

# Scan, Materialize, Simulate: A Generalizable Framework for Physically Grounded Robot Planning

Amine Elhafsi<sup>1</sup>, Daniel Morton<sup>1</sup>, Marco Pavone<sup>1,2</sup>

**Abstract**—Autonomous robots must reason about the physical consequences of their actions to operate effectively in unstructured, real-world environments. We present *Scan, Materialize, Simulate* (SMS), a unified framework that combines 3D Gaussian Splatting for accurate scene reconstruction, visual foundation models for semantic segmentation, vision-language models for material property inference, and physics simulation for reliable prediction of action outcomes. By integrating these components, SMS enables generalizable physical reasoning and object-centric planning without the need to relearn foundational physical dynamics. We empirically validate SMS in a billiards-inspired manipulation task and a challenging quadrotor landing scenario, demonstrating robust performance on both simulated domain transfer and real-world experiments. Our results highlight the potential of bridging differentiable rendering for scene reconstruction, foundation models for semantic understanding, and physics-based simulation to achieve physically grounded robot planning across diverse settings. Our project page with additional materials can be found at <https://sites.google.com/view/scan-materialize-simulate>.

## I. INTRODUCTION

In recent years, autonomous robotic systems have demonstrated remarkable capabilities, paving the way for deployment across a wide range of real-world applications. However, tasks that require a robot to reason about the physical consequences of its actions remain particularly challenging. Many existing systems tackle these challenges only in highly structured environments, under rigid assumptions, or with highly specialized algorithms; for example, custom solutions have been developed to enable robots to play physically challenging games such as Jenga [1], billiards [2], [3] and table tennis [4]. Beyond such recreational settings, diverse domains motivate the need for robust reasoning about physical interactions in unstructured environments, whether a robot is negotiating unstable rubble in disaster response, stacking irregular loads in warehouses, or handling equipment in microgravity space environments. In each of these scenarios, the robot’s success hinges not just on accurate control, but on anticipating the dynamics of the interactions between itself and the world.

Developing a methodology for physics-aware planning requires the careful integration of several capabilities. The robot must perceive and reconstruct its environment with sufficient geometric detail to support nuanced interactions. Yet bulk geometric reconstruction alone is not enough; an

understanding of the scene’s semantics is also required to segment individual entities and assign material characteristics (e.g., density, friction), which directly affect how each object reacts to contact and/or manipulation. Moreover, the system should incorporate a model to accurately predict the outcomes of both robot-object and object-object interactions, enabling the robot to plan actions that reliably anticipate the environment’s response.

Motivated by these challenges and building upon recent breakthroughs in differentiable rendering, foundation models, and physics simulation, we propose **Scan, Materialize, Simulate** (SMS), a framework for physically grounded robot action planning, comprised of the three eponymous modules: (1) **Scan**: The robot observes the scene from multiple viewpoints to acquire a set of color-and-depth (RGBD) images used to construct a detailed geometric representation using 3D Gaussian Splatting (3DGS) [5]. Visual foundation models (VFMs) perform object detection [6] and segmentation [7] on the RGBD images, allowing volumetric primitives to be associated with their corresponding objects.

(2) **Materialize**: These volumetric primitives are converted into physics-ready meshes, with estimated material attributes (e.g., density, friction) inferred by a vision-language model (VLM).

(3) **Simulate**: The reconstructed meshes are instantiated in a physics simulation wherein candidate robot actions are optimized before being enacted in the target environment.

Collectively, SMS blends the generalist, semantic understanding provided by learning-based foundation models with the principled consistency of physical simulation, enabling generalizable, semantically-aware physically grounded robot planning. We illustrate this framework in Figure 1.

## II. RELATED WORK

**Learning-Based Robot Policies:** Reinforcement learning (RL) [8]–[10] and imitation learning (IL) [11]–[14] have reigned as the dominant paradigms for developing robot policies in recent years. Although physical priors are generally not explicitly enforced, these policies must implicitly encode knowledge of the physical phenomena pertaining to the task at hand (e.g., gravity, gripper friction). This knowledge is provided implicitly via the training environments [15], [16] in the case of RL, or the training demonstrations for IL. However, these learned policies are well known to suffer significant performance degradation when confronted with out-of-distribution (OOD) data [17], [18], and we cannot expect their implicit physical knowledge to generalize to

This work was funded by the DARPA TIAMAT program.

<sup>1</sup>Amine Elhafsi, Daniel Morton, and Marco Pavone are with Stanford University, Stanford, CA 94305, USA {amine, dmorton, pavone}@stanford.edu.

<sup>2</sup>Marco Pavone is also affiliated with NVIDIA Research.

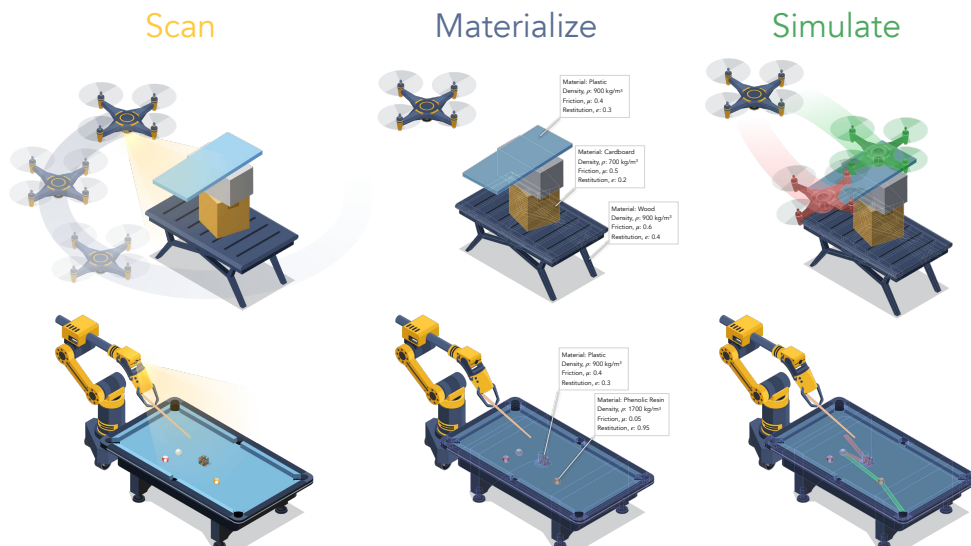


Fig. 1: Overview of the **Scan, Materialize, Simulate (SMS)** framework for physically grounded robot action planning. SMS consists of three steps: scanning to build a geometric environment model, materializing the reconstruction into a simulation-ready representation, and simulating to optimize actions in a virtual environment prior to target-environment execution.

the infinitude of possible world configurations a robot may encounter.

High-capacity vision-language-action (VLA) models [19]–[24] for robotics have also begun to emerge alongside the broader wave of foundation models [25]. Trained on massive, internet-scale, datasets, these models demonstrate strong generalization, semantic reasoning, and zero-shot performance on a variety of manipulation and navigation tasks. However, their primary strength lies in their semantic and contextual understanding of diverse tasks and scenes, while their demonstrated applications rarely extend to the physically dynamic tasks we consider in this work.

**Differentiable Rendering in Robotics:** The advent of modern differentiable rendering methods [5], [26]–[28] has provided powerful new tools for scene reconstruction in robotics. Methods built around these representations have quickly made their way into a range of robotics applications, serving as highly detailed obstacle maps for motion planning [29]–[31], providing new means for object tracking [32], [33], performing system identification and visuo-motor control [34], [35], and underpinning visually accurate virtual environments for policy training in simulation [36]–[41] and synthetic data generation [42]. Semantic features have also been integrated into these reconstructions [43]–[45], enabling robots to interpret local scene semantics and manipulate objects according to natural language task specifications [46], [47].

Of particular relevance to this work are the methods which integrate physics into the reconstructed environment geometry, such as [48]–[50], which deform the scene geometry according to the material point method (MPM) [51]. However, these methods are primarily oriented towards graphics applications, with the main objective being to produce plausible and visually appealing animated scenery. Some

of the aforementioned works such as [32], [33], [40] have explored physically simulating the motion of reconstructed scene components under predicted forces or interactions, but their focus has largely been on manipulation scenarios, often targeting specific object interactions. In contrast, our work aims to enable robots to plan and reason about physically grounded interactions, bridging the gap between physical scene reconstruction and robotic action planning.

**Statement of Contributions:** In this work, we present SMS, a novel framework for physically grounded robot action planning. SMS integrates 3D Gaussian Splatting (3DGS) for accurate reconstruction of intricate scene geometry, visual foundation models (VFMs) for semantic understanding and precise object-level segmentation, a vision-language model (VLM) for context-aware material and attribute inference, and a physics engine for consistent prediction of action outcomes based on well-understood Newtonian mechanics. By combining these components, SMS enables generalizable object recognition, scene decomposition, and reliable action-conditioned physical prediction.

We demonstrate the performance of SMS in a high-precision, billiards-inspired task, where a manipulator must strike a cue ball to direct a target ball to a goal, and a precarious landing scenario, where a quadrotor must safely perch atop an unstable stack of objects. We report performance in simulated domain transfer experiments and validate our method in the real-world with a physical quadrotor. SMS provides a planning strategy that is more generalizable and adaptable than application-specialized algorithms, and delivers superior physical reasoning compared to purely learning-based methods, especially in tasks that demand principled treatment of physical dynamics and interactions.

### III. PHYSICALLY GROUNDED PLANNING WITH SMS

In this work, we consider a setting where a robot must perform a task involving physical interaction with the environment, formulated as an optimal control problem. Since modeling the full dynamics of the world is often intractable beyond simple cases, SMS addresses this challenge by reconstructing a surrogate model of the environment. It uses a physics engine to approximate discrete-time dynamics and solves the resulting surrogate optimization problem:

$$\begin{aligned} & \underset{\mathbf{u}_{0:T-1}}{\text{minimize}} && \hat{L}(\hat{\mathbf{x}}_{0:T}, \mathbf{u}_{0:T-1}) \\ & \text{subject to} && \hat{\mathbf{x}}_{k+1} = \hat{f}(\hat{\mathbf{x}}_k, \mathbf{u}_k), \quad k = 0, \dots, T-1, \\ & && \hat{\mathbf{x}}_0 = \hat{\mathbf{x}}_{\text{init}}. \end{aligned}$$

Here,  $\hat{L}$  denotes the surrogate objective,  $\hat{f}$  represents the discrete-time surrogate dynamics provided by the physics engine, and  $\hat{\mathbf{x}}_k$  is the reconstructed environment's state at time step  $k$ . We define the discrete-time state and control trajectories as  $\hat{\mathbf{x}}_{0:T} = (\hat{\mathbf{x}}_0, \dots, \hat{\mathbf{x}}_T)$  and  $\mathbf{u}_{0:T-1} = (\mathbf{u}_0, \dots, \mathbf{u}_{T-1})$ , respectively. In the sections that follow, we outline our approach for constructing the virtual environment and solving the associated surrogate optimal control problem.

#### A. Scan

The first step of our pipeline aims to create a detailed 3D reconstruction of the scene. Precise scene geometry is necessary for physical reasoning, but object-level segmentation is essential as it enables the identification and tracking of individual entities, allowing for the simulation of their interactions with each other and with the robot.

To achieve this, the robot first observes the scene from multiple angles, collecting a sequence of  $N$  RGBD frames,  $\mathbf{F} = (\mathbf{F}_1, \dots, \mathbf{F}_N)$ , along with their associated camera poses. Frame  $\mathbf{F}_i$  consists of color  $\mathbf{I}_i \in \mathbb{R}^{H \times W \times 3}$  and depth  $\mathbf{D}_i \in \mathbb{R}^{H \times W}$ , where  $H$  and  $W$  are the image height and width, respectively. We employ OWLv2 [6] for zero-shot object detection in the initial image, and use these detections to prompt the Segment Anything Model 2 (SAM 2) [7] for segmentation and tracking across the sequence  $\mathbf{F}$ , resulting in the segmentation frames  $\mathbf{S} = (\mathbf{S}_1, \dots, \mathbf{S}_N)$ . Each segmentation frame  $\mathbf{S}_i \in \mathbb{N}^{H \times W}$  specifies an object class  $k \in \{1, \dots, K\}$  for each pixel in image  $i$ .

We reconstruct the scene using Gaussian splatting [5], which optimizes a collection of 3D Gaussians with indices  $\{1, \dots, P\}$  to minimize the difference between rendered and observed images from a set of training viewpoints. A given Gaussian  $\mathbf{G}_i$  is parametrized by its position  $\mathbf{p}_i \in \mathbb{R}^3$ , rotation matrix  $\mathbf{R}_i \in \mathbf{SO}(3)$ , scale  $\mathbf{s}_i \in \mathbb{R}^3$ , opacity  $\mathbf{o}_i \in [0, 1]$ , and color  $\mathbf{c}_i \in \mathbb{R}^3$ . Gaussian  $i$  can be projected to the pixel coordinates of viewpoint  $v$  as  $\mathbf{p}_i^{\text{image}} = \mathbf{M}_v \mathbf{p}_i$ , where  $\mathbf{M}_v \in \mathbb{R}^{3 \times 4}$  is a projection matrix mapping homogeneous world coordinates to homogeneous pixel coordinates. Given the world-to-camera rotation matrix,  $\mathbf{R}_v^{\text{cw}} \in \mathbf{SO}(3)$ , the color  $C$  at pixel coordinates  $\mathbf{u}$  can be rendered from  $m$  ordered Gaussians (by increasing distance from the camera) as

$$C(\mathbf{u}) = \sum_{j \leq m} \mathbf{c}_j \alpha_j \prod_{k < j} (1 - \alpha_k),$$

where  $\alpha_j = o_j \exp\left(-\frac{1}{2}(\mathbf{u} - \mathbf{p}_j^{\text{image}})^\top \boldsymbol{\Sigma}_j (\mathbf{u} - \mathbf{p}_j^{\text{image}})\right)$  and  $\boldsymbol{\Sigma}_j = \mathbf{R}_v^{\text{cw}} \text{diag}(\mathbf{s}_j)^2 \mathbf{R}_v^{\text{cw}\top}$ .

In [5], the parameters of the Gaussians are iteratively optimized via gradient descent to minimize the photometric loss between the rendered and observed image from a sampled training view  $\tau$ ,

$$L_{\text{photo}}^{(\tau)} = \frac{(1 - \lambda_{\text{photo}})}{HW} \|\hat{\mathbf{I}}_\tau - \mathbf{I}_\tau\|_1 + \lambda_{\text{photo}} (1 - \text{SSIM}(\hat{\mathbf{I}}_\tau, \mathbf{I}_\tau)),$$

where  $\lambda_{\text{photo}}$  is a weighting term, and SSIM is the structural similarity index measure [52]. Hereafter, we use hatted notation to indicate rendered quantities, with the non-hatted counterpart denoting the training datum. Optionally, an isotropic regularization loss can be added to penalize the Gaussians in view from becoming excessively eccentric or skinny:

$$L_{\text{isotropic}}^{(\tau)} = \frac{1}{P} \sum_{i=1}^P \mathbf{1}^\top |\mathbf{s}_i - \bar{\mathbf{s}}|,$$

where  $\mathbf{1}$  indicates a vector of ones and the bar notation indicates the mean value of the vector.

Beyond just visual appearances, we also need to accurately reconstruct geometric detail. To this end, we respectively render pixelwise depth following [53] and surface normal vectors following [54] as

$$\begin{aligned} D(\mathbf{u}) &= \sum_{j \leq m} \mathbf{p}_j^{\text{image}, z} \alpha_j \prod_{k < j} (1 - \alpha_k), \text{ and} \\ \mathbf{N}(\mathbf{u}) &= \sum_{j \leq m} \mathbf{n}_j \alpha_j \prod_{k < j} (1 - \alpha_k), \end{aligned}$$

where the normal direction is taken to be a Gaussian's shortest principal axis direction  $\mathbf{n}_j = \mathbf{R}_j \text{OneHot}(\arg \min(\mathbf{s}_j))$ . Accordingly, we introduce the respective depth and normal loss terms,

$$\begin{aligned} L_{\text{depth}}^{(\tau)} &= \frac{1}{HW} \|\hat{\mathbf{D}}_\tau - \mathbf{D}_\tau\|_1, \text{ and} \\ L_{\text{normal}}^{(\tau)} &= \frac{\lambda_{\text{normal}}}{HW} \sum_{\mathbf{u}} \|\hat{\mathbf{N}}_\tau(\mathbf{u}) - \mathbf{N}_\tau(\mathbf{u})\|_1 \\ &\quad + \frac{\lambda_{\text{scale}}}{P} \sum_{i=1}^P \|\arg \min(\mathbf{s}_i)\|_1. \end{aligned}$$

Depth maps are directly obtained from RGBD data, while normal vectors can be computed from the observed depth analytically [55]. Note we add an extra term in  $L_{\text{normal}}^{(\tau)}$  to promote the optimization of flat, disk-like Gaussians to more closely map the scene surface geometry. We control the normal mapping and Gaussian scaling regularization with  $\lambda_{\text{normal}}$  and  $\lambda_{\text{scale}}$ , respectively.

Finally, to enable entity-wise segmentation, we augment the Gaussians with randomly initialized affinity feature vectors  $\mathbf{a}_i \in \mathbb{R}^d$ , which are optimized to associate each Gaussian with one of the  $K$  entities in the scene. We render these features as

$$\mathbf{A}(\mathbf{u}) = \sum_{j \leq m} \mathbf{a}_j \alpha_j \prod_{k < j} (1 - \alpha_k),$$

and pass them through a classifier  $\phi$  (implemented as a two-layer neural network, trained jointly with the Gaussians) to obtain logits  $\mathbf{p}(\mathbf{u}) = \phi(\mathbf{A}(\mathbf{u})) \in \mathbb{R}^K$  and predicted object label at pixel  $\mathbf{u}$  as  $y(\mathbf{u}) = \text{softargmax}(\mathbf{p}(\mathbf{u}))$ . To optimize

the affinity features and the classifier, we use the cross-entropy loss between the predicted logits  $p_c$ , for each class  $c$ , and the assigned label  $\mathbf{S}(\mathbf{u})$  for each pixel  $\mathbf{u}$ :

$$L_{\text{segmentation}}^{(\tau)} = -\frac{1}{HW} \sum_{\mathbf{u}} \log \left( \frac{\exp(p_{\mathbf{S}(\mathbf{u})}(\mathbf{u}))}{\sum_{c=1}^K \exp(p_c(\mathbf{u}))} \right).$$

Similar to [53], our implementation of Gaussian splatting optimizes each frame sequentially. Upon reaching a new frame, Gaussians are seeded from a subset of the back-projected RGBD point cloud, with sampling biased towards low-density, unmapped regions, or regions with high depth error. At the first frame, points are sampled primarily from edges detected from the color image, we perform 1000 gradient descent steps. Subsequent frames are optimized for 100 steps. While optimizing a particular frame, a gradient descent step is occasionally taken for a randomly sampled previous frame to avoid catastrophic forgetting. The overall loss that we compute for frame viewpoint  $\tau$  is:

$$L^{(\tau)} = L_{\text{photo}}^{(\tau)} + L_{\text{isotropic}}^{(\tau)} + L_{\text{depth}}^{(\tau)} + L_{\text{normal}}^{(\tau)} + L_{\text{segmentation}}^{(\tau)}.$$

### B. Materialize

Once the environment is mapped and distinct entities are localized, we must convert the Gaussian splatting reconstruction into a set of simulation-ready meshes, a format compatible with most physics simulators [56]–[58]. Entity geometries are recovered by grouping Gaussians using their affinity features, with their centers forming an initial object point cloud. Since the optimized Gaussians can vary significantly in scale and orientation, this point cloud often yields nonuniform surface coverage. As such, we exploit each Gaussian’s extended geometry which was optimized to be flat, disk-like, and tightly adherent to the scene surface by locally sampling additional points from the plane defined by its two largest principal axes. This produces a uniformly dense surface point cloud for subsequent processing.

To robustly reconstruct irregularly shaped objects, we perform a Delaunay tetrahedralization [59], [60] on the densified point cloud to generate a volumetric mesh. This tetrahedralization yields the convex hull of an object, but we can recover non-convex geometry by pruning tetrahedra with edges exceeding a set, or adaptively determined, length threshold. The final object mesh is extracted by retaining only the external faces and vertices of this pruned tetrahedralization. For entities identified as approximately spherical, such as balls, we instead fit a sphere to the densified point cloud via random sample consensus (RANSAC) [61] in order to preserve rolling dynamics in simulation.

Finally, to support realistic physical behavior, we prompt OpenAI’s GPT-4o VLM with an image of each detected entity, querying it for the dominant material class along with relevant properties such as the density and the coefficients of friction and restitution. Leveraging a VLM as such enables recognition of arbitrary, previously unseen objects and taps into the broad material knowledge encoded in the large language model backbone. This approach offers far greater flexibility than manually maintained lookup tables

or fixed classifiers, allowing our system to infer physical properties for a wide range of objects encountered in real-world environments with minimal prior information.

### C. Simulate

The meshed entities are imported into a physics simulator together with a detailed model of the robot, which is readily available in a compatible format for most popular platforms. We define a task-specific objective function and optimize the robot actions using the simulator as a rigid-body dynamics constraint module, predicting the environment’s response to candidate actions.

Given the highly nonlinear and potentially discontinuous nature of these dynamic tasks, we employ the Nelder-Mead simplex optimization algorithm [62], which is a robust, gradient-free method well suited for low- to moderate-dimensional problems. The optimizer iteratively refines candidate actions until convergence or a fixed iteration budget is reached, after which the best solution found can be deployed in the target environment.

## IV. EXPERIMENTS

To assess the performance of SMS, we conduct experiments on two challenging domains: a billiards-inspired manipulation task and a quadrotor landing scenario. In both settings, we evaluate SMS using simulated domain transfer experiments with the target environment constructed in Nvidia Isaac Sim, a photorealistic platform with high-fidelity physical simulation. We also demonstrate a landing scenario with a physical quadrotor to validate our method’s real-world applicability. We briefly explain the experimental setup in these sections, but provide implementation details as well as additional results in the appendix on our project page.

### A. Billiards Scenario

In the billiards-inspired task, a Franka Research 3 (FR3) robot is positioned on a flat surface. In each scene, a cue ball is placed in front of the robot within a defined region, while a target billiard ball and a goal position are randomly sampled elsewhere in the scene. Several obstacles representing common household objects (e.g., toys, dishware, decor) of varied geometry and material are randomly selected from the photorealistic asset library in Nvidia Omniverse and scattered throughout the workspace. Entities in the scene, including the robot and floor, are manually assigned realistic physical parameters, such as friction, restitution, and mass, to ensure realistic behavior. We reconstruct each scene using 60 images, with optimization averaging 3 seconds per frame for a total mapping time of around 3 minutes.

The robot’s task is to strike the cue ball with its end effector fingers to drive the target ball as close as possible to the specified goal. The robot’s trajectory,  $\mathbf{u}_{0:T-1}$ , is parametrized by a contact position  $\mathbf{p}_{\text{contact}}$ , a contact speed  $v_{\text{strike}}$ , and strike angle  $\theta_{\text{strike}}$ . The robot executes a linear sweeping motion, accelerating to reach the contact position (i.e., the estimated cue ball center position) with the desired velocity. The strike angle determines the orientation of this

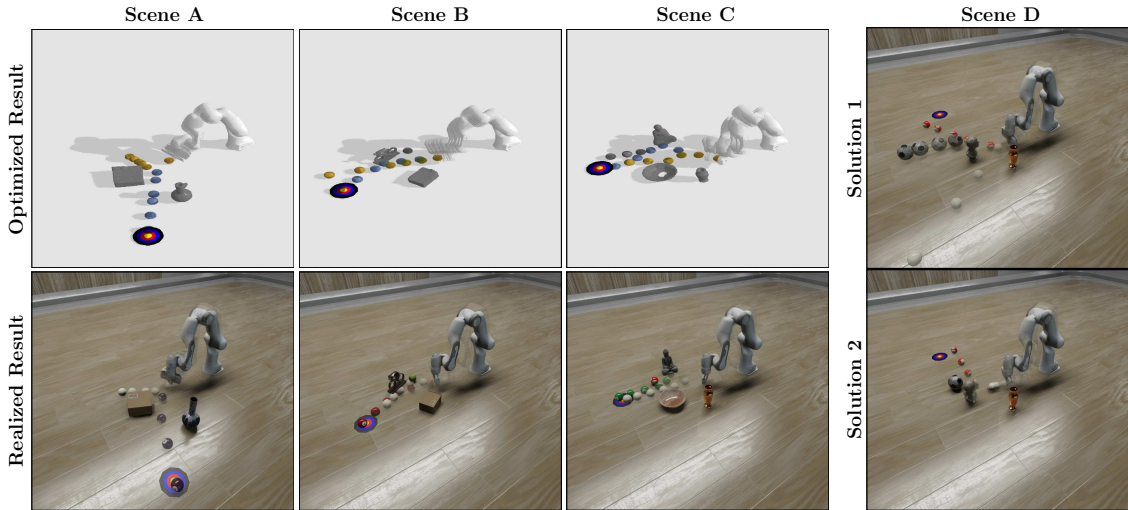


Fig. 2: Billiards scenes. **Left:** Motion composite images show the actions optimized in the reconstructed virtual environment (top) and subsequently realized in the target environment (bottom) for Scenes A, B, and C (cue and target balls colored yellow and blue, respectively, and the target is indicated by the bullseye pattern). Scene A depicts a direct shot from cue to target ball. Scene B involves a rebound off the heavy brass sculpture. Scene C features a complex double-rebound involving the Buddha statue and the billiard ball. **Right:** For the more challenging Scene D, SMS finds two distinct solutions.

linear path about the contact point. The task objective is defined as the minimum Euclidean distance between the target ball and the goal over the planning horizon, i.e.,  $\hat{L}_{\text{bill}} = \min_{t \in \{0, 1, \dots, T\}} \|\mathbf{p}_{\text{ball}, t} - \mathbf{p}_{\text{goal}, t}\|_2$ , encouraging the target ball to pass over the goal position at some point during execution, analogous to sinking a ball in pool or putting in golf. Actions are optimized using PyBullet [57] as our virtual physics environment. Nelder-Mead optimization averages 1.1 seconds per iteration and typically converges within 15 to 20 of the allotted 30 iterations.

We evaluate SMS over 18 generated scenes. Since our optimizer is sensitive to the initial guess, we optimize each scene from 30 random initializations. We present quantitative performance figures in Table I aggregated across all scenes. The table compares the task objective function predicted for the optimal result in the virtual environment and the realized objective in the target environment. We report the mean and standard deviation of the objective function across all evaluations, as well as those across the individual best and worst evaluation in each scene.

In Table I, we also include results for our baseline comparison method. This baseline searches over a dense grid of candidate contact speeds and strike angles at the cue ball position, assuming the initial cue-target ball collision is elastic. To account for future collisions with obstacles, a top-down RGBD image is used to construct a static obstacle mask and trace the post-collision path of the target ball with impacts reflecting the target ball’s normal velocity component. Since the baseline produces a single deterministic action, the performance is computed over the 18 per-scene trials and compared against SMS’s mean per-scene best performance in Table I. We find that our method’s best performing action per scene consistently outperforms this baseline. The baseline can be competitive in simple scenes

with few collisions or heavier, immobile obstacles. However, SMS especially outperforms this baseline in scenes with complex or dynamic interactions.

TABLE I: Billiards performance for SMS and the baseline. Values are mean (standard deviation) of the task objective,  $\hat{L}_{\text{bill}}$ , in centimeters. “Predicted” and “realized” refer to performance in the virtual and target environments, respectively.

Performance	SMS Predicted	SMS Realized	Baseline Predicted	Baseline Realized
<b>Overall</b>	5.5 (12.7) <sup>a</sup>	20.5 (17.9) <sup>a</sup>	–	–
<b>Per-Scene Best</b>	0.0 (0.0) <sup>b</sup>	5.5 (8.3) <sup>b</sup>	5.76 (10.2) <sup>b</sup>	28.2 (21.7) <sup>b</sup>
<b>Per-Scene Worst</b>	27.1 (19.3) <sup>b</sup>	41.2 (22.1) <sup>b</sup>	–	–

<sup>a</sup> Aggregated over all 18 scenes  $\times$  30 seeds for SMS; not applicable to deterministic baseline.

<sup>b</sup> Mean and standard deviation across 18 scenes (best or worst per scene).

To illustrate SMS in action, we showcase four diverse scenes in Figure 2, each varying in object material and geometry. These visualizations demonstrate not only the range of environments considered but also the nuanced behavior enabled by our planning approach. In Scene A, SMS selects a direct, traditional shot from cue to target ball. In Scene B, the planner exploits physical properties, opting to rebound the target ball off the heavy brass sculpture (whereas in Scene A, the lightweight cardboard box is avoided as an ineffective intermediary). Scene C exhibits even greater complexity: the solution involves an initial sharp-angle rebound off the Buddha statue, followed by a second corrective collision with the billiard ball to reach the goal - a result that could not be produced by our baseline. Notably, the actions optimized in the virtual environment translate well to execution, with predicted outcomes closely matching realized trajectories across all scenes. Finally, Scene D illustrates the method’s ability to discover multiple valid strategies, utilizing rebounds off the marble statue and soccer ball, once

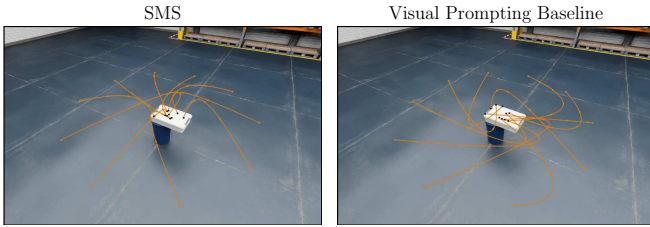


Fig. 3: Quadrotor approach comparison. **Left:** SMS optimizes approach paths that avoid the overhanging ledge. **Right:** Visual prompting does not consider propeller wash, resulting in flight over the overhang. Note several landing sites are chosen at the edges of the landing platform.

again demonstrating awareness of how object geometry and material affect momentum transfer.

For more granular results, we further compare the distributions of predicted and realized performance for all 18 scenes in our appendix. Some degradation in realized performance is observed, with some difference naturally arising due to domain mismatch, mesh reconstruction errors, and slight discrepancies in material property estimates. The impact is most pronounced in scenes with high object density, such as Scene C, where small errors compound through successive collisions, and in long-range shots as in Scene D, which amplify trajectory divergence. Nonetheless, these experiments highlight SMS’s ability to account for scene geometry and material properties, while frequently uncovering multiple effective strategies for task completion.

### B. Quadrotor Landing Scenario

In this setting, a quadrotor is tasked with safely landing on an unstable landing structure constructed by placing a flat landing platform (e.g., a cardboard box or plastic tote) atop a supporting base such as a step ladder, stool, or recycling bin. The landing platform is positioned in such a way that it significantly overhangs the base, making landing site selection critical to avoid causing the platform or structure to topple. As in the billiards scenario, objects are drawn from Nvidia’s Omniverse asset library and assigned realistic physical properties, and each scene is reconstructed using 60 images.

Beyond the quadrotor’s landing impulse, the propeller wash can also impart significant forces on the landing structure on approach. Specifically, flying over an overhanging region of the landing platform can induce moment about the base, causing it to topple before the quadrotor has a chance to land. As such, for this task we jointly optimize the quadrotor’s approach path and landing position. We parametrize the quadrotor’s approach path by the quadrotor’s (fixed) initial position  $\mathbf{p}_{\text{quad},0}$ , landing position  $\mathbf{p}_{\text{land}}$ , and an approach radius  $r_{\text{app}}$  and angle  $\theta_{\text{app}}$ . The latter two parameters define the approach waypoint  $\mathbf{p}_{\text{app}}$  relative to the landing position. The approach path is constructed as a Bézier curve with control point sequence  $(\mathbf{p}_{\text{quad},0}, \mathbf{p}_{\text{app}}, \mathbf{p}_{\text{land}})$ . We define the task objective as  $\hat{L}_{\text{land}} = \sum_{k=1}^K \|\mathbf{p}_{k,T} - \mathbf{p}_{k,0}\|_2 + \sum_{k=1}^K 2 \arccos(|\mathbf{q}_{k,T} \cdot \mathbf{q}_{k,0}|) + \|\mathbf{p}_{\text{quad},T} - \mathbf{p}_{\text{land}}\|_2$ , where  $\mathbf{p}$

indicates a position,  $\mathbf{q}$  is an orientation represented as a quaternion,  $k$  indexes an object, and  $T$  is the simulated horizon. The first two terms penalize any position and rotation changes of the landing structure, while the last encourages landing position stability.

Genesis [58] is used as the virtual physics environment and scale a provided quadrotor model to match reference real-world dimensions and mass. During simulation, the quadrotor travels to the landing position following the approach Bézier curve and cuts off the motors upon arrival. Propeller wash is modeled using smoothed-particle hydrodynamics (SPH) [63], with each propeller represented as a discrete fluid emitter with parameters estimated via a simple momentum conservation model. Running Genesis on an Nvidia RTX 4090 GPU, each Nelder-Mead optimization averages 8.2 seconds per iteration for a maximum of 30 iterations.

We evaluate SMS on four hand-designed scenes, shown in the appendix. For each scene, approach paths and landing positions are computed for the quadrotor starting from 10 uniformly spaced positions on a 2-meter radius circle around the landing platform. As a baseline, inspired by [64], we annotate candidate landing positions on an image observed by the quadrotor and prompt GPT-4o to select the best one. Finally, we annotate several candidate paths to the chosen landing site, again querying GPT-4o to select the preferred approach. Across the four scenes, SMS outperforms this baseline with respective landing success rates of 100%, 80%, 90%, and 90%, compared to 50%, 50%, 60%, and 50% for the visual prompting strategy.

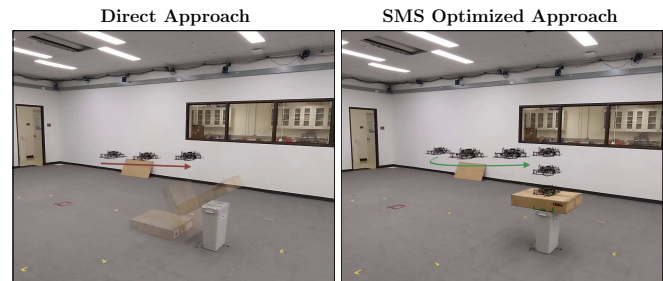


Fig. 4: Quadrotor hardware demonstration. **Left:** A direct approach causes the propeller wash to topple the landing platform. **Right:** SMS optimizes a roundabout trajectory that avoids disturbing the box and enables a successful landing.

These experiments underscore that, while a VLM provides strong semantic and visual understanding of the scene, it alone does not yield a reliable policy for safety-critical tasks like quadrotor landing. In contrast, SMS grounds the planning in physical simulation, enabling the identification of both stable landing sites and comparatively lower-risk approach trajectories as shown in Figure 3. This integration results in consistently higher success rates and robust performance, especially in scenarios where the purely visual reasoning falls short, such as when dealing with complex effects like propeller wash. Real-world quadrotor flight tests (Figure 4) further demonstrate the practical value of physically grounded planning for safe and reliable operation.

## V. CONCLUSION

In this work, we introduced SMS, a unified framework for robot planning that combines the semantic understanding and generalization ability of foundation models with the predictive accuracy of physical simulation. SMS enables object-aware scene modeling and action optimization for complex, dynamic behaviors, as demonstrated in both billiards-inspired manipulation and quadrotor landing tasks. In contrast to prior approaches, which can lack generalization, struggle with novel physical behaviors, or compromise on precision, SMS achieves robust performance in domains where both semantic understanding and physical reasoning are critical. SMS thus represents a step toward more reliable and generalizable robot autonomy. Limitations and future directions, including accelerated scene reconstruction, closed-loop feedback, and more expressive control, are discussed in the appendix.

## REFERENCES

- [1] N. Fazeli, M. Oller, J. Wu, Z. Wu, J. B. Tenenbaum, and A. Rodriguez, "See, feel, act: Hierarchical learning for complex manipulation skills with multisensory fusion," *Science Robotics*, vol. 4, no. 26, p. eaav3123, 2019. [Online]. Available: <https://www.science.org/doi/abs/10.1126/scirobotics.aav3123>
- [2] M. Greenspan, J. Lam, M. Godard, I. Zaidi, S. Jordan, W. Leckie, K. Anderson, and D. Dupuis, "Toward a competitive pool-playing robot," *Computer*, vol. 41, no. 1, pp. 46–53, 2008.
- [3] R. E. Moutaouaffiq, "Billiardbot: Physics-aware planning for robotic billiards," <https://rachad47.github.io/rwae/BilliardBot.html>, 2025.
- [4] D. B. D'Ambrosio, N. Jaitly, V. Sindhwani, K. Oslund, P. Xu, N. Lazic, A. Shankar, T. Ding, J. Abelian, E. Coumans, et al., "Robotic table tennis: A case study into a high speed learning system," in *Robotics: Science and Systems*, 2023.
- [5] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, July 2023. [Online]. Available: <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>
- [6] M. Minderer, A. Gritsenko, and N. Houlsby, "Scaling open-vocabulary object detection," *Advances in Neural Information Processing Systems*, vol. 36, pp. 72983–73007, 2023.
- [7] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024. [Online]. Available: <https://arxiv.org/abs/2408.00714>
- [8] R. S. Sutton, A. G. Barto, et al., "Reinforcement learning," *Journal of Cognitive Neuroscience*, vol. 11, no. 1, pp. 126–134, 1999.
- [9] D. Yarats, I. Kostrikov, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," in *International Conference on Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=GY6-6sTvGaf>
- [10] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3m: A universal visual representation for robot manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 892–909.
- [11] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.
- [12] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799.
- [13] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on robot learning*. PMLR, 2022, pp. 158–168.
- [14] 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*, 2024.
- [15] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa, "dm\_control: Software and tasks for continuous control," *Software Impacts*, vol. 6, p. 100022, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2665963820300099>
- [16] M. Denil, P. Agrawal, T. D. Kulkarni, T. Erez, P. Battaglia, and N. de Freitas, "Learning to perform physics experiments via deep reinforcement learning," in *International Conference on Learning Representations*, 2017. [Online]. Available: <https://openreview.net/forum?id=r1nTpv9eg>
- [17] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.
- [18] R. Sinha, A. Sharma, S. Banerjee, T. Lew, R. Luo, S. M. Richards, Y. Sun, E. Schmerling, and M. Pavone, "A system-level view on out-of-distribution data in robotics," *arXiv preprint arXiv:2212.14020*, 2022.
- [19] 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.
- [20] O. X.-E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Mad-dukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irgan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiqullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitran, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mart' in-Mart' in, R. Bajjal, R. Scalis, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin, "Open X-Embodiment: Robotic learning datasets and RT-X models," <https://arxiv.org/abs/2310.08864>, 2023.
- [21] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, et al., "Openvla: An open-source vision-language-action model," in *8th Annual Conference on Robot Learning*, 2024.
- [22] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X.

- Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, “ $\pi_0$ : A vision-language-action flow model for general robot control,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.24164>
- [23] J. Wen, Y. Zhu, Z. Tang, J. Li, Y. Peng, C. Shen, and F. Feng, “Dexvla: Vision-language model with plug-in diffusion expert for visuomotor policy learning,” *arXiv preprint arXiv:2502.05855*, 2025.
- [24] H. Huang, F. Liu, L. Fu, T. Wu, M. Mukadam, J. Malik, K. Goldberg, and P. Abbeel, “Otter: A vision-language-action model with text-aware feature extraciton,” *arXiv preprint arXiv:2503.03734*, 2025.
- [25] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He, *et al.*, “A comprehensive survey on pretrained foundation models: A history from bert to chatgpt,” *International Journal of Machine Learning and Cybernetics*, pp. 1–65, 2024.
- [26] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “Nerf: Representing scenes as neural radiance fields for view synthesis,” in *ECCV*, 2020.
- [27] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, “Plenoxels: Radiance fields without neural networks,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5501–5510.
- [28] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, July 2022. [Online]. Available: <https://doi.org/10.1145/3528223.3530127>
- [29] T. Chen, P. Culbertson, and M. Schwager, “Catnips: Collision avoidance through neural implicit probabilistic scenes,” *IEEE Transactions on Robotics*, 2024.
- [30] T. Chen, O. Shorinwa, J. Bruno, A. Swann, J. Yu, W. Zeng, K. Nagami, P. Dames, and M. Schwager, “Splat-nav: Safe real-time robot navigation in gaussian splatting maps,” *arXiv preprint arXiv:2403.02751*, 2024.
- [31] X. Lei, M. Wang, W. Zhou, and H. Li, “Gaussnav: Gaussian splatting for visual navigation,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 01, pp. 1–14, 2025.
- [32] J. Abou-Chakra, K. Rana, F. Dayoub, and N. Suenderhauf, “Physically embodied gaussian splatting: A realtime correctable world model for robotics,” in *8th Annual Conference on Robot Learning*, 2024. [Online]. Available: <https://openreview.net/forum?id=AEq0onGrN2>
- [33] M. Zhang, K. Zhang, and Y. Li, “Dynamic 3d gaussian tracking for graph-based neural dynamics modeling,” in *8th Annual Conference on Robot Learning*.
- [34] K. M. Jatavallabhula, M. Macklin, F. Golemo, V. Voleti, L. Petrini, M. Weiss, B. Considine, J. Parent-Levesque, K. Xie, K. Erleben, L. Paull, F. Shkurti, D. Nowrouzezahrai, and S. Fidler, “gradsim: Differentiable simulation for system identification and visuomotor control,” *International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: <https://openreview.net/forum?id=c.E8kFWfhp0>
- [35] R. Liu, A. Canberk, S. Song, and C. Vondrick, “Differentiable robot rendering,” in *8th Annual Conference on Robot Learning*.
- [36] J. Low, M. Adang, J. Yu, K. Nagami, and M. Schwager, “Sous vide: Cooking visual drone navigation policies in a gaussian splatting vacuum,” *arXiv preprint arXiv:2412.16346*, 2024.
- [37] 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.
- [38] 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,” *arXiv preprint arXiv:2409.10161*, 2024.
- [39] A. Quach, M. Chahine, A. Amini, R. Hasani, and D. Rus, “Gaussian splatting to real world flight navigation transfer with liquid networks,” in *8th Annual Conference on Robot Learning*.
- [40] 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.
- [41] L. Barcellona, A. Zadaianchuk, D. Allegro, S. Papa, S. Ghidoni, and E. Gavves, “Dream to manipulate: Compositional world models empowering robot imitation learning with imagination,” in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: <https://openreview.net/forum?id=3RSLW9YSgk>
- [42] L. Meyer, F. Erich, Y. Yoshiyasu, M. Stamminger, N. Ando, and Y. Domae, “Pegasus: Physically enhanced gaussian splatting simulation system for 6dof object pose dataset generation,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 10710–10715.
- [43] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik, “Lerf: Language embedded radiance fields,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19729–19739.
- [44] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21676–21685.
- [45] M. Qin, W. Li, J. Zhou, H. Wang, and H. Pfister, “Langsplat: 3d language gaussian splatting,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20051–20060.
- [46] W. Shen, G. Yang, A. Yu, J. Wong, L. P. Kaelbling, and P. Isola, “Distilled feature fields enable few-shot language-guided manipulation,” in *Conference on Robot Learning*. PMLR, 2023, pp. 405–424.
- [47] G. Lu, S. Zhang, Z. Wang, C. Liu, J. Lu, and Y. Tang, “Manigaussian: Dynamic gaussian splatting for multi-task robotic manipulation,” in *European Conference on Computer Vision*. Springer, 2024, pp. 349–366.
- [48] T. Xie, Z. Zong, Y. Qiu, X. Li, Y. Feng, Y. Yang, and C. Jiang, “Physgaussian: Physics-integrated 3d gaussians for generative dynamics,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 4389–4398.
- [49] R.-Z. Qiu, G. Yang, W. Zeng, and X. Wang, “Feature splatting: Language-driven physics-based scene synthesis and editing,” in *European Conference on Computer Vision (ECCV)*, 2024.
- [50] H. Zhao, H. Wang, X. Zhao, H. Fei, H. Wang, C. Long, and H. Zou, “Efficient physics simulation for 3d scenes via mllm-guided gaussian splatting,” 2025. [Online]. Available: <https://arxiv.org/abs/2411.12789>
- [51] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle, “The material point method for simulating continuum materials,” in *Acm siggraph 2016 courses*, 2016, pp. 1–52.
- [52] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [53] V. Yugay, Y. Li, T. Gevers, and M. R. Oswald, “Gaussian-slam: Photo-realistic dense slam with gaussian splatting,” *arXiv preprint arXiv:2312.10070*, 2023.
- [54] M. Turkulainen, X. Ren, I. Melekhov, O. Seiskari, E. Rahtu, and J. Kannala, “Dn-splatter: Depth and normal priors for gaussian splatting and meshing,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025.
- [55] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3d: A modern library for 3d data processing.”
- [56] 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.
- [57] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” <http://pybullet.org>, 2016–2023.
- [58] G. Authors, “Genesis: A universal and generative physics engine for robotics and beyond,” December 2024. [Online]. Available: <https://github.com/Genesis-Embodied-AI/Genesis>
- [59] B. Delaunay, “Sur la sphère vide. a la mémoire de georges voronoï,” *Bulletin de l’Académie des Sciences de l’URSS. Classe des sciences mathématiques et naturelles.*, no. 6, pp. 793–800, 1934.
- [60] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, “The quickhull algorithm for convex hulls,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 4, pp. 469–483, 1996.
- [61] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [62] J. A. Nelder and R. Mead, “A simplex method for function minimization,” *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.
- [63] D. Koschier, J. Bender, B. Solenthaler, and M. Teschner, “Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids,” *arXiv preprint arXiv:2009.06944*, 2020.
- [64] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, *et al.*, “Pivot: Iterative visual prompting elicits actionable knowledge for vlms,” in *International Conference on Machine Learning*. PMLR, 2024, pp. 37321–37341.