VR-Robo: A Real-to-Sim-to-Real Framework for Visual Robot Navigation and Locomotion

Shaoting Zhu[12*], Linzhan Mou[23*], Derun Li[24], Baijun Ye[13], Runhan Huang[12], Hang Zhao†[123]
[1]IIIS, Tsinghua University [2]Shanghai Qi Zhi Institute [3]Galaxea AI [4]Shanghai Jiao Tong University
[*]Equal contribution †Corresponding author **VR-Robo.github.io**

***Abstract*— Recent success in legged robot locomotion is attributed to the integration of reinforcement learning and physical simulators. However, these policies often encounter challenges when deployed in real-world environments due to sim-to-real gaps, as simulators typically fail to replicate visual realism and complex real-world geometry. Moreover, the lack of realistic visual rendering limits the ability of these policies to support high-level tasks requiring RGB-based perception like ego-centric navigation. This paper presents a *Real-to-Sim-to-Real* framework that generates photorealistic and physically interactive "digital twin" simulation environments for visual navigation and locomotion learning. Our approach leverages 3D Gaussian Splatting (3DGS) based scene reconstruction from multi-view images and integrates these environments into simulations that support ego-centric visual perception and mesh-based physical interactions. To demonstrate its effectiveness, we train a reinforcement learning policy within the simulator to perform a visual goal-tracking task. Extensive experiments show that our framework achieves RGB-only sim-to-real policy transfer. Additionally, our framework facilitates the rapid adaptation of robot policies with effective exploration capability in complex new environments, highlighting its potential for applications in households and factories.**

I. Introduction

Exploring, perceiving, and interacting with the physical real world is crucial for legged robotics, which supports many applications such as household service and industrial automation. However, conducting real-world experiments poses considerable challenges due to safety risks and efficiency constraints. Consequently, training in simulation [1]–[3] has emerged as an effective alternative, allowing robots to experience diverse environmental conditions, safely explore failure cases, and learn directly from their actions. Despite these advantages, transferring policies learned in simulated environments to real world remains challenging, owing to the significant *Sim-to-Real* reality gap [1], [4].

Substantial efforts have been made to narrow the *Sim-to-Real* gap in legged locomotion. Previous studies have employed cross-domain depth images to train agents in simulation, achieving impressive zero-shot policy transfer for both quadruped [5]–[8] and humanoid [9] robots. To further integrate RGB perception into the *Sim-to-Real* pipeline, LucidSim [10] leverages generative models within simulation [1] for visual parkour. However, these depth-based and generation-based visual policies are predominantly constrained to low-level locomotion tasks, primarily because standard simulators struggle to replicate the visual fidelity and complex geometry of real-world environments.

Recently, advanced neural reconstruction techniques such as Neural Radiance Fields (NeRF) [11] and 3D Gaussian Splatting (3DGS) [12] have emerged as promising solutions for creating *Real-to-Sim* "digital twins" from real-world data. Nonetheless, most existing approaches [13]–[17] focus on enhancing photorealism, offering limited support for physical interaction with complex terrains. Moreover, the simulations in these works often lack mechanisms for robust environment exploration, thus constraining their deployment in complex real-world scenarios that demand dynamic interaction.

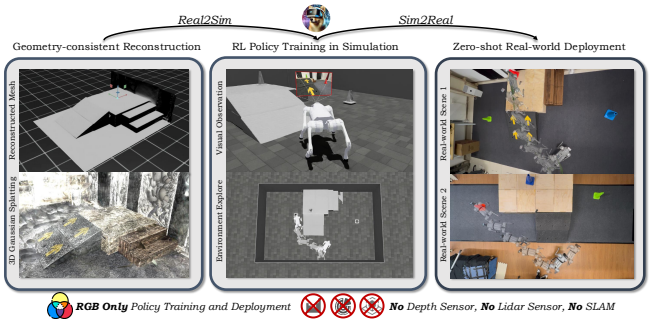

Fig. 1: **Our *VR-Robo* introduces a unified *real-to-sim-to-real* framework.** We generate photorealistic and physically interactive "digital twin" simulation environments for visual policy training, and enable zero-shot real-world deployment.

To address these challenges, we introduce VR-Robo, a novel *Real-to-Sim-to-Real* framework that enables realistic, interactive *Real-to-Sim* simulation and reinforcement learning (RL) policy training for legged robot navigation and locomotion. Given the multi-view images, we employ 3DGS [12] and foundation-model priors [18] to reconstruct geometry-consistent scenes and objects. We then propose a GS-mesh hybrid representation with coordinate alignment to create a "digital twin" simulation environment in Isaac Sim, which supports ego-centric visual perception and mesh-based physical interactions. To enable robust policy training, we introduce an agent-object randomization and occlusion-aware scene composition strategy, as illustrated in Figure 7.

We summarize our contributions as follows:

1) **A *Real-to-Sim-to-Real* framework for robot navigation and locomotion.** We propose to reduce the *Sim-to-Real* gap by training RL policies within a realistic and interactive simulation environment reconstructed only from RGB images captured from real-world.
2) **Photorealistic and physically interactive *Real-to-Sim* environment reconstruction.** We introduce a novel pipeline for transferring real-world environments into simulation using a GS-mesh hybrid representation with coordinate alignment. We further incorporate object randomization and occlusion-aware scene composition for robust and efficient RL policy training.

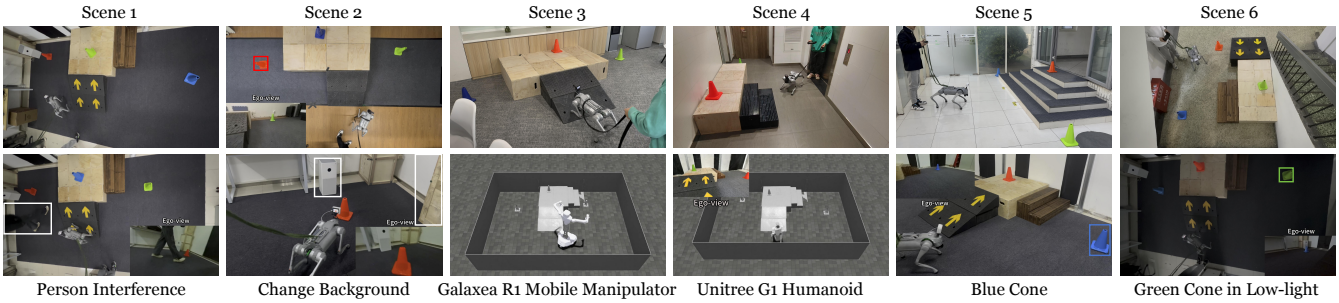

Fig. 2: Diverse experiment settings including scenes, conditions, and robot types (better viewed when zoomed in).

3) **Zero-shot *Sim-to-Real* RL policy transfer.** Through extensive experimental evaluations, we demonstrate that VR-Robo produces effective navigation and locomotion policies in complex, real-world scenarios using RGB-only observations. In addition, we have some empirical findings for RL policy training, including the generalization ability, sample complexity, and architectural choices.

II. Related Works

A. Sim-to-Real Policy Transfer

Transferring reinforcement learning (RL) policies trained in simulation to the real world remains a major challenge due to substantial domain gaps. Traditional simulator-based methods, such as domain randomization [19]–[21] and system identification [22], [23], aim to reduce the *Sim-to-Real* gap by aligning simulations with physical setups. Recent work [24], [25] proposes leveraging conditional generative models to augment visual observations for more robust agent training. LucidSim [10], for example, incorporates RGB color perception into the *Sim-to-Real* pipeline to learn low-level visual parkour by generating and augmenting image background. However, these approaches remain constrained by conventional simulators, which fail to capture the full breadth of real-world physics and visual realism necessary for high-level policy training and real-world deployment.

B. Real-to-Sim Scene Transfer

Recently, advances in scene representation and reconstruction such as Neural Radiance Fields (NeRF) [11] and 3D Gaussian Splatting (3DGS) [12] have facilitated the creation of high-fidelity digital twins that closely replicate real-world environments for *Real-to-Sim* scene transfer. For instance, NeRF2Real [13] integrates NeRF and mesh into a simulation for vision-based bipedal locomotion policy training, but struggles to achieve real-time rendering, terrain climbing, and environment exploration, thus limiting their practical application. Meanwhile, RialTo [26] augments digital twins with articulated USD representations to enable manipulative interactions. More recent works employ 3DGS to generate realistic simulations for both robot manipulation [14]–[16] and navigation [17], [27]. In contrast, VR-Robo is designed to produce photorealistic and physically interactive "digital twin" environments specifically tailored for ego-centric visual locomotion learning.

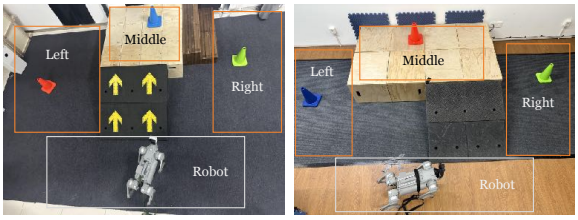

Fig. 3: **Task definition.** The robot aims to reach the target cone of a specified color in a specified spatial location, relying solely on *ego-view* RGB observation and proprioception.

III. Task definition

Our designed task requires both high-level understanding and navigation, as well as low-level motion control. As shown in Figure 3, the task is defined as reaching the target cone of a specified color within a limited time horizon. Each scene features a terrain in the center composed of a high table and a slope or stair connecting the ground to the platform.

At the start of each trial, the robot's initial position is sampled from $\mathbf{p}_{\text{robot}} \sim \mathcal{U}(\mathbb{P}_{\text{robot}})$, and its orientation is similarly randomized as $\theta_{\text{robot}} \sim \mathcal{U}(\theta_{\text{robot}})$. Each scene contains three cones with identical shapes and textures but different colors $C = \{\text{red, green, blue}\}$. These cones are placed in three designated regions: left, middle, and right, with one cone per region at a random position $\mathbf{p}_{\text{cone}} \sim \mathcal{U}(\mathbb{P}_{\text{cone}}^{\text{region}})$. Since the robot may not initially face the target cone, it must explore the environment to identify and locate the correct cone. Additionally, if the target cone is placed atop the table, the robot must navigate up the slope to access the platform.

IV. A Real-to-Sim-to-Real System for Visual Locomotion

A. Geometry-Consistent Reconstruction

Given multi-view RGB images of a scene with corresponding camera poses from COLMAP [28], we aim to generate a photorealistic and physically interactive simulation environment for agent policy training. We decoupledly reconstruct the environment and the object, and then compose them into a single Gaussian scene for policy training.

Gaussian-based 3D Scene Representation. 3D Gaussian Splatting (3DGS) represents the scene as a set of Gaussian primitives as follows:

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

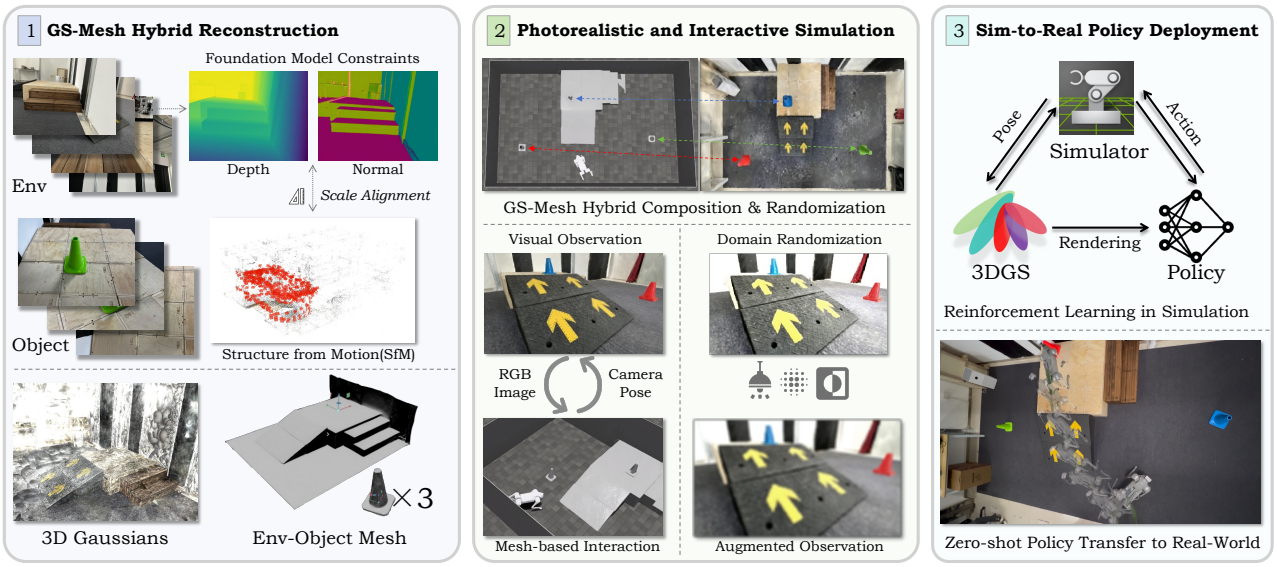

Fig. 4: **VR-Robo *real-to-sim-to-real* framework.** We build a realistic and interactive simulation environment with GS-mesh hybrid representation and occlusion-aware composition & randomization for policy training. Finally, we zero-shot transfer the RL policy trained in simulation into the real robot for ego-centric navigation and visual locomotion.

During optimization, Σ_i is reparametrized using a scaling matrix $S_i \in \mathbb{R}^3$ and a rotation matrix $R_i \in \mathbb{R}^{3\times3}$ as $\Sigma_i = R_i S_i S_i^T R_i^T$. The color value of a pixel $\mathbf{c}(x)$ can be rendered through a volumetric alpha-blending [12] process:

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

where $\alpha_i(\mathbf{x}) = o_i \mathcal{G}_i(\mathbf{x})$ denotes the alpha of the Gaussian $\mathcal{G}_i$ at $\mathbf{x} \in \mathbb{R}^3$. Meanwhile, the view-dependent color c_i of $\mathcal{G}_i$ is represented by its spherical harmonics (SH) coefficients.

Flattening 3D Gaussians for Geometric Modeling. Inspired by [29], we flatten the 3D Gaussian ellipsoid with covariance $\Sigma_i = R_i S_i S_i^T R_i^T$ into 2D flat planes to enhance the scene geometry modeling by minimizing its shortest axis scale $S_i = \text{diag}(s_1, s_2, s_3)$:

$$\mathcal{L}_{\texttt{scale}} = \frac{1}{N}\sum_{i\in N} \|\min(s_1, s_2, s_3)\|, \tag{3}$$

where N denotes the total number of planar-based Gaussian splats. Following PGSR [30], we utilize the plane representation to render both the plane-to-camera distance map $\mathcal{D}$ and the normal map $\mathcal{N}$, then convert them into unbiased depth maps by intersecting rays with the corresponding planes as:

$$\boldsymbol{D}(p)_{\texttt{render}} = \frac{\mathcal{D}}{\mathcal{N}(p)K^{-1}\tilde{p}}. \tag{4}$$

where p is the 2D position on the image plane. $\tilde{p}$ denotes the homogeneous coordinate of p, and K is the camera intrinsic.

Foundation Model Geometric Prior Constraints. In textureless regions (e.g., ground and walls), photometric loss tend to be insufficient. We employ an off-the-shelf monocular depth estimator [18] to provide dense depth prior. We address the inherent scale ambiguity between the estimated depths and the actual scene geometry by comparing them to sparse Structure-from-Motion (SfM) points, following [31]. Specifically, we align the scale of the monocular depth map $\boldsymbol{D}_{\texttt{mono}}$ with the SfM points projected depth map $\boldsymbol{D}_{\texttt{sfm}}$ using linear regression:

$$\hat{\boldsymbol{s}}, \hat{\boldsymbol{t}} = \arg\min_{\boldsymbol{s},\boldsymbol{t}} \sum_{p\in\boldsymbol{D}_{\texttt{sfm}}} \|(\boldsymbol{s} * \boldsymbol{D}(p)_{\texttt{mono}} + \boldsymbol{t}) - \boldsymbol{D}(p)_{\texttt{sfm}}\|_2^2. \tag{5}$$

Finally, we use the re-scaled depth map $\hat{\boldsymbol{s}} \times \boldsymbol{D}(p)_{\texttt{mono}} + \hat{\boldsymbol{t}}$ to regularize the rendered depth map $\boldsymbol{D}(p)_{\texttt{render}}$. Similarly, we adopt an off-the-shelf monocular normal estimator [32] to provide dense normal regularization to the rendered normal map $\mathcal{N}_{\texttt{render}}$ for accurate geometric modeling.

Multi-view Consistency Constraints. Inspired by [33], [34], a patch-based normalized cross-correlation (NCC) loss is applied between two gray renders $\{\mathbf{I}_{\texttt{render}}, \hat{\mathbf{I}}_{\texttt{render}}\}$ to force the multi-view photometric consistency:

$$\mathcal{L}_{\texttt{mv}} = \frac{1}{\|\mathcal{P}\|}\sum_{\mathbf{p}\in\mathcal{P}}\sum_{p\in\mathbf{p}}(1 - NCC(\hat{\mathbf{I}}(\mathcal{H}p), \mathbf{I}(p))). \tag{6}$$

where $\mathcal{P}$ is the set of all patches extracted from $\mathbf{I}_{\texttt{render}}$ and $\mathcal{H}$ is the homography matrix bwteen the two frames.

B. Building Realistic and Interactive Simulation

To enable the agent-environment interaction, we integrate a GS-mesh hybrid scene representation into the Isaac Sim with coordinate alignment. We further leverage agent-object randomization and occlusion-aware scene composition to advance robust and generalizable visual policy training.

GS-mesh Hybrid Representation. 1) The ***Gaussian*** representation generates photorealistic visual observations from the robot's ego-centric viewpoints. We first align the intrinsic parameters between *Sim* and *Real* by calibrating the robot camera's focal length and distortion parameters. We then obtain the camera extrinsic within the simulation coordinate system by retrieving the ego-view position and quaternion-based orientation from Issac Sim. 2) The ***Mesh*** representation facilitates physical interaction and precise collision detection. We render the depth map for each input view and leverage a Truncated Signed Distance Function (TSDF) [35] fusion

algorithm to build the corresponding TSDF field. We then extract the mesh from this TSDF field to enable physically accurate interactions within the simulation.

Coordinate Alignment. To align the reconstructed COLMAP coordinate system with the Isaac Sim environment, we first compute the homogeneous transformation matrix $T_{\text{homo}} \in \mathbb{R}^{4\times4}$ by manually matching four non-coplanar points.

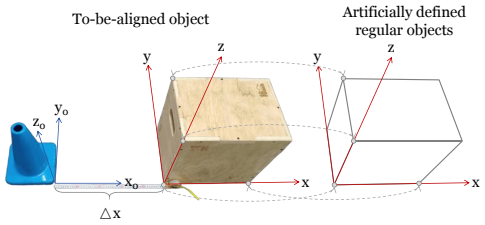

Fig. 5: **Coordinate alignment.** We first compute the transformation for a regular reference object and then apply this transformation to the irregular object of interest.

Specifically, as shown in Figure 5, we use objects with regular shapes to transform coordinate systems. For instance, to obtain the transformation matrix for the blue cone, we first manually define a rectangular block $\mathcal{B}$ with the correct coordinate system and size as the reference object. Then, using the four-point registration method, we align the block's coordinates with the reconstructed COLMAP system. The transformation matrix T_{block} is determined as:

$$T_{\text{block}} = \arg\min_{T} \sum_{i=1}^{4} \|T \cdot \mathbf{p}_i^{\mathcal{B}} - \mathbf{p}_i^{\text{COLMAP}}\|^2, \tag{7}$$

Finally, the transformation matrix for the cone, T_{cone}, is determined by translating the cone's position relative to the aligned rectangular block:

$$T_{\text{cone}} = T_{\text{block}} \cdot \Delta\mathbf{p}_{\text{cone}}. \tag{8}$$

Gaussian Attributes Adjustment. We first decompose T_{homo} into its rotation component $R_{\text{homo}} \in \mathbb{R}^{3\times3}$, translation vector $t_{\text{homo}} \in \mathbb{R}^3$, and scale factor $s_{\text{homo}} \in \mathbb{R}$. In addition, the 3D covariance matrix Σ of the Gaussian points is parameterized by a scaling matrix $S \in \mathbb{R}^3$ and a rotation matrix $R \in \mathbb{R}^{3\times3}$, such that $\Sigma = RSS^TR^T$. The mean μ, scaling S and rotation R are adjusted as follows:

$$\mu' = R_{\text{homo}}\mu + t_{\text{homo}}, \tag{9}$$

$$S' = S + log(s_{\text{homo}}), \tag{10}$$

$$R_{\text{norm}} = \frac{R_{\text{homo}}}{s_{\text{homo}}} \quad \Sigma' = R_{\text{norm}}\Sigma R_{\text{norm}}^T, \tag{11}$$

Since the spherical harmonics (SHs) for the 3D Gaussians are stored in world space (i.e., the COLMAP coordinate system), view-dependent colors change when the Gaussians rotate. To accommodate different rotations, we first extract the Euler angles α, β, γ from the rotation matrix R_{homo} and construct the corresponding Wigner D-matrix D. We then apply D to rotate the SH coefficients. Formally, each band ℓ of the SH coefficients (of length $2\ell+1$) transforms as:

$$\mathbf{C}^{(\ell)'} = D_\ell(\alpha, -\beta, \gamma)\mathbf{C}^{(\ell)}. \tag{12}$$

where $\mathbf{C}^{(\ell)}$ is the vector of the spherical-harmonic coefficients for degree ℓ and D_ℓ is the $(2\ell+1)\times(2\ell+1)$ Wigner D-matrix.

Occlusion-Aware Composition and Randomization. With the resulting mesh and 3D Gaussians, we employ an interactive mesh editor to obtain both the 3D bounding box and its corresponding transformation matrix $T_{\text{bbox}} \in \mathbb{R}^{4\times4}$. Since the object mesh and the 3D Gaussians share the same COLMAP coordinate system, we use the mesh bounding box to crop the object Gaussians and merge them into the environment coordinate space according to the transformation below:

$$T_{\texttt{obj}}^{\texttt{COLMAP-env}} = T_{\texttt{sim}}^{\texttt{COLMAP-env}} \cdot T_{\texttt{COLMAP-obj}}^{\texttt{sim}} \cdot T_{\texttt{bbox}}. \tag{13}$$

As shown in Figure 7, the merged object Gaussians are synchronized with the object mesh in Issac and can be rendered jointly with the environment Gaussians via a volumetric alpha-blending [12] process. Since they are separately reconstructed, there are no holes in the merged 3DGS scenes. By incorporating z-buffering [36], it is impossible to see those objects that are farthest away from the ego-viewer and behind other objects. Therefore, our method can achieve occlusion-aware scene composition with accurate visibility. This randomization strategy substantially increases the environment's diversity, enabling more robust and scalable agent training.

C. Reinforcement Learning in Reconstructed Simulation

The next stage of VR-Robo focuses on training a robust locomotion policy via reinforcement learning, as illustrated in Figure 6. This policy is designed to address diverse goal-tracking tasks under varying environmental conditions. Due to the limited onboard computational resources of real robots, we adopt a two-level asynchronous control strategy: the high-level policy operates at 5 Hz, while the low-level policy executes at 50 Hz. The high-level policy communicates with the low-level policy via velocity commands, represented as $V_{\text{cmd}} = (V_x, V_y, V_{yaw})$. Main experiments are conducted and demonstrated on the Unitree Go2 quadruped robot.

Low-Level Policy. The first step is to train the low-level policy, similar to previous locomotion works [8], [37]. The low-level policy takes velocity commands $V_{\text{cmd}} = (V_x, V_y, V_{\text{yaw}})$ along with robot proprioception as input, and outputs the desired joint positions. The actor network is composed of a lightweight LSTM layer combined with multiple MLP layers.

The low-level policy adopted here enables the quadruped robot to climb slopes and stairs measuring up to 15 cm in height. However, unlike previous parkour works [5]–[7], this policy does not allow the robot to directly climb onto terrain taller than 30 cm. While existing frameworks could be extended to handle higher climbs, our approach instead focuses on teaching the robot to recognize and utilize slopes or stairs to reach high platforms. This strategy is particularly useful in real-world scenarios featuring taller obstacles, where direct climbing or jumping may be infeasible for any policy.

High-Level Policy. We freeze the previously trained low-level policy and train the high-level policy independently. The high-level policy is trained using reinforcement learning (PPO) within our GS-mesh hybrid simulation environment.

State Space: The high-level policy actor network receives four types of inputs: `1) RGB Image Feature` (I_a): We employ a frozen, pre-trained Vision Transformer (ViT) [38] to

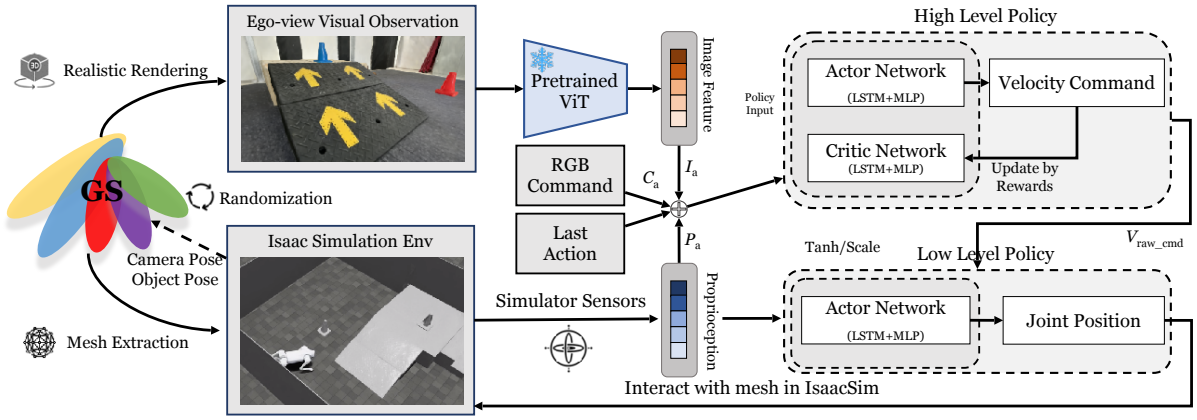

Fig. 6: **RL policy training in the reconstructed simulation environment.** The agent leverages the ego-view GS photorealistic rendering as visual observations and interacts with the GS-extracted mesh in the Issac Sim environment.

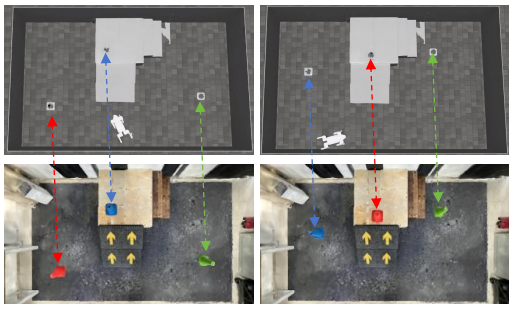

Fig. 7: **Agent-object Randomization and Scene Composition.** At the beginning of each episode, we randomly sample the positions for the robot and three cones in the Isaac Sim environment (upper row). We synchronously merge the agent and object Gaussians into the environment and compose for joint rendering (lower row).

extract features from RGB images. ViT's attention mechanism is well-suited for detecting positions and specific colors. This approach compresses the input size significantly and accelerates training. `2) RGB Command` (C_a): A RGB vector indicating the desired color of the target cone. For example, $[1,0,0]$ represents the red cone. `3) Last Action`: The action output by the policy during the previous timestep. `4) Proprioception` (P_a): This includes the robot's base angular velocity, projected gravity, joint positions, and joint velocities. We use an asymmetric actor-critic structure. The critic's observation includes the actor's observation (with noise removed) as well as the robot's world coordinate position and orientation, and the target cone's world coordinate position. This design improves the estimation of the value function and enhances the training process.

Action Space: The actor outputs the raw velocity command, $V_{\text{raw\_cmd}} = (V_x, V_y, V_{\text{yaw}})$. This raw command will pass through a tanh layer and scaled by the velocity range $V_{\text{max\_cmd}} = (V_{\text{max\_x}}, V_{\text{max\_y}}, V_{\text{max\_yaw}})$ to ensure safety. Note that the tanh layer and velocity range scaling are applied outside the actor network. The resulting velocity command $V_{\text{cmd}} = (V_x, V_y, V_{\text{yaw}})$ is sent to the low-level policy, enabling the robot to move.

Reward Design: The total reward consists of two main categories: **Task Rewards** r_T and **Regularization Rewards** r_R. r_T include `Reach_goal`, `Goal_dis`, `Goal_dis_z`, and `Goal_heading`. To be specific, the robot receives a reward if it comes close enough to the goal.

$$r_{\text{reach\_goal}} = \begin{cases} R_{\max}, & \text{if } \|\mathbf{p}_{\text{robot}} - \mathbf{p}_{\text{goal}}\|_2 \le \epsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

`Goal_dis` is based on the change in the Euclidean distance to the goal between consecutive time steps:

$$r_{\text{goal\_dis}} = d_{\text{prev}} - d_{\text{current}}, \tag{15}$$

$$d = \|\mathbf{p}_{\text{robot}} - \mathbf{p}_{\text{goal}}\|_2. \tag{16}$$

`Goal_dis_z` is based on the change in the vertical (z-axis) distance to the goal:

$$r_{\text{goal\_dis\_z}} = d_{z,\text{prev}} - d_{z,\text{current}}, \tag{17}$$

$$d_z = |z_{\text{robot}} - z_{\text{goal}}|. \tag{18}$$

`Goal_heading`: This reward encourages the robot to face the goal by minimizing the yaw angle error, defined as a linear function of the yaw difference:

$$r_{\text{goal\_heading}} = -|\psi_{\text{robot}} - \psi_{\text{goal}}|, \tag{19}$$

where ψ_{robot} is the robot's current yaw angle, ψ_{goal} is the desired yaw angle toward the goal.

In addition, regularization rewards are incorporated to further refine the robot's performance and ensure stable and efficient behavior. It contains `Stop_at_goal`, `Track_lin_vel`, `Track_ang_vel`, and `Action_l2`.

Training Process: We randomly sample the positions and orientations of the robot and cones at the beginning of each episode. The robot's camera poses and cone poses are obtained from Isaac Sim and sent to the aligned and editable 3D Gaussians mentioned above to render a photorealistic image. The policy then outputs an action based on this image, which is applied in Isaac Sim to interact with the mesh.

Domain Randomization: We apply domain randomization to reduce the sim-to-real gap. This includes three key components: `1) Camera Pose Noise`: Before requesting the 3D Gaussians to render an image, we add uniform noise to the camera pose extracted from Isaac Sim. `2) Image Augmentation`: We randomly apply brightness, contrast, saturation, and hue adjustments to the image. This is followed by Gaussian blur to simulate camera blur caused by robot

movement. Additionally, we randomly add Gaussian noise to the image with a small probability ($p = 0.05$). `3) Image Delay`: We randomly apply a 0- or 1-step delay to the rendered images.

V. Experiments

Our experiments are designed to evaluate two main aspects of VR-Robo: **1)** its ability to reconstruct realistic and interactive simulation environments with GS-mesh hybrid representation, supporting the locomotion policy training through both visual observations and mesh interactions; **2)** its capability to minimize the *Sim-to-Real* gap when deployed to the real world. We conducted extensive experiments in both simulation and real-world settings to validate these capabilities.

A. Experiment Setup

Real-to-Sim Reconstruction. We reconstruct 6 distinct indoor room environments, each characterized by specific terrains. We also randomize the environments by incorporating three cones colored red, green, and blue. We use iPads or iPhones to take photos, which are easily available.

Simulated Locomotion Training. We conduct policy training in Isaac Sim using a single NVIDIA 4090D GPU. For the low-level policy, we instantiate 4,096 quadruped robot agents in parallel and train for 80,000 iterations from scratch. In contrast, the high-level policy is trained from scratch over 8,000 iterations with 64 quadruped agents. The entire training process requires approximately three days to complete. For visual encoding, we employ the Vision Transformer model "vit_tiny_patch16_224", omitting its final classification head. The Isaac Sim simulator and the 3DGS rasterization-based renderer communicate via a TCP network.

Sim-to-Real Deployment. We deploy our policy on the Unitree Go2 quadruped robot, which is equipped with an NVIDIA Jetson Orin Nano (40 TOPS) as the onboard computer. We use ROS2 [39] for communication between the high-level policy, low-level policy, and the robot. Both policies run onboard. The robot receives desired joint positions from the low-level policy for PD control ($K_p = 40.0$, $K_d = 1.0$). We use an Insta360 Ace camera to capture RGB images. The camera captures images at a resolution of 320×180. After calibration, it has a horizontal field of view (FOVX) of 1.5701 radians and a vertical field of view (FOVY) of 1.0260 radians.

B. Simulation Experiments

We conduct comparison and ablation experiments with various baselines. For comparison experiments,

TABLE I: Comparison results in the simulation setting.

Method	SR ↑	ART (s) ↓
Ours	**100.00%**	**4.94**
Imitation Learning (IL)	8.67%	14.01
Random Background	43.33%	11.75

TABLE II: Ablation results in the simulation setting.

Method	SR ↑	ART (s) ↓
Ours	**100.00%**	**4.94**
Textured Mesh	22.00%	12.73
CNN Encoder	54.67%	10.04

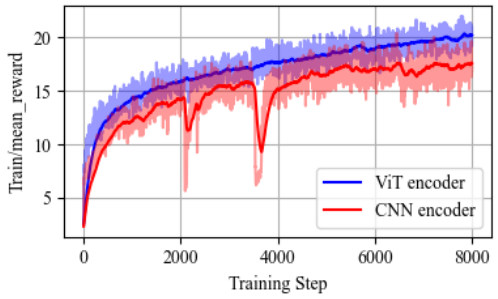

Fig. 8: **Training reward comparison.** Pretrained CNN [40] encoder struggles to extract precise semantic features, leading to unstable and worse rewards compared to ViT [38] encoder.

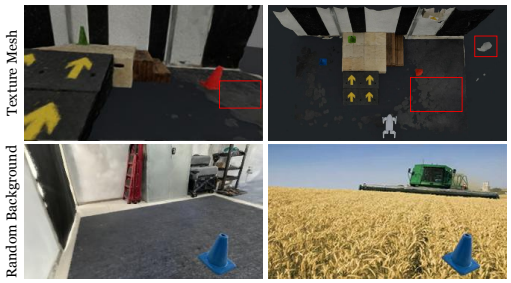

Fig. 9: Illustration of textured mesh and random background.

TABLE III: Reconstruction quality comparison.

Method	Planar 3DGS	Textured Mesh
PSNR ↑	**29.73**	27.73

- **Imitation Learning:** We collect 60 different real-world trajectories via teleoperation and train the policy through supervised learning using regression optimization.
- **Random Background:** LucidSim [10] uses generative models to produce visual observations. Since the training and deployment code has not been open-sourced, we re-implemented it by using random images from ImageNet as the background and maintaining the target object.

For ablation experiments,

- **Textured Mesh:** We use SuGaR [41] to reconstruct textured mesh and support direct mesh rendering in Isaac Sim as visual observation.
- **CNN Encoder:** We replace ViT with a CNN [40] encoder, and remove the last classification layer.

We evaluated all methods in a 'red cone reaching' task within Scene 1. The red cone is randomly placed in one of three regions: left (ground), middle (terrain), or right (ground). We generate 150 distinct cone placements, specified before any experiments, and apply the same setup for all methods to ensure a fair comparison.

To measure performance, we report both Success Rate (SR) and Average Reaching Time (ART). An episode is deemed successful if the robot comes within 0.25 m of the red cone at any point within a 15-second window. For successful episodes, we record the time taken to reach the cone; otherwise, we assign the maximum time of 15 seconds. The ART is then computed as the mean reaching time across all trials.

The comparison and ablation results are shown in Table I and Table II, respectively. Our method consistently achieves

TABLE IV: Comparison and ablation experimental results in the real-world setting.

Method	Exteroception	Success Rate ↑			Average Reaching Time (s) ↓		
		Easy	Medium	Hard	Easy	Medium	Hard
Ours	RGB	**100.00%**	**93.33%**	**100.00%**	**4.96**	**6.28**	**9.09**
Imitation Learning (IL)	RGB	0.00%	0.00%	0.00%	15.00	15.00	15.00
SARO [42]	RGB	66.67%	26.67%	0.00%	46.49	57.24	60.00
Textured Mesh	RGB	20.00%	6.67%	0.00%	12.90	14.90	15.00
CNN Encoder	RGB	73.33%	66.67%	6.67%	9.10	11.41	14.90
w/o Domain Randomization	RGB	53.33%	6.67%	0.00%	10.04	14.76	15.00

higher performance across all evaluation metrics by a large margin. In particular, the Imitation Learning (IL) baseline suffers from insufficient data samples and a lack of policy exploration. As shown in Figure 8, the CNN encoder struggles to extract precise features from the images. In this work, we reconstruct solely from RGB images. The textured mesh obtained from GS exhibits noticeable bulges in texture-less regions such as the ground and walls, as shown in Figure 9 and Table III. These artifacts degrade the rendering quality and can even hinder the robot's movement. The quality and resolution of the mesh itself significantly affect rendering fidelity, whereas GS is less sensitive to such factors. In contrast, our method crops and utilizes only the mesh of the central terrain, and renders observation from GS, effectively avoiding these issues. The Random Background setting (the bottom-right image) loses important features specific to the original scene (the bottom-left image). These scene-specific features are crucial for the robot to effectively explore the environment.

C. Real-World Experiments

We conduct qualitative experiments under varying conditions, including 6 different scenes, various target cone colors, changes in lighting conditions, random interference, background variations, and training on the different robots, as shown in Figure 2. A detailed demonstration is provided in the supplementary materials. These experiments highlight the robustness of our method, showcasing its adaptability to a wide range of environments and conditions.

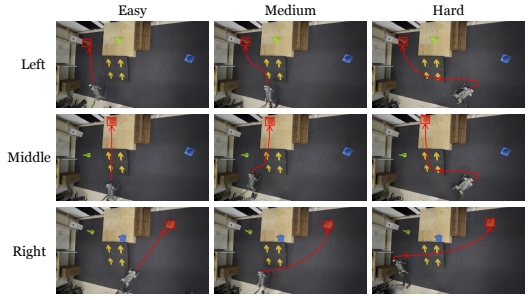

Fig. 10: Tasks of different difficulties (*easy*, *medium*, and *hard*) and different target positions (*left*, *middle*, and *right*).

We further conduct quantitative experiments, as presented in Table IV. In addition to the methods mentioned above, we conduct two additional baseline methods for real-world experiments. These methods are specifically designed in the real-world setting and cannot be directly compared in simulation:

- **SARO [42]:** SARO enables the robot to navigate across 3D terrains leveraging the vision-language model (VLM).
- **w/o Domain Randomization:** We exclude domain randomization as described in Section IV-C.

All the real-world quantitative experiments is in the 'red cone reaching' task setting in Scene 1. We categorize the task into three levels of difficulty based on the robot's starting position: *easy*, *medium*, and *hard*. For *easy* tasks, the robot starts with the target cone directly visible. For *medium* tasks, the robot needs to turn to a certain degree to see the target cone. For *hard* tasks, the robot starts very far from the cone and faces a nearly opposite direction to the target cone. We randomly select 3 sets of cone positions, with each set containing one easy, one medium, and one hard task, as shown in Figure 10. For each task, we repeat the experiment 5 times. We then calculate the Success Rate (SR) and Average Reaching Time (ART). For SARO, the maximum time is extended to 60 seconds due to the long inference time required by the Vision-Language Model (VLM). In the real robot experiments, we consider the task successful if the robot makes contact with the target cone.

Our method achieves the highest success rates across all difficulty levels and consistently records the shortest average reaching times, demonstrating its efficiency and robustness. Note that due to inherent randomness in the real-world setting, there is a failure case in the *medium* task, even though all trials succeed in the *hard* task. Among the other methods, SARO achieves moderate success on easy tasks (66.67%) but performs poorly on medium and hard tasks. This under-performance is primarily attributed to the lack of historical context and limited exploration capability in complex scenarios. If the robot cannot initially see the goal, it is unlikely to succeed.

VI. Limitations and Conclusion

A. Limitations

The current environment is limited to indoor scenes with static terrains, which constrains the diversity of scenarios. Training a locomotion policy from scratch for each task takes approximately three days, posing a significant time burden. Additionally, meshes reconstructed using only RGB input often contain structural imperfections and require manual post-processing, as they lack the fidelity that could be provided by additional sensing modalities such as depth cameras or LiDAR.

B. Conclusion

We introduce VR-Robo, a real-to-sim-to-real framework designed to train visual locomotion policies within a pho-

torealistic and physically interactive simulation environment. Extensive experimental results demonstrate that our agents successfully learn robust and effective policies for challenging high-level tasks and can be zero-shot deployed to diverse real-world scenarios. In future work, we aim to extend our framework to larger-scale and more complex environments, incorporating generative models into scene reconstruction for more generalizable policy learning.

References

[1] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[2] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[3] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, *et al.*, "Habitat 2.0: Training home assistants to rearrange their habitat," *Advances in neural information processing systems*, vol. 34, pp. 251–266, 2021.

[4] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T.-k. Chan, *et al.*, "Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai," *arXiv preprint arXiv:2410.00425*, 2024.

[5] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," *arXiv preprint arXiv:2309.05665*, 2023.

[6] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 11 443–11 450.

[7] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024.

[8] J. Wu, G. Xin, C. Qi, and Y. Xue, "Learning robust and agile legged locomotion using adversarial motion priors," *IEEE Robotics and Automation Letters*, 2023.

[9] Z. Zhuang, S. Yao, and H. Zhao, "Humanoid parkour learning," *arXiv preprint arXiv:2406.10759*, 2024.

[10] A. Yu, G. Yang, R. Choi, Y. Ravan, J. Leonard, and P. Isola, "Learning visual parkour from generated images," in *8th Annual Conference on Robot Learning*, 2024.

[11] 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," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[12] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[13] A. Byravan, J. Humplik, L. Hasenclever, A. Brussee, F. Nori, T. Haarnoja, B. Moran, S. Bohez, F. Sadeghi, B. Vujatovic, *et al.*, "Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9362–9369.

[14] H. Lou, Y. Liu, Y. Pan, Y. Geng, J. Chen, W. Ma, C. Li, L. Wang, H. Feng, L. Shi, *et al.*, "Robo-gs: A physics consistent spatial-temporal model for robotic arm with hybrid representation," *arXiv preprint arXiv:2408.14873*, 2024.

[15] X. Li, J. Li, Z. Zhang, R. Zhang, F. Jia, T. Wang, H. Fan, K.-K. Tseng, and R. Wang, "Robogsim: A real2sim2real robotic gaussian splatting simulator," *arXiv preprint arXiv:2411.11839*, 2024.

[16] Y. Wu, L. Pan, W. Wu, G. Wang, Y. Miao, and H. Wang, "Rl-gsbridge: 3d gaussian splatting based real2sim2real method for robotic manipulation learning," *arXiv preprint arXiv:2409.20291*, 2024.

[17] Z. Xie, Z. Liu, Z. Peng, W. Wu, and B. Zhou, "Vid2sim: Realistic and interactive simulation from video for urban navigation," *arXiv preprint arXiv:2501.06693*, 2025.

[18] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, "Depth anything v2," *arXiv preprint arXiv:2406.09414*, 2024.

[19] L. Wang, Y. Ling, Z. Yuan, M. Shridhar, C. Bao, Y. Qin, B. Wang, H. Xu, and X. Wang, "Gensim: Generating robotic simulation tasks via large language models," *arXiv preprint arXiv:2310.01361*, 2023.

[20] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[21] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[22] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[23] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[24] X. Li, Y. Zhang, and X. Y. Drivingdiffusion, "Layout-guided multi-view driving scene video generation with latent diffusion model," *arXiv preprint arXiv:2310.07771*, vol. 2, no. 3, 2023.

[25] Z. Chen, Z. Mandi, H. Bharadhwaj, M. Sharma, S. Song, A. Gupta, and V. Kumar, "Semantically controllable augmentations for generalizable robot learning," *The International Journal of Robotics Research*, p. 02783649241273686, 2024.

[26] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal, "Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation," *arXiv preprint arXiv:2403.03949*, 2024.

[27] A. Quach, M. Chahine, A. Amini, R. Hasani, and D. Rus, "Gaussian splatting to real world flight navigation transfer with liquid networks," *arXiv preprint arXiv:2406.15149*, 2024.

[28] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4104–4113.

[29] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2d gaussian splatting for geometrically accurate radiance fields," *arXiv preprint arXiv:2403.17888*, 2024.

[30] D. Chen, H. Li, W. Ye, Y. Wang, W. Xie, S. Zhai, N. Wang, H. Liu, H. Bao, and G. Zhang, "Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction," *arXiv preprint arXiv:2406.06521*, 2024.

[31] M. Turkulainen, X. Ren, I. Melekhov, O. Seiskari, E. Rahtu, and J. Kannala, "Dn-splatter: Depth and normal priors for gaussian splatting and meshing," *arXiv preprint arXiv:2403.17822*, 2024.

[32] G. Bae and A. J. Davison, "Rethinking inductive biases for surface normal estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 9535–9545.

[33] S. Galliani, K. Lasinger, and K. Schindler, "Massively parallel multiview stereopsis by surface normal diffusion," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 873–881.

[34] Q. Fu, Q. Xu, Y. S. Ong, and W. Tao, "Geo-neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3403–3416, 2022.

[35] 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, pp. 303–312.

[36] E. E. Catmull, *A subdivision algorithm for computer display of curved surfaces*. The University of Utah, 1974.

[37] S. Zhu, R. Huang, L. Mou, and H. Zhao, "Robust robot walker: Learning agile locomotion over tiny traps," *arXiv preprint arXiv:2409.07409*, 2024.

[38] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[39] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[40] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1314–1324.

[41] A. Guédon and V. Lepetit, "Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 5354–5363.

[42] S. Zhu, D. Li, L. Mou, Y. Liu, N. Xu, and H. Zhao, "Saro: Space-aware robot system for terrain crossing via vision-language model," *arXiv preprint arXiv:2407.16412*, 2024.