Vid2Sim: Realistic and Interactive Simulation from Video for Urban Navigation

Ziyang Xie[1,2] Zhizheng Liu[2] Zhenghao Peng[2] Wayne Wu[2] Bolei Zhou[2]
[1]University of Illinois Urbana-Champaign [2]University of California, Los Angeles

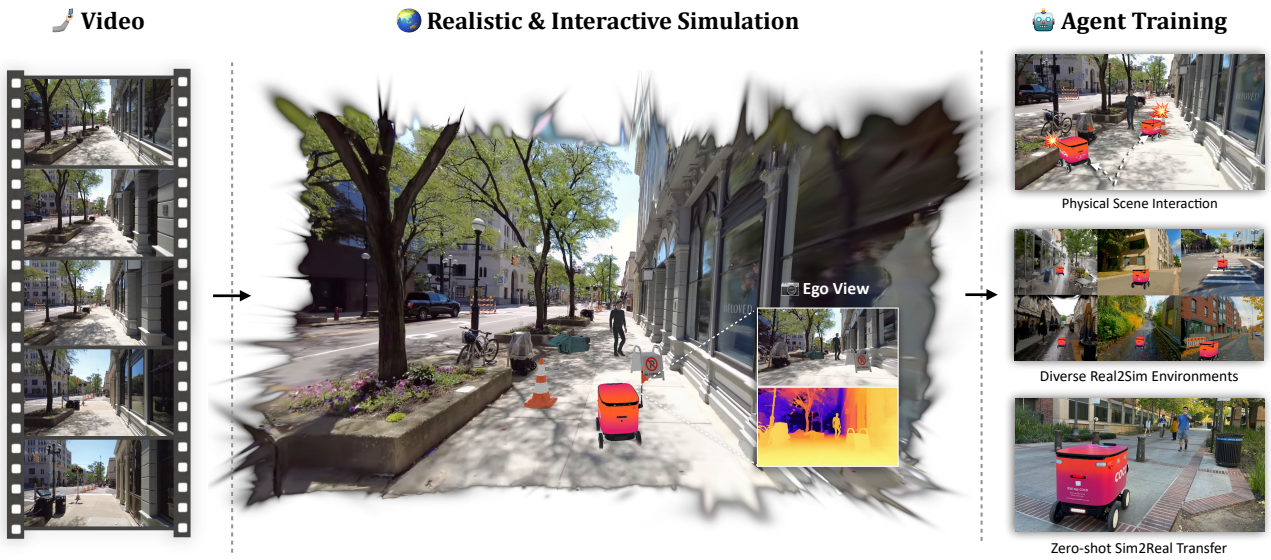


Figure 1. Vid2Sim converts monocular video captured by a hand-held camera into realistic and interactive 3D simulation environments. It facilitates RL training of navigation agents in digital twins of urban scenes and provides realistic observations like RGB and depth to reduce the sim-to-real gap. The pink mobile robot in the image is a food delivery bot that avoids collisions with pedestrians and obstacles.

Abstract

Sim-to-real gap has long posed a significant challenge for robot learning in simulation, preventing the deployment of learned models in the real world. Previous work has primarily focused on domain randomization and system identification to mitigate this gap. However, these methods are often limited by the inherent constraints of the simulation and graphics engines. In this work, we propose Vid2Sim, a novel framework that effectively bridges the sim2real gap through a scalable and cost-efficient real2sim pipeline for neural 3D scene reconstruction and simulation. Given a monocular video as input, Vid2Sim can generate photo-realistic and physically interactable 3D simulation environments to enable the reinforcement learning of visual navigation agents in complex urban environments. Extensive experiments demonstrate that Vid2Sim significantly improves the performance of urban navigation in the digital twins and real world by 31.2% and 68.3% in success rate compared with agents trained with prior simulation methods. Code and data will be made publicly available.

1. Introduction

Developing intelligent agents navigating and interacting within complex urban environments is crucial for many robotic applications like food delivery bots and assistive electric wheelchairs. Real-world experimentation is often challenging due to safety risks and efficiency constraints. In recent years, learning in simulation [32, 43, 51] has become an essential tool to provide a safe, controlled, and cost-effective alternative for training agents on complex tasks such as robotic manipulation [22, 28, 31] and autonomous driving [13, 26, 57, 64]. However, transferring the models trained in a simulated environment into the real world remains challenging due to the significant sim-to-real gap.

Substantial efforts have been made to bridge this gap. Prior approaches often leverage domain randomization [42, 50, 52] and system identification [8, 40, 49] methods that enhance the agent's robustness by simulating real-world noises and aligning agent dynamic model with the real-world settings. While these methods have achieved some success, their effectiveness is fundamentally limited by the simulator's capabilities. Traditional simulators often fail to

provide realistic observations, dynamic interactions, and diverse environmental variations. It is thus difficult to accurately reproduce the digital twins of real-world scenarios for agent training, limiting the agent's capacity to generalize effectively beyond the simulation environments. Recent advances in neural rendering techniques such as NeRF [35] and 3DGS [24] emerge as a potential solution to bridge the sim2real gap by reconstructing realistic 3D scenes from real-world data. However, most works [3, 19, 24, 35] only focus on enhancing photorealism for novel view synthesis and often fall short in constructing fully interactive environments that can support embodied agent training. While recent work Video2Game [59] attempts to extend NeRF-based neural reconstructions for interactive game development, its applications remain limited to gaming contexts, and its visual fidelity is limited by the textured mesh representation. Consequently, its effectiveness in training generalizable embodied agents remains constrained.

To tackle these issues, in this work, we present Vid2Sim, a novel real-to-sim (real2sim) framework that can convert causal videos captured in the real world into a fully realistic and physical interactive simulation environment and facilitate the reinforcement learning of urban navigation agents with minimal sim-to-real gap. By leveraging abundant web video data, our method offers a scalable and cost-efficient solution to build a realistic and diverse simulation environment for embodied agent training, as illustrated in Figure 1. Our pipeline consists of two main components: 1) geometry-consistent scene reconstruction and 2) realistic interactive simulation construction. Given a video as input, we design a geometry-consistent scene reconstruction approach that utilizes monocular cues to regularize Gaussian Splatting training in a scale-invariant way, thereby enhancing the reconstruction of fine-grained geometric details. In addition, we introduce a screen-space 2D covariance culling method to improve post-training rendering quality when the agent's camera trajectory deviates substantially from the training views.

To achieve realistic and interactive scene reconstruction, we propose a novel hybrid scene representation that combines Gaussian Splatting (GS) representation with mesh primitives. The GS provides real-time, photorealistic visuals for the agent, while the underlying mesh enables physical interactions and collision detection, which are essential for navigation training. Using Vid2Sim, we generate a diverse dataset of real2sim environments from web videos, encompassing a wide range of real-world urban settings for navigation agent training. These environments are further populated with static obstacles and other dynamic road users to replicate challenging real-world navigation scenarios. We augment these scenes through 3D scene editing and particle systems in different lighting, styles, and weather conditions to further improve the generalizability of our agents. Experiments in both simulation and real-world settings show that agents trained with Vid2Sim achieve substantially higher success rates and exhibit zero-shot sim2real transfer ability in real-world deployment. Compared to the agents trained with traditional simulation methods, Vid2Sim agent demonstrates enhanced robustness and lower collision rate, highlighting its potential as a scalable and efficient solution for bridging the sim-to-real gap. We summarize our contributions as follows:

1. **Real2Sim simulation environment from monocular video**: We introduce Vid2Sim, a novel real-to-sim (real2sim) framework that can convert monocular videos into photorealistic and physically interactive simulation environments.
2. **Geometry-consistent reconstruction with hybrid representation**: We develop a scene reconstruction method that improves scene reconstruction quality and agent visual observation quality through geometry-consistent GS training and screen-space covariance culling. We further propose a hybrid scene representation that combines GS with mesh primitives to enable photorealistic rendering and accurate physical interactions for urban navigation training.
3. **Comprehensive and diverse scene augmentation**: We present a scene composition method that integrates static obstacles and dynamic agents to construct interactive and diverse navigation environments, replicating complex real-world navigation challenges. Our method also supports extensive scene augmentation to environment layouts, lighting conditions, and weather for robust visual navigation training in complex urban environments.

2. Related Work

3D scene reconstruction 3D scene reconstruction has long been an active research topic in computer vision that aims to reconstruct the 3D scene representation from 2D image inputs. Recently, advances in neural reconstruction methods have achieved impressive results. Techniques like Neural Radiance Fields (NeRF) [35] and Gaussian Splatting (3DGS) [24] have demonstrated impressive abilities to reconstruct complex scenes from 2D inputs with photorealistic rendering. Building upon these methods, various extensions have been developed to address specific challenges, such as few-shot 3D reconstruction [10, 21, 38, 60, 61, 71] and reconstruction in unbounded, in-the-wild settings [3, 33, 68], further enhancing their applicability to real-world data. Other works [14, 19, 30, 53, 55, 56, 67] have focused on surface reconstruction task to better reconstruct object surface and capture the scene geometry. Despite these advancements, most works remain limited to photo-realistic reconstructions and cannot simulate physical interactions. In this work, we explore the possibility of extending GS

representation to create a physically interactive and realistic simulation from monocular videos, facilitating agent training in real-world consistent digital twins.

Sim-to-Real transfer The sim-to-real (sim2real) gap remains a significant challenge in deploying AI models from simulation to real-world environments. Traditional approaches like domain randomization [42, 50, 52] and system identification [8, 40, 49] aim to bridge this gap by mimicking real-world variations and aligning simulations with actual setups. However, their effectiveness is limited by the inherent fidelity of simulation environments.

Recent efforts [27, 47, 54] have leveraged image generative models to produce high-quality 2D observations for agents training. These methods often lack realistic physics, hindering physical interactions and closed-loop evaluations necessary for embodied agent training. Other studies [66, 70] attempt to integrate physics simulators with controlled 2D generation to ensure consistent observation simulation with physical interactions. However, they still rely on traditional simulators with limited environments for interaction and image layout control, preventing them from fully capturing real-world complexity. In our work, we resolve these issues by introducing a scalable and cost-efficient real2sim pipeline that can construct photo-realistic and interactive simulations from real-world videos. As a result, our method can significantly reduce the sim2real gap while avoiding the appearance and cost limitations introduced by traditional graphics simulators.

Data-driven simulation Data-driven simulation generation is crucial for creating realistic, diverse environments for embodied agents training. Traditional systems [26, 43, 48, 57, 58] use data-driven simulators to support navigation and manipulation tasks within diverse simulated environments. Despite progress, their scalability and fidelity remain limited. Recent works like Sim-on-Wheels [45] try to resolve this issue by incorporating a vehicle-in-the-loop framework that directly integrates real-world driving tests with virtual, safety-critical simulations. However, it still faces cost inefficiency and high time demands due to required real-world vehicle integration and continuous operation. Video2Game [59] leverages neural radiance fields to create interactive scenes from video for game engines, but its visual quality is constrained by textured mesh representation, and it mainly targets game development, limiting its use in embodied AI simulation. Our proposed method introduces a hybrid scene representation that combines GS with structured meshes, enabling the generation of photo-realistic, physically interactive simulations from real-world monocular video input in a data-efficient way.

3. Preliminary: 3D Gaussian Splatting

3DGS [24] has emerged as a popular point-based method for 3D scene reconstruction that explicitly represents the scene as a set of 3D Gaussian primitives. For each gaussian splat $\mathcal{G}_i(\mathbf{x})$, it's been parametrized by its means $\mu_i \in \mathbb{R}^3$, 3D covariance $\Sigma_i \in \mathbb{R}^{3\times3}$, opacity $\mathbf{o}_i$ and color $\mathbf{c}_i$ as:

$$\mathcal{G}_i(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i^{-1}(\mathbf{x}-\mu_i)\right). \quad (1)$$

During rendering, these 3D Gaussian splats $\mathcal{G}_i$ are projected onto the image plane as 2D Gaussians $\mathcal{G}_i^{'}$. The projection process can be represented as: $\Sigma_i^{'} = JW\Sigma_i W^T J^T$, where $\Sigma_i^{'}$ represent the 2D screen space gaussians covariance, W is the world-to-camera transformation matrix and J is the Jacobian of the perspective projection equation. To maintain a positive semi-definite 3D covariance Σ_i during optimization, Σ_i is then reparametrized using a scaling matrix $S \in \mathbb{R}^3$ and a rotation matrix $R \in \mathbb{R}^{3\times3}$ as $\Sigma_i = R_i S_i S_i^T R_i^T$. The color of pixel $\mathbf{c}(x)$ can then be rendered through a volumetric alpha-blending process:

$$\mathbf{c}(x) = \sum_{i\in N} T_i \mathbf{c}_i \alpha_i(\mathbf{x}), \;\; T_i = \prod_{i=1}^{i-1}(1-\alpha_i(\mathbf{x})), \quad (2)$$

where $\alpha_i(\mathbf{x}) = \mathbf{o}_i\mathcal{G}_i(\mathbf{x})$ represents the alpha value of the Gaussian Splats $\mathcal{G}_i$ at point $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{c}_i$ is the color of $\mathcal{G}_i$ evaluated by its spherical harmonics (SH) coefficients. Similarly, we can extend and render out the per-pixel median depth and normal for the Gaussian Splats as:

$$\hat{\mathbf{D}}(x) = \sum_{i\in N} T_i \mathbf{d}_i \alpha_i(\mathbf{x}), \;\; \hat{\mathbf{N}}(x) = \sum_{i\in N} \hat{\mathbf{n}}_i \alpha T_i, \quad (3)$$

where $\mathbf{d}_i$ is the i^{th} Gaussian Splat distance to the camera and $\hat{\mathbf{n}}_i$ is the normal direction based on the shortest axis direction of its covariance. Parameters of the 3DGS are then optimized by a photometric rendering loss in 2D image space.

While 3DGS effectively reconstructs visually realistic scenes, it struggles with accurate geometry reconstruction, tends to overfit training views, and cannot support physical interaction, limiting its use in interactive robotics learning. We introduce our Vid2Sim framework in the next section to overcome these challenges.

4. Vid2Sim Framework

Given a monocular video, the goal of Vid2Sim is to generate a realistic and physically interactive simulation environment for embodied navigation training with minimal sim-to-real gap. To achieve this, Vid2Sim employs a two-stage pipeline: We first reconstruct a high-quality 3D scene representation with geometry-consistent Gaussian Splatting. In the second stage, we combine the reconstructed splats and

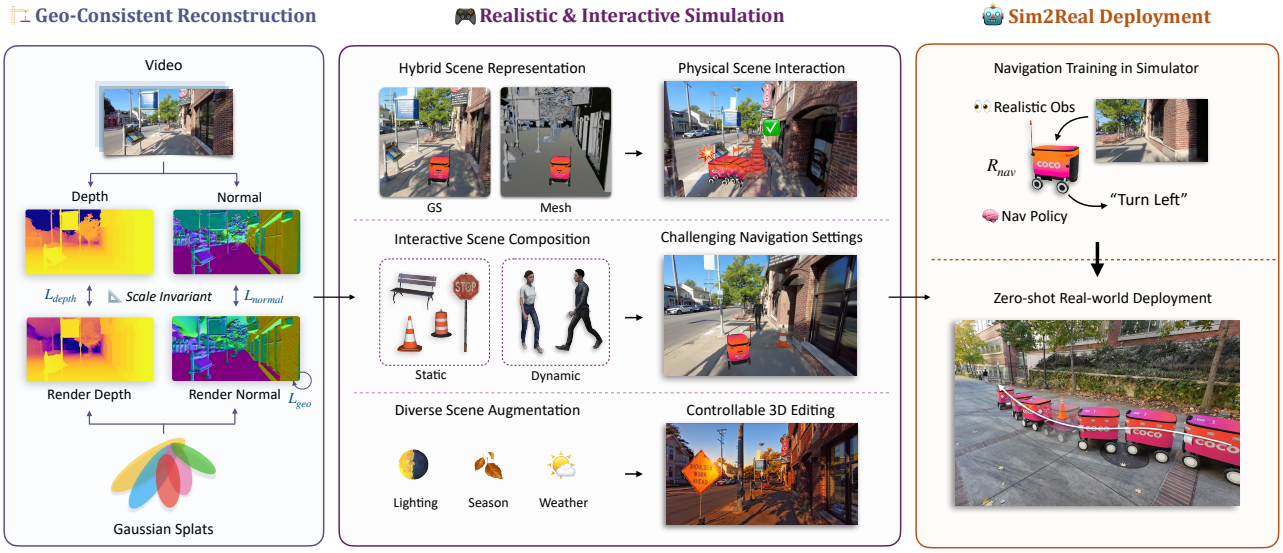


Figure 2. Vid2Sim framework consists of three key stages: (1) Geometry-consistent reconstruction for high-quality environment creation, (2) building a realistic and interactive simulation with hybrid scene representation and diverse obstacle and scene augmentation for visual navigation training, and (3) Sim2Real validation through real-world deployment.

mesh into a hybrid scene representation, building a photo-realistic and interactive training environment with diverse obstacles and augmentations to support robust visual navigation training in complex environments.

4.1. Geometry-Consistent Scene Reconstruction

Accurate geometry reconstruction is important for agent navigation training to support accurate collision detection and realistic physical interactions. However, reconstructing the high-quality 3D structure of a scene from casual monocular video footage remains challenging due to the inherent geometric uncertainties and the lack of multi-view information. To overcome this challenge, we propose a geometry-consistent reconstruction method that utilizes monocular cues to regularize GS training, enhancing geometry reconstruction for accurate agent-environment interactions.

Scale-Invariant Geometry Supervision Current advancements [5, 62, 63] in monocular depth estimation can be used to provide a strong geometry prior for in-the-wild scene reconstruction. However, most of these methods often implement a scale-shift-invariant (SSI) loss [5, 34] to predict relative depth rather than absolute metric depth. Directly minimizing the differences between rendered and predicted depth may cause ambiguities, since the Gaussian Splats is initialized with a specific depth scale from Structure-from-Motion (SfM) [44] point clouds,

As illustrated in Fig. 2, we propose to use scale-invariant losses over the depth and normal to tackle that issue. Specifically, a patch-based normalized cross-correlation (NCC) loss is applied between the rendered depth $\hat{\mathbf{D}}$ and the predicted depth $\mathbf{D}$ generated from an off-the-shelf depth estimator [63] for supervision. The patch-based NCC loss evaluates the local similarity between depth maps while being less sensitive to global scale discrepancies:

$$\mathcal{L}_{\text{depth}} = 1 - \frac{1}{\|\mathcal{P}\|} \sum_{p \in \mathcal{P}} \sum_{k=1}^{K^2} \frac{\hat{\mathbf{D}}'_{p,k} \mathbf{D}'_{p,k}}{\hat{\sigma}_p \sigma_p}, \tag{4}$$

where $\mathcal{P}$ is the set of all patches extracted from the depth map, and the inner sum over k represents the summation of the NCC scores within a single patch of size $K \times K$. $\hat{\mathbf{D}}'_{p,k}$ and $\mathbf{D}'_{p,k}$ are the mean-centered values of the rendered and predicted depths at pixel k within patch p, respectively. The $\hat{\sigma}_p$ and σ_p are standard deviations of the rendered and predicted depth maps within the patch. This approach ensures that depth alignment is based on local structural similarity rather than absolute scale, which is more robust to noise and occlusions.

For normal supervision, we also employ a scale-invariant loss that directly measures the alignment between rendered and predicted normals based on their cosine distance:

$$\mathcal{L}_{\text{normal}} = 1 - \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{\hat{\mathbf{N}}_{i,j} \cdot \mathbf{N}_{i,j}}{\|\hat{\mathbf{N}}_{i,j}\| \|\mathbf{N}_{i,j}\|}. \tag{5}$$

$\hat{\mathbf{N}}_{i,j}$ and $\mathbf{N}_{i,j}$ represent the rendered surface normal and the pseudo GT normal at pixel (i, j). The pseudo GT normal is derived from the predicted depth map $\mathbf{D}$ by applying PCA on the projected point clouds to estimate the normal direction. H and W are the height and width of the rendered normal image.

Geometry-Consistent Loss We further enhance the reconstruction geometry consistency by introducing a novel

Geometry-Consistent Loss. This loss function aims to enforce smoothness and maintain structural integrity by ensuring that the normal vectors of adjacent pixels align consistently. This is particularly important in regions with minimal depth variation where the surfaces should appear continuous. The proposed Geometry-Consistent Loss $\mathcal{L}_{\text{geo}}$ is defined as:

$$\mathcal{L}_{\text{geo}} = \frac{\sum_{i,j} w_{i,j} \cdot \left(1 - \hat{\mathbf{N}}_{i,j} \cdot \hat{\mathbf{N}}_{i+\Delta x, j+\Delta y}\right)}{\sum_{i,j} w_{i,j}}, \quad w_{i,j} = 1 - \left\| \sqrt{(\nabla_x \mathbf{D}_{i,j})^2 + (\nabla_y \mathbf{D}_{i,j})^2} \right\|. \tag{6}$$

Here, $(i + \Delta x, j + \Delta y)$ are the coordinates of adjacent pixels to pixel (i, j), which include right and bottom neighbors $(\Delta x, \Delta y \in \{0, 1\})$. Weight $w_{i,j}$ is computed from the local depth gradients, which assign higher weights to pixels in regions with less depth change to ensure normal consistency.

Inspired by 2DGS [19], we further regularize 3D gaussian into 2D disk shape to better represent the scene geometry by minimizing its shortest axis scale $S_i = \text{diag}(s_1, s_2, s_3)$ with $\mathcal{L}_{\text{scale}} = \frac{1}{N} \sum_{i \in N} \| \min(s_1, s_2, s_3) \|$, here N represents the number of Gaussian splats. By combining the scale-invariant depth loss and normal loss, our method achieves robust supervision that enhances the accuracy and consistency of 3D scene reconstructions. Our final optimization loss can be defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{rgb}} + \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{normal}} + \mathcal{L}_{\text{geo}} + \mathcal{L}_{\text{scale}}. \tag{7}$$

Screen-Space Covariance Culling During RL training, the agent must explore and interact with the environment to learn a robust navigation policy. However, such random explorations often lead to significant discrepancies between the agent's camera trajectory and the views encountered during training. These substantial changes in viewing angles can result in rendering artifacts and floaters, particularly near the ground surface. (Shown in Supp Sec. B.2) These floaters may obstruct the agent's view and be mistakenly considered as the obstacles that block the robot and create unexpected collisions.

To mitigate these issues, we propose a simple yet effective screen-space covariance culling technique that adjusts the visual input by selectively removing artifacts based on the size of Gaussian splats when rasterized to 2D space: $\|\Sigma'\|_\infty > \alpha \cdot A_{\text{img}}$, where $\|\Sigma'\|_\infty$ denotes the maximum norm of the covariance matrix and $A_{\text{img}} = H \times W$ represents the total image area with the image height H and width W. α is a proportionality constant that scales the threshold relative to the image size. This approach effectively filters out splats that exceed the defined image portion, helping to maintain visual clarity and mitigate the impact of artifacts in simulation due to view discrepancy.

4.2. Realistic and Interactive Simulation

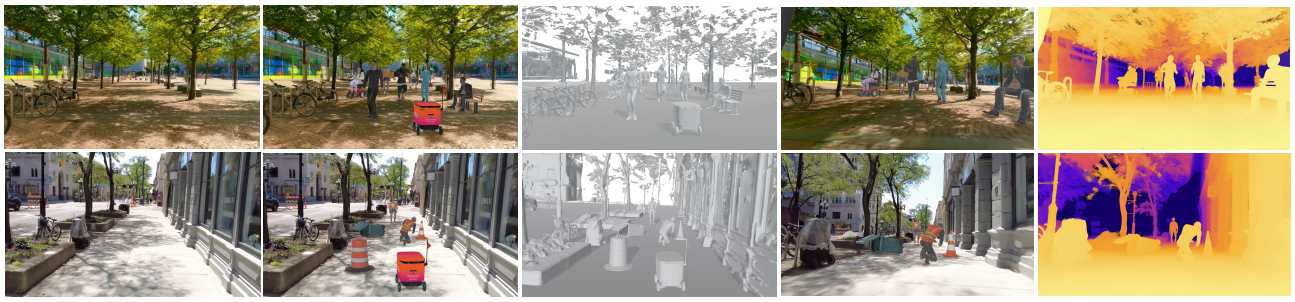
An effective simulation environment should enable agents to interact with the environment and adapt to environment changes for closed-loop evaluation. The standard GS representation, however, is unsuitable as a simulator because it lacks support for agent-environment interactions.

Hybrid Scene Representation To enhance the interactivity and realism of our simulation, as illustrated in Fig. 2, we introduce a hybrid scene representation that combines our GS representation with its scene mesh primitives to create a realistic and interactive simulation environment. As shown in Fig. 3(c), our geometry-consistent GS reconstruction enables the extraction of high-quality environment meshes. In our hybrid scene representation, the GS provides photo-realistic visual observations for our agent training, and scene mesh is baked to support physical interaction and accurate collision detection to ensure interactive agent-environment interaction.

Specifically, we use the Truncated Signed Distance Function (TSDF) [11] to export a high-quality mesh from our GS representation and employ Unity engine [16] to provide real-time physics simulation. During simulation, the GS representation and the extracted scene mesh are imported concurrently. We use a custom Unity shader to support real-time photo-realistic rendering from GS, and the mesh material is set to be invisible and operates as the collision and agent interaction primitive without impacting visual fidelity. This hybrid design combines the strength of both GS and mesh representation to support physical interaction while maintaining high-quality visual rendering for effective agent training.

Interactive Scene Composition While our hybrid scene representation offers realistic visuals and basic physical interactions, it does not fully capture the dynamic and complex situations that agents may encounter in real-world environments, such as new obstacles and other road users. To closely simulate real-world navigation scenarios, we incorporate two types of obstacles into our simulation environments: 1) static objects to simulate fixed obstacles and 2) dynamic agents to emulate other road users typically present in real-world settings.

As shown in Fig. 3, during training, static objects such as traffic cones, trash bins, traffic lights, and poles are randomly selected and placed as obstacles within the scene. These common objects are frequently found in urban environments and greatly increase the complexity and diversity of our simulation environments with real navigation challenges. We achieve seamless composition between foreground objects and the background GS scene by combining GS rasterization with mesh rendering for both RGB and depth views. The occlusion relationships between the foreground objects and background scene are handled through



(a) Real2Sim Scene (b) Insert Obstacles (c) Scene Mesh (d) Agent Observation (e) Depth Sensor

Figure 3. Interactive Scene Composition with Vid2Sim: Our method is able to combine reconstructed environments with 3D assets to create diverse simulation scenarios. Here we show the (a) original real2sim environment, (b) interactive scene composition with static and dynamic obstacles, (c) scene mesh for physical collision detection, (d) agent's RGB observations, and (e) depth rendering from our hybrid scene representation that could serve as an extra sensory modality.

Figure 4. Scene augmentation with various overall stylization

z-buffering [7] to ensure depth consistency and accurate object visibility. The dynamic obstacles like pedestrians are imported and programmed with A^* planning algorithm to move between random points within the scene following the shortest path, providing interactions that the agents must effectively predict and respond to. These introduced obstacles significantly increase the diversity of our environments and enable diverse interactions between the training agent and the environment, creating a comprehensive, data-rich setting that is crucial for effective navigation training.

Moreover, as shown in the second row of Fig. 3, our framework can simulate various safety-critical scenarios, such as falling trash cans or road construction involving roadblocks and workers. These situations are often considered out-of-distribution corner cases in autonomous navigation research [2, 69] and are extremely crucial for training a safe and robust navigation policy.

Diverse Scene Augmentation Enhancing the training environment with a multi-level augmentation approach introduces greater variability and realism to agent learning. On the scene level, we integrate and extend the popular scene editing method [36] to support video-length consistent editing with improved temporal consistency. In this way, we can augment the training environments and generate multiple diverse scene variants that capture changes in lighting, seasons, and semantics (Fig. 4). Additionally, similar to ClimateNeRF [29], Vid2Sim can also support simulating different weather conditions such as rain, fog, and snow through particle simulation. With 3D layout editing and advanced weather simulations, we can create diverse, realistic environments that enrich our dataset. This variety helps our agent develop a general and robust navigation strategy across different scenarios.

5. Experiments

To evaluate the effectiveness of Vid2Sim, we curate a dataset of real2sim environments by reconstructing 30 diverse scenes from web-sourced videos. These videos capture a range of complex urban scenarios, providing a robust foundation for in-the-wild visual navigation training. Please see the supplementary for the dataset preview. Our experiments are designed to evaluate three main aspects of Vid2Sim's effectiveness: 1) its ability to reconstruct high-quality 3D environments from monocular videos for simulation, 2) its capability to support the training of robust navigation agents, and 3) its effectiveness in minimizing the sim2real gap. We conduct extensive evaluations in both simulated and real-world environments. Results show that Vid2Sim effectively builds high-quality simulations for embodied navigation training. Agents trained with our pipeline demonstrate better performance and exhibit a significantly reduced sim-to-real gap compared with agents trained in conventional environments with mesh-based observations.

5.1. Reconstruction Evaluation

We first evaluate our geometry-consistent reconstruction method by comparing it with other state-of-the-art reconstruction methods on our Vid2Sim dataset. The Vid2Sim dataset contains 30 diverse scenes. Each scene is a 15-second video at 30 fps, containing 450 frames. In every eight frames, there is one frame reserved for testing, resulting in a total of 393 training images and 57 test images per scene. As shown in Tab. 1, our method consistently outper-

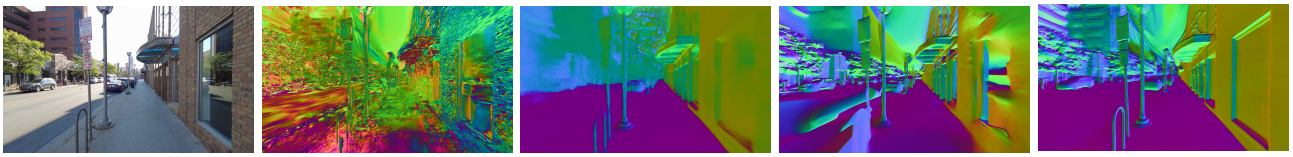


Figure 5. Surface normal renderings of different methods: results show that our approach reconstructs scene geometry with finer surface details and less artifacts compared to 3DGS [24], Video2Game [59], and 2DGS [19].

Methods	Rendering Quality			Simulation Capability		
	PSNR ↑	SSIM ↑	LPIPS ↓	Real Time	Interactive	RL Training
Instant-NGP [37]	27.50	0.827	0.240	✗	✗	✗
3DGS [24]	31.85	0.921	0.136	✓	✗	✗
2DGS [19]	30.82	0.915	0.154	✓	✗	✗
Video2Game [59]	28.32	0.834	0.275	~	✓	✗
Ours	**32.41**	**0.927**	**0.127**	✓	✓	✓

Table 1. Comparison of rendering quality and simulation capability between Vid2Sim and other methods. Our method achieves the highest reconstruction quality among other methods and offers the most comprehensive simulation capabilities.

Methods	Obs	PointNav			SocialNav		
		SR ↑	SPL ↑	Cost ↓	SR ↑	SNS ↑	Cost ↓
Mesh†	RGB	48.8%	0.496	0.34	43.2%	0.991	1.04
Vid2Sim (Oracle)	Depth	92.0%	0.937	0.57	85.6%	0.992	0.75
Vid2Sim (No Obj)	RGB	68.8%	0.695	1.45	61.6%	0.973	1.79
Vid2Sim (Static)	RGB	80.8%	0.818	0.94	71.2%	0.980	1.74
Vid2Sim (Dynamic)	RGB	**81.6%**	**0.824**	**0.86**	**74.4%**	**0.987**	**1.21**

Table 2. Evaluation of agent navigation performance in simulation. Mesh† represents only use our reconstructed mesh for visual observation to simulate the mesh-based simulation method [59].

forms all compared methods in terms of PSNR, SSIM, and LPIPS. In Fig. 5, we further show the qualitative comparison between our method and other methods in surface normal rendering to compare the surface reconstruction quality between different methods. Our method can reconstruct finer and more accurate geometric details that align with the real-world scene compared with other methods.

In supplementary we further demonstrate the effectiveness of our screen-space covariance culling method. Qualitative and quantitative results demonstrate this technique can effectively remove floater artifacts that obstruct the agent's view and significantly improve agent observation quality at extreme viewing angles.

5.2. Urban Navigation Training

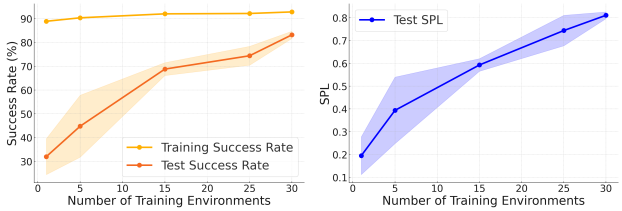
In this section, we describe our experimental setup and evaluation metrics for training and testing our agents. We aim to test the agent's navigation capability in diverse Vid2Sim environments and compare the performance with the traditional mesh-based simulator and other baseline variants. Depending on the task, the agent must learn to avoid colliding with the environment, static obstacles, and dynamic pedestrians, which reflects real-world challenges.

Experiment Setup We deploy Vid2Sim on a four-wheeled wheeled delivery robot. The robot is equipped with a front-facing RGB camera for visual observation. For comparison, an oracle agent setup with a ground-truth depth sensor is also implemented in simulation. We stack the current image observation with the past 5 timesteps to incorporate the historical information. Visual observation and the goal observation including the distance and heading angle to the goal point are then combined as the final policy observation. The agent action space includes two continuous control variables normalized in $[-1, 1]$: one for adjusting wheel rotation speed and the other for modulating speed. The simulator settings are calibrated to match real-world physical constraints to ensure a reliable sim2real transferability of the navigation policies.

Tasks and metrics We design two common navigation tasks, Point Navigation (*PointNav*) and Social Navigation (*SocialNav*), to test agent performance in both static and dynamic scenarios. In both tasks, the agent must navigate through the environment from a starting location to a goal point which are randomized across different episodes to ensure robust policy learning. For *PointNav*, the agent needs to avoid hitting the environment and static obstacles placed within the scene, while in *SocialNav*, the agent must avoid colliding with both static obstacles and other moving pedestrians by adapting its path according to their movements.

Our agents are trained across 30 unique Vid2Sim simulation environments with the off-policy reinforcement learning algorithm Soft-Actor-Critic (SAC) [15] for 1.5M steps and tested on 5 hold-out testing scenes which the agent has never seen during training. The agent's performance on each task is evaluated based on the success rate (SR) [1], success rate weighted by path length (SPL) [4], hit number (Cost) and social navigation score (SNS) [12]. All trials are rolled out 25 times to mitigate result variance. For each trial, 0 to 5 static objects are randomly sampled and placed between the agent starting point and goal point as obstacles. For social navigation dynamic pedestrians are placed within the scene and animated to walk across the movable area randomly. We report the average score across all 125 trials. Specific details including hyper-parameter settings, reward shaping, training, and testing scene descriptions are available in supplementary Sec. D.

Performance Evaluation As there is no existing method that can convert videos into a functional simulation environment for RL training, we simulate a comparable mesh-



(a) Success Rate vs Training Environments Num (b) SPL vs Training Environments Num

Figure 6. Generalization Results: This table compares (a) the success rate (SR) and (b) success rate weighted by path length (SPL) across varying numbers of training environments. Increasing the number of training environments leads to a higher test success rate and SPL, which indicates improved agents generalizability.

based approach using our reconstructed colored mesh representation for agent visual navigation, similar to the approach taken by Video2Game [59] for game development.

We report the performance of agents trained under three conditions: without obstacles, with static obstacles, and with both static and dynamic obstacles, and compare their results across these setups. Additionally, we evaluate agents trained with the oracle depth observations to better understand the capabilities of different policies. Tab. 2 shows that the agents trained with both static and dynamic obstacles achieved the best performance with an 81.6% success rate on the *PointNav* task and a 74.4% on the *SocialNav* task. It achieves a significant 32.8% and 31.2% performance improvement compared to traditional mesh-based simulation, respectively. Notably, even when trained only with static obstacles, our agent still demonstrates emergent capabilities in *SocialNav* task by reaching a 71.2% success rate.

Fig. 6 shows that our agent generalizability is substantially improved by increasing the number of training environments, where we evaluate performance across 1, 5, 15, 25, and 30 training environments. The test scene success rate increases significantly when training with 15 environments and continues to rise with additional training environments. For SPL, the trend is similar. In general, the variances of SR and SPL continue to decrease as the number of training environments increases, indicating more robust navigation performance.

5.3. Sim-to-Real Deployment

To evaluate the effectiveness of the proposed Vid2Sim pipeline in bridging the sim-to-real gap, we deploy agents trained in Vid2Sim environments to the real world in zero-shot settings. Fig. 7 demonstrates our real-world experiment settings in diverse environments. We further show a qualitative comparison between the simulation and real-world digital-twin environments in supplementary Fig. 10 to illustrate our reduced sim-to-real gap. Other real-world experiment details can be found in supplementary Sec. E

Experiment Setup We test the policy in real-world urban environments which is not included in the Vid2Sim training dataset. Each environment has a walkable region of at least 3m in width and 20m in length, sufficient for the robot to avoid obstacles. In each trial, we randomly sample starting and ending positions in the walkable region with a travel distance between 5m and 15m and evaluate the performance on the following tasks: 1) *Go Straight*: reach the goal in an empty environment without obstacles. 2) *Static Obstacle*: reach the goal while avoiding static objects placed in between the starting and goal points. 3). *Dynamic Obstacle*: reach the goal while avoiding dynamic pedestrians walking towards the robot. The robot succeeds if it reaches within 0.5m of the goal and fails if it either collides with other objects or moves out of the walkable region. We perform 20 trials for each task and report the final success rate.

Figure 7. Real-world experiment settings

Method (Env N)	Go Straight	Static Obstacle	Dynamic Obstacle
Baseline (30)	0%	0%	0%
Vid2Sim (1)	0%	30%	0%
Vid2Sim (5)	60%	40%	0%
Vid2Sim (30)	**85%**	**65%**	**55%**

Table 3. The navigation performance of trained agents in the real world. The number in the parenthesis indicates the number of environments the agent is trained on.

Results As shown in Tab. 3, the mesh reconstruction baseline fails to complete all three tasks. This shows the large sim-to-real gap with the RGB observation from mesh representation. On the contrary, Vid2Sim performs better by training with more environments, similar to what is observed in Fig 6. Note that all results are from zero-shot deployment without fine-tuning. This highlights the effectiveness of Vid2Sim in addressing the sim-to-real gap and the possibility of learning a general real-world deployable visual navigation policy in simulation by adopting the Vid2Sim pipeline to more videos.

6. Conclusion

This work introduces a novel framework Vid2Sim that can convert monocular videos into a realistic and interactive simulation environment for learning robust and generalizable urban navigation policies. Experiments show our method significantly reduces the sim-to-real gap and provides a scalable solution to create simulation environments from real-world scenes. We hope to extend this framework to many other agents like legged robots in the future.

References

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 7

[2] Jianhong Bai, Zuozhu Liu, Hualiang Wang, Jin Hao, YANG FENG, Huanpeng Chu, and Haoji Hu. On the effectiveness of out-of-distribution data in self-supervised long-tail learning. In *ICLR*, 2023. 6

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 2

[4] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020. 7

[5] Reiner Birkl, Diana Wofk, and Matthias Müller. Midas v3.1 – a model zoo for robust monocular relative depth estimation. *PAMI*, 2023. 4

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 2

[7] Edwin Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. Ph.d. dissertation, University of Utah, Salt Lake City, UT, 1974. 6

[8] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *ICRA*, 2019. 1, 3

[9] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Alexander Schwing, and Joon-Young Lee. Tracking anything with decoupled video segmentation. In *ICCV*, 2023. 1

[10] Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. Depth-regularized optimization for 3d gaussian splatting in few-shot images. In *CVPRW*, 2023. 2

[11] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 1996. 5

[12] Matt Deitke, Dhruv Batra, Yonatan Bisk, Tommaso Campari, Angel X. Chang, Devendra Singh Chaplot, Changan Chen, Claudia Pérez D'Arpino, Kiana Ehsani, Ali Farhadi, Li Fei-Fei, Anthony Francis, Chuang Gan, Kristen Grauman, David Hall, Winson Han, Unnat Jain, Aniruddha Kembhavi, Jacob Krantz, Stefan Lee, Chengshu Li, Sagnik Majumder, Oleksandr Maksymets, Roberto Martín-Martín, Roozbeh Mottaghi, Sonia Raychaudhuri, Mike Roberts, Silvio Savarese, Manolis Savva, Mohit Shridhar, Niko Sünderhauf, Andrew Szot, Ben Talbot, Joshua B. Tenenbaum, Jesse Thomason, Alexander Toshev, Joanne Truong, Luca Weihs, and Jiajun Wu. Retrospectives on the embodied ai workshop. *arXiv preprint arXiv:2210.06849*, 2022. 7

[13] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *CoRL*, 2017. 1

[14] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *CVPR*, 2024. 2

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *Deep Reinforcement Learning Symposium*, 2017. 7

[16] John K Haas. A history of the unity game engine. *Worcester Polytechnic Institute*, 2014. 5, 1

[17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 1, 2

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1

[19] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH*, 2024. 2, 5, 7

[20] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, 2011. 1

[21] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *ICCV*, 2021. 2

[22] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Proceedings of The 2nd Conference on Robot Learning*. PMLR, 2018. 1

[23] Lei Ke, Mingqiao Ye, Martin Danelljan, Yifan Liu, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Segment anything in high quality. In *NeurIPS*, 2023. 1

[24] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 2023. 2, 3, 7, 1

[25] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. 1

[26] Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1, 3

[27] Xiaofan Li, Yifu Zhang, and Xiaoqing Ye. Drivingdiffusion: Layout-guided multi-view driving scene video generation with latent diffusion model. *arXiv preprint arXiv:2310.07771*, 2023. 3

[28] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel

Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024. 1

[29] Yuan Li, Zhi-Hao Lin, David Forsyth, Jia-Bin Huang, and Shenlong Wang. Climatenerf: Extreme weather synthesis in neural radiance field. In *ICCV*, 2023. 6

[30] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *CVPR*, 2023. 2

[31] Liqian Ma, Jiaojiao Meng, Shuntao Liu, Weihang Chen, Jing Xu, and Rui Chen. Sim2real2: Actively building explicit physics model for precise articulated object manipulation. In *ICRA*, 2023. 1

[32] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021. 1

[33] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021. 2

[34] S. Mahdi H. Miangoleh, Mahesh Reddy, and Yağız Aksoy. Scale-invariant monocular depth estimation via ssi depth. In *Proc. SIGGRAPH*, 2024. 4

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

[36] Linzhan Mou, Jun-Kun Chen, and Yu-Xiong Wang. Instruct 4d-to-4d: Editing 4d scenes as pseudo-3d scenes using 2d diffusion. In *CVPR*, 2024. 6

[37] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 2022. 7

[38] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022. 2

[39] Linfei Pan, Daniel Barath, Marc Pollefeys, and Johannes Lutz Schönberger. Global Structure-from-Motion Revisited. In *ECCV*, 2024. 1, 3

[40] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *ICRA*, 2018. 1, 3

[41] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021. 2

[42] Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real single-image flight without a single real image. In *RSS*, 2017. 1, 3

[43] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *ICCV*, 2019. 1, 3

[44] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4, 1

[45] Yuan Shen, Bhargav Chandaka, Zhi-Hao Lin, Albert Zhai, Hang Cui, David Forsyth, and Shenlong Wang. Sim-on-wheels: Physical world in the loop simulation for self-driving. In *IEEE Robotics and Automation Letters*, 2023. 3

[46] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 1

[47] Alexander Swerdlow, Runsheng Xu, and Bolei Zhou. Street-view image generation from a bird's-eye view layout. *IEEE Robotics and Automation Letters*, 2024. 3

[48] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *NeurIPS*, 2021. 3

[49] Ji Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *RSS*, 2018. 1, 3

[50] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017. 1, 3

[51] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012. 1

[52] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cong Phuoc Huynh, Timo To, Eric Cameracci, Steve Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *CVPRW*, 2018. 1, 3

[53] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 2

[54] Xiaofeng Wang, Zheng Zhu, Guan Huang, Xinze Chen, Jiagang Zhu, and Jiwen Lu. Drivedreamer: Towards real-world-driven world models for autonomous driving. In *ECCV*, 2024. 3

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

[56] Qianyi Wu, Jianmin Zheng, and Jianfei Cai. Surface reconstruction from 3d gaussian splatting via local structural hints. In *ECCV*, 2024. 2

[57] Wayne Wu, Honglin He, Yiran Wang, Chenda Duan, Jack He, Zhizheng Liu, Quanyi Li, and Bolei Zhou. Metaurban: A simulation platform for embodied ai in urban spaces. *arXiv preprint arXiv:2407.08725*, 2024. 1, 3

[58] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: real-world perception for embodied agents. In *CVPR*, 2018. 3

[59] Hongchi Xia, Zhi-Hao Lin, Wei-Chiu Ma, and Shenlong Wang. Video2game: Real-time, interactive, realistic and browser-compatible environment from a single video. In *CVPR*, 2024. 2, 3, 7, 8

[60] Qingshan Xu, Xuanyu Yi, Jianyao Xu, Wenbing Tao, Yew-Soon Ong, and Hanwang Zhang. Few-shot nerf by adaptive rendering loss regularization. In *ECCV*, 2024. 2

[61] Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *CVPR*, 2023. 2

[62] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024. 4

[63] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv preprint arXiv:2406.09414*, 2024. 4

[64] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *CVPR*, 2023. 1

[65] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details for 3d gaussian splatting. *arXiv preprint arXiv:2404.10484*, 2024. 1

[66] Alan Yu, Ge Yang, Ran Choi, Yajvan Ravan, John Leonard, and Phillip Isola. Lucidsim: Learning agile visual locomotion from generated images. In *CORL*, 2024. 3

[67] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*, 2024. 2

[68] Dongbin Zhang, Chuming Wang, Weitao Wang, Peihao Li, Minghan Qin, and Haoqian Wang. Gaussian in the wild: 3d gaussian splatting for unconstrained image collections. In *ECCV*, 2024. 2

[69] Weitao Zhou, Zhong Cao, Yunkang Xu, Nanshan Deng, Xiaoyu Liu, Kun Jiang, and Diange Yang. Long-tail prediction uncertainty aware trajectory planning for self-driving vehicles. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022. 6

[70] Yunsong Zhou, Michael Simon, Zhenghao Peng, Sicheng Mo, Hongzi Zhu, Minyi Guo, and Bolei Zhou. Simgen: Simulator-conditioned driving scene generation. In *NeurIPS*, 2024. 3

[71] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *ECCV*, 2024. 2

Appendix

A. Vid2Sim dataset

Our Vid2Sim dataset includes 30 high-quality real-to-sim simulation environments. These environments are reconstructed from video clips sourced from 9 web videos recorded by individuals walking along streets with a single hand-held camera. Each clip capture includes 15 seconds of forward-facing video recorded at 30 fps, providing 450 frames per scene for reconstruction. To ensure privacy, we mask all human faces and identifiable information in the video frames. Commonly used Structure-from-Motion (SfM) methods, such as COLMAP [44], often fail to reconstruct accurate camera poses from in-the-wild videos due to significant variations in viewpoints and lighting conditions. Instead, we employ GLOMAP [39], an advanced general-purpose SfM system that is more robust and efficient than COLMAP, to obtain accurate camera poses.

In Fig. 8, We show a preview snap-shot of the 30 diverse environments in our dataset, these environments encompass a variety of urban navigation scenarios designed for more robust and generalizable navigation policy training in complex urban environments.

B. Geometry-Consistent Reconstruction

B.1. Implementation details

In this section, we mainly introduce the implementation details of our geometry-consistent reconstruction method, including training strategy, hyper-parameter settings, and other extra details.

To avoid the influence of dynamic objects within the original videos, for each frame, we generate a dynamic mask to mask out all the moving objects within the scene with an off-the-shelf video-based tracker DEVA [9]. We train our model for 30000 iterations. We use the same training hyper-parameters as 3DGS [24] and adopt the densification strategy from AbsGS [65]. All the depth, normal, and geometry-consistency loss are started to apply at the 500 iterations still the end of the training.

For the mesh extraction, we first render the depth map from each frame and fuse them into a TSDF field with KinectFusion [20], the mesh is then extracted from TSDF field with voxel size set to 0.1. To eliminate the potential dynamics issue caused by uneven ground reconstructions, we further remove the ground plane of the scene in our mesh extraction through ground plane segmentation. We found that directly applying a general segmentation model (such as SAM [25] or SAM-HQ [23]) cannot remove all the ground mesh due to inaccurate segmentation and prompt ambiguity. Instead, we propose to generate a detailed ground mask $\mathcal{M}_i$ by using the ground plane's normal direction as a prior:

$$\mathcal{M}_i = \| \arccos(\mathbf{N}_i \cdot \bar{\mathbf{n}}_i^{'}) < \delta \|, \tag{8}$$

where $\mathbf{N}_i$ indicates the rendered normal map of the i^{th} frame, $\bar{\mathbf{n}}_i^{'}$ denotes the mean normal direction of the ground surface computed from the ground segmentation mask given by the SAM-HQ [23] model and δ denotes the angle threshold. Pixels whose normal directions are within δ degrees of the mean ground surface normal direction are grouped together to generate the final ground mask.Incorporating normal priors results in a more precise ground mask, enabling clean mesh extraction without the ground surface. We set $\delta = 15$ degrees in our implementation.

B.2. Screen-space covariance culling

In this section, we systematically evaluate our screen-space covariance culling method both qualitatively and quantitatively. Results demonstrate this technique could effectively remove rendering artifacts and significantly improve agent observation quality.

Since floater artifacts often appear when the viewing angles and focal length differ significantly from the training view. Therefore, standard test views that closely align with the training view may not accurately represent real situations during agent training. For better evaluation, we simulate the agent's camera view by adjusting the test view's camera focal down by 1.5x and shifting the camera down by 1 unit. Given the fact that there is no ground-truth image available to apply common NVS evaluation metrics (e.g. PSNR and SSIM) to evaluate the quality of the culled images, we instead provide a quantitative evaluation of screen-space covariance culling by computing the Fréchet Inception Distance (FID) [17] score between the training views and the agent's rendered observations. The qualitative results show a substantial 10.7% improvement in the FID score after our screen-space covariance culling, which supports its effectiveness.

It is important to note that, due to the limited size of the evaluation dataset and the significant difference between the agent's views and the training views, the absolute FID values cannot be directly compared to those commonly reported in generative models evaluation [18, 46]. Instead, the FID score in this context serves as a relative metric for us to better understand the improvements achieved through this covariance culling process.

C. Simulation Environment Setup

Our simulation environment is built based on the Unity [16] physics engine. Our agents, hybrid scene representation,

Figure 8. Vid2Sim Dataset Preview

(a) Before 2D Covariance Culling (b) After 2D Covariance Culling

Figure 9. Screen-space Covariance Culling Comparison

Metric	w/o Culling	w Culling	Improvement (%)
FID [17]	214.47	191.54	+10.70%

Table 4. Comparison of FID scores for rendered images with and without 2D covariance culling.

and static and dynamic obstacles are imported together to form our diverse simulation environments. The ground surface is configured as a horizontal walkable area, while other scene meshes and obstacles are tagged as collidable objects. To maintain visual consistency, the ground and scene meshes are rendered invisible and serve only for collision interactions.

For the agent settings, we position the agent's camera sensors at the front of the robot, matching the real-world sensor placement. The robot size in the simulation is scaled to metric scale with the background scene adjusted proportionally. The agent operates using a bicycle dynamics model, configured with a wheelbase of 0.8 meters and a maximum turning angle of 30 degrees. We introduce random perturbations to the camera's position and rotation to simulate real-world disturbances caused by uneven ground during navigation.

In Fig. 10, We also provide an example of a digital twin environment created using our Vid2Sim pipeline that transforms a real-world environment into a simulation environment with the minimal sim-to-real gap. The agent RGB observation is provided at the top-left corner which replicates real-world observation.

(a) Vid2Sim Simulation (b) Real-world Deployment

Figure 10. Digital-twin environment for real-world scene

D. RL Training Details

In this section, we provide the training details of our RL agents. To enable agent training with popular RL frameworks like OpenAI Gym [6] and Stable-Baselines3 [41], we compiled our Unity environments and integrated them with an environment wrapper to make them compatible with the OpenAI Gym interface. The simulation system is running at 50hz and the RL training is running at 5hz.

During each episode, the agents are randomly initialized from a starting point and need to navigate to another random goal point within the scene. Typically, the distance between the starting point and the goal point is around 10∼30m. For both the PointNav and SocialNav tasks, an agent is considered successful if it reaches the goal within 0.5m.

D.1. Reward shaping

Our reward function for the agent is defined as follows:

$$R = R_{term} + c_1 R_{dist} + c_2 R_{steer} + c_3 R_{crash} + c_4 R_{time}$$

- Terminal reward R_{term}: is a sparse reward set to $+10$ if the agent successfully reaches the destination and -10 if it fails.
- Distance reward R_{dist}: is a dense reward defined as $R_{dist} = d_t - d_{t-1}$, where d_t represents the current distance between the agent to the goal point, which guide agent navigates towards the goal during training. We set the weight $c_1 = 1$.
- Steering smoothness reward R_{steer}: is a regularization reward defined as $R_{steer} = -\|s_t - s_{t-1}\| \cdot v_t$ to penalize inconsistent steer movement during agent navigation. The weight c_2 is set to 0.05
- Crash reward R_{crash} is used to penalize the collision between the agent and other objects, including the environment, static obstacles, and dynamic agents. It's a dense negative reward defined as $-1(c_t)$ where c_t denotes the collision happens at time t and $1(\cdot)$ is the indicator function. We set the weight of R_{crash} as $c_3 = 1.0$.
- Time reward R_{time} is a dense reward defined as $R_{time} = -\Delta t$, between two time steps the R_{time} is simplifies to -1 to encourage efficient navigation by penalizing longer episodes. We set the weight c_4 of R_{time} to -0.1

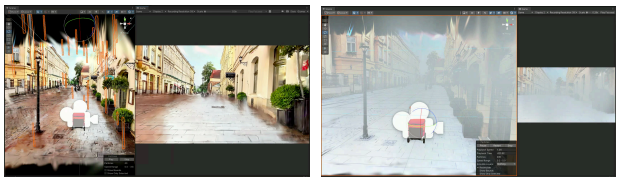
At any time step t if the $R_{term} \neq 0$, the episode will terminate. The agent is considered as failed and receives a $R_{term} = -10$ under the following conditions: 1) Navigating outside the drivable area. 2) Exceeding 3000 time steps in the episode, resulting in a timeout. 3) Accumulating more than 3 collisions within an episode. During the testing, any collision between the agent and other objects will result in a cost $+1$.

D.2. Hyper-parameter settings

We train our agent with 30 parallel environments and the training takes around 15 hours on a single NVIDIA A5000 GPU. We provide our hyper-parameter settings in Tab. 5

SAC Hyper-parameters	Value
Learning starts	10000
τ (Target critic update ratio)	0.005
Discount factor γ	0.99
SDE sample frequency	64
Batch size	256
Learning rate	3×10^{-4}
Use SDE at warmup	True
Use SDE	True

Table 5. SAC Hyper-parameters used in experiments.



Rain Simulation Fog Simulation

Figure 11. Weather simulation with particle systems in Unity

E. Sim-to-real Deployment

For the real-world experiment, we deploy navigation policies trained in Vid2Sim on the same four-wheeled delivery robot used in the simulation. The robot takes RGB images as inputs directly from an onboard camera, which has the same resolution of 1280×720 and the same intrinsic and extrinsic parameters as the sensor specifications in simulation. We resize the current image to 128×72 and stack it with the images from the past 5 timesteps to incorporate the historical information. We then combine the image inputs and the distance and heading angle to the goal point from robot odometry as the policy observation. The action output includes the normalized linear and angular velocities between -1 and 1, and we perform system identification to align the dynamics of the real robot with the simulation by re-scaling the normalized velocities in real-world units, and we use the built-in controller of the robot to convert the velocity commands to the low-level motor controls for actions.

F. Particle Simulation for Weather Editing

Apart from global scene layout editing illustrated in the main paper, we also demonstrate our ability to simulate different weather conditions like rainy and foggy weather through 3D particle simulations within the Unity environments in Fig. 11

G. Limitations and Future works

Though the proposed Vid2Sim framework can support efficient training in simulation environments with fast GS-based rendering and can achieve zero-shot sim2real deployment, building each simulation environment is time-consuming as GS requires GLOMAP [39] to initialize the point cloud and the camera poses, which takes a long time to run. Therefore, we have only collected 30 environments and experiment results have shown training with more environments can lead to better performance. In the future, we will explore more efficient ways to convert the monocular videos to GS-based simulation environments and build a larger real2sim dataset with more diverse environments. We believe such large-scale environments can further benefit the training of a generalizable navigation policy and extend our pipeline to train other embodiments like humanoids and robot dogs.

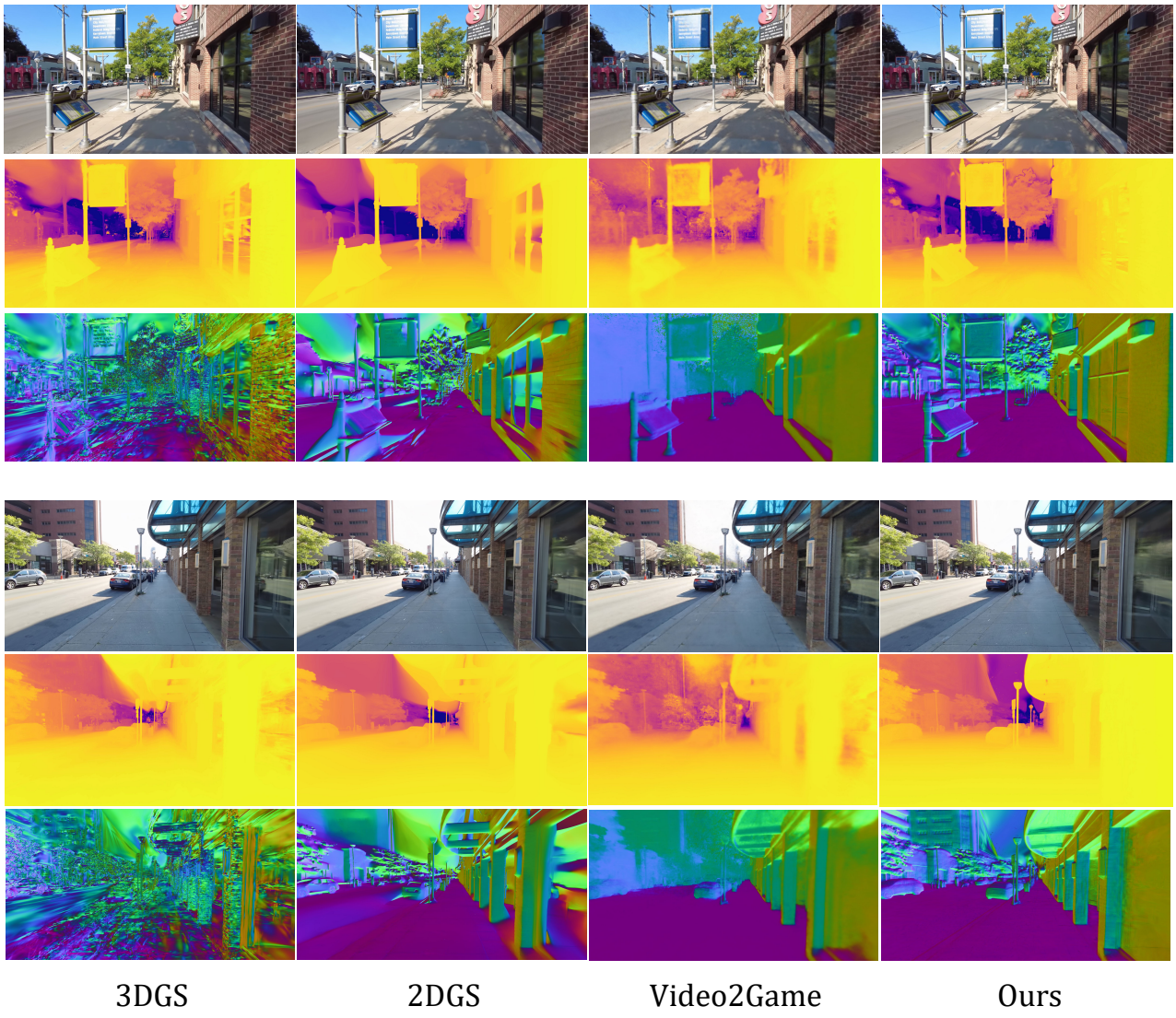


Figure 12. Reconstruction Comparison between different methods on RGB, depth map and normal map