Image2Sim: Scaling Embodied Navigation via Generative Neural Simulator

Zihan Wang[1], **Seungjun Lee**[1], **Yinghao Xu**[2], **Gim Hee Lee**[1]
[1]National University of Singapore, [2]HKUST
zihan.wang@u.nus.edu, gimhee.lee@nus.edu.sg

Abstract

Embodied navigation aims to build agents that interpret multimodal goals, reason in 3D space, and reach target destinations reliably in the real world. However, progress remains constrained by the lack of scalable, high-fidelity, and physically grounded interactive environments. Although real-world scanned datasets offer visual realism, they are limited by scale. In contrast, synthetic simulators scale more easily but often exhibit large sim-to-real gaps. We introduce Image2Sim, a real-time neural simulation framework that constructs high-quality interactive environments from posed RGB-D image sequences. The central idea is to decouple *3D spatial anchoring* from *photorealistic observation synthesis*. For scene construction, Image2Sim uses a feed-forward feature Gaussian model that lifts posed RGB-D observations into a 3D feature-Gaussian representation in a single pass. For rendering, we propose a Geometry-Aware One-Step Pixel Flow model that transforms sparse and noisy Gaussian projections into high-quality panoramic RGB-D observations. Image2Sim also serves as a fully automated embodied data engine that generates high-fidelity observations, executable actions, and diverse navigation instructions at scale. It converts large collections of videos and images into nearly 20K interactive scenes and synthesizes more than 10 million navigation training samples. Navigation models trained entirely in these neural environments achieve strong improvements on major benchmarks and transfer effectively to real-world zero-shot settings. These results suggest that scalable neural simulation can serve as a practical training substrate for embodied navigation at scale. **Project page:** github.com/MrZihan/Image2Sim

1 Introduction

Large language models [1–3], vision foundation models [4–6], and visual generative models [7–9] suggest that scaling data can lead to substantial gains in capability. However, similar scaling in embodied navigation [10–14] has not delivered comparable progress in robust real-world generalization, largely due to the lack of suitable data. A key limitation lies in the availability of scalable sources of high-fidelity, physically grounded, and interactive 3D environments.

Existing data sources expose a fundamental tradeoff between visual fidelity and scalability. Training data is typically drawn from two primary sources: real-world 3D scans [15–18] and synthetic procedural environments [19–21]. Real-world scans provide strong visual fidelity, but are expensive to acquire and difficult to scale. Procedural environments improve scalability, but often introduce substantial sim-to-real gaps due to unrealistic assets, layouts, and rendering statistics. Recent generative video models [22–24] offer another potential source of realistic visual data. However, they struggle to maintain the rigid-body consistency and explicit collision structure required for reliable closed-loop interaction. These limitations suggest that scalable embodied data requires both real-world visual fidelity and explicit 3D physical grounding.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

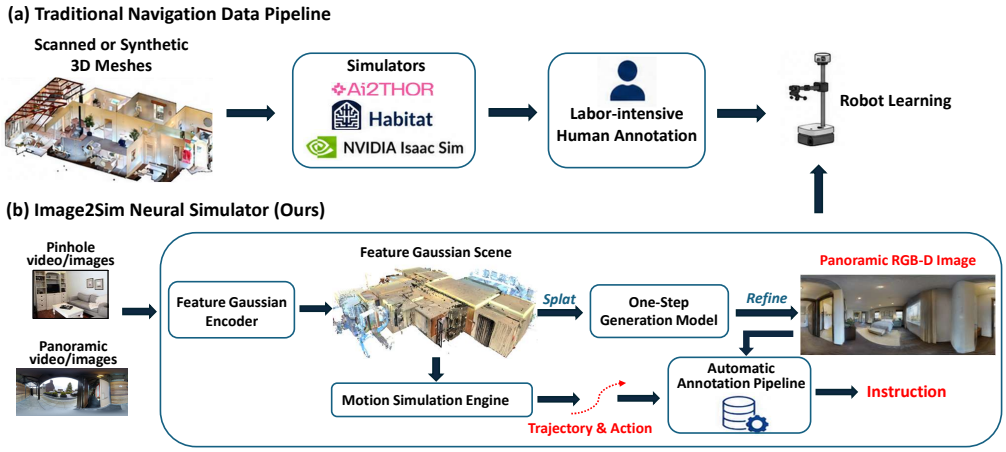


Figure 1: Comparison of (a) traditional navigation data pipeline and (b) our Image2Sim framework.

These failure modes share a common cause: existing pipelines often couple 3D geometry and visual synthesis within a single mechanism, forcing a tradeoff among scale, physical grounding, and visual fidelity. To overcome this tradeoff, we introduce Image2Sim, a real-time neural environment engine that constructs high-quality interactive scenes from posed RGB-D image sequences at scale. The key idea is to decouple 3D spatial anchoring from photorealistic observation synthesis. Specifically, Image2Sim first uses a feed-forward feature Gaussian model to lift observations into 3D feature Gaussians in a single pass, producing an explicit geometric and semantic scene representation. We further propose a Geometry-Aware One-Step Pixel Flow model to synthesize high-fidelity panoramic RGB-D observations, which formulates rendering as a probability flow ODE conditioned on multimodal priors and guided by 3DGS alpha maps. This design enables momentum-based self-distillation and compresses multi-step generation into a direct single-step mapping. However, a neural renderer alone is not enough for navigation training; executable actions and aligned language instructions are further required.

Image2Sim therefore extends from a neural renderer into a fully automated embodied data engine. Instead of relying on unconstrained generation, we integrate a white-box, geometry-aware motion engine that respects collision constraints and produces physically valid trajectories targeting long-tail and object-centric goals. A large vision-language model [25] then generates natural language instructions aligned with these paths. The framework constructs vision, action, and instruction data jointly at scale, producing over 10M navigation samples across 20K interactive scenes. Navigation models trained entirely within these neural environments show strong gains on standard embodied benchmarks and transfer effectively to real-world zero-shot settings. These results indicate that scalable neural simulation is a practical route toward embodied scaling.

In summary, our contributions are threefold:

- **A Geometry-Aware Neural Simulator.** We propose a real-time simulation framework that explicitly decouples feed-forward 3D scene construction from photorealistic panoramic RGB-D rendering. This design enables geometrically grounded, high-fidelity, and computationally efficient environments, supporting panoramic RGB-D synthesis at approximately 40 FPS on a single RTX 4090 GPU.

- **A Scalable Embodied Data Engine.** We develop a fully automated pipeline integrating object-centric target mining, physics-aware trajectory generation, and VLM-based instruction annotation. The pipeline produces over 10M navigation training samples across 20K interactive scenes.

- **Empirical Evidence for Embodied Scaling.** We demonstrate that increasing Image2Sim-generated data yields consistent navigation gains over the evaluated data scales, while models trained exclusively in our neural environments achieve strong cross-simulator performance and effective real-world zero-shot transfer. These findings validate scalable neural simulation as a practical path for improving embodied navigation.

2 Related Work

Simulation Environments for Embodied Navigation. Embodied navigation mainly relies on two simulation paradigms. Real-scanned environments such as Matterport3D, HM3D, Gibson, and Replica [15–17, 26], together with simulators such as Habitat [18, 27], provide strong geometric realism, real-world layout statistics, and physical grounding. However, high-quality 3D scanning and digital-twin post-processing are expensive and labor-intensive, limiting scene scale and diversity. Synthetic and procedural environments [19–21, 28–33] offer better scalability, controllability, and task diversity, but often suffer from sim-to-real gaps caused by unrealistic assets, layouts, and rendering statistics. Image2Sim instead constructs interactive environments directly from large-scale images and videos, aiming to combine scalable scene generation with the realism of real-world captures.

Neural Scene Representations and World Models. Neural simulation is closely related to neural scene representations and generative world models. Optimization-based methods such as NeRF [34] and 3D Gaussian Splatting [35], as well as feature-augmented variants [36], achieve high-fidelity rendering but require costly per-scene optimization, limiting their use as large-scale embodied data engines. Recent embodied simulators built on such neural scenes [37–44] inherit this scalability limitation. Feed-forward reconstruction methods [45–52] reduce scene construction cost, but often degrade under noisy inputs, partial observations, or sparse views, producing incomplete geometry and missing regions. Generative world models [23, 24, 9, 53] can synthesize realistic visual observations, but generally lack persistent 3D structure, explicit navigability, and collision-aware geometry for closed-loop interaction. Image2Sim addresses these limitations by decoupling *3D spatial anchoring* from *photorealistic observation synthesis*, using feed-forward feature Gaussians for persistent geometry and a one-step pixel-flow renderer for panoramic RGB-D generation.

Scaling Vision-Language Navigation. Scaling data has become a central direction in Vision-Language Navigation (VLN). Early progress was largely driven by human-annotated datasets [10–13, 54], which provide high-quality training supervision but remain constrained by annotation cost and the limited number of underlying 3D environments. To reduce annotation bottlenecks, subsequent works synthesize additional instructions, goals, or trajectories using pretrained vision-language models or goal-oriented data engines [55–63]. These methods substantially increase the amount and diversity of language-action supervision, but the resulting data is still grounded in a fixed pool of simulator-supported scenes [15–17]. As a result, the distribution of interactive environments remains comparatively narrow, leaving scene-scale diversity as an underexplored dimension of VLN scaling. Image2Sim addresses this dimension by converting large-scale image and video data into near 20K interactive environments, from which it generates over 10M physically grounded vision-language-action samples.

3 Method

3.1 Problem Formulation

Let a captured scene be represented by a sequence of posed RGB-D observations $\mathcal{O} = \{(I_i, D_i, \mathbf{K}_i, \mathbf{T}_i)\}_{i=1}^{N}$, where $I_i \in \mathbb{R}^{H\times W\times 3}$ is an RGB image, D_i is its aligned depth map, $\mathbf{K}_i$ denotes the camera intrinsics, and $\mathbf{T}_i \in SE(3)$ is the camera-to-world pose. For RGB-only videos, the depth, camera poses, and intrinsics can be obtained from off-the-shelf 3D foundation models [6, 66, 67].

We build Image2Sim to convert passive scene captures into executable neural simulators for scalable embodied navigation data synthesis. Here, simulation concerns static-scene navigation with geometric collision checking and kinematic motion; contact dynamics and movable-object interaction are outside our scope. As illustrated in Figure 2 and 3, the system couples three components: (i) feed-forward construction of a persistent 3D feature-Gaussian scene, (ii) geometry-aware one-step panoramic RGB-D rendering, and (iii) executable motion simulation for trajectory and instruction generation. The key design principle is to combine explicit 3D grounding with generative completion: reliable observed geometry is preserved through Gaussian splatting, and unobserved regions are completed by a conditional pixel-flow renderer.

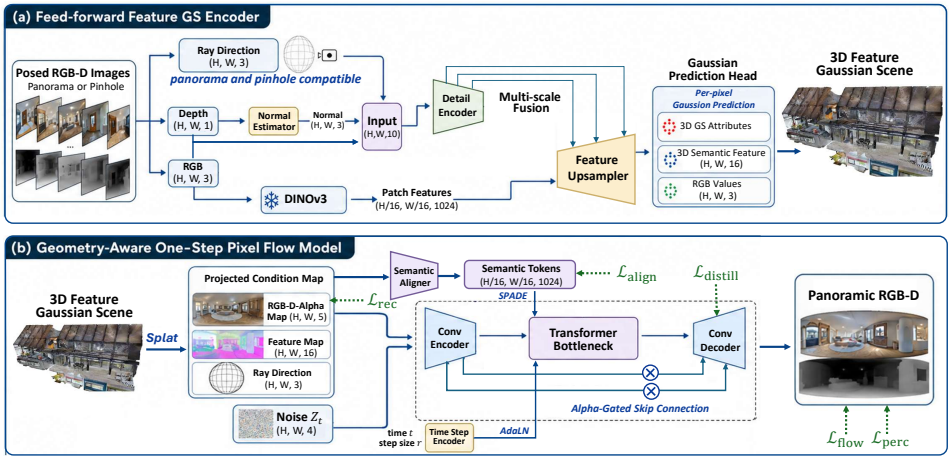


Figure 2: Framework of the feed-forward GS encoder and one-step pixel flow model.

Given observations $\mathcal{O}$, Image2Sim constructs an interactive simulator $\mathcal{S} = \{\mathcal{G}, \mathcal{M}, \mathcal{R}_\theta\}$, where $\mathcal{G}$ is a persistent 3D feature-Gaussian scene, $\mathcal{M}$ is a traversable voxel connectivity graph, and $\mathcal{R}_\theta$ maps arbitrary agent poses to panoramic RGB-D observations. $\mathcal{S}$ supports physically grounded atomic actions, collision-aware trajectory sampling, and panoramic rendering along feasible paths. Using $\mathcal{S}$, we synthesize aligned vision-language-action triplets $\mathcal{D}_{\text{nav}} = \{(X_{1:T}, a_{1:T}, Y)\}$, where $X_{1:T}$ are rendered panoramic observations, $a_{1:T}$ are executable low-level actions, and Y denotes the corresponding navigation instruction.

3.2 Feed-Forward 3D Scene Construction

Semantic-geometric pixel lifting. Image2Sim first lifts posed RGB-D frames into a persistent 3D representation without per-scene optimization. For each frame, we extract dense visual features using a dual-stream encoder. A frozen DINOv3 [5] backbone provides high-level semantic features, and a lightweight geometric detail stream preserves local details from RGB, metric depth, surface normals, and ray directions. The ray-direction encoding makes the Gaussian encoder compatible with both pinhole and panoramic cameras.

The semantic and geometric features are fused by a feature upsampler and decoded by a Gaussian prediction head into dense 3D feature Gaussians:

$$\mathcal{G} = \{g_j\}_{j=1}^{M}, \qquad g_j = (\boldsymbol{\mu}_j, \mathbf{s}_j, \mathbf{q}_j, \alpha_j, \mathbf{c}_j, \mathbf{f}_j), \tag{1}$$

where $\boldsymbol{\mu}_j \in \mathbb{R}^3$ is the Gaussian center, $\mathbf{s}_j \in \mathbb{R}^3$ is the anisotropic scale, $\mathbf{q}_j \in \mathbb{R}^4$ is the rotation, $\alpha_j \in [0, 1]$ is the opacity, $\mathbf{c}_j \in \mathbb{R}^3$ stores RGB values, and $\mathbf{f}_j$ stores compact semantic features. The Gaussian centers are deterministically unprojected from depth and voxel-downsampled, which anchors the representation to metric scene geometry. Unlike optimization-based Gaussian reconstruction, this feed-forward lifting amortizes scene construction across environments and enables real-time conversion of large image collections into executable 3D scenes.

Projection and semantic grounding. Given a target panoramic pose p, the Gaussian scene $\mathcal{G}$ can be splatted into the panorama coordinate system to obtain projected RGB, depth, feature, and opacity maps:

$$\mathbf{C}_p = [\tilde{\mathbf{I}}_p, \tilde{\mathbf{D}}_p, \tilde{\mathbf{F}}_p, \tilde{\mathbf{A}}_p, \mathbf{R}_p], \tag{2}$$

where $\tilde{\mathbf{I}}_p$, $\tilde{\mathbf{D}}_p$, and $\tilde{\mathbf{F}}_p$ are the splatted RGB, depth, and semantic feature maps, $\tilde{\mathbf{A}}_p$ is the accumulated opacity map, and $\mathbf{R}_p$ is the panoramic ray-direction map.

Observed regions are supervised by an RGB-D reconstruction loss. Projected feature maps are then tokenized by a semantic aligner Φ and aligned with DINOv3 features extracted from the target RGB panorama:

$$\mathcal{L}_{\text{rec}} = \left(\|\tilde{\mathbf{I}}_p - \mathbf{I}_p\|_1 + \|\tilde{\mathbf{D}}_p - \mathbf{D}_p\|_1\right) \odot \mathbf{M}_p, \quad \mathcal{L}_{\text{align}} = \left(1 - \frac{\Phi(\mathbf{C}_p) \cdot \mathbf{S}_p}{\|\Phi(\mathbf{C}_p)\|_2 \|\mathbf{S}_p\|_2}\right) \odot \mathbf{M}'_p. \tag{3}$$

$\mathbf{M}_p$ is a validity mask derived from ground-truth depth, $\mathbf{S}_p$ denotes ground-truth DINOv3 tokens from the target RGB image, and $\mathbf{M}'_p$ is the downsampled validity mask. This supervision encourages $\mathcal{G}$ to encode local appearance and semantic layout cues that are useful for novel-view completion.

3.3 Geometry-Aware One-Step Pixel Flow Rendering

Direct Gaussian splatting preserves observed geometry but produces holes and noisy projections under sparse captures. Conversely, unconstrained image generation can synthesize plausible pixels but lacks persistent 3D grounding. Image2Sim bridges this gap with a geometry-aware pixel-flow renderer that performs conditional generative completion under explicit 3D constraints. Its contribution is the spatially selective use of generative completion according to 3D projection reliability, rather than introducing feature Gaussians, MeanFlow, or EMA teachers individually.

Alpha-gated source state. Let $\mathbf{X}_p = [\mathbf{I}_p, \mathbf{D}_p]$ denote the target panoramic RGB-D observation at pose p. We first obtain a projected RGB-D prior: $\tilde{\mathbf{X}}_p = [\tilde{\mathbf{I}}_p, \tilde{\mathbf{D}}_p]$ and an opacity map $\tilde{\mathbf{A}}_p$ that estimates the reliability of the 3D projection. We then define a spatially varying noise scale:

$$\Sigma(\tilde{\mathbf{A}}_p) = \tilde{\mathbf{A}}_p \cdot \sigma_{\text{small}} + (1 - \tilde{\mathbf{A}}_p) \cdot \sigma_{\text{large}}. \tag{4}$$

Finally, the alpha-gated source state is constructed as:

$$\mathbf{z}_{\text{src}} = \tilde{\mathbf{A}}_p \odot \tilde{\mathbf{X}}_p + \Sigma(\tilde{\mathbf{A}}_p) \odot \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}). \tag{5}$$

High-opacity regions remain close to the projected 3D evidence. In contrast, low-opacity regions receive larger perturbations and are handled by generative completion.

Single-step pixel-flow renderer. To estimate the geometry-conditioned probability flow, we parameterize the denoised prediction $\mathbf{x}_\theta$ with a UNet-style architecture. A convolutional encoder first compresses the alpha-gated source state $\mathbf{z}_{\text{src}}$ together with the geometric condition $\mathbf{C}_p$. A deep transformer bottleneck then captures global layout dependencies and infers unobserved scene structures. Within this bottleneck, time step is injected through AdaLN [8], and semantic tokens $\Phi(\mathbf{C}_p)$ are injected through SPADE-like spatially-adaptive normalization [68] to provide high-level layout guidance. During convolutional upsampling, multi-scale skip connections are further modulated by an alpha pyramid derived from $\tilde{\mathbf{A}}_p$. These alpha-gated skips preserve reliable projected evidence in high-opacity regions and suppress noisy features in missing regions.

3.4 Training the Pixel-Flow Renderer

To generate high-quality panoramic RGB-D images in real time, we replace traditional multi-step flow matching [69] with a continuous-time MeanFlow formulation [70, 71], which trains the network to estimate the average velocity for one-step transport from the source state to the target observation.

During training, we hold out panoramic RGB-D frames $\mathbf{X}_p$ as target supervision. Given the source state $\mathbf{z}_{\text{src}}$, we define a linear probability path following the pixel MeanFlow convention [71], where $t = 0$ corresponds to the target panorama and $t = 1$ corresponds to the projected noisy source:

$$\mathbf{z}_t = (1 - t)\mathbf{X}_p + t\mathbf{z}_{\text{src}}, \qquad \mathbf{v}_{\text{target}} = \mathbf{z}_{\text{src}} - \mathbf{X}_p, \qquad t \in [0, 1]. \tag{6}$$

Following pixel MeanFlow [71], we let the network predict the denoised observation $\mathbf{x}_\theta$ and parameterize the average velocity as $\mathbf{u}_\theta = (\mathbf{z}_t - \mathbf{x}_\theta)/t$. We then construct the instantaneous velocity field $\mathbf{V}_\theta$ and flow loss:

$$\mathbf{u}_\theta = \frac{\mathbf{z}_t - \mathbf{x}_\theta}{\max(t, \varepsilon)},\ \mathbf{V}_\theta = \mathbf{u}_\theta + (t-r)\frac{d\mathbf{u}_\theta}{dt},\ \mathcal{L}_{\text{flow}} = \mathbb{E}_{r,t,\boldsymbol{\epsilon},\mathbf{X}_p}\left[\mathbf{W}(\tilde{\mathbf{A}}_p) \odot \|(\mathbf{V}_\theta - \mathbf{v}_{\text{target}})\|_2^2\right]. \tag{7}$$

Here, $0 \le r \le t \le 1$ define the integration interval, and ε prevents division by zero. The network conditions on both t and $h = t - r$; $\mathbf{x}_\theta$ parameterizes the average velocity, while transport to r uses $\mathbf{z}_r = \mathbf{z}_t - (t - r)\mathbf{u}_\theta$. At inference, $(t, r) = (1, 0)$ yields the one-step RGB-D prediction. The total derivative $\frac{d\mathbf{u}_\theta}{dt}$ captures how the predicted average velocity changes along the probability path. Instead of computing this derivative with expensive higher-order backpropagation, we use forward-mode Jacobian-vector products (JVPs) [70] to efficiently evaluate the directional derivative along the flow trajectory, and apply stop-gradient to this derivative term in the loss. The opacity-adaptive weighting map $\mathbf{W}(\tilde{\mathbf{A}}_p)$ assigns larger gradients to low-confidence regions where $\tilde{\mathbf{A}}_p \approx 0$. This explicitly concentrates the learning signal on unobserved regions that require generative completion while concurrently preventing the network from unnecessarily modifying reliable 3D projections.

Momentum-based self-distillation. Single-step generation is efficient, but can be unstable in severely occluded or weakly observed regions. Inspired by DINO [72], we introduce a momentum-based

self-distillation loss to improve stability and suppress hallucinated artifacts. We maintain a teacher network updated by the exponential moving average (EMA) of the student parameters. Unlike the student that receives the noisy alpha-gated source state, the teacher is conditioned on a privileged source state $\mathbf{z}_{\text{gt}}$ constructed from the ground-truth RGB-D panorama. The student is then encouraged to match the multi-scale decoder features of the teacher across a set of decoder levels $\mathcal{K}$:

$$\mathcal{L}_{\text{distill}} = \sum_{k \in \mathcal{K}} \tilde{w}_k \left[1 - \frac{\mathbf{f}_k^S \cdot \mathbf{f}_k^T}{\|\mathbf{f}_k^S\|_2 \|\mathbf{f}_k^T\|_2} + \beta \|\mathbf{f}_k^S - \mathbf{f}_k^T\|_2^2 \right], \tag{8}$$

where $\mathbf{f}_k^S$ and $\mathbf{f}_k^T$ are the student and teacher feature maps at level k, respectively, and $\tilde{w}_k$ are normalized layer-wise weights that emphasize deeper semantic layers. This distillation improves single-step completion ability by transferring the cleaner latent structure of the teacher to the student.

Full learning objective. The complete training objective jointly optimizes geometric projection, semantic grounding, generative completion, and single-step rendering stability: $\mathcal{L}_{\text{total}} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{align}}\mathcal{L}_{\text{align}} + \lambda_{\text{flow}}\mathcal{L}_{\text{flow}} + \lambda_{\text{distill}}\mathcal{L}_{\text{distill}} + \lambda_{\text{perc}}\mathcal{L}_{\text{perc}}$, where $\mathcal{L}_{\text{perc}}$ is a perceptual LPIPS loss [73], and the λ coefficients balance the different terms.

3.5 Motion Simulation and Trajectory Generation

High-fidelity visual rendering alone is insufficient for embodied learning: the simulator must also support physically valid agent motion. As shown in Figure 3, Image2Sim addresses this requirement by deriving a dense navigability structure directly from the reconstructed 3D scene. We voxelize the Gaussian scene into a dense grid $\mathcal{V}$ and determine traversability using both semantic labels and geometric obstacle constraints, such as floor height, free-space connectivity, and clearance from occupied regions. To account for finite-size agents, we perform GPU-parallel ray marching over the voxelized geometry to query occupancy within the agent footprint. This allows the simulator to reject invalid motions, detect geometric penetration, and support physically plausible sliding along obstacle boundaries. The resulting structure forms a traversable voxel connectivity graph $\mathcal{M} = (\mathcal{V}, \mathcal{E})$ where nodes denote traversable voxels and edges connect locally reachable neighboring states.

For trajectory generation, we plan paths over $\mathcal{M}$ using path-length and collision-aware edge costs, similar to the NavFn planner [74, 75]. The collision-aware term penalizes states close to obstacles, encouraging the planner to favor safer corridors rather than merely shortest paths. A lookahead pure-pursuit controller [76] then converts discrete graph paths into smooth executable motion. Finally, the simulator replays these trajectories to render panoramic RGB-D observations along the exact kinematic path, conditioned on the same persistent feature-Gaussian scene. Generated RGB-D observations are never fused back into the navigability map and cannot change collision boundaries or oracle actions. Cross-view consistency in fully unsupported regions is not guaranteed; Appendix F.1 discusses this limitation.

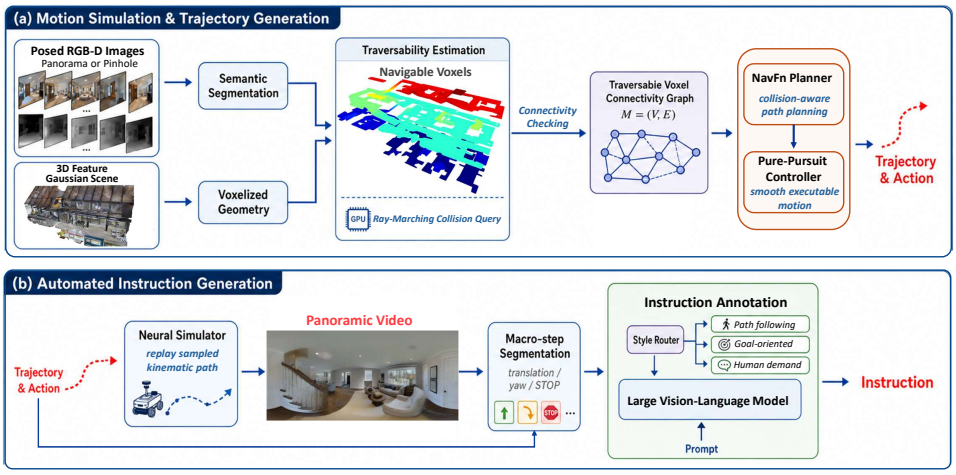


Figure 3: Framework of the motion simulation engine and automated instruction generation.

3.6 Automated Instruction Generation

Image2Sim further converts simulated trajectories into vision-language-action annotations. Each sampled feasible trajectory is replayed in the neural simulator to render panoramic RGB-D observations along the executed motion. To bridge the granularity gap between dense low-level actions and human navigation language, we segment each trajectory into semantic macro-steps according to accumulated translation, yaw rotation, and terminal STOP events. Each macro-step is mapped to a motion primitive and aligned with the panorama rendered at its starting pose, yielding a compact observation-motion sequence for language annotation. The aligned sequence is then annotated by Qwen3-VL-32B-Instruct [25] to synthesize natural-language navigation instructions. A style-conditioned annotation router further diversifies the instructions across fine-grained path following, high-level goal-oriented descriptions, and natural human demands. This automatic pipeline produces scalable embodied navigation data without manual trajectory or instruction annotation.

4 Experiments

4.1 Experimental Setup

Datasets. We assemble a source corpus of 19,936 real-world and synthetic scenes with images or videos, including RealSee3D [77], Structured3D [78], ARKitScenes [79], HM3D [16], ScanNet [80], Gibson [17], and Matterport3D [15]. Panoramic inputs are fixed at 512×1024, while pinhole images are resized to suitable multiples of 16. The pixel-flow renderer outputs 512×1024 panoramic RGB-D images. From these 20K interactive environments, our data engine synthesizes over 10M vision-language-action samples for navigation training. The source-corpus count precedes split filtering: held-out evaluation/test scenes are excluded from navigation training (Appendix A).

Image2Sim Neural Simulator. Image2Sim is trained on Matterport3D-360 [15, 81], RealSee3D [77], and Structured3D [78] for 400K iterations using 8 NVIDIA RTX 6000 Ada Generation GPUs with a per-GPU batch size of 2. We adopt a three-stage curriculum: geometry initialization of Gaussian encoder in the first 100K steps, pixel-flow training with LPIPS [73] and momentum-based self-distillation after 100K steps, and high-frequency refinement after 300K steps by freezing the convolutional encoder and transformer bottleneck. The self-distillation teacher is updated with EMA [72] decay 0.999.

Image2Nav Navigation Model. Image2Nav is our baseline navigation model, built on Qwen3-VL-4B-Instruct [25] and predicts two discrete actions per step from interleaved historical RGB observations and executed actions. We train and evaluate Image2Nav under two monocular observation settings: the standard pinhole camera (336×336, 90° HFOV/VFOV) and the ultra-wide equirectangular image (512×336, 180° HFOV and 118° VFOV). Unless otherwise specified, we use the standard pinhole camera setting for experimental analyses. Within the training input, each trajectory contains up to 32 uniformly sampled RGB observations, paired with 64 action labels. For long-horizon inference, we use a streaming sparse-dense context: 24 uniformly sampled historical frames provide global memory, and an 8-frame recent window supports dense local prediction with KV-cache reuse. Image2Nav is trained on 64 H20 GPUs with DeepSpeed ZeRO-2 [82], bfloat16 precision, and gradient checkpointing, fine-tuning only the LLM and MLP projector. Stage 1 performs 70K-step offline behavior cloning, and Stage 2 performs 10K-step online DAgger [83, 84] in Image2Sim with randomized 4–32 step chunks that mix model-predicted and oracle actions while supervising all executed spans with oracle labels. After each action, collision checking and replanning from the reached pose provide fresh oracle labels for policy-visited states (Appendix C.4).

4.2 Novel-View Rendering Evaluation

We first evaluate the novel-view rendering quality and speed across varying pose and depth noise distributions (Table 1). AnySplat [52] and Image2Sim (Pinhole) receive identical six-view RGB-D inputs; AnySplat centers are computed from the same metric depth and poses. This adapted baseline removes geometry-estimation error. SE3DS [85] and DiT360 [86] use the same GS projections; Appendix E.1 additionally evaluates the native point-cloud condition of SE3DS. The pure generative baselines, DiT360 [86] and SE3DS [85], can produce plausible pixels but do not satisfy the real-time requirement for closed-loop interaction (0.3 FPS and 3.4 FPS). Conversely, feed-forward 3DGS predictors such as AnySplat [52] operate at very high frequencies ($>$115 FPS), but suffer substantial

structural degradation when large regions are unobserved, as reflected by clear drops in PSNR [87] and SSIM.

Table 1: Novel-view rendering quality and speed. RealSee3D-Synthesis (RS3D-Synth.) has no added noise, Matterport3D-360 (MP3D-360) has medium noise, and RealSee3D-Real (RS3D-Real) has high noise. Pano and pinhole denote input formats.

Method	RS3D-Synth.				MP3D-360				RS3D-Real			
	LPIPS↓	PSNR↑	SSIM↑	**FPS↑**	LPIPS↓	PSNR↑	SSIM↑	**FPS↑**	LPIPS↓	PSNR↑	SSIM↑	**FPS↑**
DiT360 [86] (pano)	0.259	24.56	0.724	0.3	0.337	21.63	0.545	0.3	0.364	19.58	0.564	0.3
AnySplat [52] (pinhole)	0.378	20.31	0.555	**125.8**	0.450	18.32	0.508	**115.1**	0.582	15.23	0.415	**137.3**
SE3DS [85] (pano)	0.379	18.23	0.538	3.4	**0.426**	18.76	**0.536**	3.3	**0.421**	16.78	0.457	3.4
Image2Sim (pinhole)	0.345	23.56	0.583	41.1	0.432	**20.21**	0.525	37.1	0.488	17.17	0.469	43.0
Image2Sim (pano)	**0.335**	**23.88**	**0.591**	41.3	0.428	20.15	0.521	37.5	0.473	**17.43**	**0.470**	45.6

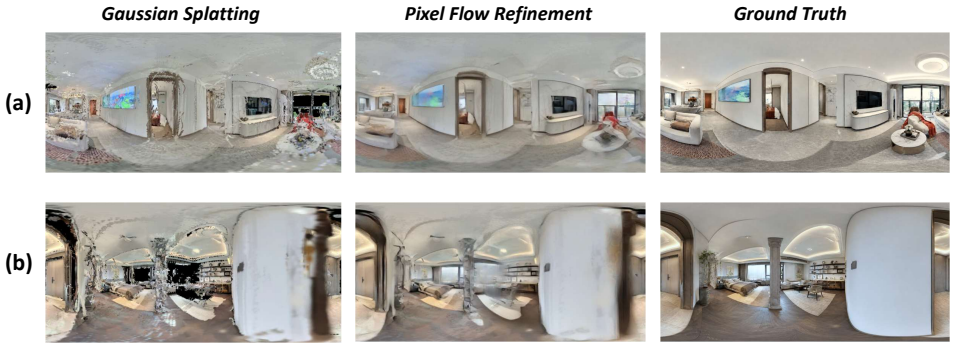


Figure 4: Visualization of Gaussian Splatting and Pixel Flow renderings in high-noise sparse scenes.

By explicitly decoupling spatial anchoring from generative completion, Image2Sim achieves a more balanced tradeoff between fidelity, robustness, and efficiency. On RealSee3D-Real [77], which contains sparse and high-noise LiDAR depth, our panoramic variant maintains 17.43 PSNR and 0.470 SSIM while running at 45.6 FPS, substantially outperforming Gaussian-only AnySplat in rendering quality. These results show that the geometry-aware pixel-flow model can complete structural blind spots under noisy reconstruction, as shown in Figure 4, while the Gaussian projection and alpha guidance help prevent unconstrained hallucination. The consistent performance of both pinhole and panoramic inputs further indicates that Image2Sim can accommodate heterogeneous capture formats, which is important for scaling neural simulation from diverse real-world videos and images.

Table 2: Navigation performance on the unseen validation splits of R2R-CE, RxR-CE, and REVERIE-CE. Image2Nav uses either a standard pinhole camera (336×336, 90° HFOV/VFOV) or an ultra-wide equirectangular view (512×336, 180° HFOV and 118° VFOV).

Methods	**RGB**	**Depth**	**R2R-CE**				**RxR-CE**				**REVERIE-CE**			
			NE↓	OSR↑	SR↑	SPL↑	NE↓	SR↑	SPL↑	nDTW↑	NE↓	OSR↑	SR↑	SPL↑
NaVid [88]	✓		5.47	49.1	37.4	35.9	-	-	-	-	6.74	36.3	26.6	20.8
Uni-NaVid [89]	✓		5.58	53.3	47.0	42.7	6.24	48.7	40.9	-	-	-	-	-
NaVILA [90]	✓		5.22	62.5	54.0	49.0	6.77	49.3	44.0	58.8	-	-	-	-
Dynam3D [91]	✓	✓	5.34	62.1	52.9	45.7	-	-	-	-	6.22	48.9	40.1	28.5
NavFoM [63]	✓		5.01	64.9	56.2	51.2	5.51	57.4	49.4	60.2	-	-	-	-
StreamVLN [92]	✓	✓	4.98	64.2	56.9	51.9	6.22	52.9	46.0	61.9	-	-	-	-
D3D-VLP [93]	✓	✓	4.73	67.2	61.3	56.1	-	-	-	-	5.36	56.9	47.5	34.7
JanusVLN [94]	✓		4.78	65.2	60.5	56.8	6.06	56.2	47.5	62.1	-	-	-	-
InternVLA-N1 [95]	✓	✓	4.83	63.3	58.2	54.0	5.91	53.5	46.1	65.3	-	-	-	-
EfficientVLN [96]	✓		4.18	73.7	64.2	55.9	3.88	67.0	54.3	68.4	-	-	-	-
DualVLN [97]	✓		4.05	70.7	64.3	58.5	4.58	61.4	51.8	70.0	-	-	-	-
Image2Nav (90° FOV)	✓		3.96	72.9	66.3	61.5	4.03	65.1	54.7	70.5	**4.90**	59.2	49.1	37.0
Image2Nav (180° FOV)	✓		**3.71**	**76.1**	**70.3**	**65.6**	**3.74**	**70.7**	**59.1**	**71.8**	5.08	**59.5**	**53.7**	**42.7**

4.3 Cross-Simulator Zero-Shot Navigation Generalization

To assess the quality of the navigation data generated by Image2Sim, we evaluate Image2Nav on the R2R-CE [11], RxR-CE [12], and REVERIE-CE [13, 91] benchmarks (Table 2). All compared baselines are trained and evaluated within the same Habitat simulator [18]. In contrast, Image2Nav uses transferred R2R/RxR trajectory-language annotations together with Image2Sim-generated data, with all training observations rendered and all actions and DAgger rollouts executed in Image2Sim. It is evaluated *zero-shot* in Habitat, without Habitat-rendered training observations or Habitat fine-tuning. This setting directly tests whether Image2Sim-generated data transfers across simulator domains rather than only improving in-domain performance.

Despite the substantial domain shift, Image2Nav achieves strong results against in-domain methods such as EfficientVLN and DualVLN; the 180° setting further improves performance. Improvements across path-following tasks and object-centric REVERIE-CE support the usefulness of the generated supervision for both instruction following and goal-oriented navigation.

4.4 Controlled Analysis of Rendering and Data Diversity

The benchmark methods differ in architecture, data, and field of view. We therefore use controlled experiments on R2R-CE Val Unseen to examine simulator quality and data diversity separately.

Navigation with different simulators. To assess the effect of simulator quality at a fixed training-data scale, we train Image2Nav using the same R2R-CE and RxR-CE training annotations across all three simulators. All training uses one epoch of behavior cloning followed by 1,000 DAgger steps on 8 H20 GPUs. The resulting policies are evaluated in native Habitat on R2R-CE Val Unseen. Image2Sim achieves 46.1 SR, nearly matching native Habitat [18] (46.3) and exceeding Habitat-GS [98] with AnySplat [52] rendering (44.3; Table 3). The GS baseline uses the native Habitat NavMesh, so its lower score is not due to weaker navigation geometry. These results support gains from simulation quality at a fixed data scale.

Table 3: Navigation performance on different simulators.

Simulator	NE↓	OSR↑	SR↑	SPL↑
Image2Sim	5.61	53.5	46.1	41.7
Native Habitat [18]	5.36	54.3	46.3	42.8
Habitat-GS/AnySplat [98, 52]	5.65	51.0	44.3	39.1

Table 4: Scene and trajectory diversity on R2R-CE Val Unseen. Behavior cloning only.

Scenes	Traj.	Epochs	NE↓	OSR↑	SR↑	SPL↑
5,000	0.5M	1	5.26	59.8	54.3	47.9
1,000	0.5M	1	5.45	56.7	51.7	42.3
5,000	0.1M	1	5.67	52.3	47.9	38.4
5,000	0.1M	5	5.58	53.4	46.1	38.2

Scene diversity versus trajectory coverage. We next fix Image2Sim and Image2Nav and vary RealSee3D-Synthesis data, using only behavior cloning on 8 H20 GPUs (Table 4). First, fixing 0.5M trajectories and increasing scenes from 1,000 to 5,000 raises SR from 51.7 to 54.3. Second, fixing 5,000 scenes and increasing trajectories from 0.1M to 0.5M raises SR from 47.9 to 54.3. Both comparisons use one epoch. Repeating 0.1M trajectories for five epochs processes the same number of training examples but yields 46.1 SR. Thus, broader scenes and richer trajectory/instruction coverage both help; repetition alone does not recover these gains (Appendix E.2).

Robustness to estimated geometry. Using the same schedule as the simulator comparison, replacing ground-truth poses and depth with Argus [99] estimates from RGB panoramas reduces Image2Sim SR from 46.1 to 44.5 (1.6 points), compared with 43.2 to 40.1 (3.1 points) for direct AnySplat [52] rendering. Appendix E.3 gives the full comparison and capture-coverage limitations.

4.5 Scaling Trends for Navigation Training Data

A central goal of Image2Sim is to alleviate the environmental data bottleneck in embodied navigation. As shown in Table 5 and Figure 5, we transfer human-annotated R2R and RxR trajectories and instructions into Image2Sim, then add generated samples. Scaling to 10M samples improves SR from 46.1 to 66.3 and SPL from 41.3 to 61.5. As the training budget grows, Image2Sim jointly expands scenes, trajectories, observations, and instructions, exposing the policy to broader layout, visual, action, and language distributions. The results show an empirical scaling trend over the evaluated budgets, rather than a fitted universal scaling law.

Table 5: R2R-CE results with increasing generated data. Every row includes R2R and RxR.

Training data	NE↓	OSR↑	SR↑	SPL↑
R2R, RxR	5.53	53.8	46.1	41.3
+ Image2Sim-**1M**	4.87	65.4	56.9	49.5
+ Image2Sim-**5M**	4.18	70.0	63.8	55.9
+ Image2Sim-**10M**	**3.96**	**72.9**	**66.3**	**61.5**

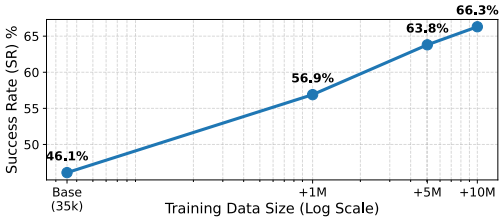


Figure 5: Navigation performance across training-data budgets.

4.6 Ablation Study

Table 6 evaluates the components of our pixel-flow renderer. Using only 3DGS projections severely degrades quality, showing the need for generative completion of sparse reconstructions. Removing semantic features weakens completion in unobserved regions. Without alpha-gated fusion, reliable projected evidence is no longer explicitly separated from uncertain regions, reducing fidelity. Removing momentum-based self-distillation also lowers quality, supporting the EMA teacher's role in stabilizing one-step generation.

Table 6: Ablation study of the pixel-flow model.

Settings	LPIPS ↓	PSNR ↑	SSIM ↑	FPS ↑
w/o pixel flow model	0.518	17.75	0.438	**80.3**
w/o semantic features	0.449	18.36	0.459	42.9
w/o alpha-gated fusion	0.447	18.57	0.506	37.5
w/o self-distillation	0.453	19.84	0.518	37.5
All settings	**0.428**	**20.15**	**0.521**	37.5

Table 7: Real-world navigation success counts.

Methods	Path following		Goal oriented	
	OSR	SR	OSR	SR
JanusVLN [94]	12/20	8/20	5/20	4/20
DualVLN [97]	13/20	8/20	8/20	5/20
Image2Nav	**14/20**	**11/20**	**13/20**	**9/20**

4.7 Real-World Navigation Evaluation

Real-world experiments use a Hello Robot Stretch 3 with inference on an RTX 4090 workstation. Each method is evaluated on the same 20 path-following and 20 goal-oriented trials across a living room, kitchen, meeting room, and office (Table 7). JanusVLN [94] and DualVLN [97] use official checkpoints; all methods share the robot, sensor, and action interface, with no real-world fine-tuning. Image2Nav improves path-following SR from 8/20 to 11/20 and goal-oriented SR from 5/20 to 9/20 over the strongest baselines. These results support zero-shot transfer, particularly for goal-oriented navigation, although the small trial count limits conclusions about deployment reliability. Appendix E.4 provides the setup and baseline training data.

5 Conclusion

We introduce Image2Sim, a generative neural simulator that converts passive visual captures into scalable, physically grounded, and interactive environments for embodied navigation. By combining feed-forward 3D feature-Gaussian reconstruction, geometry-aware one-step panoramic RGB-D generation, collision-aware motion simulation, and automated instruction generation, Image2Sim enables large-scale synthesis of vision-language-action data without manual scene modeling or trajectory annotation. Experiments show that navigation models trained with Image2Sim achieve strong benchmark performance and improved cross-simulator and real-world generalization, suggesting neural simulation as a practical foundation for scaling embodied navigation.

Limitations. Several limitations remain. First, to support real-time interaction, low computational cost, and online policy training, our compact renderer trades off model capacity and generation ability. Second, the simulator mainly handles navigation-level physical validity; richer contact dynamics, movable objects, and human–robot interactions are not yet supported. Third, reconstruction errors can affect collision geometry, and completely unobserved regions are not guaranteed to be consistent across rendered views. Finally, VLM-based instruction annotation enables scale but may introduce linguistic bias or occasional semantic mismatch.

Acknowledgments and Disclosure of Funding

This research / project is supported by the National Research Foundation (NRF) Singapore, under its NRF-Investigatorship Programme (Award ID. NRF-NRFI09-0008), and the Tier 2 grant MOET2EP20124-0015 from the Singapore Ministry of Education. We gratefully acknowledge Yujun Shen for insightful suggestions and help. We also thank the Realsee Team for providing access to the RealSee3D dataset.

Competing interests. The authors declare no additional financial relationships or competing interests relevant to this work during the 36 months prior to submission.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[3] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[4] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[5] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[6] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025.

[7] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[8] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[9] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[10] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.

[11] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*, pages 104–120. Springer, 2020.

[12] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-Across-Room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2020.

[13] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991, 2020.

[14] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33:4247–4258, 2020.

[15] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *International Conference on 3D Vision (3DV)*, 2017.

[16] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[17] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9068–9079, 2018.

[18] 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.

[19] 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. *Advances in Neural Information Processing Systems*, 35:5982–5994, 2022.

[20] Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16384–16393, 2024.

[21] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[22] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[23] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

[24] Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15791–15801, 2025.

[25] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[26] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[27] Liuyi Wang, Xinyuan Xia, Hui Zhao, Hanqing Wang, Tai Wang, Yilun Chen, Chengju Liu, Qijun Chen, and Jiangmiao Pang. Rethinking the embodied gap in vision-and-language navigation: A holistic study of physical and visual disparities. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9455–9465, 2025.

[28] Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Munoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

[29] Sihao Lin, Zerui Li, Xunyi Zhao, Gengze Zhou, Liuyi Wang, Rong Wei, Rui Tang, Juncheng Li, Hanqing Wang, Jiangmiao Pang, et al. Vlnverse: A benchmark for vision-language navigation with versatile, embodied, realistic simulation and evaluation. *arXiv preprint arXiv:2512.19021*, 2025.

[30] Kiana Ehsani, Tanmay Gupta, Rose Hendrix, Jordi Salvador, Luca Weihs, Kuo-Hao Zeng, Kunal Pratap Singh, Yejin Kim, Winson Han, Alvaro Herrasti, et al. Spoc: Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16238–16250, 2024.

[31] Yejin Kim, Wilbert Pumacay, Omar Rayyan, Max Argus, Winson Han, Eli VanderBilt, Jordi Salvador, Abhay Deshpande, Rose Hendrix, Snehal Jauhri, et al. Molmospaces: A large-scale open ecosystem for robot navigation and manipulation. *arXiv preprint arXiv:2602.11337*, 2026.

[32] Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16227–16237, 2024.

[33] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

[34] 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. *Communications of the ACM*, 65(1):99–106, 2021.

[35] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[36] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21676–21685, 2024.

[37] Ziyang Xie, Zhizheng Liu, Zhenghao Peng, Wayne Wu, and Bolei Zhou. Vid2sim: Realistic and interactive simulation from video for urban navigation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1581–1591, 2025.

[38] Jiahang Liu, Yuanxing Duan, Jiazhao Zhang, Minghan Li, Shaoan Wang, Zhizheng Zhang, and He Wang. Navgsim: High-fidelity gaussian splatting simulator for large-scale navigation. *arXiv preprint arXiv:2603.15186*, 2026.

[39] Bingchen Miao, Rong Wei, Zhiqi Ge, Shiqi Gao, Jingzhe Zhu, Renhan Wang, Siliang Tang, Jun Xiao, Rui Tang, Juncheng Li, et al. Towards physically executable 3d gaussian for embodied navigation. *arXiv preprint arXiv:2510.21307*, 2025.

[40] Gunjan Chhablani, Xiaomeng Ye, Muhammad Zubair Irshad, and Zsolt Kira. Embodiedsplat: Personalized real-to-sim-to-real navigation with gaussian splats from a mobile device. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 25431–25441, 2025.

[41] Xinhao Liu, Jiaqi Li, Youming Deng, Ruxin Chen, Yingjia Zhang, Yifei Ma, Li Guo, Yiming Li, Jing Zhang, and Chen Feng. Wanderland: Geometrically grounded simulation for open-world embodied ai. *arXiv preprint arXiv:2511.20620*, 2025.

[42] Seungyeon Yoo, Youngseok Jang, Dabin Kim, Youngsoo Han, Seungwoo Jung, and H Jin Kim. Ready-go: Real-to-sim dynamic 3d gaussian splatting simulation for environment-specific visual navigation with moving obstacles. *arXiv preprint arXiv:2602.11575*, 2026.

[43] Alejandro Escontrela, Justin Kerr, Arthur Allshire, Jonas Frey, Rocky Duan, Carmelo Sferrazza, and Pieter Abbeel. Gaussgym: An open-source real-to-sim framework for learning locomotion from pixels. *arXiv preprint arXiv:2510.15352*, 2025.

[44] Seonghwan Sim, Yeji Kim, and Sung Soo Hwang. Floor flattening of image-based 3d reconstruction for mobile robots. In *2026 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*, pages 441–446. IEEE, 2026.

[45] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19457–19467, 2024.

[46] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European conference on computer vision*, pages 370–386. Springer, 2024.

[47] Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. Freesplat: Generalizable 3d gaussian splatting towards free view synthesis of indoor scenes. *Advances in Neural Information Processing Systems*, 37:107326–107349, 2024.

[48] Botao Ye, Boqi Chen, Haofei Xu, Daniel Barath, and Marc Pollefeys. Yonosplat: You only need one model for feedforward 3d gaussian splatting. *arXiv preprint arXiv:2511.07321*, 2025.

[49] Suyoung Lee, Jaeyoung Chung, Kihoon Kim, Jaeyoo Huh, Gunhee Lee, Minsoo Lee, and Kyoung Mu Lee. Omnisplat: Taming feed-forward 3d gaussian splatting for omnidirectional images with editable capabilities. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16356–16365, 2025.

[50] Zheng Chen, Chenming Wu, Zhelun Shen, Chen Zhao, Weicai Ye, Haocheng Feng, Errui Ding, and Song-Hai Zhang. Splatter-360: Generalizable 360 gaussian splatting for wide-baseline panoramic images. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21590–21599, 2025.

[51] Chen Ziwen, Hao Tan, Peng Wang, Zexiang Xu, and Li Fuxin. Long-lrm++: Preserving fine details in feed-forward wide-coverage reconstruction. *arXiv preprint arXiv:2512.10267*, 2025.

[52] Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *ACM Transactions on Graphics (TOG)*, 44(6):1–16, 2025.

[53] Jie Chen, Yuxin Cai, Yizhuo Wang, Ruofei Bai, Yuhong Cao, Jun Li, Yau Wei Yun, and Guillaume Sartoretti. Imaginav: Scalable embodied navigation via generative visual prediction and inverse dynamics. *arXiv preprint arXiv:2603.13833*, 2026.

[54] Fengda Zhu, Xiwen Liang, Yi Zhu, Qizhi Yu, Xiaojun Chang, and Xiaodan Liang. Soon: Scenario oriented object navigation with graph-based exploration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12689–12699, 2021.

[55] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. *Advances in neural information processing systems*, 31, 2018.

[56] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13137–13146, 2020.

[57] Aishwarya Kamath, Peter Anderson, Su Wang, Jing Yu Koh, Alexander Ku, Austin Waters, Yinfei Yang, Jason Baldridge, and Zarana Parekh. A new path: Scaling vision-and-language navigation with synthetic instructions and imitation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10813–10823, 2023.

[58] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Learning from unlabeled 3d environments for vision-and-language navigation. In *European Conference on Computer Vision*, pages 638–655. Springer, 2022.

[59] Zun Wang, Jialu Li, Yicong Hong, Yi Wang, Qi Wu, Mohit Bansal, Stephen Gould, Hao Tan, and Yu Qiao. Scaling data generation in vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12009–12020, 2023.

[60] Zun Wang, Jialu Li, Yicong Hong, Songze Li, Kunchang Li, Shoubin Yu, Yi Wang, Yu Qiao, Yali Wang, Mohit Bansal, et al. Bootstrapping language-guided navigation learning with self-refining data flywheel. *arXiv preprint arXiv:2412.08467*, 2024.

[61] Zihan Wang, Yaohui Zhu, Gim Hee Lee, and Yachun Fan. Navrag: Generating user demand instructions for embodied navigation through retrieval-augmented llm. *arXiv preprint arXiv:2502.11142*, 2025.

[62] Songze Li, Zun Wang, Gengze Zhou, Jialu Li, Xiangyu Zeng, Limin Wang, Yu Qiao, Qi Wu, Mohit Bansal, and Yi Wang. Learning goal-oriented language-guided navigation with self-improving demonstrations at scale. *arXiv preprint arXiv:2509.24910*, 2025.

[63] Jiazhao Zhang, Anqi Li, Yunpeng Qi, Minghan Li, Jiahang Liu, Shaoan Wang, Haoran Liu, Gengze Zhou, Yuze Wu, Xingxing Li, et al. Embodied navigation foundation model. *arXiv preprint arXiv:2509.12129*, 2025.

[64] Shuo Wang, Yucheng Wang, Guoxin Lian, Yongcai Wang, Maiyue Chen, Kaihui Wang, Bo Zhang, Zhizhong Su, Yutian Zhou, Wanting Li, et al. Progress-think: Semantic progress reasoning for vision-language navigation. *arXiv preprint arXiv:2511.17097*, 2025.

[65] Gengze Zhou, Yicong Hong, Zun Wang, Xin Eric Wang, and Qi Wu. Navgpt-2: Unleashing navigational reasoning capability for large vision-language models. *arXiv preprint arXiv:2407.12366*, 2024.

[66] Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. *arXiv preprint arXiv:2507.13347*, 2025.

[67] Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

[68] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2337–2346, 2019.

[69] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[70] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

[71] Yiyang Lu, Susie Lu, Qiao Sun, Hanhong Zhao, Zhicheng Jiang, Xianbang Wang, Tianhong Li, Zhengyang Geng, and Kaiming He. One-step latent-free image generation with pixel mean flows. *arXiv preprint arXiv:2601.22158*, 2026.

[72] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[73] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[74] Steve Macenski, Francisco Martín, Ruffin White, and Jonatan Ginés Clavero. The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2718–2725. IEEE, 2020.

[75] Steve Macenski, Tom Moore, David V Lu, Alexey Merzlyakov, and Michael Ferguson. From the desks of ros maintainers: A survey of modern & capable mobile robotics algorithms in the robot operating system 2. *Robotics and Autonomous Systems*, 168:104493, 2023.

[76] Steve Macenski, Shrijit Singh, Francisco Martín, and Jonatan Ginés. Regulated pure pursuit for robot path tracking. *Autonomous Robots*, 47(6):685–694, 2023.

[77] Linyuan Li, Yan Wu, Xi Li, Lingli Wang, Tong Rao, Jie Zhou, Cihui Pan, and Xinchen Hui. Realsee3d: A large-scale multi-view rgb-d dataset of indoor scenes (version 1.0), 2025. URL `https://doi.org/10.5281/zenodo.17826243`.

[78] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3d: A large photo-realistic dataset for structured 3d modeling. In *Proceedings of The European Conference on Computer Vision (ECCV)*, 2020.

[79] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Yuri Feigin, Peter Fu, Thomas Gebauer, Daniel Kurz, Tal Dimry, Brandon Joffe, Arik Schwartz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

[80] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[81] Manuel Rey, Mingze Yuan Area, and Christian Richardt. 360monodepth: High-resolution 360 monocular depth estimation. in 2022 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2022.

[82] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3505–3506, 2020.

[83] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[84] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Think global, act local: Dual-scale graph transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16537–16547, 2022.

[85] Jing Yu Koh, Harsh Agrawal, Dhruv Batra, Richard Tucker, Austin Waters, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson. Simple and effective synthesis of indoor 3d scenes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 1169–1178, 2023.

[86] Haoran Feng, Dizhe Zhang, Xiangtai Li, Bo Du, and Lu Qi. Dit360: High-fidelity panoramic image generation via hybrid training. *arXiv preprint arXiv:2510.11712*, 2025.

[87] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4): 600–612, 2004.

[88] Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[89] Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-navid: A video-based vision-language-action model for unifying embodied navigation tasks. *arXiv preprint arXiv:2412.06224*, 2024.

[90] An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Xueyan Zou, Jan Kautz, Erdem Biyik, Hongxu Yin, Sifei Liu, and Xiaolong Wang. Navila: Legged robot vision-language-action model for navigation. In *RSS*, 2025.

[91] Zihan Wang, Seungjun Lee, and Gim Hee Lee. Dynam3d: Dynamic layered 3d tokens empower vlm for vision-and-language navigation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[92] Meng Wei, Chenyang Wan, Xiqian Yu, Tai Wang, Yuqiang Yang, Xiaohan Mao, Chenming Zhu, Wenzhe Cai, Hanqing Wang, Yilun Chen, et al. Streamvln: Streaming vision-and-language navigation via slowfast context modeling. *arXiv preprint arXiv:2507.05240*, 2025.

[93] Zihan Wang, Seungjun Lee, Guangzhao Dai, and Gim Hee Lee. D3d-vlp: Dynamic 3d vision-language-planning model for embodied grounding and navigation. *arXiv preprint arXiv:2512.12622*, 2025.

[94] Shuang Zeng, Dekang Qi, Xinyuan Chang, Feng Xiong, Shichao Xie, Xiaolong Wu, Shiyi Liang, Mu Xu, Xing Wei, and Ning Guo. Janusvln: Decoupling semantics and spatiality with dual implicit memory for vision-language navigation. *arXiv preprint arXiv:2509.22548*, 2025.

[95] Intern Robotics. Internvla-n1: An open dual-system vision-language navigation foundation model with learned latent plans. 2025.

[96] Duo Zheng, Shijia Huang, Yanyang Li, and Liwei Wang. Efficient-vln: A training-efficient vision-language navigation model. *arXiv preprint arXiv:2512.10310*, 2025.

[97] Meng Wei, Chenyang Wan, Jiaqi Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, et al. Ground slow, move fast: A dual-system foundation model for generalizable vision-and-language navigation. *arXiv preprint arXiv:2512.08186*, 2025.

[98] Ziyuan Xia, Jingyi Xu, Chong Cui, Yuanhong Yu, Jiazhao Zhang, Qingsong Yan, Tao Ni, Junbo Chen, Xiaowei Zhou, Hujun Bao, Ruizhen Hu, and Sida Peng. Habitat-GS: A High-Fidelity Navigation Simulator with Dynamic Gaussian Splatting. *arXiv preprint arXiv:2604.12626*, 2026.

[99] Xi Li, Linyuan Li, Yan Wu, Tong Rao, Kai Zhang, Xinchen Hui, and Cihui Pan. Argus: Metric Panoramic 3D Reconstruction for Indoor Scenes. In *European Conference on Computer Vision*, pages 628–647. Springer, 2026.

[100] Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, Shuran Song, and Fangyin Wei. SAGE: Scalable Agentic 3D Scene Generation for Embodied AI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22358–22368, 2026.

[101] Jay Zhangjie Wu, Yuxuan Zhang, Haithem Turki, Xuanchi Ren, Jun Gao, Mike Zheng Shou, Sanja Fidler, Zan Gojcic, and Huan Ling. DIFIX3D+: Improving 3D Reconstructions with Single-Step Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26024–26035, 2025.

[102] Yuxuan Zhang, Katarína Tóthová, Zian Wang, Kangxue Yin, Haithem Turki, Riccardo de Lutio, Yen-Yu Chang, Or Litany, Sanja Fidler, and Zan Gojcic. DiffusionHarmonizer: Bridging Neural Reconstruction and Photorealistic Simulation with Online Diffusion Enhancer. *arXiv preprint arXiv:2602.24096*, 2026.

A Data Analysis and Statistics

We summarize the source datasets and the generated navigation data below.

A.1 Environment Dataset Statistics

To build interactive 3D environments for Image2Sim, we aggregate diverse real-world and synthetic datasets. Table 8 summarizes their key statistics, including scenes, panoramic images, pinhole images, and navigable area. Overall, Image2Sim ingests 19,936 scenes with over 617K panoramas, 1.16M pinhole images, and nearly 1.2M m^2 of navigable space. **To prevent data leakage, we exclude 29 validation and test scenes from the 90 Matterport3D scenes during navigation training.**

Table 8: Statistics of the source corpus before navigation train/evaluation split filtering.

Dataset Name	Scenes	Pano Images	Pinhole Images	Navigable Area (m^2)
RealSee3D-Real	1,000	24,264	0	75,198.90
RealSee3D-Synthesis	9,000	274,790	0	727,493.97
Structured3D	3,496	21,812	78,461	155,112.76
ScanNet	1,613	0	132,418	7,289.50
ARKitScenes	3,345	0	957,701	18,492.32
Matterport3D	90	9,684	0	31,209.51
HM3D	900	176,917	0	105,900.34
Gibson	492	109,918	0	75,418.18
TOTAL	**19,936**	**617,385**	**1,168,580**	**1,196,115.48**

A.2 Generated Navigation Data Statistics

Based on the constructed scenes, our automated data engine synthesized over 10 million vision-language-action training samples. The generated navigation instructions cover a diverse range of text lengths and styles, effectively mitigating instruction bias. Specifically, the instruction styles are distributed evenly across three categories:

- **R2R-style (Path-following):** 3,454,136 samples (33.42%)
- **REVERIE-style (Object-oriented):** 3,461,321 samples (33.49%)
- **Demand-style (Human-centric):** 3,419,752 samples (33.09%)

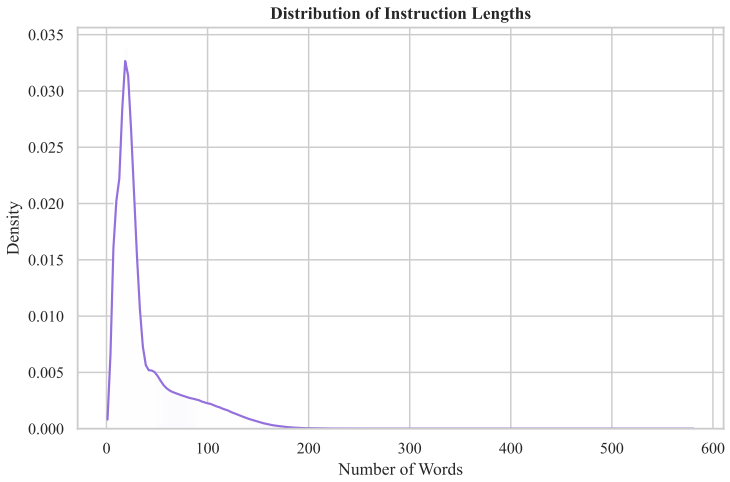


Figure 6: Distribution of instruction lengths (number of words) in the generated dataset.

As illustrated in Figure 6, the length of the generated instructions ranges from 1 to 581 words. The distribution exhibits a mean of 38.1 words and a median of 24.0 words, which reflects a healthy mix of concise commands and detailed navigational guidance.

Figure 7 and Figure 8 present the distributions of the generated physical trajectories. The trajectory lengths exhibit a mean of 8.53 meters and a median of 8.50 meters, with a standard deviation of 3.86 meters. The overall travel distances range from a minimum of 0.37 meters to a maximum of 85.72 meters to ensure the navigation policy learns both local target approaches and long-horizon exploration. In terms of agent actions, the trajectories require a mean of 54.7 steps and a median of 53.0 steps to complete, bounded by a maximum of 501 steps.

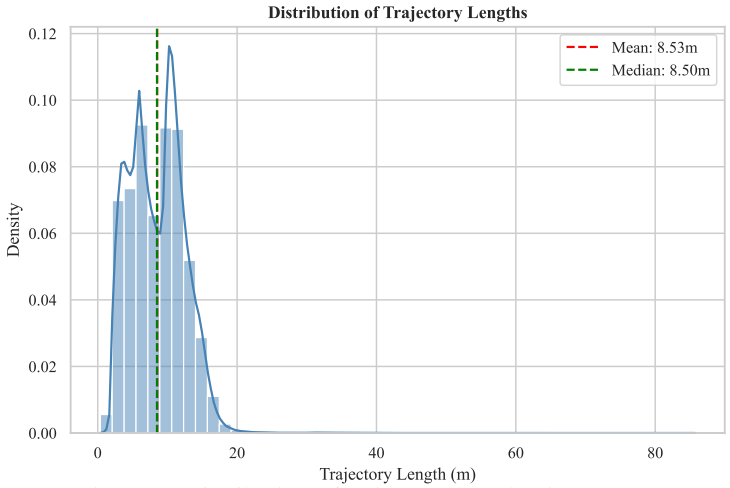


Figure 7: Distribution of trajectory lengths (in meters).

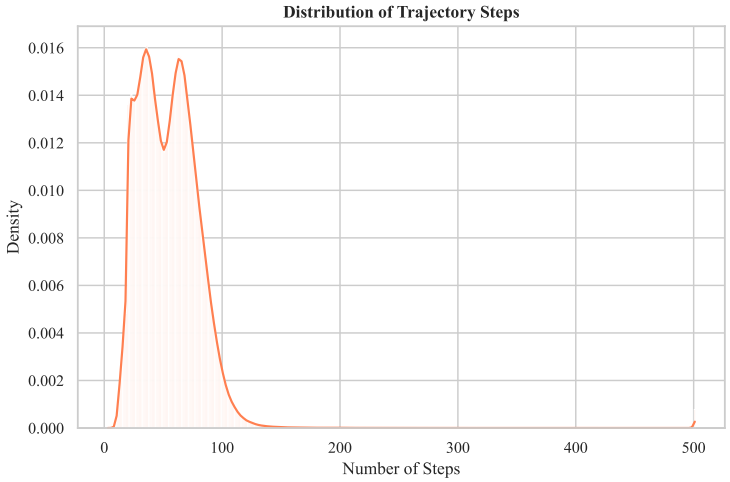


Figure 8: Distribution of trajectory steps (number of discrete actions).

B Model Architecture and Training

Figure 9 summarizes the three stages of the renderer: scene encoding, projection and semantic alignment, and pixel-flow rendering. We describe each stage and the training configuration below.

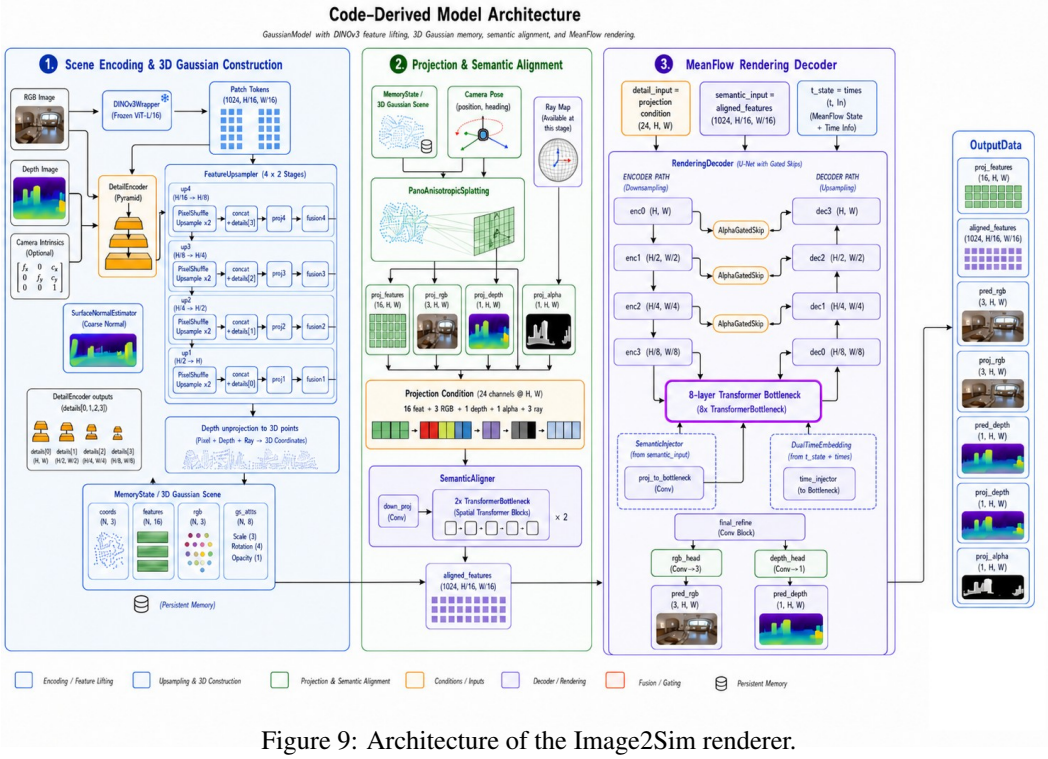


Figure 9: Architecture of the Image2Sim renderer.

B.1 Stage 1: Scene Encoding & 3D Gaussian Construction

The first stage constructs a persistent 3D Gaussian scene from input visual captures in a feed-forward manner without per-scene optimization.

Input Representation and Format-Agnostic Ray Directions. The initial inputs include RGB images, metric depth, and explicitly computed camera ray directions to ensure the `FeatureUpsampler` is robust to different camera geometries. The `get_ray_map` function dynamically generates these ray directions based on the input type:

- **Panoramic Inputs:** Operates on an equirectangular grid where the pixel coordinates are mapped to longitude $\phi \in [-\pi, \pi]$ and latitude $\theta \in [-\pi/2, \pi/2]$. The resulting unit vectors are computed as $(x, y, z) = (\cos\theta \sin\phi, \cos\theta \cos\phi, \sin\theta)$.
- **Pinhole Inputs:** Utilizes provided intrinsic matrices (focal lengths f_x, f_y and principal points c_x, c_y) to compute Z-forward anti-projections: $x = (u - c_x)/f_x, y = (v - c_y)/f_y, z = 1$, which are then L_2-normalized into direction vectors.

Surface Normal Estimation. To provide geometric priors while suppressing high-frequency noise from raw depth sensors, a `SurfaceNormalEstimator` computes coarse surface normals. The depth map D is first smoothed via a 3×3 average pooling. Normals are then extracted using 5×5 Sobel filters (S_x, S_y) scaled by $1/48$, which provides a larger receptive field than standard 3×3 gradients. A constant Z-bias of 0.5 is appended before L_2-normalization to suppress lateral random noise.

Dual-Stream Feature Extraction and Fusion. The visual and geometric contexts are extracted via a dual-stream design:

- **Semantic Stream:** A frozen `DINOv3Wrapper` (ViT-L/16) extracts rich semantic patch tokens at a resolution $H/16$ with 1024 channels.
- **Detail Stream:** A `DetailEncoder` processes the 10-channel concatenated input (RGB, depth, normals, ray maps) to build a spatial feature pyramid at $\{H, H/2, H/4, H/8\}$ resolutions.

The `FeatureUpsampler` fuses these streams across four stages. Each stage utilizes a `PixelShuffleUpsample` operation and fuses features using a `PanoBlock` (incorporating depth-wise Pano-convolutions and ConvNeXt-V2-style inverted bottlenecks with Global Response Normalization). The network maps these into 16-dimensional scene features and 8-dimensional 3D Gaussian attributes. Rotation quaternions q are factorized into a base rotation q_{base} (aligned to the surface normal) and a learned residual q_{res} to ensure stable initialization. These attributes are unprojected into a persistent `MemoryState`.

B.2 Stage 2: Projection & Semantic Alignment

Given a target camera pose, the 3D representation is mapped back into the 2D observation space via `PanoAnisotropicSplatting`, producing a set of raw projections: 16-channel features, 3-channel RGB, 1-channel depth, and a 1-channel alpha (opacity) map.

Semantic Aligner. To ensure strict semantic fidelity, a `SemanticAligner` processes a 24-channel condition tensor (splatted features, RGB, depth, alpha, and target ray map). It applies strided `PanoConv2d` layers to downsample the spatial resolution to $H/16$ and refines the tokens using two `TransformerBottleneck` blocks, generating aligned contextual tokens (1024 channels).

B.3 Stage 3: Pixel Flow Rendering Decoder

The `RenderingDecoder` synthesizes the photorealistic panoramic RGB-D observation using a continuous-time MeanFlow formulation.

Encoder Path and Deep Bottleneck. The projection condition is combined with the noisy flow state z_t and downsampled to a resolution of $H/16$. Here, an 8-layer `TransformerBottleneck` infers global scene structures.

- **Semantic Injection:** A `SemanticInjector` utilizes SPADE-like spatial modulation to infuse DINOv3-aligned tokens into the bottleneck. A simple SiLU and 1×1 convolution predict scale (γ) and shift (β) parameters, applied as $F'_{spatial} = \text{LayerNorm}(F_{spatial}) \odot (1 + \gamma) + \beta$.
- **Temporal Conditioning:** A `DualTimeEmbedding` module encodes both the absolute integration time t and step size $h = t - r$ using sinusoidal embeddings. These are fused via an MLP to modulate the bottleneck features.

Alpha-Gated Skip Connections. The decoder upsamples features across four stages. To prevent noisy 3DGS projections from corrupting generated regions, standard U-Net skips are replaced with `AlphaGatedSkip` modules. An `AlphaPyramid` generates multi-scale alpha masks (using 2×2 average pooling) to guide each decoder level. The gating mechanism works as follows:

- **Adapter:** Encoder features F_{enc} are aligned to the decoder space using a 1×1 `PanoConv2d`, LayerNorm, and GELU activation to get F'_{enc}.
- **Mixer:** A gate g is predicted by concatenating the decoder feature F_{dec}, adapted encoder feature F'_{enc}, and the corresponding multi-scale alpha mask α:

$$g = \sigma(W_{gate}[F_{dec} \oplus F'_{enc} \oplus \alpha]) \tag{9}$$

- **Zero-Initialization Prior:** Crucially, the 1×1 convolution W_{gate} is zero-initialized with a constant positive bias of 2.0. This yields an initial gate value of $\sigma(2.0) \approx 0.88$, which encourages the network to strongly trust the 3DGS projection during early training to accelerate convergence. The decoder generation branch $(1 - g)$ is only emphasized in regions where the projection is unreliable.

The final output of the skip connection is computed as $F_{out} = g \odot F'_{enc} + (1 - g) \odot F_{dec}$.

Prediction Heads. The network branches into an RGB Head (Sigmoid activation) and a Depth Head. The Depth Head explicitly uses a ReLU activation instead of Sigmoid to accommodate unconstrained metric depth ranges and prevent gradient vanishing in expansive environments.

B.4 Hyperparameters and Loss Weights

The Image2Sim neural simulator is trained using the AdamW optimizer with an initial learning rate of 1×10^{-4} and a weight decay of 1×10^{-4}. The learning rate is decayed following a Cosine Annealing schedule down to a minimum of 1×10^{-6} over a total of $400K$ training iterations. The model is trained using Distributed Data Parallel (DDP) with a per-GPU batch size of 2.

We adopt a three-stage curriculum learning strategy to stabilize the one-step rendering and distillation process. The loss weights dynamically adjust across these stages to prioritize different learning objectives:

- **Stage 1: Geometry Initialization (0 - 100K iterations).** The network focuses on grounding the 3D representation. The self-distillation is disabled. The perceptual loss weight λ_{perc} is set to 1.0.
- **Stage 2: Flow and Distillation (100K - 300K iterations).** The EMA teacher is introduced (decay rate $\gamma = 0.999$) to guide the student. The distillation loss weight $\lambda_{distill}$ is set to 2.0. To enhance visual fidelity, the perceptual loss weight λ_{perc} is increased to 2.0, and an adaptive spatial focus weight is applied to the flow loss to emphasize unobserved regions.
- **Stage 3: High-Frequency Refinement (300K - 400K iterations).** The encoder backbone, feature upsampler, semantic aligner, and the deep transformer bottleneck are completely frozen. The network purely optimizes the decoder to sculpt high-frequency pixel details. The perceptual loss weight λ_{perc} is further increased to 4.0.

For the base loss terms throughout all stages, the reconstruction loss weight λ_{rec} and semantic alignment loss weight λ_{align} are both set to 0.1. The flow loss λ_{flow} incorporates a saturation penalty term with a coefficient of 0.2 to prevent color degradation.

C Motion Simulation and Online Supervision

The motion engine extracts navigable space from input geometry, constructs a connectivity graph, and executes collision-aware kinematic actions (Figure 10).

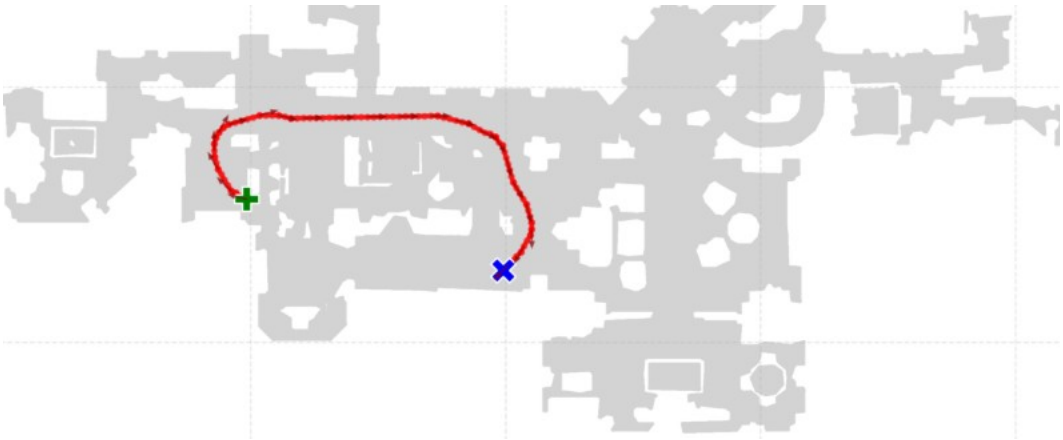


Figure 10: Visualization of navigable voxels and path planning.

C.1 Navigable Area Extraction and Denoising

Before path planning, we extract valid navigable regions from the raw RGB-D observations. We employ Mask2Former (Swin-Large) to segment semantic navigable areas (e.g., floors, rugs, sidewalks,

and paths) alongside door masks. To ensure geometric validity and eliminate hallucinations, we process the extracted point cloud through a rigorous cleaning pipeline:

- **Geometric Filtering:** The point cloud is downsampled to a 0.05m voxel size. We estimate surface normals using a 5×5 Sobel operator and filter out points with a Z-component (vertical axis) smaller than 0.5, which effectively removes vertical walls that leaked into the semantic mask.
- **RANSAC Calibration:** We apply RANSAC plane fitting to robustly identify the ground plane and calibrate the scene's coordinate system to ensure the Z-axis strictly aligns with gravity.
- **Obstacle Denoising and Adaptive Filling:** To prevent noisy obstacle points from digging artificial holes in the floor, we apply radius outlier removal. The remaining valid obstacles project a 2D safety buffer. Finally, we project the ground points into a 2D grid (0.05m resolution) and apply morphological closing to adaptively fill holes caused by occlusions or depth sensor failures, re-projecting the filled regions back to 3D using the calibrated plane equation.

C.2 Topology and Geometry-Aware Graph Construction

Using the cleaned navigable point cloud, we construct a dense, traversable voxel connectivity graph $\mathcal{M} = (\mathcal{V}, \mathcal{E})$. First, the scene is voxelized into a 3D grid with a resolution of 0.05m. We compute a 2D clearance map using the Euclidean Distance Transform (EDT) to explicitly measure the distance from any navigable voxel to the nearest obstacle.

To establish locally reachable neighboring states, we perform GPU-parallel ray marching. For each node, a KD-Tree queries candidate neighbors within a 0.3m search radius. We then cast vectorized rays between candidate pairs, where an edge is retained only if the line-of-sight is entirely free of collisions within the 3D occupancy grid.

The path planning relies on a collision-aware cost formulation instead of merely the shortest path. The traversal cost between node i and node j is defined as:

$$cost_{i \to j} = d_{i,j} + w_{safe} \cdot \left(\max \left(0, \frac{r_{safe} - c_j}{r_{safe}} \right) \right)^2 + p_{collision}$$

where $d_{i,j}$ is the physical distance, c_j is the EDT clearance at node j, r_{safe} is a safety margin ($5\times$ the agent radius), and w_{safe} is the safety penalty weight. The term $p_{collision}$ introduces a soft linear penalty for severe penetration. This design encourages the planner to favor safer corridors and avoid tight corners. The resulting sparse graph is solved efficiently using Dijkstra's algorithm. **Specifically, given a 3D starting coordinate and a target destination as inputs, the global planner utilizes this graph to output an optimal collision-free sequence of discrete 3D waypoints as shown in Figure 10**.

C.3 Kinematic Simulation and Closed-Loop Control

The discrete paths generated by the planner are initially smoothed using a 5-point sliding window average to eliminate the zigzag artifacts inherent to voxel grids. We then utilize a lookahead pure-pursuit controller to convert the path into smooth, executable motion primitives.

Our simulator assumes a finite-size agent with a radius of 0.15m and a camera eye height of 1.25m. The discrete action space consists of `MOVE_FORWARD` (0.25m step size), `TURN_LEFT` (15°), `TURN_RIGHT` (15°), and `STOP`, matching standard settings of the Habitat agent. **All these physical and kinematic parameters are fully configurable by the user to accommodate diverse robotic embodiments.** During the physical rollout, the simulator enforces strict collision constraints:

- **Geometric Penetration:** At each micro-step, a local KD-tree query checks for obstacles within the agent footprint. Motions exceeding a maximum step height of 0.15m or violating the radius clearance are rejected.
- **Wall Sliding:** If a direct forward movement is blocked, the simulator computes the center of mass of the obstructing points. Using the cross-product between the heading vector of the agent and the normal of the obstacle, the engine supports physically plausible sliding along obstacle boundaries.

- **Stuck Detection:** We maintain a 4-step sliding window of the state of the agent. If the net spatial displacement is less than 0.1m and the total yaw rotation is less than 15° over 4 steps, the agent is deemed stuck.

C.4 Closed-Loop DAgger

During DAgger, either a policy or oracle action is executed according to the mixing schedule. After each action, collision checking updates the reached position and heading, and the planner recomputes a path and smoothed oracle action from that state. Supervision therefore follows policy-visited states after deviations from the demonstration. This geometry-based execution supports static navigation; contact dynamics and movable objects are outside its scope.

D Instruction Generation and Pipeline Efficiency

To construct a high-quality, large-scale vision-language-action dataset, we develop a fully automated pipeline comprising global object mining, topologically-aware trajectory sampling, and multi-modal instruction annotation.

D.1 Global Object Mining and 3D Grounding

For goal-oriented tasks (e.g., REVERIE and human demands), the agent must locate specific semantic targets. We utilize a pre-trained open-vocabulary detector, YOLO-World, configured with a comprehensive indoor vocabulary. To ensure a broad coverage, the mining process is executed across the navigable area from different viewpoints.

To reliably ground 2D detections into the 3D scene, we apply a filtering mechanism:

- **Adaptive Confidence and Scale:** We relax the confidence threshold to 0.08 but enforce a minimum area of the bounding box of 300 pixels to capture small and long-tail objects while rejecting noise.
- **Depth Validation:** Bounding boxes are projected into 3D using the median depth of the central patch and the camera intrinsic matrix. Detections with a depth greater than 10m are discarded to ensure spatial accuracy.
- **Duplicate Suppression:** A 0.5m Euclidean distance threshold is applied to merge overlapping detections of the same category.

D.2 Topologically-Aware Trajectory Sampling

To prevent the navigation policy from overfitting to specific path lengths or dense clusters of a single scene, we implement a robust trajectory sampling strategy over the pre-built connectivity graph.

- **Dynamic Scene Capacity:** The maximum number of sampled trajectories per scene is dynamically scaled based on the physical navigable area bounded between 50 and 1000 episodes to prevent topological redundancy in small rooms.
- **Length-Bucketed Distribution:** Target paths are explicitly sampled using a roulette wheel selection across three length buckets: short (3-6m, 20%), medium (6-10m, 30%) and long (10-15m, 50%). This ensures that the model learns both local manipulation approaches and long-horizon exploration.
- **Spatial Diversity:** We employ Furthest Point Sampling against historical start and end points within the scene to maximize global spatial coverage.
- **Long-Tail Goal Targeting:** At the destination, the agent queries the mined object database. Target objects are selected using an Inverse Document Frequency (IDF) weighting scheme, which heavily penalizes frequent categories to aggressively sample rare long-tail objects. Valid targets must fall within a strict Euclidean distance (1.5m for manipulable items, 3.0m for large objects) and pass a geodesic distance check to prevent selecting objects behind walls.

Once the start and end points are determined, the optimal path is calculated using our global planner. A physical rollout is then executed using the step function of the simulator. During this rollout,

a 4-step sliding window monitors the agent state. If the accumulated spatial displacement is less than 0.1m and the yaw rotation is less than 15°, the agent is classified as stuck and the trajectory is truncated.

D.3 Multi-Modal Instruction Annotation Pipeline

Directly translating dense kinematic actions (e.g., 0.25m forward steps) into natural language often yields overly rigid instructions. To bridge this granularity gap, our annotation pipeline formulates the task as an interleaved image-text reasoning process using the Qwen3-VL (32B) model.

- **Odometry-Based Semantic Chunking:** We aggregate the high-frequency physical steps into semantic macro-steps based on accumulated odometry. A new macro-step boundary is triggered when the agent accumulates a spatial displacement of 1.0 meters, a yaw rotation of 45°, or executes a terminal `STOP` action.
- **Interleaved Image-Action Prompting:** For the beginning of each macro-step, we extract a 180° ultra-wide field-of-view image from rendered panorama. The accumulated kinematic movement of that step is heuristically mapped into a concise text description (e.g., "Walk forward", "Walk forward while turning right", "Turn left"). The VLM is then prompted with a strictly alternating sequence of observations and movements: `[Image 1]` $\rightarrow$ `[Movement 1]` $\rightarrow$ `[Image 2]` $\cdots \rightarrow$ `[STOP]`.
- **Joint Quality Assurance and Conditioned Generation:** We integrate automated dataset filtering directly into the generation process. Before writing the instruction, the VLM must explicitly output a `REASONING` block evaluating consecutive frames for fatal simulation errors, such as visual freezing (agent stuck) or unnatural clipping through solid walls.
- **Decoupled Prompt Matrix:** If the trajectory passes inspection, the VLM applies a decoupled prompt matrix. The prompt dynamically combines syntax and tone constraints to route the generation into one of three distinct styles: fine-grained R2R-style (sequential and path-following), REVERIE-style (object-centric and passive voice), or human-centric Demand (casual interaction requests). Strict anti-hallucination rules are enforced, which requires every mentioned object to be visually grounded in the provided frames.

D.4 Pipeline Efficiency

Table 9 reports component runtimes. VLM instruction annotation is the dominant offline cost; it is performed once per instruction and can be parallelized. Scene-construction timings assume available posed RGB-D; raw-RGB reconstruction adds external preprocessing. Online policy training is evaluated through DAgger.

Table 9: Data-generation component runtimes, measured separately. Rendering uses an RTX 4090; instruction generation uses one H20.

Component	Runtime
Feature-GS construction	$<$ 5 s / scene
Navigability-map construction	$<$ 5 s / scene
Planning and oracle generation	$<$ 10 ms / trajectory
Panoramic RGB-D rendering	$\sim$ 40 FPS
Qwen3-VL-32B instruction generation	$\sim$ 10 s / instruction on one H20

E Additional Experimental Results and Protocols

E.1 Rendering Baselines and Conditioning

For Table 1, DiT360 [86] and SE3DS [85] take Gaussian-projected images as conditions and output panoramic RGB. AnySplat [52] receives six 90° pinhole RGB-D views projected from the panoramas, exactly as Image2Sim (pinhole). Its Gaussian centers are unprojected from the same metric depth and camera poses; the resulting Gaussians are rendered panoramically. This geometry-adapted baseline controls for input geometry rather than evaluating native unposed-RGB reconstruction.

Table 10 additionally tests SE3DS with its native point-cloud condition. GS conditioning generally improves its results, especially on noisy RealSee3D-Real. Image2Sim achieves higher PSNR and throughput across the three datasets, while SE3DS retains an advantage on some LPIPS/SSIM metrics. The results support a quality–speed tradeoff. DiT360 serves as an image-quality reference at approximately 0.3 FPS.

Table 10: SE3DS [85] conditioning control under the rendering protocol of Table 1. Image2Sim uses panoramic inputs.

Dataset	Method / condition	LPIPS↓	PSNR↑	SSIM↑	FPS↑
RealSee3D-Synthesis	SE3DS / GS	0.379	18.23	0.538	3.4
	SE3DS / point cloud	0.396	17.79	0.541	3.4
	Image2Sim / GS	0.335	23.88	0.591	41.3
Matterport3D	SE3DS / GS	0.426	18.76	0.536	3.3
	SE3DS / point cloud	0.437	18.56	0.522	3.4
	Image2Sim / GS	0.428	20.15	0.521	37.5
RealSee3D-Real	SE3DS / GS	0.421	16.78	0.457	3.4
	SE3DS / point cloud	0.515	14.98	0.436	3.4
	Image2Sim / GS	0.473	17.43	0.470	45.6

E.2 Navigation Data and Diversity Controls

The 90° Image2Nav benchmark and the 10M scaling result use the same model and training mixture: transferred R2R/RxR trajectory-language annotations plus Image2Sim-generated data. All training observations, actions, and rollouts come from Image2Sim. The smaller-budget controls in Section 4.4 are separate runs.

In Table 4, 1,000 scenes with 0.5M trajectories provide about 500 trajectories per scene, whereas 5,000 scenes with 0.1M provide only 20. Their SR difference (51.7 versus 47.9) therefore does not rank the intrinsic importance of scene and trajectory diversity. At a fixed trajectory budget, more scenes improve SR; at a fixed scene count, more unique trajectories outperform repeating the smaller dataset for five epochs. Trajectory and language diversity remain coupled because every trajectory has an instruction.

RealSee3D-Synthesis averages $80.8\,\mathrm{m}^2$ of navigable area per scene, while generated trajectories average 8.53 m and 54.7 actions (Appendix A). Area-dependent scene capacity, length buckets, spatially spread endpoints, and IDF-weighted goals promote coverage (Appendix D.2), explaining why additional trajectories can remain useful within the same scenes.

E.3 RGB-Only Inputs and Capture Coverage

We replace all Matterport3D ground-truth poses and depth with estimates from RGB panoramas using Argus [99], which was not trained on Matterport3D. All four policies use the same R2R/RxR annotations, one behavior-cloning epoch, and 1,000 DAgger steps on 8 H20 GPUs, with evaluation on R2R-CE Val Unseen. Image2Sim loses 1.6 points in both SR and SPL, compared with 3.1 and 4.4 points for direct AnySplat rendering (Table 11).

Table 11: Navigation with ground-truth or RGB-estimated geometry. Direct AnySplat rendering here differs from Habitat-GS/AnySplat with a native NavMesh in Table 3.

Renderer	Pose / depth	NE↓	OSR↑	SR↑	SPL↑
Image2Sim	Ground truth	5.61	53.5	46.1	41.7
Image2Sim	Argus prediction [99]	5.74	51.3	44.5	40.1
AnySplat [52]	Ground truth	5.70	50.8	43.2	38.0
AnySplat	Argus prediction	6.06	46.9	40.1	33.6

Capture coverage. Structured3D averages roughly six panoramas per scene and RealSee3D-Real roughly 24. Sparse coverage leaves occlusions, partially observed objects, low-resolution textures,

and noisy geometry (Figure 4). Image2Sim completes local gaps within captured spaces; it cannot reliably reconstruct an entirely unseen room or an area absent from every input.

E.4 Real-World Deployment

The Stretch 3 carries a head-mounted Intel RealSense D435i; Image2Nav uses its RGB stream. Inference runs on a remote RTX 4090 workstation with 64 GB RAM, connected to the robot over WiFi. Each method uses the same robot, sensor, action interface, and 20 trials per instruction type across a living room, kitchen, meeting room, and office. These objects and layouts are absent from training, and no method receives real-world fine-tuning or environment adaptation.

JanusVLN [94] and DualVLN [97] use official checkpoints. JanusVLN is trained on R2R-CE/RxR-CE, 155K ScaleVLN [59] trajectories (about 9.21M image-action pairs), and 14K DAgger trajectories (about 1.49M pairs) rendered in Habitat. DualVLN uses about 450K expert clips, 240K DAgger samples, and general multimodal data; its InternData-N1 source [95] contains over 0.2M trajectories and 50M egocentric images from more than 3,000 scenes. The comparison thus controls deployment conditions, not training data. Table 7 reports success counts without repeated-seed uncertainty or a measured failure-category breakdown.

F Scope, Consistency, and Related Systems

F.1 Rendering Consistency and Collision Geometry

Opacity-based perturbations and alpha-gated skips preserve reliable projections while concentrating generation in weakly observed regions. Feature-GS semantics, depth, camera rays, semantic alignment, and self-distillation further constrain completion. Every view is conditioned on the same persistent scene, without a temporal generator, allowing arbitrary pose queries. This encourages consistency but cannot guarantee agreement in fully unsupported regions or identical outputs under independently sampled noise.

The navigability map is built from input geometry using outlier filtering, ground/obstacle identification, clearance margins, and morphological closing. Generated RGB-D is never fused into this map: a hallucinated opening cannot create free space or alter collision boundaries. It can nevertheless mislead the policy or instruction annotator, and reconstruction or geometric-filling errors can affect the map itself.

F.2 Comparison with Other Simulation Pipelines

Table 12 compares reported scene-construction requirements. Image2Sim converts sparse, heterogeneous captures into executable static navigation scenes without per-scene fitting. The systems differ in source assets, hardware, tasks, and policies; their construction costs and navigation scores are not matched comparisons. Table 3 provides the fixed-data simulator control.

Procedural scene generation. NVIDIA SAGE [100] composes task-conditioned scenes using generative tools and simulation-based critics; it is distinct from GS-based SAGE-3D [39]. Image2Sim preserves the layout and appearance of captured scenes, subject to coverage, and also uses synthetic RealSee3D and Structured3D inputs. Procedural generation and capture-based simulation are complementary. The Image2Nav/D3D-VLP [93] benchmark difference does not isolate scene-source quality because models, objectives, and training mixtures differ.

GS artifact removal. Difix3D+ [101] enhances existing NeRF/GS renders and can distill them into the scene through iterative updates. DiffusionHarmonizer [102] enhances reconstructed-scene renderings with temporal conditioning. Image2Sim instead jointly generates panoramic RGB-D from sparse captures and couples rendering with navigability extraction and online oracle generation. Its one-step pixel-space renderer requires neither a VAE nor per-scene render–enhance–distill fitting. These are differences in system inputs and roles; no matched-hardware comparison is reported.

Table 12: Scene-construction requirements of GS-based navigation systems. Scene counts refer to reported source corpora or demonstrations; Image2Sim excludes held-out scenes from navigation training.

System / scenes	Inputs and preparation	Construction cost / scope
Image2Sim 19,936 scenes	Sparse posed RGB-D: typically 6–30 panoramas or tens to hundreds of pinhole images	< 10 s per scene for feature-GS and navigability; no per-scene fitting
NavGSim [38] 1 large scene	Roughly 6K RGB-LiDAR frames; hierarchical GS fitting, chunk merging, cleanup, and alignment	> 5 h of scene-specific construction in the reported configuration
Habitat-GS [98] 120 scenes	Prepared GS assets and navigation geometry (NavMeshes or geometric proxies)	Reconstruction cost unspecified; rendering and interaction for supplied assets
SAGE-3D [39] 1,000 scenes	Artist-created geometry; approximately 3K RGB-D renders per scene for GS fitting; object meshes and collision bodies	Asset preparation and per-scene GS fitting; construction time unspecified

G Broader Impacts and Release Information

Benefits and limitations. Lower-cost navigation environments can broaden access to embodied-learning research and reduce physical data collection. Indoor captures may reveal private information, while dataset and VLM biases can affect instructions and policies. Reconstruction errors and visual hallucinations motivate independent deployment validation; geometric collision checks alone do not guarantee safety. Training and annotation also consume substantial computing resources; complete energy and carbon accounting is not reported.

Release and third-party assets. Code and reproduction instructions are available at `https://github.com/MrZihan/Image2Sim`; data and checkpoints are linked at `https://huggingface.co/datasets/MrZihanWang/Image2Sim-V1`, subject to access requests. Code uses Apache-2.0; generated data and non-code artifacts use CC BY-NC 4.0 unless specified otherwise (`LICENSE_NOTICE.MD`). Third-party assets retain their original terms (Table 13) and must be obtained separately where required. The repository covers data preparation, training, evaluation, and custom-capture import; raw RGB requires an external geometry estimator.

Table 13: Owner-maintained sources for third-party dataset and model terms.

Asset	Terms and access information
Matterport3D	`https://niessner.github.io/Matterport/`
HM3D	`matterport/habitat-matterport-3dresearch`
RealSee3D	`https://github.com/realsee-developer/RealSee3D`
Structured3D	`https://github.com/bertjiazheng/Structured3D`
Gibson	`https://github.com/StanfordVL/GibsonEnv`
ScanNet	`https://github.com/ScanNet/ScanNet`
ARKitScenes	`https://github.com/apple/ARKitScenes`
DINOv3	`https://github.com/facebookresearch/dinov3`
Qwen3-VL	`https://github.com/QwenLM/Qwen3-VL`

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction describe the simulation and data engine. Section 4.4 separates controlled evidence from benchmark comparisons, and the limitations distinguish static navigation from richer physical interaction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The main-paper limitations and Appendices E.3, F.1, and G discuss capture coverage, reconstruction noise, hallucination, evaluation scope, bias, and compute.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This paper presents a simulation system and empirical findings, not new theoretical guarantees. The flow formulation follows the cited MeanFlow work.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Sections 3 and 4 and the technical appendices specify the architecture, data, loss weights, training budgets, and evaluation protocols; the public code provides implementation and reproduction instructions.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code and instructions are public, but data/checkpoints require the stated access process, and source datasets retain third-party restrictions. Appendix G explains this access route; unrestricted access to every asset is not claimed.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and Appendices A, B.4, and E describe the data splits, optimizer, curriculum, model settings, and separate controlled-experiment budgets.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Results are point estimates and real-world success counts. Repeated-seed variability, error bars, and significance tests are not reported; the limited scope of this evidence is stated.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: GPU types/counts, principal training iterations, and selected component timings are reported, but complete per-experiment wall-clock, memory, and total GPU-hour accounting is not provided.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses attributed research assets under their applicable terms and reports robot-navigation experiments rather than a human-subject study. Appendix G addresses privacy, bias, and deployment risks.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: Appendix G discusses benefits for embodied-learning access and risks related to privacy, annotation bias, hallucination, robot deployment, and computing cost.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [Yes]

Justification: Appendix G identifies the data access process and third-party licensing restrictions. The paper does not equate simulation collision checks with safe autonomous real-world deployment.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Datasets and pretrained models are cited, and Appendix G links owner-maintained terms and access information. Project licensing does not override third-party restrictions.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Appendix A documents the data composition and statistics. Appendix G provides code, data/checkpoint access, reproduction instructions, and release terms.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The study does not use crowdsourced annotation or human research participants. Navigation instructions are generated automatically by a VLM.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No human-subject study is reported; the real-world evaluation concerns robot navigation.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The method and Appendix D describe Qwen3-VL-32B-Instruct for automatic instruction generation; Section 4 describes Qwen3-VL-4B-Instruct as Image2Nav initialization.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.