# BUILD REALISTIC AND INTERACTIVE 3D SCENE SIMULATION FROM IN-THE-WILD VIDEOS

BY

ZIYANG XIE

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2025

Urbana, Illinois

Adviser:

Professor David Forsyth

## ABSTRACT

This thesis proposes a unified system for constructing realistic and interactive 3D scene simulations directly from in-the-wild monocular videos. Given unstructured video input, the system reconstructs complete virtual environments that support high-fidelity rendering and real-time physical interaction. The pipeline is designed to enable a broad range of applications, including embodied agent training, sim-to-real transfer, and virtual content creation.

The pipeline addresses key challenges in 3D scene simulation through four core components: (1) a geometry-consistent background reconstruction module that combines Structure-from-Motion, 3D Gaussian Splatting, and learned geometric priors to reconstruct consistent large-scale environments; (2) a tri-branch foreground object modeling framework that supports object-level reconstruction, retrieval from large-scale 3D asset databases, and generation via 3D generative models; (3) a scene composition, relighting and rendering module that ensures photorealistic composition of foreground and background with consistent placement, lighting and shadows; and (4) a simulation layer that enables realistic sensor simulation and interactive physics-based interaction within the simulation environment.

A key feature of the system is its extensibility: each component is designed to incorporate future advances in neural rendering, generative modeling, and geometry reconstruction. This design enables the pipeline to serve both as a practical simulation pipeline and as a research platform for complex, real-world 3D scene simulation. By bridging unconstrained video input with high-fidelity interactive simulation, this work offers a scalable and generalizable framework for building realistic virtual environments from everyday video data.

# TABLE OF CONTENTS

# CHAPTER 1: INTRODUCTION

3D scene simulation is a fundamental problem in computer vision and computer graphics. It includes tasks such as scene reconstruction, object modeling, scene composition, and rendering. Traditional methods often rely on computer-generated imagery (CGI) to manually build 3D scenes. This process is not scalable and the results are often not realistic.

With the increasing availability of monocular video data and recent progress in learning-based scene reconstruction, there is growing interest in generating realistic 3D scenes directly from videos, without relying on complex graphics pipelines or manually designed assets.

In this thesis, I present a comprehensive 3D scene simulation pipeline built upon my previous publications [1, 2, 3] and projects [4]. This work proposes a scalable pipeline that converts monocular video or image into a realistic and interactive 3D simulation environment. The system integrates recent advances in 3D scene reconstruction, neural rendering and foreground-background decomposition to produce high-fidelity and interactive reconstructions which enables various downstream tasks and applications such as embodied agent training and virtual 3D content creation.

## 1.1 MOTIVATION AND PROBLEM DEFINITION

Realistic 3D simulation environments are essential for many computer vision and robotics tasks. Applications such as embodied navigation, autonomous driving, and virtual content creation require controllable and realistic 3D scenes. However, building such environments is expensive. It usually involves collecting dense sensor data or manually designing assets. This limits the scalability and diversity of the simulated scenes.

Monocular videos, on the other hand, are cheap to collect and widely available. They contain rich visual information about real-world scenes. If we can convert monocular video into a high-quality interactive 3D simulation environment, it would provide a low-cost and scalable way to generate training and testing environments for learning-based methods.

The central goal is to convert a monocular video sequence into a 3D simulation environment. Given a video

$$\mathcal{V} = \{I_t\}_{t=1}^{T}, \quad I_t \in \mathbb{R}^{H\times W\times 3}, \tag{1.1}$$

the task is to recover a general 3D scene representation.

$$\mathcal{S} = \{\mathcal{B}, \mathcal{F}, \mathcal{P}\}, \tag{1.2}$$

where $\mathcal{B}$ represents the static background geometry and appearance, $\mathcal{F} = \{F_k\}$ is a set of foreground objects with 3D shapes, appearances and trajectories and $\mathcal{P} = \{P_t\}$ is the set of camera poses corresponding to each frame $I_t$.

The reconstructed scene $\mathcal{S}$ should support: 1) Novel-view rendering from novel camera poses. 2) Foreground object manipulation and insertion. 3) Basic agent-scene interaction for simulation and training. This thesis focuses on building a pipeline that satisfies the above requirements using only monocular video without relying on manual annotations.

## 1.2 RELATED WORK AND KEY CHALLENGES

Reconstructing a usable 3D simulation environment from monocular video involves multiple stages. These include estimating camera poses, reconstructing the scene, extracting geometry surface, and embedding the scene into a simulation engine. Each of these problems has been studied individually, but combining them into a unified and robust pipeline remains difficult. This section reviews representative methods for each sub-task and discusses the limitations they introduce when applied to the simulation task.

### 1.2.1 Pose Estimation from Video Sequences

Camera pose estimation is a prerequisite for most geometry-based reconstruction methods. Traditional structure-from-motion (SfM) methods [5, 6] recover camera parameters and sparse 3D point clouds by feature matching and bundle adjustment to minimize reprojection errors. These methods assume that feature correspondences are reliable and that adjacent frames have sufficient overlap. These assumptions often fail in real video sequences, especially in dynamic scenes or those captured with noises.

To improve robustness in these cases, learning-based methods have been proposed. DUSt3R [7] and MASt3R [8] propose to jointly predicts pair-wise camera parameters and corresponding pointmap using a unified network. MonST3R [9] adapted DUSt3R [7] paradigm to handle dynamic objects by fine-tuning on a small set of dynamic videos. Fast3R [10] and VGGT [11] further improve computational efficiency for long videos by using a transformer structure to replace the time-consuming global post-optimization for pair-wise pointclouds.

These methods make fewer assumptions about the sequence quality and can generalize better to unconstrained settings. However, their accuracy still remains insufficient for downstream high-fidelity 3D reconstruction tasks.

### 1.2.2 Scene Reconstruction and Novel View Synthesis

Given estimated poses, neural scene representations have become the dominant approach for dense reconstruction and novel view synthesis. NeRF [12] models the scene as a continuous volumetric field and can generate photorealistic images under novel viewpoints. 3DGS [13] improves efficiency by representing the scene as a set of 3D Gaussians, enabling fast rendering and better scalability. While these representations achieve high visual quality, they are optimized for rendering and not directly usable in simulation. They lack explicit surfaces, object separation, or support for physical interaction.

### 1.2.3 Surface Reconstruction from Learned Representations

To support downstream simulation tasks such as collision checking and agent interaction, the learned scene must be converted into an explicit surface representation. For representation-wise, signed distance function (SDF) is widely adopted in recent works [14, 15] to enhance geometry representation and surface extraction. For optimization, several works [16, 17, 18] propose to enhance surface extraction by introducing additional constraints based on primitive shapes, geometry consistency, or visibility priors.

These methods aim to preserve the fidelity of the original representation while producing outputs compatible with standard graphics and physics engines that support more advanced simulations and applications.

### 1.2.4 Simulation with Reconstructed Scenes

Recent work has explored the use of reconstructed 3D environments in simulation. UniSim [19] adapts neural representations to support sensor-level closed-loop simulation in driving scenarios. S-NeRF++ [2] further extends the simulation capability with foreground object decomposition and insertion to support customized precise control. Vid2Sim [3] focuses on creating realistic and interactive urban environments for embodied navigation training with a minimal sim-to-real gap.

### 1.2.5 Challenges in 3D Scene Simulation

Despite progress in each component, several key challenges remain. Pose estimation from monocular video is unreliable in scenes with motion blur, occlusion, or low-texture regions. Neural representations such as NeRF [12] and 3DGS [13] do not produce explicit surfaces or object boundaries, which are necessary for downstream tasks. Converting these representations into mesh geometry requires additional steps that may introduce artifacts or lose structural detail. Separating dynamic foreground objects from the static background and re-composing a given foreground object naturally into the background is also difficult.

Furthermore, integrating the reconstructed scenes into simulation engines introduces practical constraints on data format, scale handling, and physics compatibility, which are not addressed in most reconstruction pipelines. Addressing these challenges requires a system that jointly considers reconstruction quality, scene structure, and simulation compatibility.

# CHAPTER 2: PIPELINE OVERVIEW

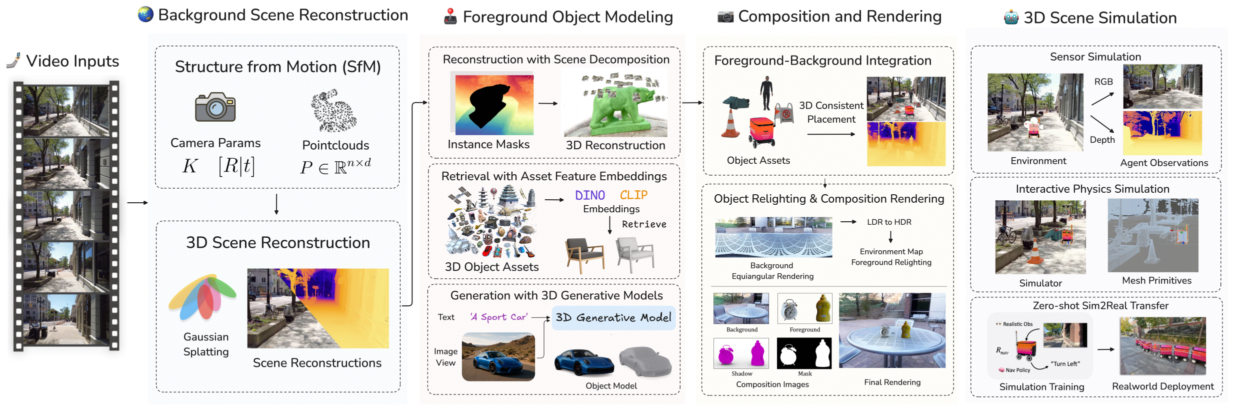

Figure 2.1: Overview of the pipeline.

Figure 2.1 shows an overview of our proposed 3D simulation pipeline. The system consists of 4 main stages: background scene reconstruction, foreground object modeling, scene composition and rendering, and 3D scene simulation.

**1) Background Scene Reconstruction (Chap. 3):** The pipeline starts with a monocular video input. We first apply Structure-from-Motion (SfM) to estimate camera intrinsic $K$ and extrinsics $[R|t]$, where rotation $R \in SO(3)$ and translation $t \in \mathbb{R}^3$, and reconstruct a set of sparse 3D pointclouds $P \in \mathbb{R}^{n\times 3}$. These parameters provide the foundation for background scene reconstruction. We use neural reconstruction methods such as 3D Gaussian Splatting [13] with extra geometry prior to produce a realistic representation of the static scene with dense geometry and realistic appearance.

**2) Foreground Object Modeling (Chap. 4):** For the foreground objects, we support three different modeling approaches: 1) Object Reconstruction with Decomposition (Sec. 4.1): For objects present in the input monocular video, we extract and reconstruct them by decomposing the scene into background and foreground regions using instance segmentation. Each foreground object is reconstructed from its multi-view observations. Scene decomposition allows us to separate these objects from the background, enabling independent manipulation and insertion within the simulated environment.

2) Object Retrieval from Asset Database (Sec. 4.2): For objects belonging to known categories or are difficult to reconstruct due to non-rigid dynamics or occlusion, we retrieve matching 3D assets from large-scale databases (e.g., Objaverse [20, 21]). Given a text or image input, to closely match the object semantics and geometry, we leverage vision-foundation

models (CLIP [22] and DINOv2 [23]) to extract and match their semantic and geometry features based on feature similarity. This approach provides high-quality object geometry and texture without the need for manual modeling.

3) Object Generation with 3D Generative Models (Sec. 4.3): When retrieval fails or novel objects are needed, we use off-the-shelf 3D generative models to synthesize assets. These models take text prompts or single-view images as input and generate diverse, controllable 3D shapes that can be inserted into the scene.

**3) Scene Composition and Rendering: (Chap. 5)** With both the reconstructed background and the modeled foreground objects, we compose them into a unified interactive scene. This process includes 3D-consistent placement of foreground objects within the background geometry. To naturally compose foreground objects with consistent lighting, we use their 3D locations as light probing points. At each location, we render a background environment map from the reconstructed scene. We apply environment map inpainting and LDR-to-HDR conversion to complete missing lighting regions and recover intensity. This environment map captures surrounding illumination and is then enhanced and used to relight the foreground objects and render their shadows accordingly. The final scene is composed and rendered with photorealistic quality, where shadows, blending, and occlusions are handled to maintain both spatial and visual consistency.

**4) 3D Scene Simulation (Chap. 6):** In the final stage, we convert the reconstructed scenes into realistic and interactive simulation environments. Our system supports both RGB and depth sensor simulation to generate realistic visual observations. Foreground objects are embedded with physical properties, allowing interaction and dynamic behavior during simulation. The entire environment can be built directly from monocular videos, forming a scalable real-to-sim pipeline.

We demonstrate this simulation pipeline across both outdoor and indoor scenarios with two practical applications. For outdoor scenes, such as urban environments, we enable embodied agent training and visual navigation tasks. This setup also provides a promising solution for reducing the sim-to-real gap, allowing agents trained entirely in simulation to generalize to real-world environments without additional fine-tuning. For indoor scenes, we demonstrate virtual object insertion that supports realistic object composition and interaction, enabling applications such as virtual product placement for e-commerce and immersive AR/VR content creation.

# CHAPTER 3: BACKGROUND SCENE RECONSTRUCTION

The goal of 3D scene simulation is to create environments that are both realistic and interactive. To achieve this, we first need to reconstruct the background scene. This component serves as the structural foundation of the simulated world. It defines the layout of the space, the shape of the surfaces, and the global appearance of the environment.

In simulation, a well-reconstructed background is important for both visual realism and functional interaction. It provides spatial context for navigation, perception, and physics. It also supports tasks such as object placement, relighting, and agent training. Without a high-quality background, the simulation will appear incomplete and unrealistic.

We aim to reconstruct the background from monocular video inputs. This task is inherently challenging due to unknown camera poses, noisy dynamics, limited viewpoints, and complex geometry and lighting conditions. These factors make it hard to recover a complete and accurate 3D structure. Scenes often include textureless surfaces, occlusions, and changing lighting, which further increase the challenge.

To address this, we design a multi-step pipeline. We begin with Structure-from-Motion (SfM) [5, 6] to estimate camera poses and recover sparse 3D points for initialization. We then apply 3D scene reconstruction methods [1, 3] with geometry prior to reconstruct a high-quality and realistic scene representation. We use neural rendering and geometry-based techniques to supervise the model to improve visual quality and geometry accuracy. Finally, we extract mesh surfaces from the reconstructed geometry to enable physics simulation and interaction.

## 3.1 STRUCTURE FROM MOTION

We first apply an off-the-shelf Structure-from-Motion (SfM) method to estimate the camera poses and recover sparse 3D scene geometry. Given a set of images or a monocular video input, SfM recovers the camera intrinsics $K$ and the extrinsics $[R|t]$, where $R \in SO(3)$ and a set of sparse pointclouds $P$. SfM works by first detecting keypoints in each image and matching them across views using local descriptors such as SIFT. These correspondences are used to estimate pairwise relative camera poses. It then use a global bundle adjustment to jointly refine the camera parameters and 3D point positions to minimize reprojection errors.

We employ GLOMAP [6], an enhanced version of COLMAP [5] for Structure-from-Motion (SfM). GLOMAP offers improved scalability and efficiency over traditional incremental SfM methods. It achieves accuracy and robustness comparable to COLMAP while being significantly faster.

### 3.1.1 Compared with Learning based Method

Recent works such as DUSt3R [7] and MASt3R [8] propose learning-based approaches for Structure-from-Motion. These methods take a pair of input images and use neural networks to directly predict dense correspondence point maps in the coordinate system of the first image. Despite their super simple and straightforward design, learning-based SfM methods have shown competitive performance compared to traditional methods and perform particularly robust in noisy real-world conditions, including low-texture regions, lighting variation, and large viewpoint changes.

In this section, we compare these methods with our SfM-based pipeline in terms of camera pose estimation and 3D reconstruction quality. This comparison helps justify our design choice to adopt a geometric SfM method (GLOMAP [6]) with dynamic masks filtering. Our results show that while learning-based methods are promising, traditional geometric pipelines with strong filtering and optimization still offer better consistency and accuracy in diverse scenes and set a solid foundation for dense 3D scene reconstruction.

To quantitatively evaluate the camera pose reconstruction performance of different methods on in-the-wild videos. We tested the method's performance on SiT [24] dataset which features dynamic, real-world trajectories through busy daily environments. This introduces realistic challenges such as motion blur, occlusion, and scene dynamics.

As shown in Table 3.1, our method GLOMAP† achieves the lowest relative pose error (RPE) in both translation and rotation. COLMAP fails to register consistent poses under these dynamic scenes, while DUSt3R and MASt3R perform worse than GLOMAP†. This is mainly because MASt3R and DUSt3R are trained only on static scenes and tend to struggle when applied to dynamic environments with moving objects and occlusions. With dynamic object masks generated as described in Sec. 3.2.3, our method is more robust and accurate when dealing with dynamic motions. The masks allow our method to focus on static background regions, improving pose estimation stability and filtering out misleading signals from dynamic foreground elements.

| **Methods** | **RPE (Translation) ↓** | **RPE (Rotation) ↓** |
|---|---|---|
| COLMAP [5] | Failed | Failed |
| DUSt3R [7] | 0.125 | 0.695 |
| MASt3R [8] | 0.060 | 0.597 |
| GLOMAP† | **0.059** | **0.535** |

Table 3.1: SfM evaluation on the SiT dataset. We report RPE for translation and rotation over 5 video sequences of scenes with 200 frames each. † represents using dynamic mask

In addition, both methods rely on a computation-heavy global optimization process, which is a fully connected graph in $\mathcal{O}^2$ complexity. When processing long video sequences, they often run into excessive runtime overhead and high memory consumption, making them impractical for reconstructing large-scale scenes.

### 3.1.2 Discussion

Although DUSt3R [7] and MASt3R [8] do not perform well in our current settings, the direction of learning-based Structure-from-Motion remains promising. Recent advances such as MegaSAM [25], CUT3R [26], and VGGT [11] demonstrate strong potential in improving robustness, scalability, and efficiency. These methods offer better handling of textureless regions, large viewpoint changes, and ambiguous motion. With continued progress in dense correspondence learning and efficient optimization, it is promising to replace the current geometric SfM module with these next-generation methods. This could further improve the system's robustness and runtime performance, especially for long videos (>250 frames).

## 3.2 MULTI-VIEW 3D RECONSTRUCTION

After recovering the camera poses and sparse 3D points using SfM, we reconstruct a dense and realistic representation of the scene. The goal is to fill in missing geometry, recover surface details, and produce high-quality appearance across viewpoints. We build a general reconstruction pipeline based on 3D Gaussian Splatting (3DGS) [13] and uses geometric priors to supervise GS training to recover high-quality 3D scenes for simulation.

### 3.2.1 Preliminary: 3D Gaussian Splatting

3DGS [13] 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:

$$\mathcal{G}_i(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i^{-1}(\mathbf{x}-\mu_i)\right). \tag{3.1}$$

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 $\Sigma_i$ during optimization, $\Sigma_i$ 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:

$$\mathbf{c}(x) = \sum_{i \in N} T_i \mathbf{c}_i \alpha_i(\mathbf{x}), \quad T_i = \prod_{i=1}^{i-1} (1 - \alpha_i(\mathbf{x})), \tag{3.2}$$

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:

$$\hat{\mathbf{D}}(x) = \sum_{i \in N} T_i \mathbf{d}_i \alpha_i(\mathbf{x}), \quad \hat{\mathbf{N}}(x) = \sum_{i \in N} \hat{\mathbf{n}}_i \alpha T_i, \tag{3.3}$$

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 can produce visually realistic scenes, it often lacks accurate geometry, overfits the training views, and does not support physical interaction. These issues limit its use in interactive robotics learning. In the next section, we introduce our framework to address these challenges.

### 3.2.2 Reconstruct Background Scene with Geometry Prior

Accurate geometry reconstruction is important for realistic simulation and accurate interactions. However, reconstructing the high-quality 3D structure of a scene from a 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 high-quality environment reconstructions.

To improve geometry reconstruction from monocular input, we introduce a geometry-consistent reconstruction framework, that guides the training of 3D Gaussian Splatting using monocular cues. As shown in Fig 3.1, we extract depth and surface normal maps from the input video using an off-the-shelf depth estimation model [27]. These outputs serve as geometry priors. During training, we render both depth and normal maps from the current 3D Gaussian scene representation according to Eq. 3.3. We then supervise them using the extracted priors with consistency losses. The depth loss $L_{depth}$ and normal loss $L_{normal}$ encourage the rendered geometry to match the predicted structure. We also apply an edge-

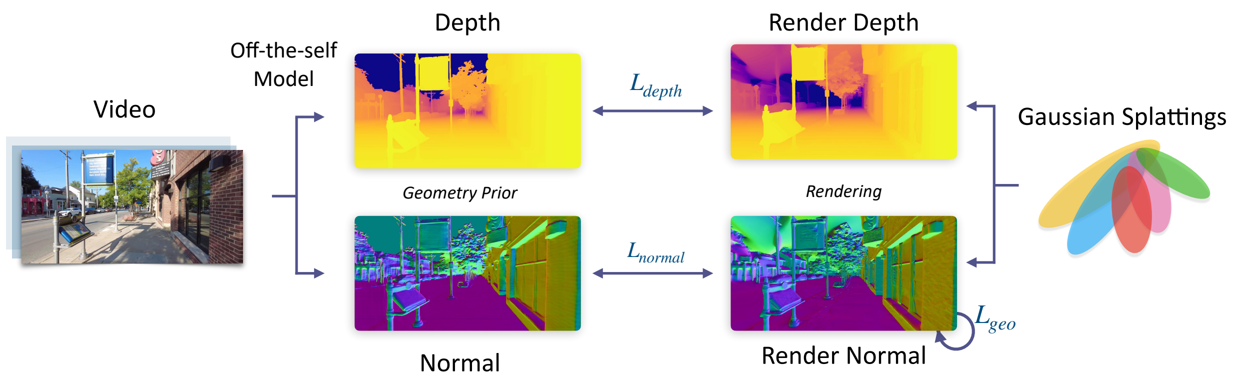

Figure 3.1: Geometry prior for reconstruction supervision.

aware geometry smoothing loss $L_{smooth}$ to ensure consistent geometry reconstruction.

Specifically, we propose to use scale-invariant losses over the depth to tackle that issue. A patch-based normalized cross-correlation (NCC) loss is applied between the rendered depth $\hat{\mathbf{D}}$ and the predicted depth $\mathbf{D}$ for supervision. The patch-based NCC loss evaluates the local similarity between depth maps while being less sensitive to global scale discrepancies:

$$\mathcal{L}_{\text{depth}} = 1 - \frac{1}{\|\mathcal{P}\|} \sum_{p \in \mathcal{P}} \sum_{k=1}^{K^2} \frac{\hat{\mathbf{D}}'_{p,k} \mathbf{D}'_{p,k}}{\hat{\sigma}_p \sigma_p}, \tag{3.4}$$

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 $\sigma_p$ 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:

$$\mathcal{L}_{\text{normal}} = 1 - \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{\hat{\mathbf{N}}_{i,j} \cdot \mathbf{N}_{i,j}}{\|\hat{\mathbf{N}}_{i,j}\| \|\mathbf{N}_{i,j}\|}. \tag{3.5}$$

$\hat{\mathbf{N}}_{i,j}$ 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.

We further improve geometry consistency by introducing a novel edge-aware geometry smoothing loss. This loss encourages smooth surfaces by aligning the normal vectors of neighboring pixels. It helps preserve structural continuity, especially in regions with small depth changes where the surface should remain smooth and connected. The proposed loss $\mathcal{L}_{\text{smooth}}$ is defined as:

$$\begin{aligned} \mathcal{L}_{\text{smooth}} &= \frac{\sum_{i,j} w_{i,j} \cdot \left(1 - \hat{\mathbf{N}}_{i,j} \cdot \hat{\mathbf{N}}_{i+\Delta x, j+\Delta y}\right)}{\sum_{i,j} w_{i,j}}, \\ w_{i,j} &= 1 - \left\| \sqrt{(\nabla_x \mathbf{D}_{i,j})^2 + (\nabla_y \mathbf{D}_{i,j})^2} \right\|. \end{aligned} \tag{3.6}$$

where $(i + \Delta x, j + \Delta y)$ 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 [16], 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

$$\mathcal{L}_{\text{scale}} = \frac{1}{N} \sum_{i \in N} \| \min(s_1, s_2, s_3) \|, \tag{3.7}$$

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.

The commonly used photometric loss, which combines L1 and D-SSIM losses between rendered images $\hat{I}$ and ground-truth images $I$, is also applied to supervise Gaussian Splatting training.

$$\mathcal{L}_{rgb} = (1 - \lambda)\mathcal{L}_1(\hat{I}, I) + \lambda \mathcal{L}_{D-SSIM}(\hat{I}, I) \tag{3.8}$$

Our final optimization loss can be defined as three parts, the photometric loss for RGB rendering $\mathcal{L}_{rgb}$, the geometry consistency loss for geometry regularization $\mathcal{L}_{geo} = \mathcal{L}_{depth} + \mathcal{L}_{normal} + \mathcal{L}_{smooth}$ and a scale loss term to regularize splats shape $\mathcal{L}_{scale}$. The final loss combination can be defined as:

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_{\text{rgb}} + \lambda_2 \mathcal{L}_{\text{geo}} + \lambda_3 \mathcal{L}_{\text{scale}}. \tag{3.9}$$

where $\lambda_i,\ i \in \{1, 2, 3\}$ is the corresponding loss weight to balance the loss term.

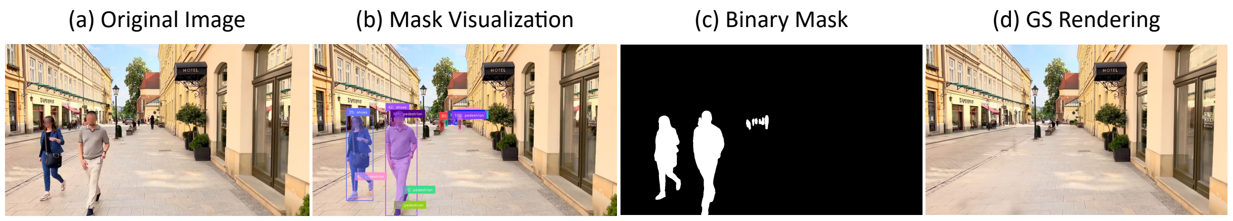

Figure 3.2: **Dynamic Mask Visualization:** We visualize the (a) Original Image (b) the dynamic objects being detected and segmented by the SAM2 [31] model, (c) the binary mask being extracted for reconstruction training and (d) the final rendering of the clean static background scene from the GS reconstruction.

### 3.2.3 Dynamic Object Handling

In most real-world videos that are captured from daily life or collected from the web, may contain many dynamic objects such as people, vehicles, and animals. These moving elements introduce noise into the scene and make it difficult for 3D reconstruction algorithms to produce clean and consistent results.

Current 3D reconstruction methods [12, 13] are designed for static environments. They assume that the scene remains unchanged across different views. When dynamic content is present, these methods often produce artifacts such as ghosting, floaters, or incorrect geometry. Although some 4D reconstruction techniques [28, 29, 30] exist, they typically only work on simple scenes and still struggle to recover complex dynamic motions accurately.

To address this issue, we filter out dynamic regions during the optimization process and focus only on reconstructing the static background for simulation while filtering out the dynamic parts. We use a dynamic object mask to exclude noisy regions from the reconstruction. To generate this mask, we apply an off-the-shelf video instance segmentation model [31] to each frame. With the help of prior semantic labels, we identify and segment dynamic objects in the scene. This allows us to isolate the background and prevent dynamic content from affecting the reconstruction results. As mentioned in Sec. 3.1.1, we also apply the same dynamic masks during the SfM stage to make the algorithm more robust and produce more accurate camera poses by focusing only on static features.

As shown in Fig 3.2, By removing the influence of dynamic objects, we obtain a clean and consistent background reconstruction. This provides a robust foundation for later steps such as foreground object modeling, relighting, and interaction.

| Methods | Rendering Quality | | | Simulation Capability | | |
|---|---|---|---|---|---|---|
| | PSNR ↑ | SSIM ↑ | LPIPS ↓ | Real Time | Interactive | RL Training |
| Instant-NGP [32] | 27.50 | 0.827 | 0.240 | ✗ | ✗ | ✗ |
| 3DGS [13] | 31.85 | 0.921 | 0.136 | ✓ | ✗ | ✗ |
| 2DGS [16] | 30.82 | 0.915 | 0.154 | ✓ | ✗ | ✗ |
| Video2Game [33] | 28.32 | 0.834 | 0.275 | ∼ | ✓ | ✗ |
| Ours [3] | **32.41** | **0.927** | **0.127** | ✓ | ✓ | ✓ |

Table 3.2: Comparison of rendering quality and simulation capability between our method and other methods. Our method achieves the highest reconstruction quality among other methods and offers the most comprehensive simulation capabilities.

### 3.2.4 Experiment and Results

We evaluate the effectiveness of our scene reconstruction pipeline on in-the-wild videos using monocular inputs. The dataset contains 30 diverse complex outdoor 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. The goal is to reconstruct a dense and realistic 3D scene that captures both geometry and appearance. We focus on comparing different reconstruction methods in terms of visual quality, geometric accuracy, and simulation usability.

As shown in Tab. 3.2, our method consistently outperforms all compared methods in terms of PSNR, SSIM, and LPIPS. In Fig. 3.3, we further show the qualitative comparison between our method and other methods in RGB, depth, and surface normal rendering to compare the reconstruction quality between different methods. Our method can reconstruct finer texture and more accurate geometric details that align with the real-world scene compared with other methods.

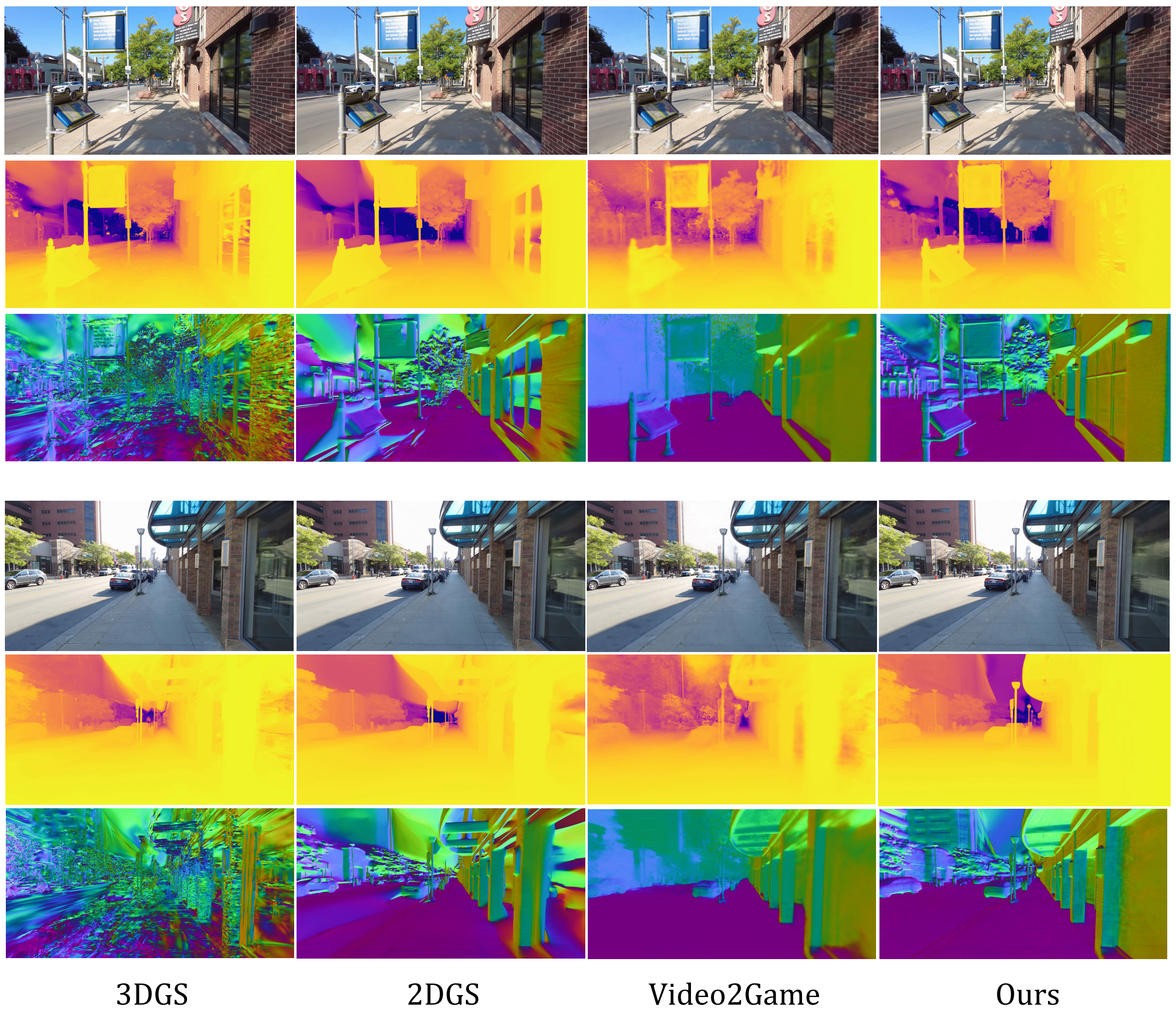

Figure 3.3: Reconstruction Comparison between different methods on RGB, depth map and normal map. Our method shows more accurate and consistent results.

## 3.3 MESH SURFACE RECONSTRUCTION

While neural representations such as 3DGS offer high visual quality and real-time rendering, they do not provide an explicit mesh structure. For physics-based simulation and interaction, a mesh surface is often required. Meshes allow physical reasoning, collision detection, and integration with simulation engines.

To obtain a mesh, we convert the reconstructed scene into a signed distance field. We first render depth maps from multiple viewpoints across the scene. These depth maps are fused into a volumetric truncated signed distance field (TSDF [34]). Each voxel stores the signed distance to the nearest surface, with truncation to handle noise and occlusions. We apply a fusion algorithm to accumulate observations from different views and build a consistent SDF volume. After fusion, we extract the final surface using the Marching Cube [35] algorithm.

### 3.3.1 Results

In Fig. 3.4, we also visualize and compare the normal rendering and reconstructed meshes quality between our methods and previous works. Our method generates cleaner and more complete surfaces, which are suitable for downstream simulation tasks. We observe improved geometric continuity, fewer artifacts, and higher fidelity.

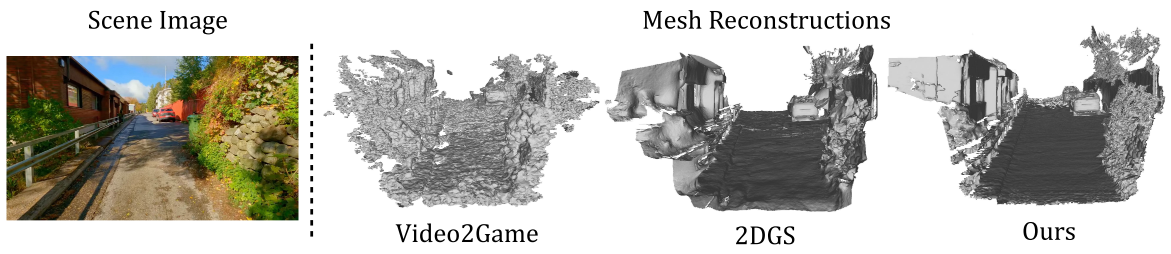

Figure 3.4: Mesh Reconstruction Comparison

# CHAPTER 4: FOREGROUND OBJECTS MODELING

Apart from the background scene, foreground objects also play a key role in creating interactive and controllable 3D simulations. These objects can be broadly categorized into two types: static object such as bottles, furniture, and traffic signs, and dynamic objects such as moving vehicles and pedestrians. To model these diverse objects, we support three approaches: direct reconstruction from the input video, retrieval from 3D asset databases, and generation using 3D generative models.

Each method is suited for different use cases. Reconstruction is effective when the object is mostly rigid and clearly observed across multiple views in the input sequence. Retrieval is preferred when the object belongs to a known category with available assets, or when its motion is complex and hard to reconstruct accurately. Generation is used as a fallback when neither reconstruction nor retrieval can produce a suitable 3D model.

We first apply video instance segmentation models such as SAM2 [31] to decompose the input scene into foreground and background. Then, based on the object property, we choose one of the three modeling methods. The final reconstructed, retrieved, or generated 3D models are then composed into the scene for rendering and interaction (Sec. 5).

## 4.1 OBJECT RECONSTRUCTION WITH SCENE DECOMPOSITION

When a foreground object is mostly rigid and visible across multiple frames in the input video, we aim to reconstruct it directly from the observed images. This approach is both practical and efficient, especially when modeling real-world scenes without relying on external databases. Reconstructing such objects provides high-fidelity geometry and appearance, and enables their physical placement and manipulation within the simulated environment.

We categorize the reconstruction cases based on object motion—whether the object remains static or exhibits dynamic motion relative to the camera.

### 4.1.1 Static Objects

For static foreground objects such as bottles, chairs, parked cars or road signs, we adopt a straightforward approach. After instance segmentation separates the object from the background, we treat it similarly to how we reconstruct the background scene. Specifically, we apply a mask to remove background pixels and reconstruct only the foreground object using neural rendering methods such as NeRF [12] or 3D Gaussian Splatting [13].

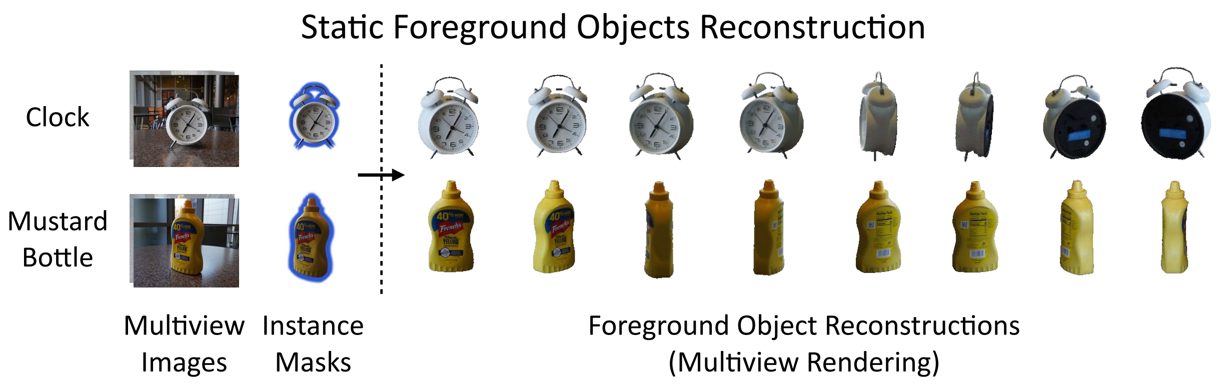
Static Foreground Objects Reconstruction
Clock
Mustard Bottle
Multiview Images
Instance Masks
Foreground Object Reconstructions (Multiview Rendering)

Figure 4.1: **Static Foreground Object Reconstruction:** Given multi-view RGB images and instance masks, we reconstruct clean foreground of static objects such as a clock and a mustard bottle. The resulting models can be rendered from novel viewpoints and inserted into simulation environments.

This approach follows the strategy used in our previous work FieldFusion [4], where static objects are reconstructed in the world coordination system using masked RGB images and known camera poses. This process results in a clear foreground object reconstruction that can later be inserted into any simulated scene with precise control over its position and appearance.

**Results** Figure 4.1 shows qualitative results of our static foreground object reconstruction pipeline. Given a set of multi-view RGB frames and their corresponding instance masks, we can reconstruct clean Gaussian splats for different static items. The reconstructed models preserve fine details and complete geometry from various angles. In this example, we show a clock and a mustard bottle reconstructed from short video sequences. Despite having limited views and cluttered backgrounds, our system accurately recovers both the shape and texture of the foreground objects, demonstrating its effectiveness in isolating and modeling static items from real-world captures.

In Table 4.1, we compared the 3D Gaussian-Splatting [13] with the NeRFacto [36] methods of their foreground reconstruction quality. We segment out the foreground object and only evaluate the foreground region instead of comparing the entire image with ground truth with its background. This approach ensures a more precise evaluation of the foreground reconstruction. We assess the quality of foreground reconstruction using two real-world objects: a clock and a mustard bottle. Both objects are captured under natural lighting and in cluttered indoor environments, providing a realistic benchmark for evaluating reconstruction performance in the wild.

In our evaluation, 3D Gaussian Splatting demonstrates superior reconstruction quality compared to NeRFacto, particularly in capturing fine geometric and texture details. Nonetheless, the NeRF-based method still produces sufficiently high fidelity results for downstream scene composition, achieving PSNR scores above 30.

| Method | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
|---|---|---|---|
| NeRFacto [36] | 31.12 | 0.97 | 0.028 |
| 3DGS [13] | **39.04** | **0.99** | **0.012** |

Table 4.1: Static Foreground Object Reconstruction Results:

### 4.1.2 Dynamic Rigid Objects

While static foreground objects can be directly reconstructed from multi-view observations within the scene, in-the-wild videos often contain lots of dynamic objects. (e.g. moving cars) At first glance, reconstructing such moving objects may seem infeasible using standard 3D techniques and require complex 4D scene reconstruction methods. The motion introduces inconsistencies that violate static scene assumptions and make optimization difficult. However, we can actually bypass this problem and address this challenge by converting the scene into a 'static' one through a coordinate transformation process.

As described in our previous work S-NeRF [1], to handle dynamic rigid objects, we can track their motion over time using a 3D object detector, which provides the per-frame pose of each object. Let $P_b$ denote the bounding box pose of the target object, and let $P_i$ denote the pose of camera $i$. To express the camera pose relative to the target object, we transform the coordinate system by setting the object's center as the origin:

$$\hat{P}_i = (P_i P_b^{-1})^{-1} = P_b P_i^{-1}, \quad P = \begin{bmatrix} R & T \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{4.1}$$

Here, $P$ represents the original transformation matrix. $\hat{P}_i$ is the new relative transformation from the camera to the object center.

As illustrated in Fig. 4.2, we show an illustration of this transformation process. With this transformation, the global camera trajectory is converted into the object's local coordinate frame, where the object remains stationary while the camera moves around it—thus turning a dynamic object into a static one from the reconstruction's point of view.

Once transformed, we treat the object as if it were static and apply the same reconstruction pipeline used for static objects to reconstruct it.

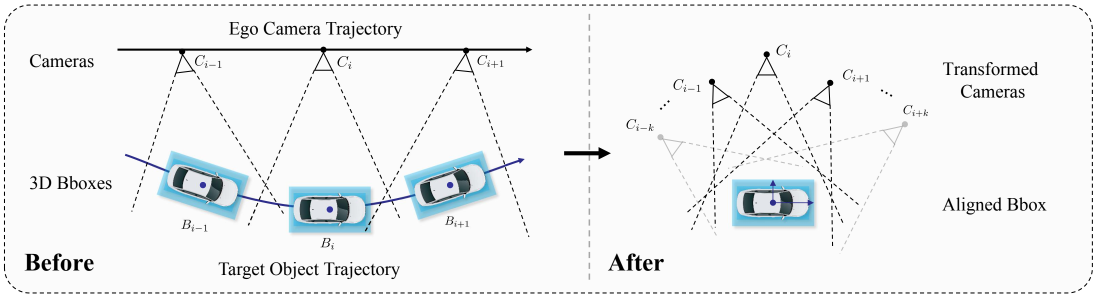

Figure 4.2: **Camera transformation for dynamic object reconstruction:** Left: In the original world coordinate system, the object moves across frames, causing view inconsistency. Right: Estimating the object's 3D bounding boxes over time and aligning them, we transform the camera poses into a object-centric coordinate system ready for reconstruction.

**Results** To further demonstrate the effectiveness of our method, we apply it to reconstruct dynamic vehicles from monocular driving videos in the nuScenes [37] dataset, which is a widely use self-driving dataset where containing abundant dynamic rigid objects, particularly moving vehicles.

Since conventional 3D reconstruction methods fail to recover dynamic foreground, To validate the effectiveness of our approach, we compare our method against GeoSim [38] a method that reconstructs dynamic objects using aggregated LiDAR point clouds and category-specific shape priors. We provide both qualitative and quantitative comparisons in Fig. 4.3 and Tab. 4.2

Our experiments show that, even without the extra data requirements, our method can yield consistent and high-fidelity object reconstructions that significantly outperform GeoSim [38]. The resulting models preserve the accurate shape, texture, and identity of the vehicles, while the GeoSim fails to. These results highlight the ability of our system to recover dynamic foreground assets directly from monocular videos, making it more scalable and generalizable for in-the-wild video inputs.

| **Methods** | **PSNR↑** | **SSIM↑** | **LPIPS↓** |
|---|---|---|---|
| NeRF [12] | – | – | – |
| GeoSim [38] | 12.24 | 0.623 | 0.322 |
| Ours [1] | **18.00** | **0.736** | **0.226** |

Table 4.2: Comparison on Moving Vehicles

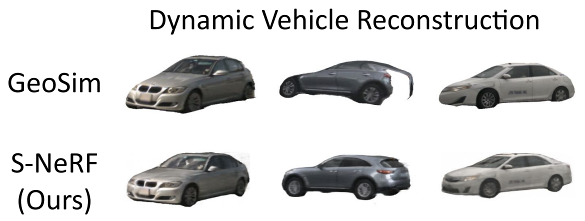

Figure 4.3: Moving Vehicles Reconstruction

## 4.2 OBJECT RETRIEVAL WITH ASSETS FEATURE EMBEDDINGS

While object reconstruction provides high-fidelity geometry when the object is well observed and mostly rigid, it is not always feasible. In many real-world scenarios, foreground objects may appear only briefly, be heavily occluded, or exhibit complex dynamics (e.g. door open or pedestrian walking). In such cases, direct modeling becomes unreliable or incomplete. To handle these situations, we turn to object retrieval from large-scale 3D asset libraries, which provide a practical and scalable alternative for completing scene composition. Object retrieval is also efficient when the object belongs to a common category with many existing assets, such as cars, chairs, or tables.

We first extract multi-view images of the target object using its instance masks across frames. Given the segmented views, we compute two types of feature embeddings per image. We use CLIP [22] to capture high-level semantic information and DINOv2 [23] to capture fine-grained texture and geometry. We average the per-view features across all frames to obtain one CLIP embedding and one DINOv2 embedding for the object. This joint feature representation reflects both the object's category and its visual appearance.

Due to the large scale of the Objaverse [21] dataset, we implement a two-stage retrieval pipeline. In the first stage, we perform coarse filtering by encoding the caption of each asset using CLIP and comparing it with the CLIP feature of the target object. We compute cosine similarity and select the top-$K$ most similar candidates. This stage efficiently narrows down the search space by leveraging high-level semantic alignment. In the second stage, we perform fine-grained matching based on visual appearance. We precompute and store multi-view DINOv2 features for each candidate asset, then compare them with the query object's mean DINOv2 embedding using cosine similarity. The asset with the highest top 5 similarity is selected as the final retrieval result.

## 4.3 OBJECT GENERATION WITH 3D GENERATIVE MODELS

When neither reconstruction nor retrieval yields a suitable 3D model, we resort to object generation using 3D generative models. This method allows us to synthesize novel foreground assets that match the desired semantics or appearance, enabling full scene composition even when real object instances are missing or incomplete in the input video. Generation fills the long-tail gap, handling rare categories, occluded objects, and previously unseen appearances.

We use an off-the-shelf image-to-3D generation model [39] conditioned on the RGB image of the target object to generate the object 3D mesh with texture. Given a cropped image of the object and its instance mask, the generation model synthesizes a textured 3D mesh that matches the visible shape and appearance of the object. This allows us to recover plausible 3D geometry from a single view.

However, in many real-world scenes, the object is only partially visible due to occlusions or truncation. To improve generation in such cases, we first apply a vision-language model (e.g., Molmo [40]) to caption the object and predict its category. Then, we inpaint the invisible regions of the object using a pretrained image inpainting model [41] conditioned on the generated caption to complete the occluded or missing regions in a realistic and context-consistent way. The inpainted image is then fed into the 3D generation model, enabling more complete geometry synthesis. This pipeline effectively transforms a partial observation into a textured 3D mesh representations that is compatible with both rendering and physical simulation.

**Results** Fig 4.4 shows a single-view object generation example. From an input image, we segment the object and generate a textured 3D mesh using TripoSG [39]. The generated mesh produces consistent views from multiple angles and can be directly used in interactive simulations.

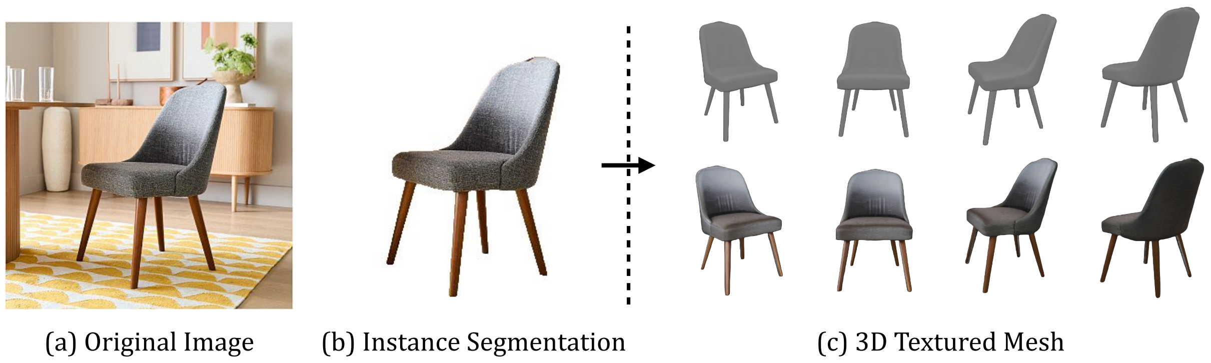

Figure 4.4: Generate Foreground Object 3D Mesh from Image

With these three modeling approaches—reconstruction, retrieval, and generation, our system provides a flexible and generalizable framework for foreground object modeling. Depending on object observability, category familiarity, and motion complexity, we choose the most suitable method to obtain a 3D model that can be placed into the simulation. These objects, once reconstructed or synthesized, are now ready to be integrated into the reconstructed background. In the next chapter, we describe how we compose these elements into a coherent and photorealistic 3D scene that supports rendering and interaction.

# CHAPTER 5: SCENE COMPOSITION AND RENDERING

After reconstructing both the background scene and the foreground objects, the next step is to compose them into a unified simulation environment. This process ensures that the reconstructed elements not only co-exist spatially but also appear visually consistent when rendered together. A realistic simulation requires more than geometry—it demands coherent composition, plausible lighting, and photorealistic rendering.

This chapter introduces our scene composition and rendering pipeline, as illustrated in Fig. 5.1. We begin by describing how reconstructed foreground objects are inserted into the background scene with spatial consistency in Sec 5.1. Then, in Sec 5.2, we discuss how lighting is estimated from the environment and used to relight the inserted objects. Finally, in Sec 5.3 we present our approach to photorealistic rendering, addressing challenges such as shadow casting and handling occlusion.

## 5.1 FOREGROUND-BACKGROUND COMPOSITION

To simulate realistic environments, foreground objects must be placed within the reconstructed background in a way that respects 3D spatial layout. Our system inserts foreground models into the background by aligning them in the world coordinate system. This alignment is performed using the estimated object poses from the original video or desired placements defined in the simulation scenario. We formally define the composed scene at each frame $t$ as:

$$\mathcal{C}_t = \mathcal{B} \cup \bigcup_{k=1}^{K} T_t^{(k)}(F_k) \tag{5.1}$$

where $\mathcal{B}$ is the reconstructed background scene, $F_k \in \mathcal{F}$ is the $k$-th foreground object, and $T_t^{(k)}$ is the spatial transformation that aligns the foreground and background coordination system and places object $F_k$ into the scene at time $t$. $\mathcal{C}_t$ denotes the composed scene at frame $t$, which integrates the static background with all transformed foreground objects under their corresponding placements.

For static objects, the transformation is typically time-invariant. Once reconstructed or retrieved, the object is aligned to the background geometry using a manually specified placement. This placement can be defined in terms of a target 3D position and orientation in the world coordinate system. The same transformation is applied across all time steps. i.e. $T_t^{(k)} = T^{(k)}$, where the object remains fixed within the scene.

For dynamic objects, the transformation varies over time to simulate motion. We define

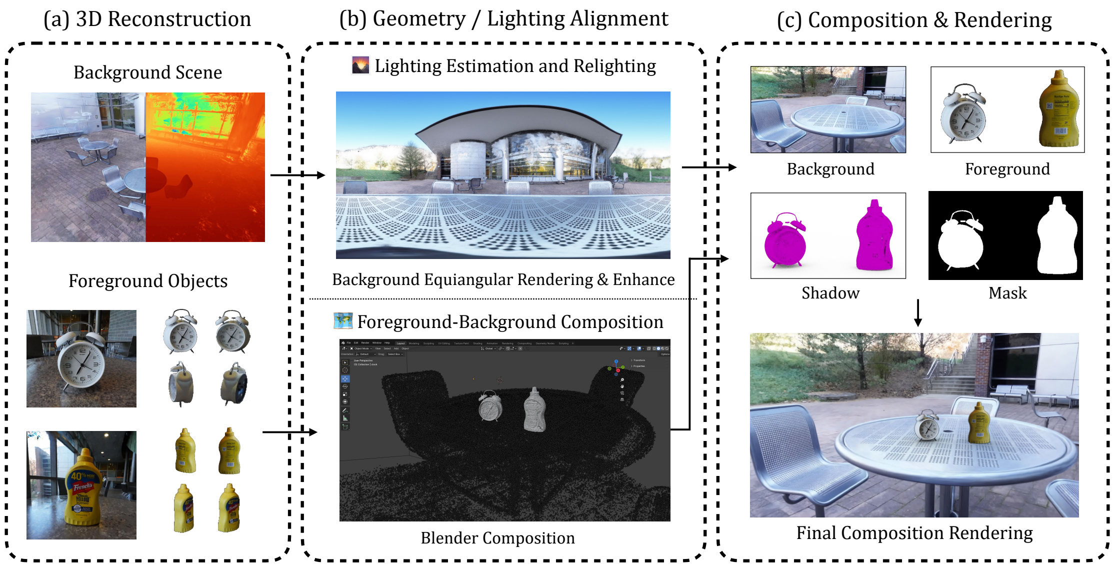
(a) 3D Reconstruction
Background Scene
Foreground Objects
(b) Geometry / Lighting Alignment
Lighting Estimation and Relighting
Background Equiangular Rendering & Enhance
Foreground-Background Composition
Blender Composition
(c) Composition & Rendering
Background
Foreground
Shadow
Mask
Final Composition Rendering

Figure 5.1: Overview of the Scene Composition and Rendering Pipeline: (a) We begin with reconstructed background scenes and foreground objects. (b) Foreground objects are placed into the background with geometric alignment, and lighting is estimated from the environment to ensure consistent relighting. (c) The final composite scene is rendered photorealistically by composing background, relit foreground and its shadows together.

a trajectory for each object in the world coordinate system, specifying how its position and orientation evolve over frames. This trajectory can be derived from the original video via pose estimation or manually scripted to simulate specific behaviors. The object is then inserted frame by frame using the corresponding transformation $T_t^{(k)}$, ensuring smooth motion and accurate rendering from all viewpoints.

A practical engineering problem in foreground-background integration is how to determine the correct transformation and placement for each foreground object. A common approach is to leverage a 3D graphics engine, such as Blender [42] or Three.js [43], to import both the reconstructed background and foreground assets into the same scene. Using the visual interface, users can manually adjust the position, rotation, and scale of each object to ensure proper alignment with the background geometry.

After aligning all foreground objects with the background in the world coordinate system, we render the final scene by composing the background and foreground outputs based on depth, following a z-buffer-like strategy. This can be done either through a 3D simulator or via 2D image-space composition when the foreground representations are implicit or incompatible with the graphic engine (e.g. NeRF [12]) or 3DGS [13]. We describe the rendering process in detail in Section 5.3.

## 5.2 LIGHTING ESTIMATION AND OBJECT RELIGHTING

Once foreground objects are spatially integrated into the background scene, a key factor for visual realism is lighting consistency. Without proper relighting, inserted objects often appear visually detached from the scene due to mismatched illumination and shading.

Fortunately, the 3D reconstruction of the background can serve as an effective source for light probing to support foreground relighting. Since the background geometry and appearance capture how light interacts with the scene, we can use it to estimate local lighting conditions at the intended object placement. Specifically, we render a localized environment map from the background scene at the object's position $p_o$ and and cast rays $\mathbf{r}(\theta, \phi)$ uniformly over the unit sphere using spherical coordinates. For each direction, the radiance $\mathcal{E}(\theta, \phi)$ is computed by accumulating the contributions from visible 3D Gaussians:

$$\mathcal{E}(\theta, \phi) = \sum_{i=1}^{N} T_i(\theta, \phi)\alpha_i(\mathbf{r})c_i(\mathbf{r}) \tag{5.2}$$

where $\alpha_i(\mathbf{r})$ is the opacity of the $i$-th Gaussian splat along the ray, $c_i(\mathbf{r})$ is the view-dependent color of the Gaussian evaluated via spherical harmonics, $T_i = \prod_{j=1}^{i-1}(1 - \alpha_j)$ is the accumulated transmittance before reaching the $i$-th Gaussian splat, $N$ is the number of Gaussians splatted along the ray. This yields a full 360-degree equirectangular environment map centered at $p_o$, capturing the scene's surrounding illumination.

However, there are still two issues remain to be solved before this environment map can be used for lighting-aware object relighting in the next rendering stage. First, the background reconstruction is often incomplete, especially in monocular videos with forward-facing motion. In such cases, the reconstructed geometry may not cover the full 360-degree field of view, resulting in missing regions in the environment map. This leads to underestimation of illumination from unseen directions and can introduce shading artifacts or inconsistencies during relighting. Second, the rendered environment map is typically in low dynamic range (LDR), which does not preserve the true intensity distribution of real-world lighting. Relighting with LDR images fails to capture accurate highlights and shadows, particularly on reflective or specular surfaces. In contrast, high dynamic range (HDR) illumination is required for photorealistic shading.

To address these limitations, we apply two enhancements before using the environment map for relighting: environment map inpainting (Sec. 5.2.1) to fill missing regions and LDR-to-HDR conversion (Sec. 5.2.2) to recover plausible lighting intensities.

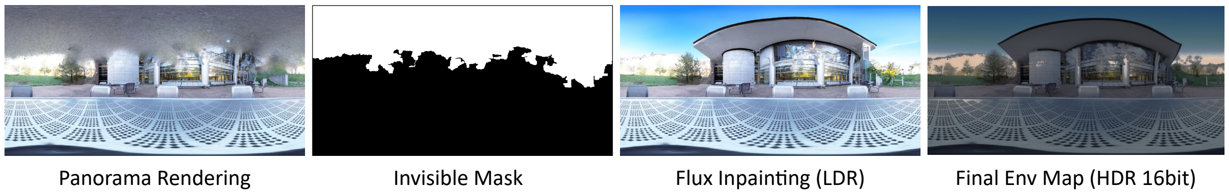

Figure 5.2: Environment map enhancing process

### 5.2.1 Environment Map Inpainting

We first apply environment map inpainting to fill the missing regions and complete the illumination context. The visibility mask, indicating regions of missing or uncertain geometry in the environment map, can be computed based on the accumulated opacity along each rendering ray. Specifically, we define visibility confidence at direction $(\theta, \phi)$ as the total alpha accumulated from the Gaussian splats along the corresponding ray:

$$V(\theta, \phi) = \sum_{i=1}^{N} T_i(\theta, \phi)\alpha_i(\mathbf{r}) \tag{5.3}$$

Pixels in the environment map with $V(\theta, \phi) < \tau$ are considered unreliable and marked as missing, where $\tau$ is a visibility threshold. The resulting binary mask is then used to guide the inpainting process.

For panorama inpainting, we adopt a latent diffusion-based pipeline using Flux-Fill [41] pipeline with a pre-trained lora [44, 45] fine-tuned on panorama images to perform inpainting conditioned on the original panorama image and the visibility mask.

As illustrated in Fig. 5.2, the upper part of the panorama rendering is not visible in the original background reconstruction due to limited vertical field of view. By applying panorama inpainting, we are able to plausibly complete these unseen regions, restoring missing sky, overhead lighting, and global illumination cues. The inpainted environment map better captures the full lighting context needed for realistic object relighting, especially in scenes with sparse camera coverage.

### 5.2.2 LDR to HDR Conversion

After completing the missing regions in the environment map through panorama inpainting, the resulting image still resides in low dynamic range (LDR). However, realistic relighting—particularly for reflective or specular surfaces—requires a high dynamic range (HDR) illumination representation to accurately capture light intensity and contrast.

To address this, we apply an off-the-shelf LDR-to-HDR reconstruction model [46]. The model is trained to convert low dynamic range (LDR) images to high dynamic range (HDR) image with a two stage pipeline. As shown in Fig. 5.2, the output is a 16-bit HDR environment map that retains the completed structure from the inpainting step while reconstructing plausible luminance levels for relighting. The final HDR environment map further serves as the illumination source in our relighting pipeline, ensuring more accurate shading on inserted foreground objects.

## 5.3 PHOTOREALISTIC SCENE RENDERING

With all foreground objects placed and relit in a consistent lighting environment, the final step is to render the composed scene with photorealistic quality. This involves integrating both the foreground and background in a way that respects visibility, occlusion, and shading coherence. To achieve this, we adopt a layered rendering approach that considers the visibility and depth of each element in the scene. We first render the background and each foreground object independently, then compose them based on per-pixel depth to determine visibility. The final output is a seamless image where all visual elements align in lighting, geometry, and perspective, ready for use in simulation.

### 5.3.1 Shadow Rendering

Composing an object into a scene without shadows significantly harms visual realism. Even with accurate lighting and geometry, the lack of shadows makes the object appear disconnected from the environment—often interpreted by the human eye as “fake” or floating. Shadows not only signal contact between objects and surfaces but also convey the direction, intensity, and softness of the light source. To ensure perceptual realism, shadow rendering is a critical part of our scene composition pipeline. As illustrated in Fig. 5.3, adding shadows to the inserted object greatly improves its visual integration, making it appear naturally grounded within the scene.

We simulate shadows based on both the lighting conditions estimated from the background and the geometry of the inserted object. When the foreground object is represented as an explicit mesh, shadow rendering is straightforward using standard graphics engine techniques such as shadow mapping or ray tracing. The environment map rendered from the background (Sec. 5.2) is used as the illumination input, and the engine computes shadows accordingly based on light occlusion. For reconstructed foreground objects represented by NeRF [12] or 3DGS [13], we extract an approximate surface mesh using the marching

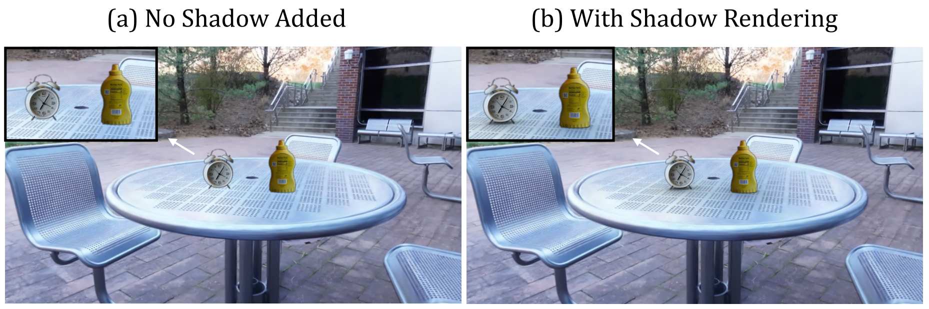

Figure 5.3: **Shadow Comparison:** (a) Direct composition without shadow. (b) Our composition with our shadow rendering. Our shadow rendering technique significantly enhances the quality of the composition.

cubes [35] algorithm. This conversion enables compatibility with standard graphics engines and allows us to reuse mesh-based shadow rendering pipelines. While the extracted mesh may not perfectly capture fine-grained details, it is sufficiently accurate to simulate realistic contact shadows and directional shading under the estimated lighting environment.

The reconstructed foreground mesh is further used to support downstream physics simulation and interaction, enabling realistic object dynamics, collision handling, and embodied agent training within the composed environment.

### 5.3.2 Photorealistic Rendering and Composition

Given the camera pose $P_t \in \mathcal{P}$ at time $t$, we render each foreground object $F_k$ by applying its placement transformation $T_t^{(k)}$ and relighting it using the corresponding environment map $E_t^{(k)}$. The rendering of the foreground object $F_k$ and background scene $\mathcal{B}$ are denoted as:

$$I_t^{(k)} = \mathcal{R}\left(F_k \mid T_t^{(k)}, E_t^{(k)}, P_t\right), \quad I_b = \mathcal{R}\left(\mathcal{B} \mid \mathcal{P}_t\right) \tag{5.4}$$

where $\mathcal{R}$ represents the rendering function.

The shadow image cast by the foreground object is:

$$S_t^{(k)} = \mathcal{S}\left(F_k,\ \mathcal{B} \mid T_t^{(k)}, E_t^{(k)}, P_t\right), \tag{5.5}$$

where $\mathcal{S}$ define the shadow rendering function. We assume that shadows are only applied to the background surface and not to occluded or foreground-covered regions. The shadow-modulated background image $I_t^b$ is then defined by compositing the original background with all object shadows:

$$I_t^b = I_b + \sum_{k=1}^{N} S_t^{(k)} \tag{5.6}$$

To generate the final composed image, we select the visible surface at each pixel by comparing depth values across all rendered elements—this includes the background and multiple foreground objects. Let the background image and its depth be $(I_b,\ D_b)$ and let the foreground objects be rendered as a set of image-depth pairs: $(I_t^k, D_t^k),\ k \in \{1, 2, \ldots, N\}$.

We define the complete set of image-depth pairs at time $t$ as:

$$\left\{ (I_t^{(0)}, D_t^{(0)}), (I_t^{(1)}, D_t^{(1)}), \ldots, (I_t^{(N)}, D_t^{(N)}) \right\}, \quad \textit{where } (I_t^{(0)}, D_t^{(0)}) = (I_t^b, D_t^b) \tag{5.7}$$

For each pixel $(x, y)$ we determine the closest visible surface by finding the minimum depth:

$$k^*(x, y) = \arg \min_{k \in 0, \ldots, N} D_t^{(k)}(x, y) \tag{5.8}$$

The final composed image is then obtained by selecting the corresponding pixel value from the winning layer:

$$I_c^t(x, y) = I_t^{(k^*(x,y))}(x, y) \tag{5.9}$$

This z-buffer-like composition ensures correct occlusion handling and produces a photorealistic scene with consistent depth, lighting, and shadow integration.

### 5.3.3 Results

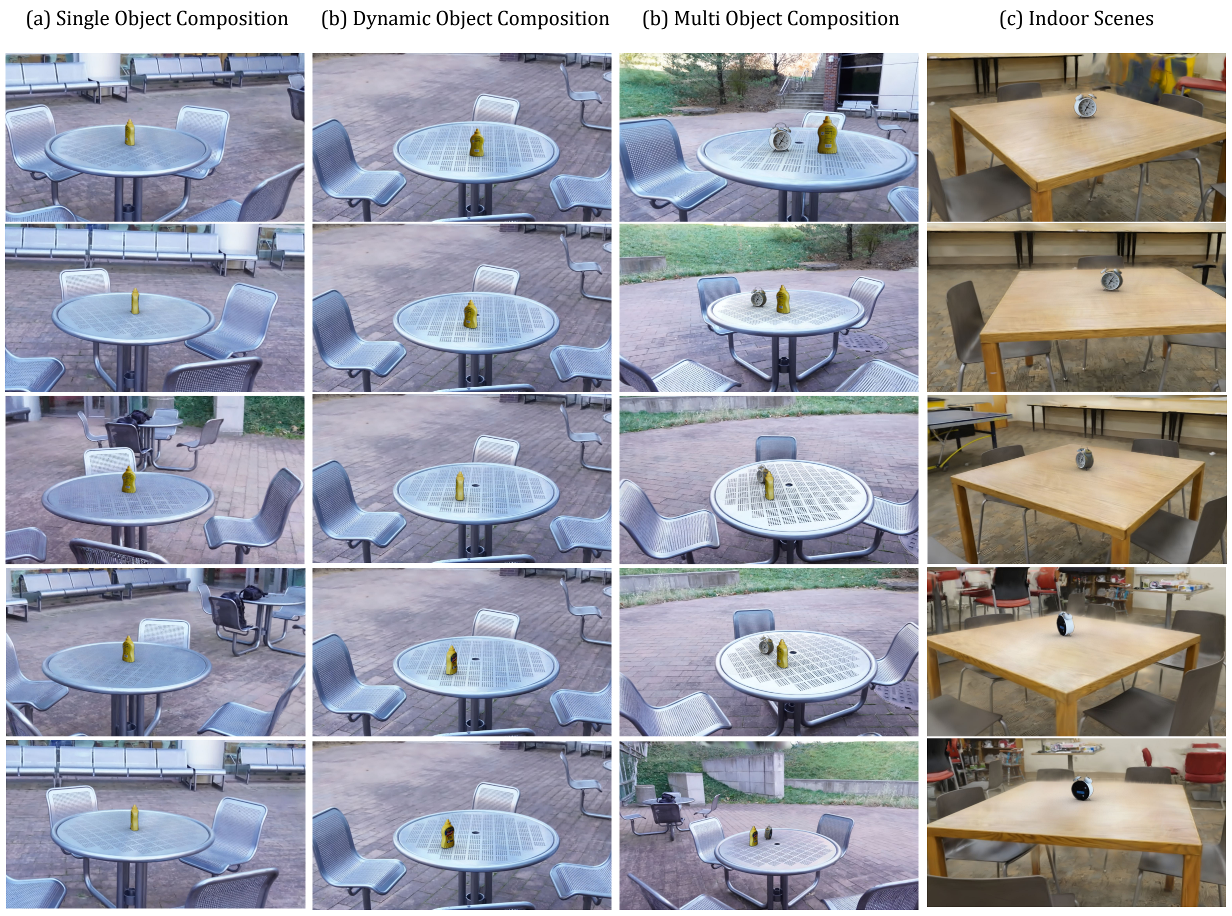
(a) Single Object Composition
(b) Dynamic Object Composition
(b) Multi Object Composition
(c) Indoor Scenes

Figure 5.4: Composition Results

In Fig. 5.4, we showcase several types of composition outcomes: (a) Single Object Composition, where a single static object is integrated into a scene and we change the render camera position, and (b) Dynamic Object Composition, where we keep the camera position fixed while change the object position (c) Multi Objects Composition, which features multiple objects, including scenarios with inter-occlusions among them. Additionally, in (d), we demonstrate the adaptability of our method by compositing our foreground object into a different indoor environment.

Our method enables natural and high-realism composition and blending of foreground objects into background scenes, while also supporting flexible manipulation of object placement, motion, and interaction within the scene.

# CHAPTER 6: 3D SCENE SIMULATION

With the reconstructed background, modeled foreground objects, and the composed scene rendering in place, the final step in our pipeline is to convert the static visual content into an interactive 3D simulation environment. This simulation capability enables the use of real-world videos not just for visualization, but also for downstream tasks such as embodied agent training, object interaction, and scene-aware content generation.

To demonstrate the simulation capabilities enabled by our pipeline, we take our work Vid2Sim [3] as an example. Vid2Sim showcases how monocular video can be transformed into a realistic and interactive simulation environment that supports embodied navigation tasks in urban environments.

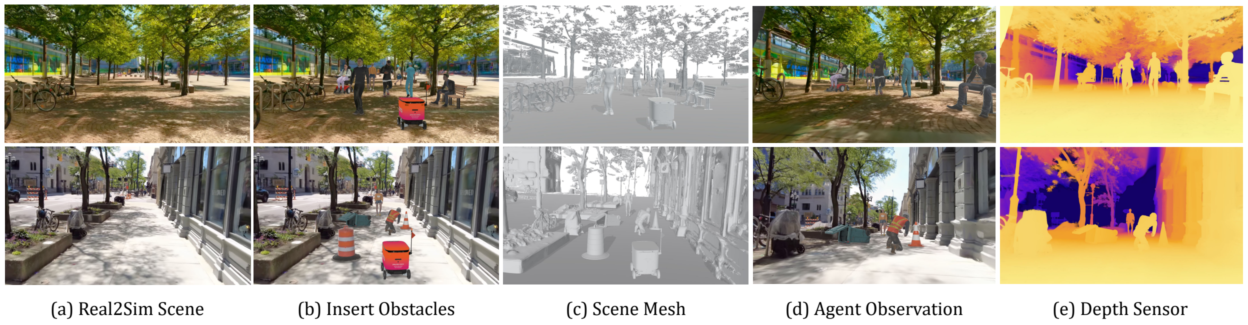

Figure 6.1: Vid2Sim [3] Simulation Pipeline

Figure 6.1 illustrates the full simulation setup from Vid2Sim [3] enabled by our pipeline. Starting from a monocular video, we reconstruct a high-fidelity outdoor scene (a), which serves as the base for generating realistic simulations. Foreground objects such as traffic barrels and delivery robots can be inserted into the scene to create dynamic and interactive scenarios (b). The underlying geometry of both the background and inserted objects is represented as a unified mesh (c), which enables collision modeling and physical simulation. An embodied agent equipped with a virtual camera navigates the scene and captures realistic RGB observations (d), consistent with the geometry and lighting of the environment. Additionally, we simulate depth sensors by extracting the composed depth values from the rendered scene (e), providing geometrically accurate input for downstream perception tasks. Together, these components demonstrate the sensor realism and flexibility supported by our real-to-sim framework.

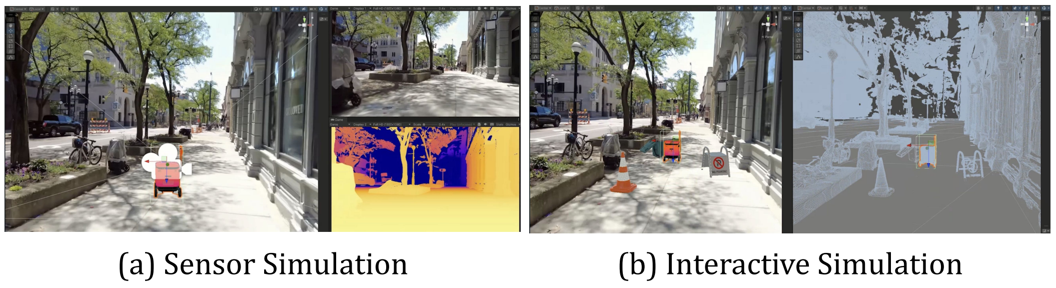

Figure 6.2: Sensor simulation and Interactive scene simulation in Unity [47]

## 6.1 REALISTIC SENSOR SIMULATION

A key aspect of this simulation is the accurate emulation of real-world sensing. Thanks to the realism of our reconstruction pipeline, the simulated RGB and depth observations closely resemble those captured by physical sensors in real environments. By leveraging high-fidelity geometry and appearance reconstructed from monocular video, our system produces sensor outputs that are both visually consistent and geometrically accurate. These simulated sensor outputs provide agents with realistic perception signals that reflect the geometry, appearance, and dynamics of the original scene.

The RGB sensor simulation follows the same rendering and composition process described in Equation (5.9), where foreground and background layers are composed based on depth in a z-buffer-like manner. The depth sensor simulation can be achieved in a similar manner to RGB rendering. After rendering the scene from the agent's viewpoint, we retain the per-pixel depth values corresponding to the closest visible surface. These depth values are extracted from the same multi-layer rendering process used for visual composition, ensuring alignment between geometric and visual observations.

By reusing the index of the closest layer $k^*(x,y)$ defined in Equation (5.9), we can express the composed depth as:

$$D_c^t(x,y) = D_t^{(k^*(x,y))}(x,y) \tag{5.11}$$

The resulting depth maps can be directly used to simulate range sensors such as stereo cameras or LiDAR-like scans, supporting accurate spatial reasoning for downstream tasks.

In Fig. 6.2, we present simulation results rendered within the Unity [47] engine. As shown in Fig. 6.2(a), our environment supports high-fidelity sensor simulation integrated directly into the interactive loop. The top-right view displays the RGB observations captured by the agent's onboard camera, while the bottom-right view shows the corresponding depth map. These sensor outputs are updated in real time as the agent moves through the environment, enabling closed-loop interaction with photorealistic and geometrically accurate feedback.

## 6.2 INTERACTIVE SCENE SIMULATION

While realistic sensor simulation enables agents to perceive the environment in a visually and geometrically consistent manner, passive observation alone is insufficient for many downstream tasks. For applications such as navigation, manipulation, or embodied learning, agents must be able to act within and affect the environment. This requires a simulation system that not only renders visual inputs but also supports real-time interaction, physical dynamics, and response to agent behavior.

To support such active simulation, our system converts the reconstructed scene—including both the background geometry and inserted foreground objects—into simulation-ready assets with physics properties. The background mesh is used to define static physical boundaries and walkable surfaces, while each inserted object is associated with a mesh collider and assigned dynamic properties, such as mass and friction. This setup allows agents to physically interact with the scene. For example, navigating around obstacles or triggering object motion during navigation.

Physics simulation is handled by integrating our reconstructed scenes into modern 3D engines such as Unity [47] or Isaac Sim [48], which provide built-in physics solvers and collision detection. In these environments, we instantiate a controllable agent and attach virtual sensors to match the setup described in Section 6.1. As the agent moves through the scene, the engine simulates physical interactions and returns updated sensory observations in real-time. Crucially, the inserted foreground objects are not just visual decorations but are fully integrated into the simulation. For example, a delivery robot or traffic cone added to the scene can be interacted with, pushed aside, or used as a dynamic obstacle during agent training. The physical realism enabled by our mesh-based reconstruction and composition ensures that these interactions are stable and responsive.

Figure 6.2(b) demonstrates an example of agent-scene interaction. The left side shows a third-person camera view of the agent navigating the environment, while the right side visualizes the mesh layout of the reconstructed scene. In this scenario, the agent physically interacts with a foreground object, a rubbish bin, by making contact and pushing it forward. This interaction is enabled by the physics-aware mesh representation of both the agent and scene components.

## 6.3 DOWNSTREAM APPLICATIONS

The interactive simulation environments enabled by our pipeline open up a wide range of downstream applications. In this section, we highlight two representative applications: (1) embodied navigation learning in large-scale outdoor scenes (Sec. 6.3.1), and (2) virtual object insertion in indoor environments (Sec. 6.3.2). These tasks demonstrate how our simulation system supports both agent-centric learning and content-aware scene editing across diverse environments.

### 6.3.1 Embodied Navigation Learning in Outdoor Scene

Outdoor navigation is a fundamental task in embodied AI, requiring agents to understand spatial layouts, avoid obstacles, and reach goal locations under real-world constraints. To support this, we transform monocular driving videos into interactive outdoor simulations using our Vid2Sim pipeline [3]. The reconstructed city-scale environments are equipped with accurate geometry, realistic appearance, and physical constraints, allowing agents to navigate through complex street layouts.

**Dataset and Experiment Setup.** We evaluate our approach using the Vid2Sim dataset, which contains 30 diverse real-world urban scenes reconstructed from web city-walking videos. Each scene spans 15 seconds of footage at 30 frames per second, resulting in 450 frames per video. The dataset includes a wide range of urban layouts, lighting conditions, and structural complexity to support robust policy training and generalization.

In each training scene, we insert a variety of static obstacles such as traffic cones, poles, and bins, along with dynamic agents like pedestrians using scripted planners. This setup simulates realistic navigation scenarios with both environmental clutter and moving objects. The agent is tasked with navigating from a random start point to a target goal, avoiding collisions along the way.

The agent receives RGB or depth observations from its onboard sensors, along with goal-relative information such as heading and distance. Actions are defined in continuous control space, representing velocity and steering commands. Agents are trained using the Soft Actor-Critic (SAC [49]) algorithm for 1.5 million steps across 30 environments. Performance is evaluated on 5 held-out testing scenes using standard metrics including success rate (SR), success weighted by path length (SPL), and average collision cost.

**Simulation Navigation Results** 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

| Methods | Obs | PointNav |  |  | SocialNav |  |  |
|---|---|---|---|---|---|---|---|
|  |  | SR $\uparrow$ | SPL $\uparrow$ | Cost $\downarrow$ | SR $\uparrow$ | SNS $\uparrow$ | Cost $\downarrow$ |
| Mesh$^\dagger$ | RGB | 48.8% | 0.496 | 0.34 | 43.2% | 0.991 | 1.04 |
| Vid2Sim | 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 6.1: Evaluation of agent navigation performance in simulation. Mesh$^\dagger$ represents only use our reconstructed mesh for visual observation to simulate the mesh-based simulation method [33].

avoid colliding with both static obstacles and other moving pedestrians by adapting its path according to their movements.

As there is no existing method that can convert videos into a functional simulation environment for RL training, we simulate a comparable mesh-based approach using our reconstructed colored mesh representation for agent visual navigation, similar to the approach taken by Video2Game [33] 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 depth observations to better understand the capabilities of different policies. Tab. 6.1 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.

**Realworld Deployment Results** 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. The robot is tasked with navigating through real-world spaces containing static and dynamic obstacles. We evaluate performance across three tasks: *Go Straight*, *Static Obstacle Avoidance*, and *Dynamic Obstacle Avoidance.*

As shown in Tab. 6.2, The Vid2Sim-trained agents consistently outperform mesh-based baselines, especially when trained with a larger number of environments. In the most robust configuration (trained with 30 environments), agents achieve 85% success in *Go Straight*, 65% in *Static Obstacle*, and 55% in *Dynamic Obstacle* tasks. These results highlight Vid2Sim's ability to generate generalizable navigation policies and significantly reduce the sim-to-real

| 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 6.2: 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.

gap. These results demonstrate the potential of our simulation pipeline to support realistic, interactive training environments that transfer effectively to the real world. For more technical details, experimental results, and implementation insights, we refer the reader to the full Vid2Sim paper [3].

### 6.3.2 Virtual Object Insertion in Indoor Scene

While the previous application focuses on navigation in large-scale urban scenes, our simulation pipeline is equally effective for indoor environments. In this setting, the focus shifts from agent mobility to scene-level augmentation and object-aware content manipulation. By leveraging high-quality 3D reconstructions from monocular indoor videos, we enable the insertion of virtual objects into realistic indoor layouts for tasks such as content creation and interactive AR/VR experiences.

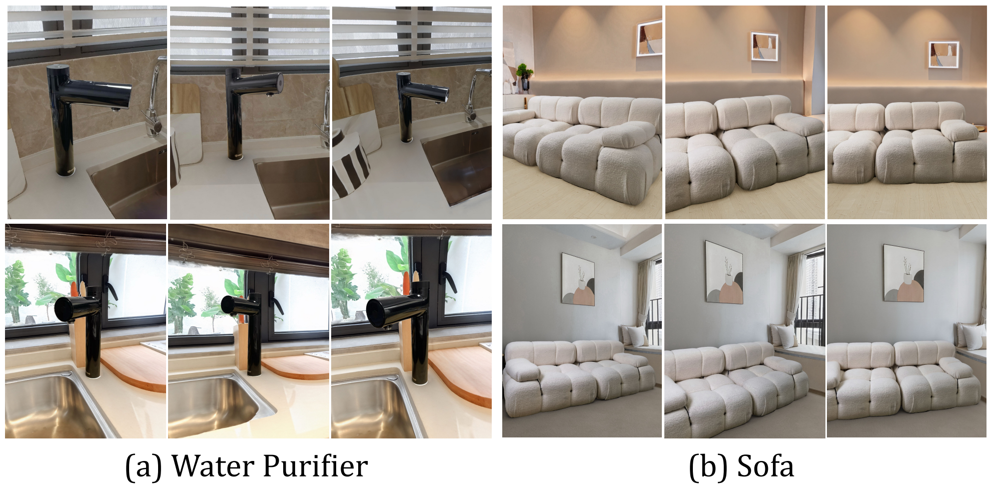

Figure 6.3: Virtual objects insertion in indoor scenes with multiview rendering

We apply our approach to a set of indoor scenes, where the camera explores different rooms. Using the same reconstruction and composition pipeline, we extract the scene mesh and compose virtual assets such as furniture, water purifiers, and small appliances into plausible positions. Fig. 6.3 shows our multi-view composition results. Virtual objects ((a) water purifier, (b) sofa) are inserted into a reconstructed indoor scene and rendered from different views, with consistent lighting and appearance. This shows the flexibility of our system for indoor content augmentation, supporting use cases like product placement, interior design, and training data generation for embodied AI.

### 6.3.3 Discussion

Together, these two applications, outdoor navigation and indoor object insertion, not only span distinct domains and task requirements but also demonstrate the flexibility and generalizability of our simulation pipeline. The outdoor navigation task involves large-scale, dynamic environments where agents must make real-time decisions based on realistic RGB and depth inputs. The challenges in this setting include long-range spatial reasoning, obstacle avoidance, and policy learning under diverse lighting and structural conditions. Our pipeline successfully supports this use case by converting monocular videos into physically grounded simulations with accurate geometry, dynamic object support, and photorealistic sensor simulation. The ability to train and deploy navigation policies that transfer robustly to real-world settings further validates the practical value of the simulation.

In contrast, the indoor object insertion task focuses on scene-level content generation and manipulation in visually constrained and static environments. Here, the emphasis shifts toward visual realism, lighting consistency, and semantic plausibility of object placement. Our system enables seamless integration of virtual assets into reconstructed scenes, producing multi-view consistent results suitable for AR/VR, digital content creation, or dataset augmentation. This highlights the pipeline's strength in precise mesh reconstruction, lighting-aware composition, and fine-grained control over scene layout.

By supporting both agent-based interaction in large-scale, dynamic scenes and precise, content-aware manipulation in static, structured environments, our simulation system demonstrates broad applicability across a wide spectrum of real-world scenarios. This capability opens opportunities for applications in robotics, vision-language grounding, virtual training platforms, interior design, and interactive media.

# CHAPTER 7: FUTURE WORK AND DISCUSSION

While our pipeline effectively reconstructs realistic and interactive 3D scenes from monocular videos, several limitations remain, and there is room for further improvement. Overcoming these challenges is crucial for extending the system's applicability to a broader range of environments and tasks. In this chapter, we highlight three promising future directions for improving the system and discuss potential extensions.

## 7.1 CONTROLLED 3D SCENE GENERATION

Although the current system can reconstruct background scenes from video, it is still constrained by the input data quality and camera coverage. It is difficult to scale beyond what is captured in the video, and especially challenging to synthesize entirely new or unrealistic environments. Extending the pipeline to support 3D scene generation with generative models [50, 51, 52, 53] presents a promising research direction.

One promising direction for integrating generative models into 3D scene reconstruction is to use them as priors for inpainting occluded regions and generating novel views, which helps reduce artifacts and improve completeness. Recent works [54, 55] have explored this approach to enhance the quality of reconstructed scenes.

Another approach is to directly leverage generative models, mostly video generation models [53, 56, 57, 58], for controlled novel view synthesis based on the given scene primitives. These models [59, 60, 61] perform view-consistent inpainting conditioned on sparse 3D point clouds, typically generated from video depth estimation and reprojected using predefined camera trajectories. This enables scene editing and view extrapolation without relying on dense 3D representations such as NeRF or 3DGS.

Such integration opens up new possibilities for structured scene generation. It allows high-level control over spatial layout, appearance, and style, and makes it feasible to construct diverse environments for simulation. Combining generative modeling with structured primitives or geometry-guided representations may provide a scalable and controllable path forward for 3D scene generation.

## 7.2 LEARNING OBJECT PLACEMENT

Another limitation of our current system lies in the integration of foreground objects into the reconstructed background scene. At present, foreground object placement is manually performed in 3D space. This process is not only time-consuming but also requires significant human effort and domain knowledge to ensure physical plausibility and semantic consistency.

Despite the critical role of object placement in simulation and content creation, the academic community has explored this problem only sparsely. The most notable recent progress is Fireplace [62], a system developed by Google that leverages the 3D reasoning capabilities of multimodal large language models (MLLMs) to guide object placement in complex scenes. This work highlights a promising direction—using MLLMs to perform high-level spatial reasoning from natural language and visual cues.

Future extensions of our pipeline could adopt a similar approach by integrating learned object placement modules. These models could take as input the scene geometry, semantic context, and task intent, and output plausible 3D placement proposals. The predictions could be refined with additional constraints such as support surfaces, visibility, and inter-object relationships.

By introducing automatic and context-aware placement strategies, our pipeline can significantly reduce manual effort and improve scene realism and diversity. This capability is especially crucial for generating large-scale simulation environments with dynamic and varied layouts.

## 7.3 SIMULATING NON-RIGID OBJECTS PHYSICS

A key limitation of our current simulation framework is that it only supports rigid-body dynamics. While this is sufficient for basic navigation and scene composition, it significantly limits the realism and applicability of the environment for tasks involving deformable objects or soft material interactions. Many real-world scenarios—especially in robotic manipulation—require reasoning about non-rigid dynamics, such as grasping soft items, folding cloth, or interacting with flexible, human-made environments.

To bridge this gap, future extensions of our pipeline should consider simulating non-rigid physics. Recent advances in differentiable physics engines, particularly those based on the Material Point Method (MPM), offer promising tools for this purpose. MPM-based methods, such as PhysGaussian[63] and PhysDreamer [64], allow accurate simulation of soft materials and support novel view synthesis and simulation for robotics and control applications.

To support real-time training, the integration can be combined with neural surrogates or approximated using hierarchical solvers that prioritize high-resolution updates only where needed. By extending the simulation to include non-rigid physics, our pipeline can better reflect real-world complexity and support learning policies that generalize to more diverse and physically realistic tasks.

# CHAPTER 8: CONCLUSION

In this thesis, we presented a complete pipeline for reconstructing and simulating realistic 3D scenes from monocular videos. Our system combines recent advances in neural rendering, 3D reconstruction, and generative modeling to produce high-fidelity and interactive environments that support agent training and content creation.

We showed how the pipeline handles both background scene reconstruction and foreground object modeling. By integrating structured primitives, mesh reconstruction, and generative models, the system supports scalable and flexible simulation. We demonstrated several applications, including outdoor navigation tasks for embodied agents and indoor scene editing through virtual object insertion. Our experiments validate the effectiveness of the system across a wide range of scenes and tasks. The modular design of the pipeline allows continuous upgrades as new models and techniques emerge, making the framework adaptable to future research and deployment needs.

By bridging video-based reconstruction and interactive simulation, this work takes a step toward creating more accessible and realistic virtual environments from everyday data sources. This foundation supports a broad range of downstream tasks and opens up endless applications in simulation, robotics, virtual reality, and content generation.

## REFERENCES

[1] Z. Xie, J. Zhang, W. Li, F. Zhang, and L. Zhang, “S-nerf: Neural radiance fields for street views,” in *International Conference on Learning Representations (ICLR)*, 2023.

[2] Y. Chen, J. Zhang, Z. Xie, W. Li, F. Zhang, J. Lu, and L. Zhang, “S-nerf++: Autonomous driving simulation via neural reconstruction and generation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2025.

[3] Z. Xie, Z. Liu, Z. Peng, W. Wu, and B. Zhou, “Vid2sim: Realistic and interactive simulation from video for urban navigation,” *The IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2025.

[4] Z. Xie and B. Li, “Fieldsfusion: Harmonious radiance fields composition,” https://github.com/ZiYang-xie/FieldFusion, 2023.

[5] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Conference on Computer Vision and Pattern Recognition*, 2016.

[6] L. Pan, D. Barath, M. Pollefeys, and J. L. Schönberger, “Global structure-from-motion revisited,” in *European Conference on Computer Vision*, 2024.

[7] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, “Dust3r: Geometric 3d vision made easy,” in *The IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2024.

[8] V. Leroy, Y. Cabon, and J. Revaud, “Grounding image matching in 3d with mast3r,” *arXiv preprint arXiv:2406.09756*, 2024.

[9] J. Zhang, C. Herrmann, J. Hur, V. Jampani, T. Darrell, F. Cole, D. Sun, and M.-H. Yang, “Monst3r: A simple approach for estimating geometry in the presence of motion,” in *International Conference on Learning Representations (ICLR*, 2024.

[10] J. Yang, A. Sax, K. J. Liang, M. Henaff, H. Tang, A. Cao, J. Chai, F. Meier, and M. Feiszli, “Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[11] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, “Vggt: Visual geometry grounded transformer,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[12] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “Nerf: Representing scenes as neural radiance fields for view synthesis,” in *ECCV*, 2020.

[13] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, 2023.

[14] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[15] M. Yu, T. Lu, L. Xu, L. Jiang, Y. Xiangli, and B. Dai, "Gsdf: 3dgs meets sdf for improved rendering and reconstruction," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[16] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2d gaussian splatting for geometrically accurate radiance fields," in *SIGGRAPH 2024 Conference Papers*, 2024.

[17] A. Guédon and V. Lepetit, "Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[18] Z. Yu, T. Sattler, and A. Geiger, "Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes," *ACM Transactions on Graphics*, 2024.

[19] Z. Yang, Y. Chen, J. Wang, S. Manivasagam, W.-C. Ma, A. J. Yang, and R. Urtasun, "Unisim: A neural closed-loop sensor simulator," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[20] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi, "Objaverse: A universe of annotated 3d objects," *arXiv preprint arXiv:2212.08051*, 2022.

[21] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre, E. VanderBilt, A. Kembhavi, C. Vondrick, G. Gkioxari, K. Ehsani, L. Schmidt, and A. Farhadi, "Objaverse-xl: A universe of 10m+ 3d objects," *arXiv preprint arXiv:2307.05663*, 2023.

[22] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning (ICML)*, 2021.

[23] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[24] J. Bae, J. Kim, J. Yun, C. Kang, J. Choi, C. Kim, J. Lee, J. Choi, and J. W. Choi, "Sit dataset: socially interactive pedestrian trajectory dataset for social navigation robots," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[25] Z. Li, R. Tucker, F. Cole, Q. Wang, L. Jin, V. Ye, A. Kanazawa, A. Holynski, and N. Snavely, "MegaSaM: Accurate, fast and robust structure and motion from casual dynamic videos," *arXiv preprint*, 2024.

[26] Q. Wang, Y. Zhang, A. Holynski, A. A. Efros, and A. Kanazawa, "Continuous 3d perception model with persistent state," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[27] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, "Depth anything v2," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[28] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, "D-NeRF: Neural Radiance Fields for Dynamic Scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[29] A. Cao and J. Johnson, "Hexplane: A fast representation for dynamic scenes," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[30] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang, "4d gaussian splatting for real-time dynamic scene rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[31] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.

[32] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics*, 2022.

[33] H. Xia, Z.-H. Lin, W.-C. Ma, and S. Wang, "Video2game: Real-time, interactive, realistic and browser-compatible environment from a single video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[34] B. Curless and M. 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.

[35] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *ACM siggraph computer graphics*, 1987.

[36] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa, "Nerfstudio: A modular framework for neural radiance field development," in *ACM SIGGRAPH 2023 Conference Proceedings*, 2023.

[37] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “nuscenes: A multimodal dataset for autonomous driving,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[38] S. D. S. W. X. Y. S. M. S. X. E. Y. R. U. Yun Chen, Frieda Rong, “Geosim: Realistic video simulation via geometry-aware composition for self-driving,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[39] Y. Li, Z.-X. Zou, Z. Liu, D. Wang, Y. Liang, Z. Yu, X. Liu, Y.-C. Guo, D. Liang, W. Ouyang et al., “Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models,” *arXiv preprint arXiv:2502.06608*, 2025.

[40] M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park, M. Salehi, N. Muennighoff, K. Lo, L. Soldaini, J. Lu, T. Anderson, E. Bransom, K. Ehsani, H. Ngo, Y. Chen, A. Patel, M. Yatskar, C. Callison-Burch, A. Head, R. Hendrix, F. Bastani, E. VanderBilt, N. Lambert, Y. Chou, A. Chheda, J. Sparks, S. Skjonsberg, M. Schmitz, A. Sarnat, B. Bischoff, P. Walsh, C. Newell, P. Wolters, T. Gupta, K.-H. Zeng, J. Borchardt, D. Groeneveld, J. Dumas, C. Nam, S. Lebrecht, C. Wittlif, C. Schoenick, O. Michel, R. Krishna, L. Weihs, N. A. Smith, H. Hajishirzi, R. Girshick, A. Farhadi, and A. Kembhavi, “Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models,” *arXiv preprint arXiv:2409.17146*, 2024.

[41] B. F. Labs, “Flux,” https://github.com/black-forest-labs/flux, 2024.

[42] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[43] R. Cabello and contributors, “Three.js – javascript 3d library,” 2010. [Online]. Available: https://threejs.org

[44] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “LoRA: Low-rank adaptation of large language models,” in *International Conference on Learning Representations (ICLR)*, 2022.

[45] S. Yang, J. Tan, M. Zhang, T. Wu, Y. Li, G. Wetzstein, Z. Liu, and D. Lin, “Layerpano3d: Layered 3d panorama for hyper-immersive scene generation,” *arXiv preprint arXiv:2408.13252*, 2024.

[46] S. Sharif, R. A. Naqvi, M. Biswas, and S. Kim, “A two-stage deep network for high dynamic range image reconstruction,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[47] Unity Technologies, “Unity,” 2025, game development platform. [Online]. Available: https://unity.com/

[48] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[49] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *Deep Reinforcement Learning Symposium*, 2017.

[50] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[51] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *The International Conference on Learning Representations (ICLR)*, 2023.

[52] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," *arXiv preprint arXiv:2112.10752*, 2021.

[53] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, V. Jampani, and R. Rombach, "Stable video diffusion: Scaling latent video diffusion models to large datasets," *arXiv preprint arXiv:2311.15127*, 2023.

[54] X. Liu, C. Zhou, and S. Huang, "3dgs-enhancer: Enhancing unbounded 3d gaussian splatting with view-consistent 2d diffusion priors," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[55] K. Schwarz, N. Mueller, and P. Kontschieder, "Generative gaussian splatting: Generating 3d scenes with video diffusion priors," *arXiv preprint arXiv:2503.13272*, 2025.

[56] W. Kong, Q. Tian, Z. Zhang, R. Min, Z. Dai, J. Zhou, J. Xiong, X. Li, B. Wu, J. Zhang, K. Wu, Q. Lin, A. Wang, A. Wang, C. Li, D. Huang, F. Yang, H. Tan, H. Wang, J. Song, J. Bai, J. Wu, J. Xue, J. Wang, J. Yuan, K. Wang, M. Liu, P. Li, S. Li, W. Wang, W. Yu, X. Deng, Y. Li, Y. Long, Y. Chen, Y. Cui, Y. Peng, Z. Yu, Z. He, Z. Xu, Z. Zhou, Z. Xu, Y. Tao, Q. Lu, S. Liu, D. Zhou, H. Wang, Y. Yang, D. Wang, Y. Liu, J. Jiang, and C. Zhong, "Hunyuanvideo: A systematic framework for large video generative models," *arXiv preprint arXiv:2412.03603*, 2024.

[57] WanTeam, A. Wang, B. Ai, B. Wen, C. Mao, C.-W. Xie, D. Chen, F. Yu, H. Zhao, J. Yang, J. Zeng, J. Wang, J. Zhang, J. Zhou, J. Wang, J. Chen, K. Zhu, K. Zhao, K. Yan, L. Huang, M. Feng, N. Zhang, P. Li, P. Wu, R. Chu, R. Feng, S. Zhang, S. Sun, T. Fang, T. Wang, T. Gui, T. Weng, T. Shen, W. Lin, W. Wang, W. Wang, W. Zhou, W. Wang, W. Shen, W. Yu, X. Shi, X. Huang, X. Xu, Y. Kou, Y. Lv, Y. Li, Y. Liu, Y. Wang, Y. Zhang, Y. Huang, Y. Li, Y. Wu, Y. Liu, Y. Pan, Y. Zheng, Y. Hong, Y. Shi, Y. Feng, Z. Jiang, Z. Han, Z.-F. Wu, and Z. Liu, "Wan: Open and advanced large-scale video generative models," *arXiv preprint arXiv:2503.20314*, 2025.

[58] Z. Yang, J. Teng, W. Zheng, M. Ding, S. Huang, J. Xu, Y. Yang, W. Hong, X. Zhang, G. Feng et al., “Cogvideox: Text-to-video diffusion models with an expert transformer,” *arXiv preprint arXiv:2408.06072*, 2024.

[59] X. Ren, T. Shen, J. Huang, H. Ling, Y. Lu, M. Nimier-David, T. Müller, A. Keller, S. Fidler, and J. Gao, “Gen3c: 3d-informed world-consistent video generation with precise camera control,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[60] M. Yu, W. Hu, J. Xing, and Y. Shan, “Trajectorycrafter: Redirecting camera trajectory for monocular videos via diffusion models,” *arXiv preprint arXiv:2503.05638*, 2025.

[61] J. Bai, M. Xia, X. Fu, X. Wang, L. Mu, J. Cao, Z. Liu, H. Hu, X. Bai, P. Wan, and D. Zhang, “Recammaster: Camera-controlled generative rendering from a single video,” *arXiv preprint arXiv:2503.11647*, 2025.

[62] I. Huang, Y. Bao, K. Truong, H. Zhou, C. Schmid, L. Guibas, and A. Fathi, “Fireplace: Geometric refinements of llm common sense reasoning for 3d object placement,” 2025.

[63] T. Xie, Z. Zong, Y. Qiu, X. Li, Y. Feng, Y. Yang, and C. Jiang, “Physgaussian: Physics-integrated 3d gaussians for generative dynamics,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[64] T. Zhang, H.-X. Yu, R. Wu, B. Y. Feng, C. Zheng, N. Snavely, J. Wu, and W. T. Freeman, “PhysDreamer: Physics-based interaction with 3d objects via video generation,” in *European Conference on Computer Vision*. Springer, 2024.