

Robot-GST: geometry-aware spatial-temporal robot policy representation and evaluation

Anonymous Author

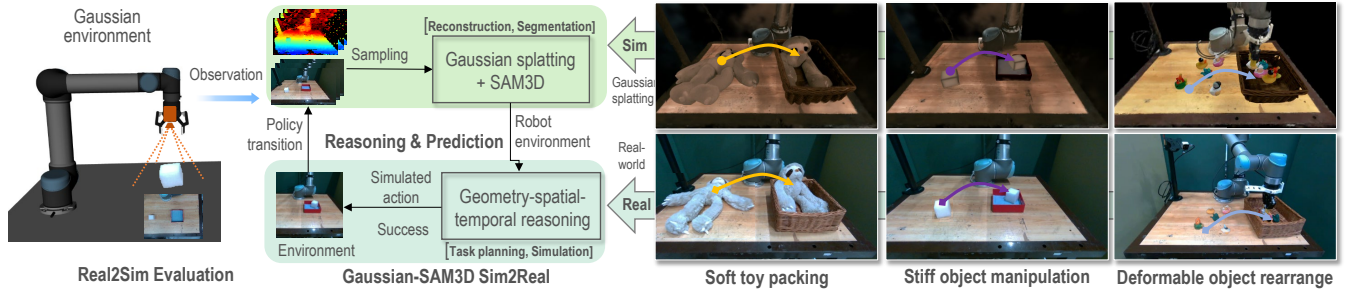


Fig. 1: **Overview of Robot-GST: Geometry-aware Spatio-Temporal Policy Representation and Evaluation.** Our system builds a unified, geometry-aware manipulation framework by combining 3D Gaussian Splatting with SAM3D segmentation, enabling prediction, reconstruction, part-aware reasoning, and robust sim-to-real transfer.

Abstract—Robotic manipulation policies are advancing rapidly with increasing reliance on vision-language models for end-to-end decision making. However, reliable deployment remains challenging because many policies lack explicit mechanisms for predicting task outcomes and evaluating whether generated actions will achieve desired final states, causing execution errors to accumulate during long-horizon manipulation. We present Robot-GST, a geometry-aware spatio-temporal behaviour representation and evaluation framework that constructs a Gaussian-SAM robotic environment for real-to-sim policy verification and improves the reliability of real-world manipulation deployment. Our approach constructs a high-fidelity robotic environment from RGB-D observations using 3D Gaussian Splatting and SAM3D, enabling “simulation and evaluation before acting.” It integrates visual observations and language instructions with spatio-temporal reasoning for long-horizon task planning using large vision-language models. To bridge high-level planning and real-world execution, we introduce Gaussian-aware final-state estimation through geometric sampling and state-based trajectory planning. Before execution, candidate action sequences are simulated and evaluated in the Gaussian-SAM environment to filter infeasible behaviours. We validate our approach on representative manipulation tasks involving rigid, soft, and deformable objects, including cube placing, toy packing, and duck rearrangement, demonstrating that geometry-aware spatio-temporal reasoning and state-aware execution improve manipulation reliability across different object categories. Our results suggest that combining geometry-aware reconstruction with high-quality rendering and simulation provides a scalable approach for evaluating robotic manipulation behaviours. Website: <https://robot-gst.github.io>

I. Introduction

Robot manipulation policies have advanced rapidly across a wide range of tasks, driven by recent progress in vision-based and vision-language-based learning approaches [1], [2], [3]. However, training large-scale robot policies remains expensive and challenging [4], [5], despite their ability to directly map language and visual observations to robot actions.

However, many existing policies lack explicit mechanisms for predicting post-action states, making it difficult to evaluate whether generated actions will achieve desired outcomes before execution [6]. Moreover, evaluating robot policies still largely relies on real-world experiments, which are time-consuming, costly, and difficult to scale [7].

Vision-language-conditioned keypoint-affordance representations enable robotic manipulation by using pretrained models to generate semantic keypoints from visual observations and language instructions for path planning [8], [9], [10]. However, transferring these generated paths to real-world environments remains challenging due to limited geometric awareness and sim-to-real discrepancies. These methods typically rely on sparse keypoints and stable correspondence, which can limit robustness under deformation, occlusion, or complex interactions [11]. In addition, many vision-language-guided task plans do not explicitly reason about object final states, limiting their application to tasks requiring collision-aware planning or contact-rich manipulation of deformable objects [12].

Simulation before action provides a scalable approach that evaluates candidate behaviours before deployment on physical robots. *Gaussian Splatting* [13] represents scenes as explicit, anisotropic 3D Gaussians that can be rendered efficiently and support rapid editing and semantic annotation. Building on this representation, systems such as GraspSplats [14] fuse hierarchical, language-aligned features with depth supervision to enable open-vocabulary part localisation and real-time grasp proposals. More recently, real-to-sim evaluation pipelines [15] have combined photorealistic 3D Gaussian reconstructions with soft-body digital twins to correlate simulated rollouts with real-world success, providing scalable benchmarking but without coupling perception and reasoning for policy synthesis. Furthermore, physics-based simulators [15], [16],

[17] provide a framework for reconstructing deformable objects, simulating interactions, and evaluating policies.

We propose a geometry-aware spatio-temporal behaviour representation and evaluation framework that integrates visual observations and language instructions with spatio-temporal reasoning for state-aware task planning. After generating keypoint-based manipulation trajectories, we construct a Gaussian-SAM3D robotic simulation environment to evaluate candidate behaviours before execution. We evaluate our method on representative manipulation tasks, including rigid cube placement, soft-toy packing, and deformable-object rearrangement. Compared with existing methods, our approach significantly improves the alignment between simulated and real-world environments, achieving a structural similarity index measure (SSIM) of 0.044. Furthermore, evaluating policies in the photorealistic simulation environment improves the average real-world task success rate by 44.67% after policy correction, confirming the practical benefits of our approach for sim-to-real robot policy evaluation.

Our contribution can be summarised as follows:

- **A Gaussian-SAM3D-based simulation and evaluation environment**, built from RGB-D images using 3D Gaussian Splatting and SAM3D, enabling high-fidelity real-to-sim evaluation with photorealistically reconstructed objects;
- **A geometry-aware spatio-temporal behaviour representation for multi-stage manipulation**, which embeds visual observations and language tasks into spatio-temporal reasoning to produce high-level plans that explicitly account for object geometry and future state evolution;
- **Gaussian-aware final-state estimation and state-based low-level trajectory planning**, in which candidate action sequences are simulated and evaluated in the simulation environment before execution, enabling near-perfect real-world deployment across tasks.

II. Related Work

A. Language task decomposition and relational reasoning

Large Language Models (LLMs) provide strong semantic priors for decomposing natural-language instructions into executable subgoals and reasoning about task dependencies [18], [19], [20]. Beyond explicit planning, recent vision-language-action (VLA) policies leverage large-scale pretraining to execute diverse real-world tasks from language and visual inputs while reducing dependence on task-specific supervision [21]. Frameworks such as ReKep [8] formalise long-horizon manipulation as relational keypoint constraints, automatically generating constraint code and keypoints from language and RGB-D inputs. However, these methods primarily reason over sparse geometric structures and do not explicitly maintain dense 3D object representations that capture object parts, shapes, and evolving physical states. Mark-based visual prompting (MOKA) enables open-world planning with minimal training by converting image-space marks into structured waypoints [9]. SoFar [22] further connects language instructions with 6-DoF semantic orientations, such as insertion directions, to enforce geometric alignment constraints. However, such approaches generally

do not explicitly model object final states, limiting their application to tasks requiring collision-aware planning or contact-rich manipulation.

B. Robotic scene representation and reconstruction

Recent advances in 3D scene modelling have transformed robotic perception. Implicit radiance-field methods [23], [24] unify appearance and semantics across views but remain computationally expensive and difficult to edit, limiting their use for real-time manipulation. In contrast, *3D Gaussian Splatting* (3DGS) [13], [25] represents scenes as explicit, anisotropic Gaussians that can be updated or rendered at interactive rates. Recent feature-augmented variants extend 3DGS by distilling dense foundation-model features for downstream semantics [26]. Extensions such as GraspSplats [14] fuse hierarchical, language-aligned features with depth supervision to achieve open-vocabulary part localisation and real-time grasp sampling. These explicit models offer high efficiency and editability for dynamic scenes, while SLAM-oriented 3DGS systems further enable continuous scene updates for robotics [27], [28]. Nevertheless, these representations remain largely perception-centric and provide limited mechanisms for structured temporal or relational reasoning required for multi-step interaction planning.

C. Policy simulation and evaluation

Recent 3DGS-based pipelines reconstruct scenes from real-world captures and couple them with approximate dynamics to estimate sim-to-real correlation and to select or refine candidate behaviours before deployment [15], [29]. Beyond individual pipelines, several recent works have begun systematising real-to-sim evaluation for robot manipulation policies through standardised benchmarks and reusable environment-generation tools [30], [31]. These approaches exploit 3DGS as a efficient renderer for policy observations while relying on auxiliary geometric proxies or physics engines to simulate contact-rich interactions. Digital twins have also proven valuable for reconstructing and simulating real-world objects [16]. However, existing frameworks often struggle to capture complex real-world dynamics because they rely on predefined physical parameters or approximate interaction models [7]. Physics-informed simulators, therefore, provide a promising alternative. For example, PhysTwin builds physical digital twins from sparse video inputs, enabling accurate reconstruction and resimulation of deformable objects [15]. However, existing approaches typically focus on individual components, such as semantic scene reconstruction, task reasoning, or simulation, without integrating geometry-aware representation, long-horizon reasoning, and pre-execution behaviour evaluation into a unified framework.

III. Methods

We describe how Gaussian representations and SAM3D construct a geometry-aware robotic environment for manipulation planning and evaluation. Our approach addresses two challenges: (i) obtaining high-fidelity geometric and semantic

scene representations, which we address via 3DGS reconstruction and multimodal feature alignment, and (ii) evaluating rigid and deformable object manipulation behaviours within the reconstructed GS environment. Addressing the second gap requires ensuring collision-free motion between rigid objects and the environment (including contacted objects), while deformable-object (including contacted objects), while deformable-object manipulation must satisfy constraints imposed by final-state estimation.

A. 3D Gaussian Splatting for robotic scene reconstruction

Gaussian Splatting Reconstruction: To reconstruct a photorealistic and geometry-consistent robotic scene from short multi-view RGB-D scans, we adopt 3D Gaussian Splatting (3DGS) as the scene representation. We model the scene as a collection of anisotropic Gaussian primitives:

$$\mathcal{G} = \{g_i\}_{i=1}^N, \quad g_i = (\mu_i, \mathbf{r}_i, \Sigma_i, \mathbf{c}_i, \alpha_i, \mathbf{f}_i) \quad (1)$$

where $\mathcal{G} = \{g_i\}_{i=1}^N$ denotes a set of N Gaussians. $\mu_i \in R^3$ denotes the Gaussian centre, $\mathbf{r}_i \in \text{SO}(3)$ encodes its orientation, $\Sigma_i \in R^{3 \times 3}$ denotes its covariance, and $(\mathbf{c}_i \in R^3, \alpha_i \in [0, 1])$ denote the colour and opacity, respectively.

We further associate each Gaussian with a semantic feature vector $\mathbf{f}_i \in R^{d_f}$ extracted from foundation vision models (e.g., SAM3D, DINOv2, and CLIP). Given a query y , object- and part-level activations $s_i^{\text{obj}}(y)$ and $s_i^{\text{part}}(y)$ are derived from $\mathbf{f}_i$ for language-conditioned segmentation and manipulation.

Given calibrated cameras and a set of images $\{I_i\}_{i=1}^N$, we render the scene with a differentiable splatting renderer using front-to-back alpha compositing:

$$I_i^* = \sum_i \mathbf{T}_i(u) \alpha_i \mathbf{w}_i(u) \mathbf{c}_i, \quad \mathbf{T}_i(u) = \sum_{j < i} (1 - a_j(u)), a_i(u) = \alpha_i \mathbf{w}_i(u) \quad (2)$$

where $\mathbf{w}_i(u)$ is the 2D splat weight at pixel u (obtained by projecting the 3D Gaussian), and $\mathbf{T}_i(u)$ is the accumulated transmittance along the viewing ray. We optimise the Gaussian parameters using a reconstruction objective $\mathcal{L}$, optionally regularised with a depth prior and feature distillation, weighted by λ_d and λ_f .

$$\mathcal{L} = \sum_{i=1}^N \|I_i^* - I_i\|_2^2 + \lambda_d \|D - D^*\|_1 + \lambda_f \|F - F^*\|_2^2 \quad (3)$$

where I^* is the rendered image, D^* is the rendered depth, D is an aligned dense depth prior that stabilises geometry under sparse views, and F and F^* denote rendered and target semantic features for object-centric interaction.

Mesh model: Given an input image I , we obtain an object mask M using SAM3. We then follow SAM3D to construct a geometric model $p(O, R, t, s \mid I, M)$, where $O \in R^{64^3}$ denotes a coarse (voxelised) object shape, $R \in R^6$ is a 6D rotation representation, $t \in R^3$ is the translation, and $s \in R^3$ is the scale. Importantly, SAM3D incorporates calibrated depth observations, enforcing metric scale during object

reconstruction and ensuring that the estimated translation t and scale s match real-world object dimensions. Active voxels from the coarse shape O are used to refine the object shape S and texture T by learning the conditional distribution $p(S, T \mid I, M, O)$. The latent representation L of S and T is then decoded into a mesh $\mathcal{M}$ via a mesh decoder D_m and into Gaussians $\mathcal{G}$ via a Gaussian-splat decoder D_g :

$$\mathcal{M} = D_m(L(S, T)); \quad \mathcal{G} = D_g(L(S, T)) \quad (4)$$

The mesh $\mathcal{M}$ provides explicit geometry for collision reasoning, while $\mathcal{G}$ enables efficient photorealistic rendering and scene composition. Compared with meshes produced by SAM3D alone, the resulting mesh offers a favourable trade-off between fidelity, accuracy, and reconstruction time. The static GS environment, mesh-based objects, and the robot model enable compositional scene construction, allowing the robot to manipulate objects in a dynamic environment.

B. Keypoint affordance constraints

Keypoint and relational constraints: Given a natural-language task l , we process annotated scene images I (with object masks M and labels) using a vision-language model to generate a keypoint-affordance representation $\mathbf{k}$. We construct a discrete set of scene keypoints:

$$\mathbf{k} = \mathcal{F}(I, M, l, D); \quad \mathbf{k} = [\mathbf{k}_1, \dots, \mathbf{k}_K] \in R^{K \times 3} \quad (5)$$

where each $\mathbf{k}_i$ denotes a 3D point associated with a target object O . We express a relational constraint as a scalar-valued function:

$$f : R^{K \times 3} \rightarrow R, \quad f(\mathbf{k}) \leq 0 \Rightarrow \text{constraint satisfied.} \quad (6)$$

Here, f encodes proximity (distance-to-contact), alignment (axis/normal consistency), or relative-pose relations between keypoints (e.g., “rearrange deformable toys from the table into a basket”). Each stage is represented by two types of constraints: (i) *sub-goal* constraints $\mathcal{C}_{\text{sub}}^{(i)} = \{f_{\text{sub},1}^{(i)}, \dots, f_{\text{sub},n_i}^{(i)}\}$, which specify grasp and release conditions, and (ii) *sub-path* constraints $\mathcal{C}_{\text{path}}^{(i)} = \{f_{\text{path},1}^{(i)}, \dots, f_{\text{path},m_i}^{(i)}\}$, which impose *path-feasibility* requirements during motion. Importantly, the VLM generates executable constraint representations, avoiding task-specific hand-crafted logic and enabling downstream optimisation.

Spatial constraints with task decomposition: We decompose a language task into a sequence of stages $\mathcal{H} = \{1, \dots, h\}$, each represented by relational keypoints and geometric constraints. For stage h , let $\mathcal{K}_h = \{\mathbf{k}_m\} \subset R^3$ denote the task-relevant keypoints for the target object O , and let $\mathcal{C}_h = \{c_\ell\}$ denote a set of differentiable spatial constraints encoding subgoals such as distance, alignment, orientation, or visibility. Each stage defines an optimisation problem over the decision variables $\zeta_h = (T_h, \gamma_h)$:

$$\begin{aligned} \zeta_h^* &= \arg \min_{\zeta_h} J_h(\zeta_h \mid \mathcal{G}, l) \\ &= \sum_{\ell=1}^{L_h} w_{h,\ell} \phi_{h,\ell} \left([f_{h,\ell}(K_h, \zeta_h; \mathcal{G}, l)]_+ \right) + \lambda_{\text{reg}} \Omega(\zeta_h) \end{aligned} \quad (7)$$

where $T_h \in \text{SE}(3)$ and γ_h denote the end-effector pose (position and orientation) and grasp/gripper parameters at stage h . Here, L_h is the number of constraint terms. The ℓ -th inequality constraint function $f_{h,\ell}$ is satisfied when $f_{h,\ell}(\cdot) \leq 0$. $\phi_{h,\ell}(\cdot)$ is a nonnegative penalty on constraint violation, and $w_{h,\ell} > 0$ is the corresponding weight. $\Omega(\zeta_h)$ is a regulariser on ζ_h (e.g., smoothness or deviation from a nominal pose/grasp), with weight λ_{reg} . Finally, $[x]_+ = \max(0, x)$ denotes the violation magnitude of an inequality constraint (zero if satisfied).

C. Geometry-aware spatio-temporal reasoning and planning

Object state prediction: We represent object O_i by its world-frame translation $\mathbf{p}_i \in R^3$ and unit-quaternion orientation $\mathbf{q}_i \in H$ (with H denoting the set of unit quaternions). The relational-spatial constraints define target grasp and release positions ($\mathbf{p}_{\text{grasp}}, \mathbf{p}_{\text{release}}$). Let $\mathbf{p}_{\mathbf{c}_i} \in R^3$ denote a representative object centre (e.g., its centroid) used for association. We select the target object index t via nearest-neighbour matching:

$$O_t = \arg \min_{O_i} \|\mathbf{p}_{\mathbf{c}_i} - \mathbf{p}_{\text{grasp}}\|_2. \quad (8)$$

To approximate stable object transport, we assume the object–end-effector relative transform remains fixed after grasping. Let $(\mathbf{p}_{ee}, \mathbf{q}_{ee})$ be the current end-effector pose, and let $(\mathbf{p}_0, \mathbf{q}_0)$ be the grasped object’s pose at the grasp moment. Let $R_{ee} \in R^{3 \times 3}$ be the rotation matrix corresponding to $\mathbf{q}_{ee}$, and let $\otimes$ denote quaternion multiplication. Then:

$$\mathbf{p}_{\text{rel}} = R_{ee}^{-1}(\mathbf{p}_0 - \mathbf{p}_{ee}), \quad \mathbf{q}_{\text{rel}} = \mathbf{q}_{ee}^{-1} \otimes \mathbf{q}_0. \quad (9)$$

To make placement robust to reconstruction noise, partial occlusion, and narrow receptacles, we perform a local search around a nominal end-effector pose predicted by the stage constraints. Let $\mathbf{p}_{\text{final}} \in R^3$ and $\mathbf{q}_{\text{final}} \in H$ denote the nominal end-effector translation and orientation. We generate candidate end-effector poses by sampling bounded positional perturbations $\Delta \mathbf{p}$:

$$\mathbf{p}_{\text{rand}} = \mathbf{p}_{\text{final}} + \Delta \mathbf{p}, \quad \Delta \mathbf{p} \sim \mathcal{U}(-\tau_p, \tau_p)^3, \quad (10)$$

where $\mathcal{U}$ is the uniform distribution and $\tau_p > 0$ bounds the translation noise per axis. We also sample a bounded rotational perturbation $\Delta \phi \in R^3$ in axis–angle form with $\|\Delta \phi\|_2 \leq \tau_r$ (for a scalar bound $\tau_r > 0$), and apply it to obtain a perturbed orientation $\mathbf{q}_{\text{rand}}$. Each candidate thus defines a hypothesised release pose for the manipulated object.

Candidate sampling and collision checking: We explore feasible release poses by sampling bounded perturbations $\Delta \mathbf{p}$ and rotational noise $\Delta \phi$. For each candidate, we evaluate geometric feasibility by checking for collisions with the static scene geometry. We build a signed distance field (SDF) $\mathcal{S} : R^3 \rightarrow R$, where $\mathcal{S}(\mathbf{x})$ denotes the distance from point $\mathbf{x}$ to the closest scene surface (positive in free space). For a set of object surface points $\{\mathbf{x}_m\}$ transformed by the candidate pose (as implied by the end-effector pose and the fixed relative transform), we compute

$$d(\mathbf{x}_m) = \mathcal{S}(\mathbf{x}_m), \quad (11)$$

and declare a candidate collision-free if

$$\min_{\mathbf{x}_m} d(\mathbf{x}_m) \geq \epsilon_{\text{col}}, \quad (12)$$

where $\epsilon_{\text{col}} > 0$ is a collision margin. Among feasible candidates, we rank solutions using a stability-oriented heuristic (e.g., minimal rotational deviation from a desired target orientation) and then invoke the motion planner to generate a collision-free trajectory that satisfies both the stage subgoal constraints and the path constraints.

D. Robot behaviour simulation and evaluation

We verify candidate low-level action sequences in a Real-to-Sim environment that is *metrically aligned* with the real scene (i.e., the same scale and robot base frame), enabling direct trajectory verification without additional normalisation. The environment maintains complementary representations: (i) a Gaussian scene for photorealistic rendering and observation consistency, and (ii) an explicit geometric proxy (mesh/TSDf/SDF extracted from the reconstruction and SAM3D objects) for collision and contact reasoning.

Trajectory parameterisation. We represent a candidate manipulation trajectory as

$$\tau = \{u_t\}_{t=1}^T, \quad u_t = (\mathbf{p}_t, \mathbf{q}_t, g_t), \quad (13)$$

where $\mathbf{p}_t \in R^3$ and $\mathbf{q}_t \in H$ denote the commanded end-effector position and unit-quaternion orientation in the robot base frame, and $g_t \in \{0, 1\}$ is the gripper open/close command.

Real-to-Sim rollout model: Let x_t denote the simulator state (robot joint configuration, gripper state, and object states) at step t . We compute joint targets via inverse kinematics

$$\theta_t = \text{IK}(\mathbf{p}_t, \mathbf{q}_t), \quad (14)$$

then advance the state using a simulator transition model

$$x_{t+1} = F(x_t, \theta_t, g_t), \quad (15)$$

where F combines URDF kinematics with physics/contact simulation. Rigid objects are simulated with rigid-body dynamics and collision meshes reconstructed from the scene/object geometry; deformable objects are simulated with a soft-body model (PhysTwin [15]) to approximate deformation and contact response.

Verification criteria (simulate before acting): We consider τ feasible if it is executable, collision-safe, and satisfies the task goal at the terminal state. We define the acceptance indicator

$$I(\tau) = \prod_{t=1}^T I(\text{IK}(\mathbf{p}_t, \mathbf{q}_t)) \cdot \prod_{t=1}^T I(\delta(x_t) \geq \epsilon_{\text{col}}) \cdot I(\Psi(x_T, l) = 1) \quad (16)$$

where $\delta(x_t)$ is the minimum signed distance between all rigid bodies (robot, gripper, rigid objects) and the static scene geometry at step t , computed from the reconstructed mesh/TSDf/SDF proxy; $\epsilon_{\text{col}} > 0$ is a safety margin; and $\Psi(x_T, l)$ is a task-specific terminal predicate evaluated on the final simulated state. We execute feasible trajectories τ on the real robot when $I(\tau) = 1$.

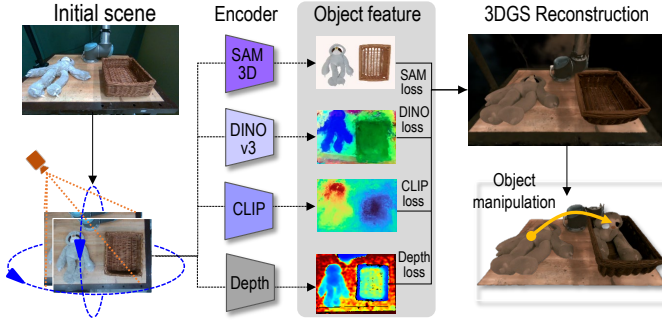


Fig. 2: **3DGS reconstruction of robotic scenes.** Multi-view RGB-D observations are processed using four complementary encoders: SAM3D for object-level segmentation, DINOv3 for semantic correspondence, CLIP for vision-language alignment, and depth cues for geometric structure. The resulting features enable high-fidelity 3D scene reconstruction.

IV. Experiments

Experimental setup: The robotic setup consists of a UR5 robot arm equipped with wrist and side RealSense cameras (D435 and D405), connected to a workstation with an NVIDIA GeForce RTX 4090 GPU. We evaluate our method on three manipulation tasks involving rigid, soft, and deformable objects:

- 1) Cube placing: the robot grasps a rigid cube from the table and places it into a small box. A trial is considered successful if the cube is placed inside the box without collision.
- 2) Toy packing: the robot picks up a soft toy (sloth) from the table and places it into a basket. Success is defined as the toy being fully placed in the basket.
- 3) Deformable toy rearrangement: the robot picks up a set of deformable duck toys from the table and rearranges them in a basket. Placing all toys into the basket is considered a success.

To create a robotic environment for behaviour verification and evaluation, we adopt a compositional scene-construction approach. We first construct the static environment from visual and geometric data, then build the robot’s virtual representation, and finally create mesh-based representations of the manipulated objects.

3DGS reconstruction: As shown in Fig. 2, the scene is captured from multiple viewpoints to obtain RGB images and depth measurements, enabling consistent modelling of appearance and metric geometry. We obtain object-level segmentation using SAM3D by lifting and fusing 2D masks into a coherent 3D representation. DINOv3 features provide semantic correspondence across views, while CLIP introduces language-aligned semantic information. Depth cues impose explicit geometric constraints. We integrate these multimodal features into an object-centric representation for geometry-aware scene reconstruction. We use the resulting representation to reconstruct the scene via 3D Gaussian Splatting, producing a continuous, photorealistic 3D model. Since all components are represented as Gaussian primitives, we align the static scene, the robot, and the objects within a shared

world frame.

We align the reconstructed GS scene with the robot URDF model using ICP, enabling consistent pose mapping between real and simulated environments. We sample points per link to ensure sufficient coverage. Before ICP, we use SuperSplat to adjust the scan pose such that the centre of the robot base frame coincides with the origin of the world frame. The static background GS is in the same frame as the robot GS, and we apply the same ICP-estimated transform. After aligning the static background, the robot, and the objects into a unified GS scene, forward kinematics updates the robot state in both joint and Cartesian spaces. During task execution (e.g., packing a sloth), the pose of the real robot is mapped to its counterpart in the GS scene: joint and gripper meshes are updated according to the robot’s rigid-body motion, and the robot in the GS environment is driven by the command sequence produced by the robot policy to evaluate task performance.

Metrics: To evaluate GS reconstruction quality, we report standard photometric rendering metrics (PSNR, SSIM, and LPIPS) in Table I [32]. The best result is in bold, and the second-best is underlined.

Baseline methods: We compare our reconstruction with representative 3DGS-based approaches. Specifically, we compare against GScream [33], VR-GS [34], Decoupled GS [35], POGS [36], D3DGS [37], and Embodied GS [38] on a robotic scene. We select these methods based on publicly reported results for robotic scene reconstruction or on the availability of source code for benchmarking.

TABLE I: **Average rendering performance on robotic scene reconstruction.** Our method achieves competitive performance across rendering metrics.

Methods	PSNR[db]↑	SSIM↑	LPIPS(vgg)↓
GScream [33]	17.82	0.590	0.56
VR-GS [34]	24.13	0.833	0.32
Decoupled GS [35]	27.32	<u>0.905</u>	0.30
POGS [36]	19.66	<u>0.869</u>	<u>0.098</u>
D3DGS [37]	12.05	0.399	0.23
Embodied GS [38]	14.95	0.389	0.36
Ours	<u>24.64</u>	0.963	0.044

Our method, with an SSIM of 0.963 and an LPIPS of 0.044, outperforms the baselines and ranks second on PSNR (24.64), offering a significant advantage for high-fidelity reconstruction of the robotic environment. Fig. 3 provides a qualitative comparison of our approach with POGS and Embodied GS (both of which use robotic tasks for benchmarking). Our method produces high-quality GS reconstructions with reduced artefacts in objects and robotic scenes, which are critical for policy simulation and evaluation. In contrast, POGS and Embodied GS struggle to render fine details and exhibit substantial noise in the reconstructed environment.

Geometry-aware spatio-temporal reasoning and planning: As shown in Fig. 4, given a natural-language task (e.g., packing a sloth into a basket), the system feeds an image captured by the wrist camera into a segmentation network. The system segments, masks, and labels task-relevant objects

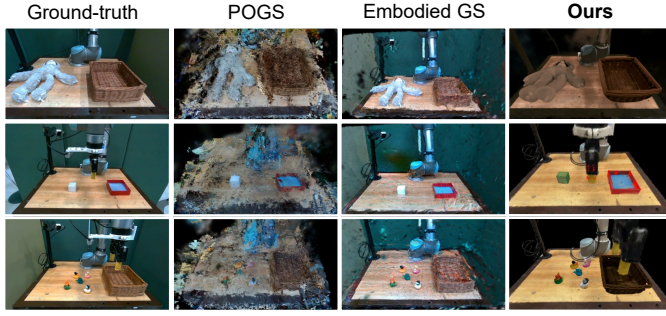


Fig. 3: **GS reconstruction rendering examples:** Our method achieves the best GS reconstruction performance compared with POGS and Embodied GS.

(e.g., the sloth and basket) with text and bounding boxes to produce an annotated image. The annotated image is then fed into a large language model (GPT-5) to generate keypoint candidates for object manipulation based on object geometry, with grasping and target keypoints indicated by points 1, 2, 3, 4, 5, and 6, respectively. The pre-trained model is then prompted to decompose the language task into $\mathcal{H} = 3$ subtasks (stages), each described in text. Here, we follow the prompt-design practice of [9]. For each subtask, we use 2D keypoint affordances, depth observations, and pose relations as inputs to subgoal and subpath constraint functions. Our method estimates feasible final object states by sampling end-effector poses while holding it, and candidates are sampled as described in Sec. III-C. These candidates chain keypoint-based subgoals together to generate robot trajectories.

Before execution on the real robot, we evaluate the generated manipulation trajectory in a 3DGS-based simulator, as shown at the bottom of Fig. 4. The robot follows the planned trajectory (blue line) to grasp the object (e.g., a sloth) at the grasp keypoint (blue point: keypoint 1). The gripper closes at the grasp point, after which the robot moves the object along the placement trajectory (green line). When part of the sloth (a soft object) contacts the target object (the basket), the embodied physics constraints of the PhysTwin-based simulator deform the contacted body part inside the basket. Specifically, the robot places the sloth’s main body into the basket at the target point (green point: keypoint 3). Other body parts (e.g., legs and arms) contact the basket edge, and simulated contact forces deform them within the basket. If the simulation is successful, the policy can be deployed on the real robot.

Robot policy evaluation in the simulator and rollout on the real robot: Fig. 5 shows results of behaviour evaluation in the simulator for cube packing and sloth packing (Rows 1 and 3). The point clouds of the cube (green) and box (red) are used for collision checking while the robot follows the generated trajectories. Specifically, our method measures the distances between the cube and box point clouds, with distances below a predefined threshold indicating a collision. The simulation results show that the robot can place the cube into the box by following the generated trajectory. The robot then executes the corresponding trajectory in the real

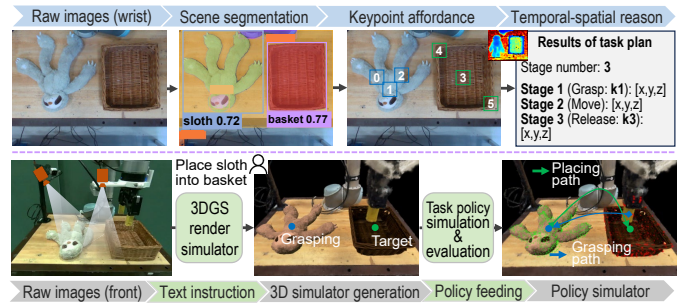


Fig. 4: **Geometry-aware spatio-temporal planning and policy evaluation.** Top: vision–language–conditioned keypoint-affordance representation and task planning in a 3D coordinate frame; bottom: policy evaluation in the constructed simulator using reconstructed robotic scenes.

world and successfully completes the task (Row 2). Similarly, we evaluate the sloth-packing policy in simulation before real-world rollout. Compared with rigid-object manipulation, deformable-object manipulation relies more heavily on contact interactions. The simulator therefore applies contact forces to deform the object and keep it within the basket. However, the simulated contact forces cannot exactly match those in the real world, which can lead to discrepancies between simulation and real-world deployment, particularly for body-part contact.

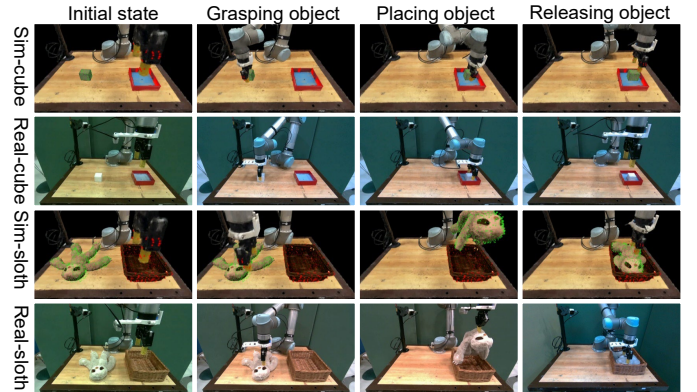


Fig. 5: **Simulation and real-world experiments for rigid and soft-object manipulation.** Rows 1 and 3 show simulation results, while Rows 2 and 4 show real-world results for cube placing and sloth packing, respectively.

To address discrepancies between simulation and real-world rollout, we implement a behaviour evaluation and replanning scheme that enables the robot to recover from failures. As shown in Fig. 6, a policy evaluated in simulation (orange) fails in the real world, leaving part of the sloth outside the basket. The control script then resets the scene and replans the task. The new plan (green) is evaluated in the simulator, and the results indicate that the sloth can be placed in the basket. Finally, the sim-to-real rollout on the real robot succeeds.

For each task, we record success over 10 trials, varying object positions across trials. If any subtask in the sequence fails (e.g., duck rearranging), subsequent subtasks are not executed. Fig. 7 (left) shows the correlation between success rates in simulation and real-world experiments across all tasks. Table II reports task success rates with and without replanning

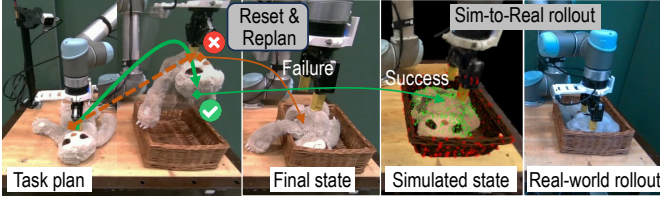


Fig. 6: **Policy evaluation and replanning.** A robot policy (orange) fails (the sloth is not fully packed into the basket), triggering replanning (green), which the simulator evaluates. If successful, the policy is rolled out on the robot.

TABLE II: **Success rate of tasks with/without replanning in real and simulation environments.**

Task	Without replanning		Replanning	
	sim	real	sim	real
Place cube into a box	80.0%	80.0%	100.0%	90.0%
Package sloth into a basket	60.0%	40.0%	100.0%	80.0%
Arrange ducks into a basket	60.0%	46.0%	80.0%	70.0%
Average	66.7%	55.3.0%	93.3%	80.0%

in simulation and the real world. We observe a positive correlation between simulation and real-world success rates. Simulation success rates are slightly higher than in real-world deployment due to the sim-to-real gap. Compared with results without replanning, simulation-based replanning yields higher success rates: object manipulation (cube and ducks) increases by 20 percentage points, while sloth packing increases by 40 percentage points. For real-robot rollouts, the success rate is lower than in replanning-enabled simulation but higher than in both simulation and real-world deployment without replanning. The average success rates across all tasks in simulation and real-world deployment are 66.7% and 55.3%, respectively. With simulation-based replanning, they increase to 93.3% and 80%, respectively.

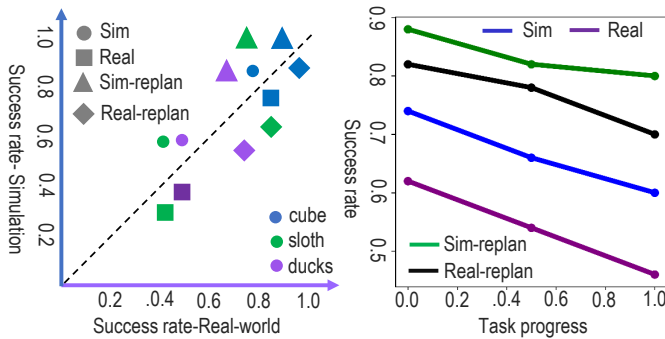


Fig. 7: Left: **Correlation of success rate in simulation and real-world experiments**; Right: **Success rate of each control step (subtask) measured on the physical robot over 10 trials for toy duck rearrangement.**

Toy duck rearrangement is a long-horizon, multi-stage task in which each episode requires placing five ducks into a basket, with each duck requiring three control steps (e.g., grasp, move, and release). We record the subtask success rate over 10 trials and normalise the results to [0, 1], as

shown in Fig. 7 (right). The results demonstrate that success rates degrade substantially as subtasks and stages progress. The task success rate in simulation is higher than in real-world deployment, reflecting the common sim-to-real gap. Replanning in both simulation and real-world deployment yields higher subtask-level consistency. This improvement is attributable to simulation and replanning, which assess task feasibility prior to action. Fig. 8 shows results for ducks 1–5 in simulation and real-world deployment from left to right. The top, middle, and bottom rows show the initial scene, grasping the duck, and placing the duck into the basket, respectively, in both the simulation and the real world.

V. Conclusions

We present Robot-GST, a geometry-aware spatio-temporal behaviour representation and evaluation framework for robotic manipulation that enables simulation-based behaviour verification before execution. Robot-GST reconstructs metrically aligned, photorealistic scenes from short RGB-D captures using 3D Gaussian Splatting with SAM3D-based object understanding, combining photorealistic rendering with explicit geometric proxies for collision and contact reasoning. For language-specified tasks, it grounds long-horizon behaviour generation in 3D keypoint affordances and relational constraints, simulates candidate behaviours, and evaluates them in the reconstructed environment. During execution, Gaussian-aware final-state estimation provides target states for efficient low-level trajectory planning, reducing error accumulation. Experiments on rigid, soft, and deformable tasks demonstrate improved closed-loop reliability and a positive relationship between simulation and real-world performance. Limitations arise from reconstruction and physics fidelity issues under occlusion, dynamic scene changes, and imperfect contact or deformation models, motivating future work on improved physical modelling and uncertainty-aware planning.

References

- [1] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” *The International Journal of Robotics Research*, vol. 44, no. 10–11, pp. 1684–1704, 2025.
- [2] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu *et al.*, “Octo: An open-source generalist robot policy,” *arXiv preprint arXiv:2405.12213*, 2024.
- [3] T. Wu, S. Li, J. Gong, C. Guo, X. Li, S. Mu, and W. Ding, “Ce: A unified interface for cross-embodiment visuomotor policy learning in 3d space,” *IEEE Robotics and Automation Letters*, 2026.
- [4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, “Openvla: An open-source vision-language-action model,” *arXiv preprint arXiv:2406.09246*, 2024.
- [5] P. Intelligence, A. Amin, R. Aniceto, A. Balakrishna, K. Black, K. Conley, G. Connors, J. Darpinian, K. Dhabalia, J. DiCarlo *et al.*, “ $\pi_{0.6}^*$: a vla that learns from experience,” *arXiv preprint arXiv:2511.14759*, 2025.
- [6] Y. Zhao, F. Jiang, T. Liu, B. Zeng, Y. Liu, L. Wang, and W. Luo, “Difficulty-estimated policy optimization,” *arXiv preprint arXiv:2602.06375*, 2026.
- [7] K. Zhang, S. Sha, H. Jiang, M. Loper, H. Song, G. Cai, Z. Xu, X. Hu, C. Zheng, and Y. Li, “Real-to-sim robot policy evaluation with gaussian splatting simulation of soft-body interactions,” *arXiv preprint arXiv:2511.04665*, 2025.

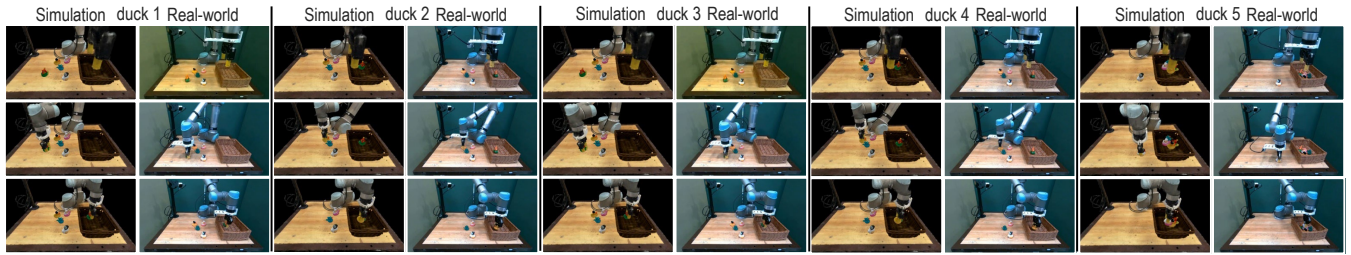


Fig. 8: **Simulation and real-world experiments for duck toy rearrangement.** Left→Right shows results for ducks 1–5. “Simulation” and “Real-world” show results from policy simulation and policy rollout on the physical robot.

- [8] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, “Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation,” *arXiv preprint arXiv:2409.01652*, 2024.
- [9] F. Liu, K. Fang, P. Abbeel, and S. Levine, “Moka: Open-world robotic manipulation through mark-based visual prompting,” *arXiv preprint arXiv:2403.03174*, 2024.
- [10] W. Yuan, J. Duan, V. Blukis, W. Pumacay, R. Krishna, A. Murali, A. Mousavian, and D. Fox, “Robopoint: A vision-language model for spatial affordance prediction for robotics,” *arXiv preprint arXiv:2406.10721*, 2024.
- [11] E. Aljalbout, J. Xing, A. Romero, I. Akinola, C. R. Garrett, E. Heiden, A. Gupta, T. Hermans, Y. Narang, D. Fox *et al.*, “The reality gap in robotics: Challenges, solutions, and best practices,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 9, 2025.
- [12] M. U. Din, W. Akram, L. S. Saoud, J. Rosell, and I. Hussain, “Vision language action models in robotic manipulation: A systematic review,” *arXiv preprint arXiv:2507.10672*, 2025.
- [13] B. Kerbl, G. Kopanas, T. Leimkühler, G. Drettakis *et al.*, “3d gaussian splatting for real-time radiance field rendering,” *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.
- [14] M. Ji, R.-Z. Qiu, X. Zou, and X. Wang, “Graspsplats: Efficient manipulation with 3d feature splatting,” *arXiv preprint arXiv:2409.02084*, 2024.
- [15] H. Jiang, H.-Y. Hsu, K. Zhang, H.-N. Yu, S. Wang, and Y. Li, “Phys-twin: Physics-informed reconstruction and simulation of deformable objects from videos,” in *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2025, pp. 7219–7230.
- [16] J. Abou-Chakra, L. Sun, K. Rana, B. May, K. Schmeckpeper, M. Vittoria Minniti, and L. Herlant, “Real-is-sim: Bridging the sim-to-real gap with a dynamic digital twin for real-world robot policy evaluation,” *arXiv e-prints*, pp. arXiv–2504, 2025.
- [17] A. Badithela, D. Snyder, L. Zha, B. Mikhail, M. O’Kelly, A. Dixit, and A. Majumdar, “Reliable and scalable robot policy evaluation with imperfect simulators,” *arXiv preprint arXiv:2510.04354*, 2025.
- [18] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, “Do as i can, not as i say: Grounding language in robotic affordances,” *arXiv preprint arXiv:2204.01691*, 2022.
- [19] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, “Inner monologue: Embodied reasoning through planning with language models,” *arXiv preprint arXiv:2207.05608*, 2022.
- [20] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, “Code as policies: Language model programs for embodied control,” in *2023 IEEE International conference on robotics and automation (ICRA)*. IEEE, 2023, pp. 9493–9500.
- [21] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choremanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” *arXiv preprint arXiv:2307.15818*, 2023.
- [22] Z. Qi, W. Zhang, Y. Ding, R. Dong, X. Yu, J. Li, L. Xu, B. Li, X. He, G. Fan *et al.*, “Sofar: Language-grounded orientation bridges spatial reasoning and object manipulation,” *Advances in neural information processing systems*, vol. 38, pp. 76367–76412, 2026.
- [23] 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.
- [24] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik, “Lerf: Language embedded radiance fields,” in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2023, pp. 19672–19682.
- [25] 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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2024, pp. 20310–20320.
- [26] S. Zhou, H. Chang, S. Jiang, Z. Fan, Z. Zhu, D. Xu, P. Chari, S. You, Z. Wang, and A. Kadambi, “Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields,” in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2024, pp. 21676–21685.
- [27] C. Yan, D. Qu, D. Xu, B. Zhao, Z. Wang, D. Wang, and X. Li, “Gs-slam: Dense visual slam with 3d gaussian splatting,” in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2024, pp. 19595–19604.
- [28] Y. Li, Y. Fang, Z. Zhu, K. Li, Y. Ding, and F. Tombari, “4d gaussian splatting slam,” in *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2025, pp. 25019–25028.
- [29] M. N. Qureshi, S. Garg, F. Yandun, D. Held, G. Kantor, and A. Silwal, “SplatSim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting,” in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 6502–6509.
- [30] M. Sedlacek, P. Yefanov, G. Ponomatkin, J. Bardhan, S. Pilc, M. Fourmy, E. Kazakos, C. G. Snoek, J. Sivic, and V. Petrik, “Realm: A real-to-sim validated benchmark for generalization in robotic manipulation,” *IEEE Robotics and Automation Letters*, 2026.
- [31] Y. Fang, Y. Yang, X. Zhu, K. Zheng, G. Bertasius, D. Szafrir, and M. Ding, “Rebot: Scaling robot learning with real-to-sim-to-real robotic video synthesis,” in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025, pp. 11351–11358.
- [32] E. Sandström, Y. Li, L. Van Gool, and M. R. Oswald, “Point-slam: Dense neural point cloud-based slam,” in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2023, pp. 18387–18398.
- [33] Y. Wang, Q. Wu, G. Zhang, and D. Xu, “Learning 3d geometry and feature consistent gaussian splatting for object removal,” in *European conference on computer vision*. Springer, 2024, pp. 1–17.
- [34] Y. Jiang, C. Yu, T. Xie, X. Li, Y. Feng, H. Wang, M. Li, H. Lau, F. Gao, Y. Yang *et al.*, “Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality,” in *ACM SIGGRAPH 2024 conference papers*, 2024, pp. 1–1.
- [35] M. Wang, Y. Zhang, W. Xu, R. Ma, C. Zou, and D. Morris, “Decoupledgaussian: Object-scene decoupling for physics-based interaction,” in *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2025, pp. 11361–11372.
- [36] J. Yu, K. Hari, K. El-Refai, A. Dalal, J. Kerr, C. M. Kim, R. Cheng, M. Z. Irshad, and K. Goldberg, “Persistent object gaussian splat (pogs) for tracking human and robot manipulation of irregularly shaped objects,” in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 3211–3218.
- [37] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan, “Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis,” in *2024 International Conference on 3D Vision (3DV)*. IEEE, 2024, pp. 800–809.
- [38] J. Abou-Chakra, K. Rana, F. Dayoub, and N. Suenderhauf, “Physically embodied gaussian splatting: A visually learnt and physically grounded 3d representation for robotics,” in *8th Annual Conference on Robot Learning*, 2024.