EmbodiedSplat: Personalized Real-to-Sim-to-Real Navigation with Gaussian Splats from a Mobile Device

Gunjan Chhablani[1]*, Xiaomeng Ye[2], Muhammad Zubair Irshad[3], Zsolt Kira[4]
[1,2,4]Georgia Tech, [3]Toyota Research Institute
{[1]chhablani.gunjan, [3]muhammadzubairirshad}@gmail.com, {[2]xye87, [4]zkira}@gatech.edu

Abstract

The field of Embodied AI predominantly relies on simulation for training and evaluation, often using either fully synthetic environments that lack photorealism or high-fidelity real-world reconstructions captured with expensive hardware. As a result, sim-to-real transfer remains a major challenge. In this paper, we introduce EmbodiedSplat, a novel approach that personalizes policy training by efficiently capturing the deployment environment and fine-tuning policies within the reconstructed scenes. Our method leverages 3D Gaussian Splatting (GS) and the Habitat-Sim simulator to bridge the gap between realistic scene capture and effective training environments. Using iPhone-captured deployment scenes, we reconstruct meshes via GS, enabling training in settings that closely approximate real-world conditions. We conduct a comprehensive analysis of training strategies, pre-training datasets, and mesh reconstruction techniques, evaluating their impact on sim-to-real predictivity in real-world scenarios. Experimental results demonstrate that agents fine-tuned with EmbodiedSplat outperform both zero-shot baselines pre-trained on large-scale real-world datasets (HM3D) and synthetically generated datasets (HSSD), achieving absolute success rate improvements of 20% and 40% on real-world Image Navigation task. Moreover, our approach yields a high sim-vs-real correlation (0.87–0.97) for the reconstructed meshes, underscoring its effectiveness in adapting policies to diverse environments with minimal effort. Project page: https://gchhablani.github.io/embodied-splat.

1. Introduction

Recent advancements in Embodied AI have demonstrated impressive performance in simulated environments [17, 18, 32, 44, 46]. However, translating these capabilities to physical robots remains a significant challenge, primarily due to limitations in simulation fidelity and accessibility [21]. A key bottleneck is the sim-to-real gap, where handcrafted or synthetic simulation environments (e.g. HSSD [25]) struggle to capture the complexity and variability of real-world settings, necessitating the use of real-world reconstructions for effective policy training. On the other hand, real-world datasets such as Matterport3D [8] and HM3D [41] rely on expensive capture equipment and labor-intensive reconstruction pipelines, making large-scale scene collection and adaptation impractical for many applications. Furthermore, it is difficult to fully capture the variability of potential deployment environments with these datasets.

Developments in 3D scene representations, particularly 3D Gaussian Splatting (GS) [24], have shown promise to-

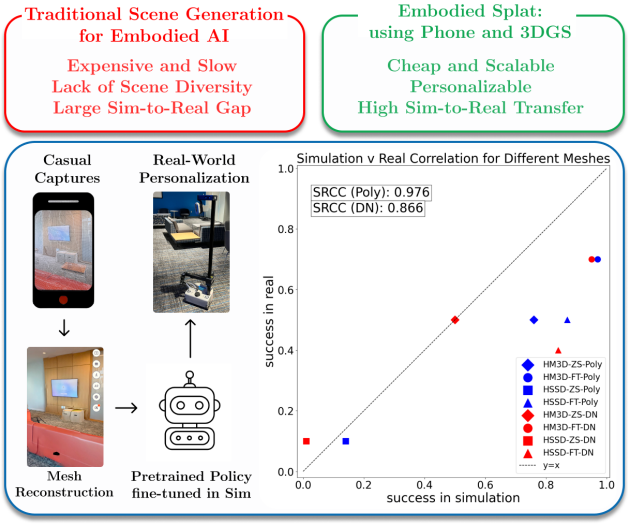


Figure 1. **Overview of EmbodiedSplat:** Mobile phone captures are used to generate reconstructed meshes via 3D Gaussian Splatting (GS). Agents are trained within these reconstructed environments in simulation before being deployed in the real world, enabling effective sim-to-real transfer. Our analysis demonstrates a strong sim-to-real correlation across both types of generated meshes, highlighting their ability to bridge the gap between simulation and real-world performance.

*Work done as a student at Georgia Tech, currently at Waymo.

Work	Goal Type	E2E	Real	S2R
SplatNav [11]	Point/Language	×	✓	×
GaussNav [27]	Image [26]	×	×	×
Ours	Image	✓	✓	✓

Table 1. Comparison against recent works using GS for navigation."E2E" indicates whether the policy is trained end-to-end, "Real" denotes evaluation on real-world robots, and "S2R" indicates demonstrated sim-to-real transfer.

wards reducing the effort needed to capture new scenes, enabling high-quality scene reconstruction from casual mobile phone captures. These approaches offer a strong ability to handle complex geometry, perform novel-view view synthesis, and provide high visual fidelity. Building upon this foundation, methods like DN-Splatter [49] have further enhanced mesh reconstruction quality through depth-and-normal regularization. However, their potential for training robot navigation policies and deploying them in the real-world has remained largely unexplored.

In this work, we explore the following question: *Can low-effort cellphone video captures be leveraged to generate meshes that facilitate the training of Embodied AI navigation policies for effective transfer to the target environment?* In other words, we seek to *personalize* models by training them directly on the target distribution, as represented by 3D models built from low-cost and low-effort data collects of the deployment environment. Some recent works indeed explore GS for navigation - GaussNav [11, 27] explores Instance-Image Navigation [26] in simulation using GS; SplatNav [11] explores using GS-based collision meshes for collision avoidance and localization in a single real-world environment using a modular policy for a drone. In contrast, as shown in Tab. 1, we explore end-to-end policy learning for image-goal navigation and sim-to-real transfer in indoor environments. To the best of our knowledge, we are the first to explore an approach to solve the personalized real-to-sim-to-real problem using Gaussian Splats for indoor image-goal navigation.

To achieve the above goal, we present a comprehensive framework for leveraging open-source 3D Gaussian Splatting [24] (and compare with Polycam [38]). The central premise of our work is that it is possible to quickly capture the scenes in which a robot will be deployed, using readily available consumer-grade hardware, and seamlessly integrate them into simulation environments (i.e. Habitat-Sim [39]). Policies can then be trained in these simulated environments, leading to improved sim-to-real transfer. Our approach combines the accessibility of smartphone-based scene capture with recent advances in depth-aware 3D scene representation, enabling rapid training and deployment of navigation policies in new, realistic environments.

However, there are several challenges and open questions, ranging from reconstruction quality to finetuning strategies, to enabling successful transfer. Building on the work of Silwal et al. [46] as a baseline, we test our framework across a range of scenes captured in a university environment, which is out-of-distribution for typical pre-trained policies. We analyze factors affecting transfer performance, including trade-offs between mesh generation pipelines, mesh quality for policy training, and training strategies (e.g., zero-shot vs. fine-tuning). Through rigorous evaluation and real-world robot experiments, we show that our approach yields significant gains in real-world image-goal navigation.

Our key contributions can be summarized as follows:

1. An **efficient and cost-effective** pipeline for **bridging the real-to-sim gap in navigation**, enabling the creation of high-quality simulation scenes from low-cost, consumer-grade iPhone captures using depth-aware 3D Gaussian Splats (GS) and Polycam.
2. **Comprehensive evaluations** conducted in both simulation and real-world environments, **assessing zero-shot and fine-tuned policies** on our `Captured` scenes. Our results **demonstrate substantial improvements in sim-to-real transfer**, emphasizing the effectiveness of fine-tuning on high-fidelity reconstructions.
3. **In-depth analysis** addressing key research questions, **exploring the relationship between reconstruction quality, pre-training scenes, downstream navigation performance, and training strategies**. For example, a notable finding is that overfitting policies on a single high-fidelity scene reconstruction in simulation yields reasonable real-world performance. Our findings provide **valuable insights into how reconstruction fidelity influences policy generalization** and its applicability to real-world scenarios.
4. An open-source codebase and dataset to facilitate further research in this domain and reproducibility of results.

Through this work, we aim to make high-quality scene collection and agent training more accessible, as well as enable easy development of personalized agents.

2. Related Work

2.1. Scene Datasets

Recent years have seen rapid progress in the development of high-quality 3D scene datasets for embodied AI research. The Matterport3D dataset [8] provides 10,800 panoramic views from 194,400 RGB-D images across 90 building-scale scenes, complete with surface reconstructions and semantic annotations. Building upon this, HM3D [41] expanded the scale to 1,000 building-scale 3D reconstructions from diverse real-world locations, though some issues with mesh quality were noted. The Gibson dataset [53] introduced another collection of real-world scans, while

synthetic datasets like HSSD [25] (with 211 realistic environments) demonstrated that smaller but higher-quality datasets can sometimes be better than larger ones for agent training. Another dataset that is commonly used is ReplicaCAD [47] which provides synthetic variations in layouts of a single scene. In this work, we use HM3D [41] for training our zero-shot baseline, as it is one of the largest photorealistic indoor-scene datasets captured using Matterport [1] cameras. Additionally, we use HSSD [25] as the synthetic counterpart for another zero-shot baseline. Note, however, that these scenes do have a bias towards apartments, and we therefore test our methods on scenes within a university environment, which is out-of-distribution for these datasets. The `MuSHRoom` dataset [42] provides multi-sensor captures of 10 real-world scenes, offering benchmarks for reconstruction and novel view synthesis methods. While this dataset has not been used for embodied tasks in prior works, our dataset collection method is inspired by the `MuSHRoom` dataset's iPhone captures. We evaluate our agents, as well as reconstruction strategies, on these scenes to benchmark our depth and normal encoders.

2.2. Embodied Navigation in Indoor Environments

Embodied navigation encompasses various forms of goal-directed navigation tasks [2, 6, 9, 26, 31, 50, 55, 56]. This study primarily investigates ImageNav [2, 55, 56] in indoor environments. Recent advancements in image-goal navigation have explored a variety of approaches to enhance performance and generalization. Notably, pre-trained visual representations [32, 46, 55, 56] have shown substantial promise in improving performance on the ImageNav task. In particular, Silwal et al. [46] demonstrate remarkable zero-shot success rates (90%) on real-world ImageNav, leveraging VC-1 [32], fine-tuned end-to-end with data augmentation. Inspired by this, we use the setup in Silwal et al. [46] for our policy training and evaluation.

Sim-to-real transfer remains a core challenge in Embodied AI beyond leveraging pre-trained visual encoders. Kadian et al. [21] proposed the Sim-vs-Real Correlation Coefficient (SRCC) to quantify how well PointNav performance in simulation predicts real-world results. We build on this idea to evaluate the sim-to-real predictivity of our `Captured` meshes.

A closely related work is Phone2Proc [13], which demonstrates enhanced sim-to-real ObjectNav performance by generating layouts from iPhone RoomPlan API captures. While similar to our approach in utilizing iPhone captures, our methodology differs in that we capture the entire room without focusing on layouts, nor do we perform any post-processing of scenes or generate multiple variations of the same scene for agent training. Instead, we generate meshes from our captures, fine-tune pre-trained policies for ImageNav task, and deploy them subsequently.

2.3. 3D Scene and Mesh Reconstruction

In this work, we adopt DN-Splatter [49] for its simplicity and superior performance (and compare to meshes from Polycam [38], which is not open-source). DN-Splatter uses depth-and-normal regularization to improve mesh quality. While mesh reconstruction has also been explored with NeRFs [35], we choose GS for 3D reconstruction due to their fast training and rendering speeds, improving the overall efficiency of the pipeline. For an expanded list of related works on this topic, please see Appendix D.1.

2.4. 3D Representations for Embodied AI/Robotics

There have been works which explore sim-to-real manipulation [7, 28, 52]. In this work, we attempt to solve the problem of sim-to-real navigation, laying the foundation for a diverse range of embodied tasks, such as object-goal navigation, rearrangement, and mobile manipulation, which require room-, apartment-, and building-level scene representations. Capturing high-quality 3D Gaussian Splats (GS) is relatively straightforward for tabletop manipulation tasks, whereas large-scale scene reconstruction poses greater challenges — a key aspect of this work. Recent works also explore the use of GS in solving the embodied navigation problem. GaussNav [27] presents a semantic Gaussian-based map reconstruction for the HM3D-Instance ImageNav [26] task. Splat-Nav [11] introduces a two-stage pipeline for planning and pose estimation through Gaussian Splat Maps. Tab. 1 highlights the differences between our approach and recent works utilizing GS for navigation. By capturing the entire scene using mobile phones, we enable personalization to specific scenes, as well as the potential to scale scene collection in the future. Some works [33] explore automated 3D scene capture using robot cameras in simulation, but these methods require complex, embodiment-dependent strategies that are time-consuming and difficult to scale. In contrast, our approach leverages human captures in the real-world, offering a fast and scalable solution without the need for intricate planning. For more related work on this topic, please see Appendix D.2.

3. Methodology

The overall pipeline for bridging and integrating a real-world scene with Habitat-Sim [39] is shown in Fig. 2. In the following subsections, we discuss each stage of the pipeline in detail. Sec. 3.1 discusses the datasets used and the details of scene captures. Sec. 3.2 discusses the second stage, where these captures are converted to meshes. Sec. 3.3 discusses how ImageNav episodes are created, which is a crucial part of integrating the scene with Habitat Simulator [39]. Appendix H discusses agent training and evaluation details. For details on real-world deployment setup, refer to Appendix A.

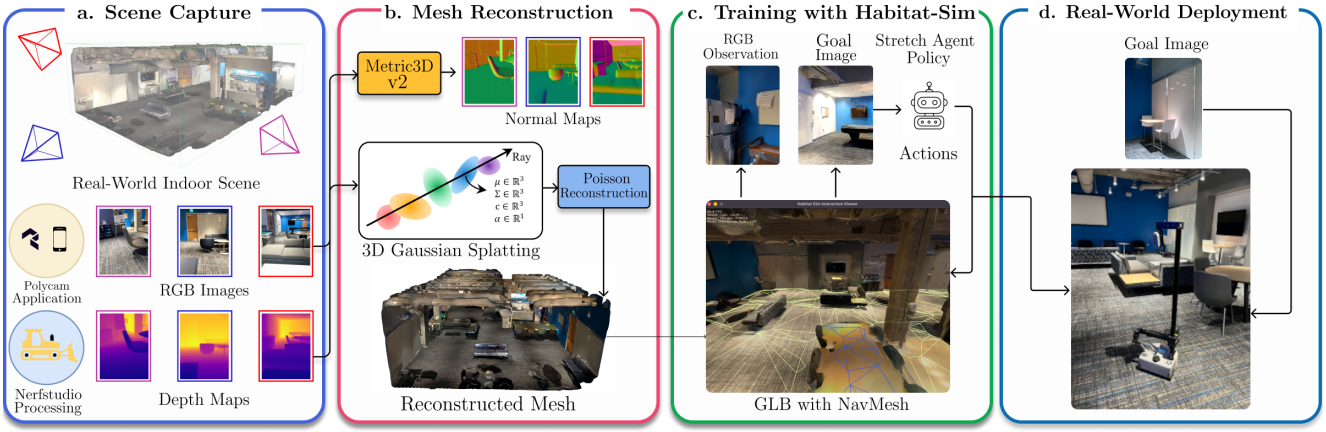


Figure 2. **The EmbodiedSplat Pipeline**: Pipeline for integration of real-world captures with Habitat-Sim [39] and subsequent deployment. The first stage (a.) involves capturing the scene using Polycam [38] and Nerfstudio [48] which produces RGB frames, associated iPhone GT depth maps, and poses. In the second stage (b.), we use DN-Splatter [49] to train GS using depth and normal regularization, with normals from Metric3D-V2 [16] monocular encoder. A mesh (`.ply`) is created using Poisson reconstruction from the GS. In the third stage (c.), the mesh is processed and loaded into Habitat-Sim [39] for training the agent in simulation. In the last stage (d.), the policy is deployed in the real-world in the same scene for image-goal navigation.

3.1. Datasets and Scene Captures

Scene Datasets: We use two datasets for pre-training ImageNav policies, which serve as zero-shot baselines and pre-trained policies for our scenes, both in simulation and real-world evaluations. Specifically, we use the HM3D [41] dataset which consists of apartment-scale scenes split into 800 training and 100 validation scenes, and the HSSD [25] dataset split into 134 training and 33 validation scenes.

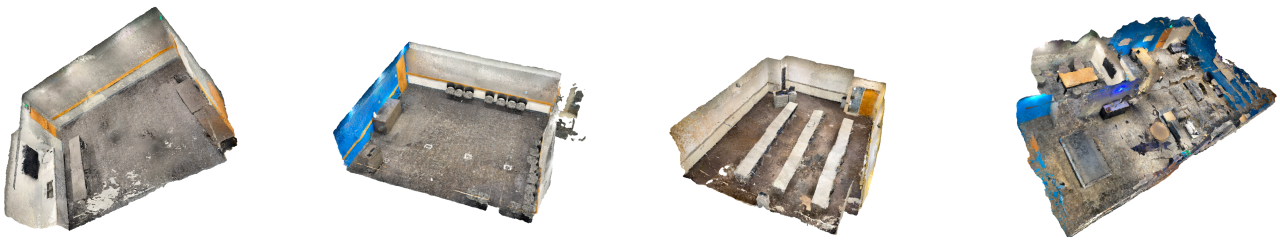
`Captured` Scenes: To evaluate the feasibility of the pipeline and conduct real-world experiments, we capture scenes from a university environment (classroom, community lounges, conference rooms, etc). For custom data collection with an iPhone, we follow the procedure used for collecting the `MuSHRoom` dataset [42]. Specifically, we use an iPhone 13 Pro Max to record the iPhone RGB-D data using the Polycam application [38]. Polycam provides an assistive interface to ensure that all the details of the scene have been sufficiently covered during the capture. We use the default settings with Polycam and export the raw data exposed by the application. Subsequently, we use Nerfstudio [48] to process the RGB-D data and sample 1000 aligned RGB-depth frames with low blur scores and corresponding poses. Additionally, Polycam also provides a mesh with its exported data. We use this mesh for comparison purposes, referred to as POLYCAM meshes throughout the paper. Unlike Ren et al. [42], we use a manually-held iPhone for capture instead of a gimbal, enabling us to evaluate whether the process remains effective and easily replicable without relying on expensive capture equipment. Each capture requires 20-30 minutes of recording with Polycam. We repeat this process for different indoor scenes, which we refer to as `lounge`, `classroom`, `conf_a`, and `conf_b`. The scale of these of these scenes is similar to those in the `MuSHRoom` dataset (1-3 rooms) [42].

`MuSHRoom` Dataset: We use `MuSHRoom` [42] dataset for benchmarking. The dataset consists of long and short sequences of 10 indoor scenes, captured with iPhone, Kinect, and Faro Scanner. In our work, we only use the iPhone long-sequences for training and evaluation purposes. We benchmark different normal encoders on all 10 scenes (see Appendix G) for DN-Splatter [49]. For agent training and evaluation, we primarily use three scenes - `honka`, `sauna`, `activity` - which have varied scale and complexity.

3.2. 3D Scene and Mesh Reconstruction

For mesh reconstruction, we use DN-Splatter [49] as our method of choice for its superior performance on mesh reconstruction in comparison to others, and simplicity of its integration with Habitat-Sim [39]. DN-Splatter leverages depth-normal regularization, and smoothness losses to maintain geometric consistency during Gaussian Splat training. For mesh reconstruction, the rendered depth and normal maps are back-projected from training views to create a point-set for meshing.

We use the default hyperparameters from DN-Splatter [49] and train the GS for 30,000 iterations. We use sensor depth $\lambda_d = 0.2$, and enable the depth smoothness and normal losses. Metric3D-V2 [16] is used as the normal encoder instead of Omnidata [14, 23], as empirical results demonstrate it yields higher-quality meshes. For further discussion on depth and normal encoder selection, see Appendix G.



(a) conf_b (b) conf_a (c) classroom (d) lounge

Figure 3. Reconstructed meshes of the Captured scenes after DN-Splatter [49] training and Poisson reconstruction.

After training GS and converting into ply meshes, we convert the meshes to glb in Blender (see Appendix E) before loading them in Habitat-Sim [39]. The final meshes for our Captured scenes are shown in Fig. 3 and referred to as DN mesh hereafter to contrast against the POLYCAM mesh counterpart. It takes approximately 20-30 minutes per capture, and 1-2 hours of training with DN-Splatter [49] to generate these meshes, which is significantly lesser compared to the cost and several hours of capture and processing with Matterport [1] cameras.

3.3. ImageNav Episode Generation

Pre-training Scene Datasets: For the HM3D [41] and HSSD [25] datasets, we use the predefined train-validation split, generating 10,000 episodes for each training scene and 25 episodes for each validation scene (see Appendix B).

Captured and MuSHRoom Scenes: For scenes with DN meshes and POLYCAM meshes, we select only the largest navmesh island within each scene. We ensure that the largest island covers most, if not all, of the navigable areas in the scene. This step is crucial due to the relatively smaller sizes of these scenes compared to HM3D scenes. Hence, we generate only 1000 training episodes and 100 evaluation episodes per scene. By selecting the largest navmesh island, we minimize the risk of incorrectly treating entities other than the floor as navigable.

Evaluation Metric: We consider Success Rate (SR), the fraction of successful episodes out of all the episodes, as the primary metric for performance evaluation of our policies. An episode is considered successful if the agent stops within 1m of the goal location before the maximum number of steps (1000 for simulation, 100 for real) are over.

4. Experiments

This section aims to answer the following research questions:

1. How does a pre-trained policy perform on Captured and MuSHRoom scenes? (Sec. 4.1)
2. Does fine-tuning on the Captured and MuSHRoom scenes improve performance? (Sec. 4.2).
3. Does the performance in simulation transfer to the real-world? (Sec. 4.3).

Further experiments and analyses are discussed in Sec. 5.

4.1. How does a pre-trained policy perform on Captured and MuSHRoom scenes?

In this experiment, we evaluate the pre-trained policies that achieved the highest validation success rates on the HM3D [41] (83.08% val SR) and HSSD [25] (63.15% val SR), trained for 600M and 1200M steps, respectively. Note that the performance on HM3D is consistent with the simulation results reported in Silwal et al. [46] and represents the current state-of-the-art for this task. For agent embodiment, policy training and evaluation details, please refer to Appendix H. We assess the zero-shot generalization of these policies within simulation, evaluating their performance across different meshes for individual scenes.

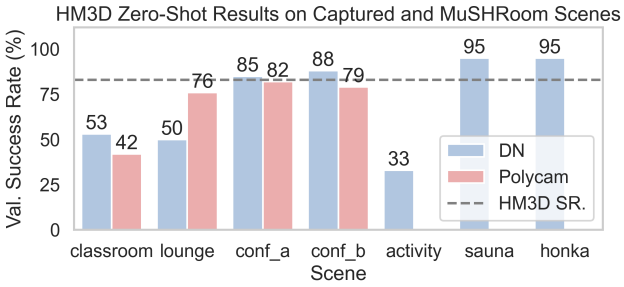


Figure 4. **Zero-shot val. SR for HM3D pre-trained policy**.

Fig. 4 presents the results of the zero-shot evaluation across individual scenes for the HM3D pre-trained policy. Performance varies significantly based on scene size, complexity, and mesh type. For smaller scenes such as conf_a and conf_b, the policy demonstrates relatively high success rates in both DN mesh and POLYCAM settings, achieving 85% (DN) and 82% (POLYCAM) for conf_a and 88% (DN mesh) and 79% (POLYCAM) for conf_b.

In contrast, performance declines in larger environments such as classroom and lounge. In classroom, success rates drop to 53% with DN mesh and 42% with POLYCAM, while in lounge, the policy achieves 50% (DN

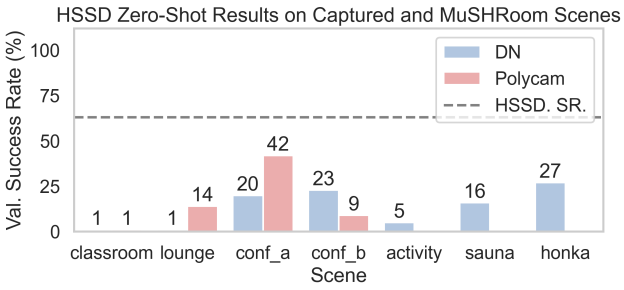


Figure 5. **Zero-shot val. SR for HSSD pre-trained policy**.

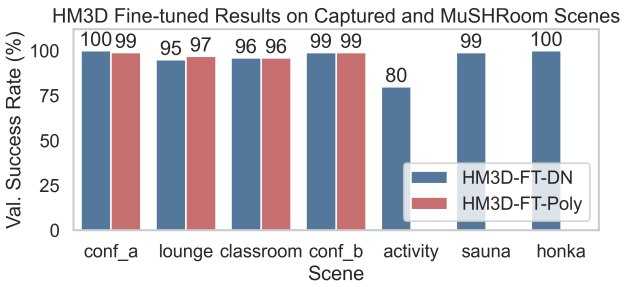


Figure 6. **Fine-tuned validation SR from HM3D pre-trained policy**: The policy is fine-tuned and evaluated on the same mesh.

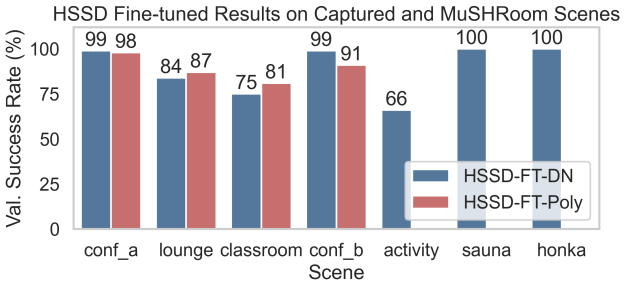


Figure 7. **Fine-tuned validation SR from HSSD pre-trained policy**: The policy is fine-tuned and evaluated on the same mesh.

mesh) and 76% (POLYCAM). These results highlight the challenge of transferring the policy to more complex, large-scale environments that differ from the training data. Since HM3D consists primarily of apartment-style scenes, the policy has not been exposed to classrooms or community lounges during training. Both DN and POLYCAM meshes yield similar success rates on average($\sim$ 60%), suggesting no clear advantage of one mesh type over the other in aligning with HM3D. Additionally, we evaluate the policies on three scenes from the `MuSHRoom` [42] dataset. Due to the unavailability of POLYCAM meshes for `MuSHRoom`, we conduct evaluations solely on DN meshes. The policy achieves a 95% success rate in the `sauna` and `honka` scenes, suggesting promising potential for performance improvements through additional scene refinements (such as using a gimbal during capture). However, in the larger `activity` scene, performance declines, indicating challenges in generalizing to larger environments.

Fig. 5 reveals similar trends for the HSSD pre-trained policy, though with significantly lower success rates overall. This degradation in performance can be attributed to the synthetic nature of HSSD scenes, which lack the realism and scale necessary for effective generalization. The policy performs particularly poorly on `classroom`, achieving only 1% success on both DN mesh and POLYCAM, while performance in `lounge` is slightly better, reaching 14% success with POLYCAM. The policy exhibits slightly improved performance on other scenes, but it is hard to conclude which mesh type best aligns with HSSD.

4.2. Does fine-tuning on the `Captured` and `MuSHRoom` scenes improve performance?

In this section, we investigate whether fine-tuning the pre-trained policies for only 20M additional steps improves performance in simulation. The learning rate is set to $2.5e{-}6$ for the LSTM policy and $6e{-}7$ for the visual encoder, following a fine-tuning strategy similar to that of Deitke et al. [13]. We fine-tune the pre-trained policy on the training episodes for a single scene and evaluate on the corresponding validation episodes.

The results are presented in Fig. 6 and Fig. 7. We observe that fine-tuning significantly improves performance across all tested scenes. For the pre-trained HM3D policy, fine-tuning for 20M steps results in success rates approaching 90%+ across different meshes. Similarly, for the HSSD pre-trained policy, fine-tuning leads to substantial improvements, with most policies achieving success rates of 80%+ in respective scenes. In particular, performance gains are particularly pronounced in larger, more complex environments such as `classroom` and `lounge`, which differ significantly from apartment-style scenes in HM3D and HSSD. These results indicate that our pipeline can effectively be used to collect diverse scene data at scale. Furthermore, they suggest that agents can be quickly fine-tuned (only 20M steps vs 100M+ steps for training from scratch) on these generated meshes to improve performance on specific scene types, enhancing personalization beyond the original training distribution. We present results on additional scenes in Appendix J.

4.3. Does the performance in simulation transfer to the real-world?

We evaluate both zero-shot and fine-tuned policies in the real-world `lounge` scene on a Stretch robot. During the evaluation, each episode is capped at 100 steps, with 10 distinct start-and-goal locations sampled within the scene (see Appendix A for deployment details). To assess performance, we record the number of steps taken and final

distance to the goal at the end of each episode, determining whether the agent successfully completes the task.

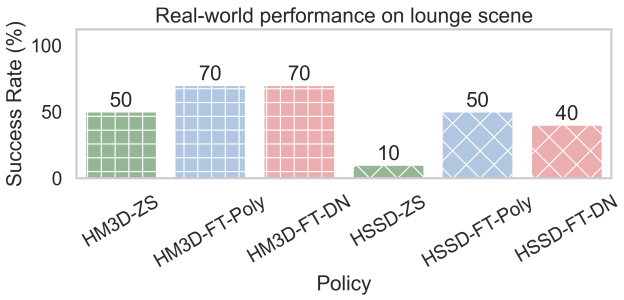


Figure 8. **Real world results of zero-shot and fine-tuned models on `lounge` scene**. HM3D-ZS (real dataset) leads to a better success rate than HSSD-ZS (synthetic dataset). We observe increased real-world success with fine-tuning on DN and POLYCAM meshes for both HM3D and HSSD policies.

Fig. 8 presents results across 10 evaluation episodes. The zero-shot HM3D policy achieves a 50% success rate, demonstrating our hypothesized lack of generalization. This is in contrast with the results reported in Silwal et al. [46] showing 90% zero-shot success rate in the real-world. We attribute this discrepancy to the structural and semantic differences between the `lounge` and the apartment-style scenes typically encountered in HM3D. Fine-tuning on the POLYCAM and DN mesh reconstructions of this scene improves performance, with success rates increasing to 70%. For HSSD, zero-shot performance is significantly lower at 10%, while fine-tuned policies improve success rates to 50% with POLYCAM and 40% with DN mesh.

Fig. 1 illustrates the Sim-to-Real Correlation Coefficient (SRCC) [21] between simulation and real-world performance. The observation suggests that improvements in evaluation performance on DN and POLYCAM meshes in simulation translate to improved real-world performance. This demonstrates that our approach can efficiently adapt policies to novel real-world environments. We discuss real-world statistics (number of steps, distance) in Appendix I.

5. Ablations and Analysis

In this section, we systematically investigate critical factors influencing policy performance. First, we assess whether pre-training on large-scale datasets such as HM3D or HSSD is necessary for effective real-world transfer (Sec. 5.1). Next, we analyze the impact of validation PSNR and scene geometry on the zero-shot performance (Sec. 5.2). Finally, we examine whether continuous training on large-scale datasets improves zero-shot performance on our `Captured` scenes, shedding light on the relationship between dataset characteristics and generalizability (Sec. 5.3).

5.1. Is it necessary to pre-train over large-scale datasets?

In this experiment, we investigate the necessity of pre-training on large-scale datasets by "overfitting" policies directly on POLYCAM and DN meshes with a policy trained from scratch (not pre-trained on large-scale datasets such as HM3D or HSSD) for ∼100M steps. Training and validation data are derived from ImageNav episode generation, as described in Sec. 3.3. To ensure diversity, start-goal locations for training and evaluation episodes are randomly sampled. The best validation success rates for all policies on their respective scenes are presented in Fig. 9. As expected, the overfitted policies achieve near-perfect performance in simulation, maintaining high success rates. For a detailed discussion on why overfitting results seem better than fine-tuning in this scenario, please see Appendix L.

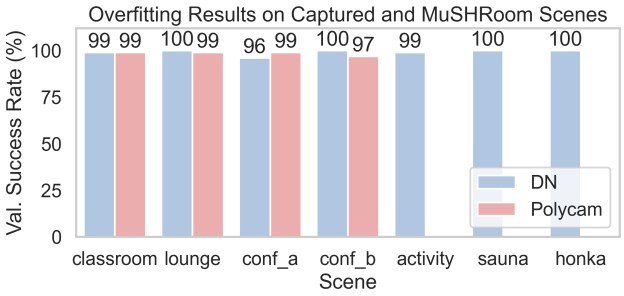


Figure 9. **Validation SR for overfitted policies on respective scenes**: Polices are not pre-trained on HM3D/HSSD.

We further evaluate these overfitted policies in the real-world `lounge` scene using POLYCAM and DN meshes. Surprisingly, the overfitted policy trained on the POLYCAM mesh achieves a 50% success rate in real-world evaluations. In contrast, the policy trained on the DN mesh achieves only 10% success. This result suggests that our mesh-based training approach can yield non-zero real-world performance, even without large-scale pre-training. We attribute the performance gap between POLYCAM and DN-trained policies to differences in visual fidelity—POLYCAM meshes preserve more visual detail by directly utilizing original images to reconstruct the scene, whereas DN meshes are based on GS which use the learned colors for the 3D Gaussians.

5.2. How is the scale and validation PSNR of the scene related to the final performance?

Fig. 10 illustrates the correlation between HM3D zero-shot validation success rates on DN meshes and two key factors: the Peak Signal-to-Noise Ratio (PSNR) of the corresponding 3D Gaussian Splats (GS) and the scale of the scene, measured as the average shortest distance between start-goal locations in validation episodes. We observe a negative correlation between success rate (SR) and average

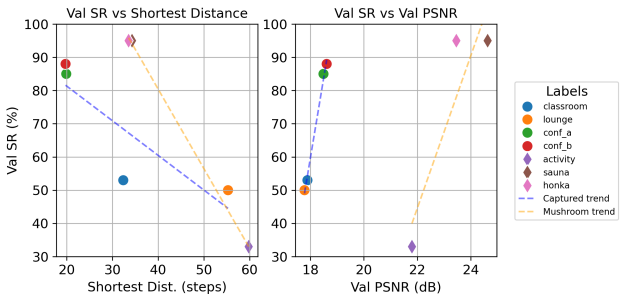


Figure 10. **HM3D zero-shot validation success rates on DN meshes vs. validation GS PSNRs vs. average shortest distances of validation episodes**. The zero-shot SR is inversely correlated with the scale of the scene, while directly correlated with val. PSNR on the trained GS.

shortest distance—indicating that as the scale of the scene increases, the zero-shot success rate declines. Conversely, a positive correlation is seen between SR and PSNR, where higher validation PSNR values correspond to improved success rates. Notably, different trend lines emerge for `MuSHRoom` captures and our own `Captured` scenes. `MuSHRoom` captures generally exhibit higher validation PSNRs, likely due to the use of a stabilized gimbal during data collection. Future work will further investigate the impact of capture stability on policy performance.

5.3. Does continuous training on large-scale datasets improve zero-shot performance?

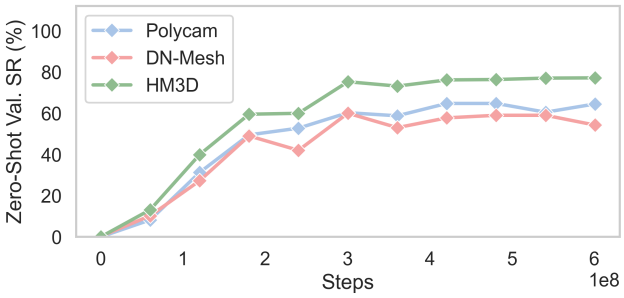


Figure 11. **Average zero-shot success rates on respective validation sets** over different stages of pre-training on HM3D.

Fig. 11 presents average zero-shot success rates across HM3D validation set, POLYCAM meshes, and DN meshes at various stages of HM3D policy pre-training. This analysis aims to determine whether continuous performance improvements on HM3D validation scenes translate to improved zero-shot performance on our `Captured` scenes. We observe that while performance initially increases, it begins to deteriorate or plateau at approximately 400M steps, despite continued improvements on HM3D validation scenes.

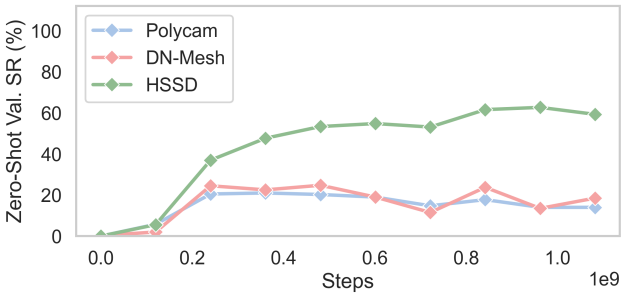


Figure 12. **Average zero-shot success rates on respective validation sets** over different stages of pre-training on HSSD.

Fig. 12 shows a similar experiment conducted using the HSSD dataset. Up to 300M steps, improvements in HSSD validation performance correspond to slight improvements in zero-shot performance on our `Captured` scenes. However, performance plateaus for `Captured`, showing no further gains despite continued improvement on HSSD validation scenes. For HM3D pre-training, POLYCAM meshes outperform DN meshes in success rates, while for HSSD, the trend reverses slightly, highlighting the impact of pre-training dataset characteristics on zero-shot generalization.

6. Conclusion

In this work, we introduced a comprehensive pipeline for bridging the gap between real-world and simulated environments in training embodied agents using 3D Gaussian Splats (GS) and Polycam. By leveraging the `MuSHRoom` dataset and custom iPhone-captured scenes, we demonstrated an efficient and scalable approach to policy personalization, leveraging 3D scene reconstruction from low-effort collects, enabling high-quality training for the ImageNav task. Our pipeline facilitates accessible and replicable scene collection without requiring specialized hardware or significant costs, making it a practical solution for large-scale embodied AI research. We evaluated overfitted, zero-shot, and fine-tuned policies, showing that fine-tuning pre-trained policies on real-world scene reconstructions improves sim-to-real transfer. We also analyzed the differences between GS-generated DN meshes and POLYCAM meshes, finding that POLYCAM more closely resemble real-world scenes.

This work lays the foundation for seamlessly integrating real-world scene captures into simulation, expanding the applicability of embodied AI systems. Another promising avenue is integrating GS directly into training, replacing visual observations and goal images to enhance learning efficiency. Furthermore, we aim to extend the use of Gaussian Splats to more complex embodied AI tasks, such as rearrangement and mobile manipulation, broadening its impact across diverse real-world applications.

References

[1] Capture, share, and collaborate the built world in immersive 3D — matterport.com. https://matterport.com/. [Accessed 08-03-2025]. 3, 5

[2] Ziad Al-Halah, Santhosh K. Ramakrishnan, and Kristen Grauman. Zero experience required: Plug & play modular transfer learning for semantic visual navigation, 2022. 3

[3] Gwangbin Bae and Andrew J. Davison. Rethinking inductive biases for surface normal estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3, 4

[4] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields, 2021. 2

[5] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth, 2023. 4

[6] Guillaume Bono, Leonid Antsfeld, Boris Chidlovskii, Philippe Weinzaepfel, and Christian Wolf. End-to-end (instance)-image goal navigation through correspondence as an emergent phenomenon, 2023. 3

[7] Arunkumar Byravan, Jan Humplik, Leonard Hasenclever, Arthur Brussee, Francesco Nori, Tuomas Haarnoja, Ben Moran, Steven Bohez, Fereshteh Sadeghi, Bojan Vujatovic, and Nicolas Heess. Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields, 2022. 3, 2

[8] 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, 2017. 1, 2

[9] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Object goal navigation using goal-oriented semantic exploration, 2020. 3

[10] Lawrence Yunliang Chen, Chenfeng Xu, Karthik Dharmarajan, Muhammad Zubair Irshad, Richard Cheng, Kurt Keutzer, Masayoshi Tomizuka, Quan Vuong, and Ken Goldberg. Rovi-aug: Robot and viewpoint augmentation for cross-embodiment robot learning. In *Conference on Robot Learning (CoRL)*, Munich, Germany, 2024. 2

[11] Timothy Chen, Ola Shorinwa, Joseph Bruno, Javier Yu, Weijia Zeng, Keiko Nagami, Philip Dames, and Mac Schwager. Splat-nav: Safe real-time robot navigation in gaussian splatting maps, 2024. 2, 3

[12] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels, 2024. 2

[13] Matt Deitke, Rose Hendrix, Luca Weihs, Ali Farhadi, Kiana Ehsani, and Aniruddha Kembhavi. Phone2proc: Bringing robust robots into our chaotic world, 2022. 3, 6

[14] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10786–10796, 2021. 4, 3

[15] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering, 2023. 2

[16] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. 2024. 4, 3

[17] Muhammad Zubair Irshad, Chih-Yao Ma, and Zsolt Kira. Hierarchical cross-modal agent for robotics vision-and-language navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021. 1

[18] Muhammad Zubair Irshad, Niluthpol Chowdhury Mithun, Zachary Seymour, Han-Pang Chiu, Supun Samarasekera, and Rakesh Kumar. Semantically-aware spatio-temporal reasoning agent for vision-and-language navigation in continuous environments. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 4065–4071, 2022. 1

[19] Muhammad Zubair Irshad, Sergey Zakharov, Katherine Liu, Vitor Guizilini, Thomas Kollar, Adrien Gaidon, Zsolt Kira, and Rares Ambrus. Neo 360: Neural fields for sparse view synthesis of outdoor scenes. 2023. 2

[20] Muhammad Zubair Irshad, Mauro Comi, Yen-Chen Lin, Nick Heppert, Abhinav Valada, Rares Ambrus, Zsolt Kira, and Jonathan Tremblay. Neural fields in robotics: A survey. *arXiv preprint arXiv:2410.20220*, 2024. 2

[21] 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? *IEEE Robotics and Automation Letters*, 5(4):6670–6677, 2020. 1, 3, 7

[22] Ivan Kapelyukh, Yifei Ren, Ignacio Alzugaray, and Edward Johns. Dream2real: Zero-shot 3d object rearrangement with vision-language models, 2024. 2

[23] Oğuzhan Fatih Kar, Teresa Yeo, Andrei Atanov, and Amir Zamir. 3d common corruptions and data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18963–18974, 2022. 4

[24] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 2

[25] Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Schacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X Chang, and Manolis Savva. Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation. *arXiv preprint arXiv:2306.11290*, 2023. 1, 3, 4, 5

[26] Jacob Krantz, Stefan Lee, Jitendra Malik, Dhruv Batra, and Devendra Singh Chaplot. Instance-specific image goal navigation: Training embodied agents to find object instances, 2022. 2, 3

[27] Xiaohan Lei, Min Wang, Wengang Zhou, and Houqiang Li. Gaussnav: Gaussian splatting for visual navigation, 2024. 2, 3

[28] Yulong Li and Deepak Pathak. Object-aware gaussian splatting for robotic manipulation. In *ICRA 2024 Workshop on 3D Visual Representations for Robot Manipulation*, 2024. 3, 2

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 3

[30] JunEn Low, Maximilian Adang, Javier Yu, Keiko Nagami, and Mac Schwager. Sous vide: Cooking visual drone navigation policies in a gaussian splatting vacuum, 2025. 2

[31] Arjun Majumdar, Gunjan Aggarwal, Bhavika Devnani, Judy Hoffman, and Dhruv Batra. Zson: Zero-shot object-goal navigation using multimodal goal embeddings, 2023. 3

[32] Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Yecheng Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Pieter Abbeel, Jitendra Malik, Dhruv Batra, Yixin Lin, Oleksandr Maksymets, Aravind Rajeswaran, and Franziska Meier. Where are we in the search for an artificial visual cortex for embodied intelligence?, 2024. 1, 3

[33] Pierre Marza, Laetitia Matignon, Olivier Simonin, Dhruv Batra, Christian Wolf, and Devendra Singh Chaplot. Autonerf: Training implicit scene representations with autonomous agents, 2023. 3, 2

[34] Wugang Meng, Tianfu Wu, Huan Yin, and Fumin Zhang. Beings: Bayesian embodied image-goal navigation with gaussian splatting, 2024. 2

[35] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. 3, 2

[36] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 2

[37] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. UniDepth: Universal monocular metric depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 4

[38] Polycam. Polycam. Accessed: 2024-11-10. 2, 3, 4

[39] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander Clegg, Michal Hlavac, So Yeon Min, et al. Habitat 3.0: A co-habitat for humans, avatars, and robots. In *The Twelfth International Conference on Learning Representations*, 2023. 2, 3, 4, 5, 1

[40] Mohammad Nomaan Qureshi, Sparsh Garg, Francisco Yandun, David Held, George Kantor, and Abhisesh Silwal. Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting, 2024. 2

[41] Santhosh K. Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X. Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai, 2021. 1, 2, 3, 4, 5

[42] Xuqian Ren, Wenjia Wang, Dingding Cai, Tuuli Tuominen, Juho Kannala, and Esa Rahtu. Mushroom: Multi-sensor hybrid room dataset for joint 3d reconstruction and novel view synthesis, 2024. 3, 4, 6, 2

[43] Meta AI Research. Home robot. `https://github.com/facebookresearch/home-robot`, 2023. GitHub repository. 1

[44] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research, 2019. 1

[45] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. 3

[46] Sneha Silwal, Karmesh Yadav, Tingfan Wu, Jay Vakil, Arjun Majumdar, Sergio Arnaud, Claire Chen, Vincent-Pierre Berges, Dhruv Batra, Aravind Rajeswaran, Mrinal Kalakrishnan, Franziska Meier, and Oleksandr Maksymets. What do we learn from a large-scale study of pre-trained visual representations in sim and real environments?, 2024. 1, 2, 3, 5, 7, 4

[47] 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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3

[48] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 4

[49] Matias Turkulainen, Xuqian Ren, Iaroslav Melekhov, Otto Seiskari, Esa Rahtu, and Juho Kannala. Dn-splatter: Depth and normal priors for gaussian splatting and meshing, 2024. 2, 3, 4, 5

[50] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *International Conference on Learning Representations*, 2019. 3

[51] Yaniv Wolf, Amit Bracha, and Ron Kimmel. Gs2mesh: Surface reconstruction from gaussian splatting via novel stereo views, 2024. 2

[52] Yuxuan Wu, Lei Pan, Wenhua Wu, Guangming Wang, Yanzi Miao, and Hesheng Wang. Rl-gsbridge: 3d gaussian splatting based real2sim2real method for robotic manipulation learning, 2024. 3, 2

[53] Fei Xia, Amir Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents, 2018. 2

[54] Hongchi Xia, Zhi-Hao Lin, Wei-Chiu Ma, and Shenlong Wang. Video2game: Real-time, interactive, realistic and

browser-compatible environment from a single video, 2024. 2

[55] Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline visual representation learning for embodied navigation, 2022. 3

[56] Karmesh Yadav, Arjun Majumdar, Ram Ramrakhya, Naoki Yokoyama, Alexei Baevski, Zsolt Kira, Oleksandr Maksymets, and Dhruv Batra. Ovrl-v2: A simple state-of-art baseline for imagenav and objectnav, 2023. 3, 4

[57] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2, 2024. 4

[58] Justin Yu, Kush Hari, Kishore Srinivas, Karim El-Refai, Adam Rashid, Chung Min Kim, Justin Kerr, Richard Cheng, Muhammad Zubair Irshad, Ashwin Balakrishna, et al. Language-embedded gaussian splats (legs): Incrementally building room-scale representations with a mobile robot. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13326–13332. IEEE, 2024. 2

[59] Justin Yu, Kush Hari, Karim El-Refai, Arnav Dalil, Justin Kerr, Chung-Min Kim, Richard Cheng, Muhammad Z. Irshad, and Ken Goldberg. Persistent object gaussian splat (pogs) for tracking human and robot manipulation of irregularly shaped objects. *ICRA*, 2025. 2

[60] Siting Zhu, Guangming Wang, Dezhi Kong, and Hesheng Wang. 3d gaussian splatting in robotics: A survey, 2024. 2

EmbodiedSplat: Personalized Real-to-Sim-to-Real Navigation with Gaussian Splats from a Mobile Device

Supplementary Material

A. Real-world Deployment on Stretch Robot

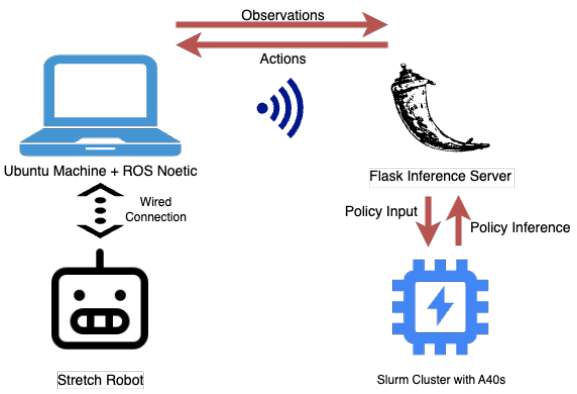


Figure 13. Framework for real-world evaluation on a Stretch robot.

Since the stretch robot is not equipped with GPUs, we develop a framework for deploying a policy on a remote cluster, and performing inference using POST requests from the robot. An overview of this arrangement is shown in Fig. 13. Specifically, we create a Flask server that runs on a remote compute node, port-forwarded to an Ubuntu 20.04 machine with ROS noetic installed. The same machine is connected to a Stretch robot via ethernet connection. We use the `home-robot` [43] repository as framework to convert discretized actions from the policy into continuous space using the "StretchNavigationClient". A script running on the machine consumes the ROS topics for image observations from the robot, and passes this along with the goal image to the policy and receives the next action based on these observations.

For evaluation in real-world, we use random start-and-goal locations, place the Stretch robot at the goal location, and capture a goal image. Then, we mark these locations using colored masking tape on the floor. The policy is deployed with the goal image passed to the agent during the episode as above, keeping the start location and pose similar for each evaluation. The episode stops when the agent outputs a STOP action or the maximum number of steps (100 in our case) are reached. Finally, the distance of the base is measured from the the goal location using a measuring tape, and the episode is marked successful if the agent reaches within $1m$ of the goal location before stopping. The goal images used are shown in Fig. 15.

Figure 14. Hello Robot Stretch in `lounge` scene

Figure 15. Goal images used in `lounge` scene

B. Generating ImageNav episodes

To generate ImageNav episodes, we build upon Habitat's [39, 44] scripts for PointNav generation. During training, the PointNav goal locations are paired with an Image-goal sensor to retrieve the image corresponding to the goal location prior to the start of the episode.

Habitat [39, 44] uses navmesh islands to define navigable areas in simulation. For episode generation, we leverage the stretch robot parameters [46] to compute the navmeshes.

Valid start and goal locations are sampled during episode generation, following the approach in [46]. The validity checks ensure that the goal is reachable, the distance from the start location to the goal is non-zero, and the navmesh island radius exceeds $2m$, thereby avoiding navigation on objects like beds or tables.

C. Examples for DN and POLYCAM rendering

Fig. 16 and Fig. 17 shows the rendering differences for the images in Habitat-Sim [39] for DN and POLYCAM meshes. DN meshes have diffused and darker colors, and some holes for regions where capture was not sufficient. In contrast, polycam produces a smoother and more realistic mesh.

Figure 16. DN (`conf_b`)

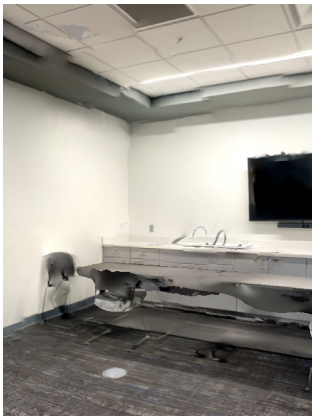


Figure 17. POLYCAM (`conf_b`)

D. Expanded Related Work

D.1. 3D Scene and Mesh Reconstruction

Recent advancements in 3D scene reconstruction have led to the development of several notable approaches. Neural Radiance Fields (NeRF) [35] and subsequent methods [4, 19, 36] have focused on training and improving neural scene representation techniques. The foundational 3D Gaussian Splatting (GS) method [24] has inspired a variety of extensions. DN-Splatter [49] improves reconstruction quality by incorporating depth and normal regularization with monocular networks. Gaussian Surfels [12] introduces a technique for flattening 3D Gaussians into 2D surfels, resulting in enhanced mesh reconstruction. GS2Mesh [51] incorporates real-world knowledge through stereo-matching to generate smoother meshes. SuGaR [15] proposes a fast mesh extraction method using surface-aligned Gaussian splatting.

D.2. 3D Scene Representation in Robotics

Several studies have explored the use of neural scene representations [33, 54] and 3D Gaussian Splatting (GS) for robotic tasks in simulation. SplatGym [40] presents a neural simulator for training robotic control policies using Gaussian Splatting. BEINGS [34] leverages 3D Gaussian Splatting as a scene prior to predict future observations and refine navigation strategies. A concurrent work, SOUS-VIDE [30], uses a simple drone dynamics model within a high visual fidelity 3DGS reconstruction for training an end-to-end policy and show sim-to-real transfer. We note that our work is a complementary effort focused on image-goal navigation, out-of-domain generalization, the effects of reconstruction quality and training strategies.

Recent research has also emphasized the application of neural scene representations for real-world robotic applications [7, 10, 22]. Object-Aware Gaussian Splatting [28] proposes semantic and dynamic 3D representations for robotic manipulation. RL-GSBridge [52] introduces a mesh-based 3D Gaussian Splatting method for zero-shot sim-to-real transfer in manipulation tasks. LEGS [58] and POGS [59] distil semantic features from foundation models into GS to enable downstream navigation and manipulation applications. While these studies investigate Gaussian Splatting for sim-to-real manipulation, our work focuses on using it for personalized sim-to-real Image-Goal Navigation. For additional methods utilizing Neural Fields and Gaussian Splatting in robotics, we refer to the comprehensive surveys [20, 60] which examines the applications of Neural fields in robotics.

E. Mesh Processing using Blender

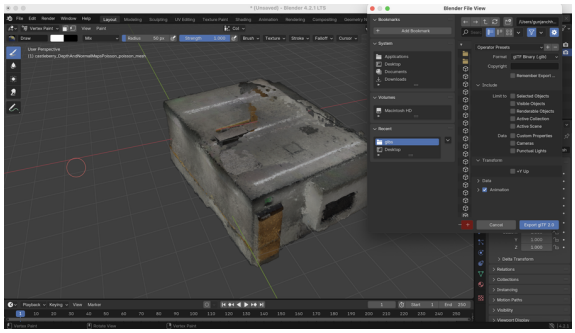


Figure 18. Converting the `.ply` files generated by DN-Splatter [49] to `.glb` files using Blender. Transform `+Y Up` is kept unchecked following conventions of Habitat-Sim [39].

DN-Splatter [49] gives us Poisson reconstructed meshes from the Gaussian Splats in `.ply` format. In order to load these meshes in Habitat-Sim [39], we need to convert these to `.glb` format. In order to do so, we import the `.ply` in Blender and then export them as `.glb` files while ensuring that `+Y Up` transform is unchecked, following the conventions used in Habitat-Sim.

F. Episode Statistics for Generated Datasets on `Captured` Scenes

Tab. 2 presents the shortest path lengths (Min, Max, and Avg.) for both training and validation episodes across the different types of GS-based scenes used in our work.

Notably, the scale of the `Captured` scenes is comparable to that of the `MuSHRoom` [42] scenes, suggesting a similar range of complexity in terms of environment size and

Scene Type	Scene	Train			Val		
		Min	Max	Avg.	Min	Max	Avg.
MuSHRoom	activity	7	127	59.623	17	120	59.82
	sauna	5	60	34.615	7	57	34.31
	honka	6	61	33.563	16	63	34.16
Captured	lounge	8	99	51.431	18	96	55.3
	classroom	7	78	35.161	11	70	32.39
	conf_b	5	42	20.766	5	34	19.77
	conf_a	5	46	19.996	6	43	19.92

Table 2. Shortest path lengths for episodes across MuSHRoom [42] and Captured scenes.

structure. In future, we will attempt to collect apartment-scale scenes which require considerably larger number of steps (∼200).

Within the Captured scenes, there is variation in scale, with certain scenes such as lounge having much larger path lengths compared to others like conf_b or conf_a, which are conference rooms.

Both GS training and agent training is likely to be influenced by this scale variability, as the agent must learn to navigate environments that may range from relatively simple, smaller spaces to more expansive, complex ones.

G. Depth and Normal Encoder Selection for DN-Splatter

Since we do not have ground-truth meshes for our Captured scenes, we use all 10 MuSHRoom [42] scenes as a proxy for evaluation of different depth and normal encoders towards 3D scene reconstruction.

We evaluate various depth encoders using monocular depth predictions and assess their reconstruction performance based on several metrics following MuSHRoom [42]: accuracy (Acc), completion (Comp), Chamfer distance (C-L1), normal consistency (NC), and F-score. The depth scale for each encoder is defined by the ground truth (GT) from an iPhone, with a transform (scale and bias) learned via gradient optimization to convert the monocular depths from the encoders into the appropriate scale.

As shown in Tab. 3, the GT depth from iPhone provides the best performance across most metrics, achieving the highest normal consistency (0.815) and F-score (0.748). Among the tested depth encoders, GT consistently outperforms others, confirming its robustness for reconstruction tasks. Since the GT depth encoder performed the best overall, we selected the same for our subsequent evaluations for normal encoders. As shown in Tab. 4, the Metric3D-v2 [16] encoder achieves the highest F-score (0.752) when used with GT depth, outperforming other normal encoders - DSINE [3] and Omnidata [14]. Therefore, for 3D reconstruction of our Captured scenes, we use the GT-depth and Metric3D-v2 normals.

H. Agent, Training, and Evaluation Details

Agent: We employ the Hello Robot Stretch robot embodiment as our agent, following the setup in [46]. In the Habitat Simulator, the agent is modeled as a cylinder with a height of 1.41m and a radius of 0.3m. The RGB camera sensor is positioned at a height of 1.31m from the ground and is vertically aligned. The sensor outputs images with a resolution of 640×480 (H×W) and a horizontal field of view of 42°. The goal sensor is configured with identical parameters to the RGB sensor. During training, the agent's rotation at the goal location is randomly sampled.

Training and Evaluation: We train our agents using DD-PPO [50] with ImageNav reward, with each environment generating 64 frames per rollout. The training process includes 2 PPO [45] epochs, each consisting of 2 mini-batches. Following [46], we use the AdamW optimizer [29] with a weight decay of 10^{-6} and a learning rate of 2.5×10^{-4}, unless stated otherwise. The visual encoder is kept unfrozen during training and data augmentation is applied. The visual encoder learning rate is set to 1.5×10^{-6} for HM3D and other settings, and 1.5×10^{-5} for HSSD. The goal and observation visual encoders share the same weights. The number of environments per GPU is set to 10 for HM3D/HSSD, and 8 for Captured scenes due to computational constraints. Policies are trained to convergence across all training setups and datasets. Checkpoints are saved approximately every $\sim 3M$ steps and evaluated on the validation sets of the corresponding datasets. The best checkpoint is selected based on the highest success rate (SR) achieved on the validation set. For training, we utilized 16 NVIDIA A40 GPUs per policy per dataset. For simulation-based evaluations, we utilize a single A40 GPU with one environment for Captured scenes and MuSHRoom scenes, and 20 environments for HM3D/HSSD. For real-world evaluations, we employ a single environment and a single A40 GPU. For details on real-world setup, please refer to Appendix A.

Visual Encoder: The observation and goal images are resized to 160×120 before being input into the VC-1-Base visual encoder [32]. Patch embeddings are

Depth Type	Acc ↓	Comp ↓	C-L1 ↓	NC ↑	F-score ↑
DepthAnything-v2 (Indoor) [57]	0.054	0.090	0.072	0.800	0.688
GT	**0.047**	0.090	**0.068**	**0.815**	**0.748**
Metric3D-v2 [16]	0.052	0.089	0.071	0.804	0.700
UniDepth-v2 [37]	0.049	**0.087**	**0.068**	0.813	0.719
ZoeDepth [5]	0.061	0.091	0.076	0.780	0.621

Table 3. Average metrics for `MuSHRoom` Scenes computed against Faro-Scanner ground-truths for different depth types. The normal encoder used is Omnidata [14]. Ground truth from iPhone leads to the highest F-scores.

Normal Type	Depth Type	Acc ↓	Comp ↓	C-L1 ↓	NC ↑	F-score ↑
DSINE [3]	GT	0.046	0.090	0.068	0.815	0.750
Metric3D-v2 [16]	GT	0.046	0.090	0.068	**0.818**	**0.752**
Omnidata [14]	GT	0.047	0.090	0.068	0.815	0.748

Table 4. Average metrics for `MuSHRoom` scenes computed against Faro-Scanner ground truths for different normal encoders. The depth used is the ground truth from iPhone. Metric3D-v2 outperforms other normal encoders.

processed through a "compression layer" [56], which comprises of a 2D convolution followed by group normalization, to generate the final embeddings.

Policy: The policy is implemented as a 2-layer LSTM that takes as input the previous action embedding, visual observation embeddings, and goal image embeddings following [46]. It outputs one of the following discrete actions: `MOVE_FORWARD`, `TURN_LEFT`, `TURN_RIGHT`, or `STOP`.

Reward Function: The reward function is adapted from [46, 56], using the same hyperparameters as in [46]: success weight $c_s = 5.0$, angle success weight $c_a = 5.0$, goal radius $r_g = 1.0$, angle threshold $\theta_g = 25°$, and slack penalty $\lambda = 0.01$. The collision penalty is defined as $c_{coll} = 0.03$.

The full reward function is detailed in Eq. (1).

$$
\begin{aligned}
r_t = & \, c_s \times ([d_t < r_g] \wedge [a_t = \text{STOP}]) \\
& + c_a \times ([\theta_t < \theta_g] \wedge [a_t = \text{STOP}]) \\
& + \left(\hat{\theta}_{t-1} - \hat{\theta}_t\right) \\
& + (d_{t-1} - d_t) - \gamma \\
& - c_{\text{coll}} \times [\text{collision = true}]
\end{aligned} \tag{1}
$$

I. Real-world metrics for successful episodes

Tab. 5 shows the distance-to-goal (in centimeters) and the number of steps taken on average for the successful episodes for each type of policy we deploy in the real world. We note that some of these average values may not include enough samples to be statistically significant. Therefore, we refrain from making conclusions about which policy is more efficient based on these, and provide the values for book-keeping purposes.

Policy	D2G (cm)	#Steps
HM3D	44.20	45.60
HM3D-FT-Poly	48.29	58.14
HM3D-FT-DN	29.64	52.86
HSSD	98.50	38.00
HSSD-FT-Poly	37.10	54.00
HSSD-FT-DN	40.87	36.25

Table 5. Average distance to goal and number of steps in the real-world `lounge` scene for successful episodes for each policy.

J. Simulation results on additional real-world scenes

In Tab. 6, we present additional results for HM3D zero-shot and fine-tuned policies across several scenes. Five of these are from the `MuSHRoom` dataset—`koivu`, `classrm2`, `kokko`, `coffeerm`, and `vr_rm`. We also include one real-world scene, `conf_c`, that we captured using our methodology. Fine-tuning consistently improves performance across all mesh types. The improvements are modest in smaller scenes (e.g., `koivu`, `coffeerm`), where the zero-shot policy already performs well. In contrast, larger scenes (e.g., `conf_c`, `vr_rm`) benefit significantly from fine-tuning.

We also visualize the relationship between HM3D zero-shot success rate, validation PSNR, and average shortest path distance in Fig. 19. We observe a consistent trend with the previous results: the zero-shot success rate decreases as the size of the scene increases, and improves as the validation PSNR of the trained Gaussian Splat (GS) increases.

An exception is `vr_rm`, which appears as an outlier in the `MuSHRoom` dataset and warrants further investigation. Aside from this case, the remaining scenes from `MuSHRoom` generally show improved zero-shot performance as PSNR increases.

Scene	Mesh	HM3D-ZS	HM3D-FT
koivu	DN	0.87	0.98
classrm2	DN	0.61	0.97
kokko	DN	0.66	0.99
coffeerm	DN	0.90	1.00
vr_rm	DN	0.39	0.99
conf_c	DN	0.16	0.84
	Poly	0.50	0.98

Table 6. Validation success rates on additional scenes from MuSHRoom and Captured datasets.

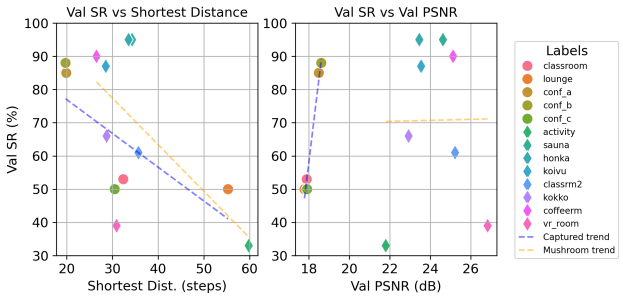


Figure 19. **HM3D zero-shot validation success rates on DN meshes vs. validation GS PSNRs vs. average shortest distances of validation episodes.** The zero-shot SR is inversely correlated with scene scale and directly correlated with GS validation PSNR. vr_rm is an outlier in the MuSHRoom dataset and requires further analysis.

K. Failure modes for HM3D zero-shot on lounge scene in simulation

To assess the role of visual fidelity, we analyze zero-shot HM3D policy trajectories and identified failure modes in the lounge scene using both DN and POLYCAM meshes (Tab. 7) in simulation. We found significant increase in "Maximum Steps Reached" failures with the DN mesh, indicating low visual fidelity prevents the policy from matching goal images— a key limitation of the pre-trained policy.

Termination Reason	**POLYCAM**	**DN**
Target Reached (success)	74	55
Early Stop (Goal Not Visible)	3	5
Early Stop (Goal Visible)	11	16
Early Stop (Similar Goal Visible)	4	5
Maximum Steps Reached	8	19

Table 7. Sim failure analysis for HM3D-ZS on lounge.

L. Overfitting vs. Fine-Tuning

In this section, we clarify why the overfitting results shown in Fig. 9 may appear stronger than the fine-tuning results in Fig. 6. Fine-tuning requires significantly fewer steps because the pre-trained policy already captures general navigation behaviors. We also deliberately limit the number of fine-tuning steps to preserve the policy's ability to generalize to real-world environments—leveraging knowledge acquired during large-scale pre-training across diverse scenes.

This pre-training not only enables better generalization in the real world (see Fig. 8) but also substantially reduces overall training time, which is critical for rapid deployment.

In contrast, our overfitting experiments involve training policies exclusively on simulated versions of specific real scenes. These policies typically achieve 80–90% success within 20–30M steps, but their performance is limited to the training environment. For completeness, we early stop around 100M steps, although most gains saturate earlier.

Tab. 8 reports the best validation success rates of overfitted policies at 20M steps. Despite strong performance in simulation, these policies fail to generalize to real-world settings due to factors such as lighting variations, sensor noise, and actuation inaccuracies.

Scene	DN val. SR.	Poly val. SR.
lounge	0.74	0.74
classrm	1.00	0.85
conf_a	0.99	1.00
conf_b	1.00	1.00
honka	1.00	—
sauna	0.99	—
activity	0.79	—

Table 8. Overfitted validation success rates after 20M steps.