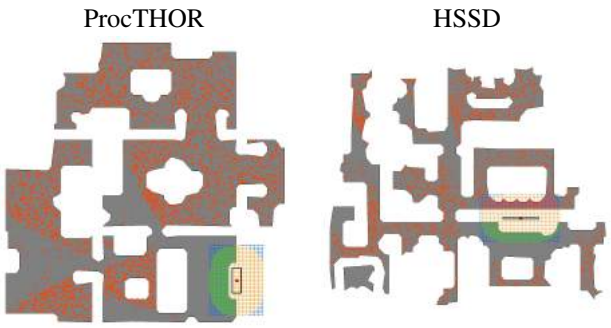

Figure 11. **ObjectNav episode generation visualization.** We show generated episode goal and starting positions for TVs in example scenes from HSSD and ProcTHOR. The goal object is outlined with a black box, the valid viewpoints are shown in green, and the episode start positions are shown with orange points.

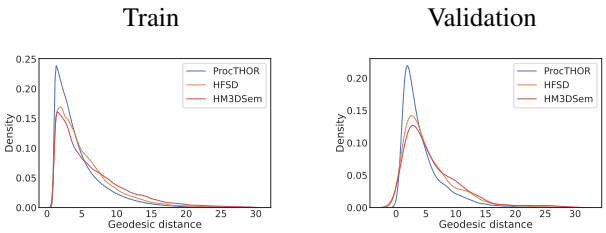

Figure 12. **Geodesic distance distribution of episode datasets.** We compare the distribution of geodesic distances across the episode datasets of ProcTHOR [2], HSSD, and HM3DSem [9]. Note how the differences in scene size distributions lead to significantly higher numbers of (easier) low geodesic distance episodes in ProcTHOR compared to fairly similar distributions between HSSD and HM3DSem.

C. Analysis & experiment details

Hierarchical clustering algorithm details. Given the object co-occurrence matrix C we obtain as described in the main paper, we first compute the dissimilarity matrix $D = 1 - C$. Then, we compute the distance for each unique pair $D(i, j)$, constructing an $n * (n - 1)/2$-dimensional vector. This vector is then used for hierarchical clustering with the farthest point algorithm to compute the distance between clusters and output a linkage matrix. We use SciPy's hierarchical clustering implementation to do this and form flat clusters. Cutting into flat cluster requires a distance threshold (maximum distance between clusters). We use a threshold $t = 0.8$ to compute the similarity scores reported in the main paper.

ProcTHOR-122. We disentangle scene dataset scale and scene dataset realism by creating a version of the ProcTHOR-10K dataset [2] that matches the scale of HSSD, as measured by number of scenes and navigable area. We sample 122 scenes from ProcTHOR-10K, matching the navigable area

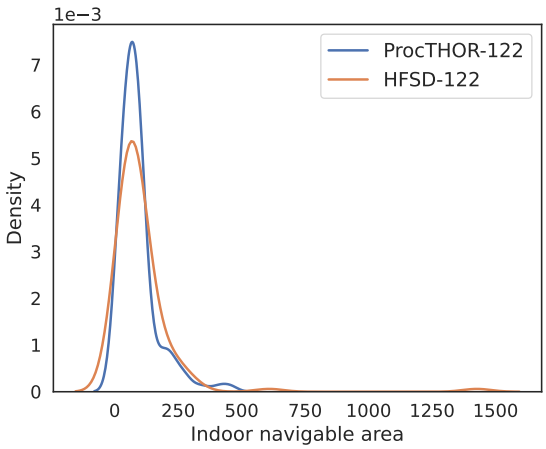

Figure 13. **Navigable area distributions for comparable scale datasets.** We prepare a 122-scene subset of ProcTHOR-10K [2] that matches HSSD's training set in scale in terms of number of scenes and the navigable area distribution. We refer to this scene dataset as ProcTHOR-122.

distribution of HSSD's training dataset as closely as possible. The navigable area distributions of these scale-matched scene datasets are in Figure 13.

HSSD-60 and ProcTHOR-60. To measure the impact of scene dataset scale, we also create variants of HSSD and ProcTHOR-122 with 60 scenes. We do this by randomly sampling 60 scenes out of 122. We refer to these scene dataset variants as HSSD-60 and ProcTHOR-60.

D. Training plots and finetuning results

Training and evaluation plots. In the main paper we reported zero-shot performance of agents trained on different datasets in **??**. Here, we present the agent training plots as well as validation set performance plots during training (on the same dataset's validation set). Figure 14 shows these plots. All agents reach validation set convergence by approximately 200M steps of experience. The results in the main paper use the agent checkpoint with highest validation set SPL from each training run. We also plot zero-shot performance of agents on HM3DSem and MP3D validation datasets across number of training steps in Figure 15. These plots show that overall HSSD-pretrained agents generalize better to real-world 3D scanned scenes than ProcTHOR and iTHOR-pretrained agents.

Finetuning results. In addition to the zero shot generalization experiments which were the focus of our work, we also present experiments with agents finetuned on the target dataset. The agents are pre-trained on variants of the HSSD and ProcTHOR training datasets and then finetuned

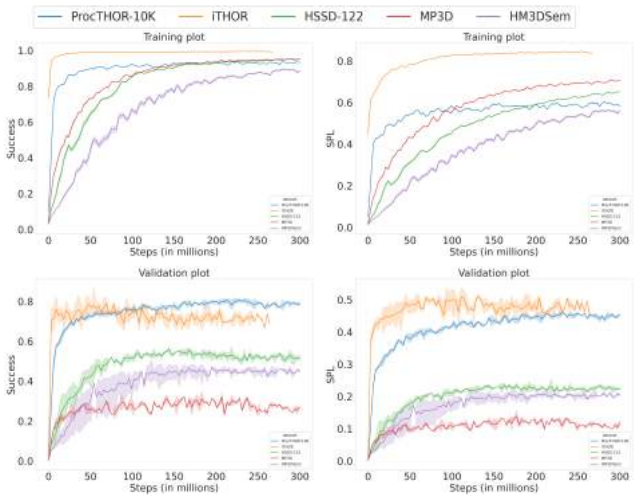

Figure 14. **Agent training plots.** We provide plots of agent performance during training on each dataset's training set and validation set. These plots correspond to the zero-shot agents presented in **??** of the main paper. Each agent is trained on the indicated dataset (iTHOR, ProcTHOR, HSSD, HM3DSem, and MP3D) to convergence. The plots show results from three independent training runs. Validation set performance for all agents saturates by approximately 200M steps of experience. Note that agents differ on when they reach convergence, with iTHOR agents doing so significantly faster likely due to the simplicity of the one-room scenes in the dataset.

on the HM3DSem training set. Specifically, for each agent, we finetune the agent checkpoint that has the best zero-shot performance on the HM3DSem validation set in terms of the SPL metric. See Table 1 for a summary of the results. In the table we compare these finetuned agents against the performance of an agent trained directly on HM3DSem. We find that all agents converge to similar levels of performance after finetuning, irrespective of the pre-training dataset. Performance in terms of the success metric ranges between 47.85 and 48.48, while the combined success and efficiency performance as measured by SPL ranges between 22.16 and 23.1, with HSSD agents retaining a lead over ProcTHOR agents. This trend is not surprising as finetuning on the target dataset is expected to reduce performance gaps due to differences between the original training datasets.

E. Agent failure case analysis

Inspired by Ramrakhya et al. [6], we analyze common failure cases when evaluating the HSSD-trained agent on the HM3DSem [9] val set (after fine-tuning on the HM3DSem train set). We randomly sample 100 val set episodes where the agent failed to succeed and analyze the modes of failure by classifying the agent's performance into the following failure cases:

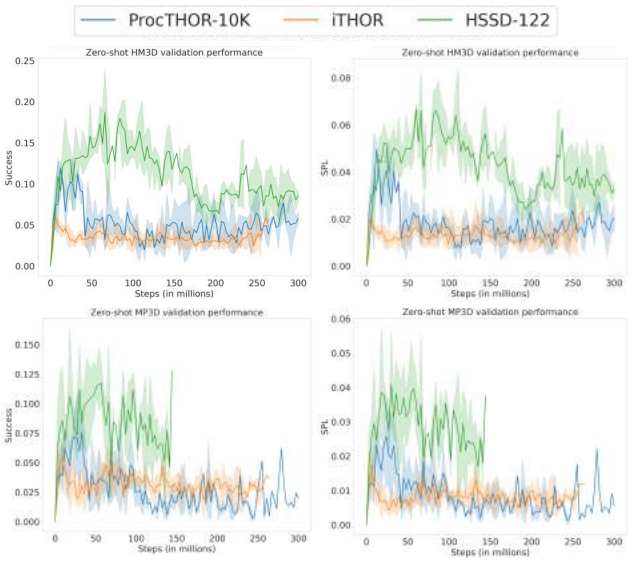

Figure 15. **Zero-shot evaluation on HM3DSem and MP3D.** We plot zero-shot success and SPL on HM3DSem (top row) and MP3D (bottom row) validation scenes for agents pretrained on each synthetic scene dataset across number of training steps. Each line summarizes performance for agent checkpoints from three independent training runs evaluated zero-shot on HM3DSem and MP3D. We see that HSSD-pretrained agents perform better throughout compared to both ProcTHOR and iTHOR-pretrained agents.

Pre-training dataset	Success ↑	SPL ↑
HM3DSem	48.10	22.16
ProcTHOR-60	47.85	22.37
HSSD-60	**48.13**	**22.76**
ProcTHOR-122	**48.48**	22.79
HSSD-122	48.23	**23.10**
ProcTHOR-10K	48.32	21.80

Table 1. **Finetuned agent performance.** Performance of HSSD and ProcTHOR pre-trained agents on HM3DSem validation set scenes after finetuning on the HM3DSem training set scenes. All agents converge to comparable performance, though HSSD agents retain a small lead in combined success and efficiency (SPL).

Exploration failure (33%): agent does not explore some part of the house and therefore fails to come across the goal object. A common cause is excessive looping behavior in one part of the house, i.e. repeatedly visiting the same region.

Incorrect prediction (19%): agent stops in front of an object that is not the goal (e.g. stopping in front of a green toy when the goal was a plant).

Inter-floor navigation (14%): agent is spawned on a floor that does not have any goal instances. It needs to change floors to find the object.

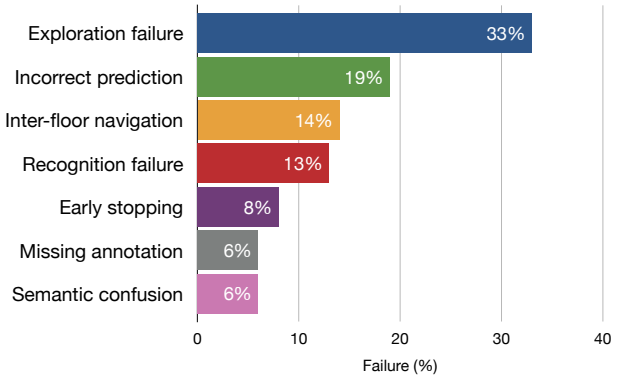

Figure 16. **Failure case analysis.** Breakdown of 100 randomly selected failure cases for agent pre-trained on HSSD, fine-tuned on the HM3DSem [9] train set, and evaluated on the HM3DSem val set. We see that agents are most likely to fail due to inefficacy in exploring the scene (i.e. exploration failures).

Recognition failure (13%): agent sees the object clearly when exploring, but does not navigate to it.

Early stopping (8%): agent finds the object but stops a few centimeters too far from it.

Missing annotation (6%): agent navigates to a valid goal object that unfortunately has not been annotated in the HM3DSem scene, causing the episode to be deemed unsuccessful.

Semantic confusion (6%): agent navigates to an incorrect but semantically similar object category (e.g. navigating to an armchair instead of a sofa).

We plot the corresponding distribution of failure cases in Figure 16. A major cause of failure is inability to effectively explore the scene. Agents are likely to show better performance if annotations are improved and if objects can be found on the same floor.

References

[1] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3164–3174, 2020. 2

[2] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems*, 2022. 1, 2, 5, 6, 7, 8

[3] Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Cao Li, Zengqi Xun, Chengyue Sun, Yiyun Fei, Yu Zheng, Ying Li,

et al. 3D-FRONT: 3D Furnished Rooms with layOuts and semaNTics. *arXiv preprint arXiv:2011.09127*, 2020. 2, 7

[4] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3D-Future: 3D Furniture shape with TextURE. *arXiv preprint arXiv:2009.09633*, 2020. 2

[5] 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 preprint arXiv:1712.05474*, 2017. 1, 2

[6] Ram Ramrakhya, Dhruv Batra, Erik Wijmans, and Abhishek Das. Pirlnav: Pretraining with imitation and rl finetuning for objectnav. *arXiv preprint arXiv:2301.07302*, 2023. 8

[7] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied AI research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019. 1

[8] 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. *Advances in neural information processing systems*, 2021. 2

[9] Karmesh Yadav, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Theo Gervet, John Turner, Aaron Gokaslan, Noah Maestre, Angel Xuan Chang, Dhruv Batra, Manolis Savva, et al. Habitat-Matterport 3D semantics dataset. *arXiv preprint arXiv:2210.05633*, 2022. 1, 5, 6, 7, 8, 9