VR-GS: A Physical Dynamics-Aware Interactive Gaussian Splatting System in Virtual Reality

Ying Jiang*
UCLA
USA
HKU
Hong Kong
anajymua@gmail.com

Chang Yu*
UCLA
USA
g1n0st@live.com

Tianyi Xie*
UCLA
USA
tianyixie77@g.ucla.edu

Xuan Li*
UCLA
USA
xuanli1@g.ucla.edu

Yutao Feng
University of Utah
USA
Zhejiang University
China
fytal0n@gmail.com

Huamin Wang
Style3D Research
China
wanghmin@gmail.com

Minchen Li
CMU
USA
minchernl@gmail.com

Henry Lau
HKU
Hong Kong
hyklau@hku.hk

Feng Gao†
Amazon
USA
fenggo@amazon.com

Yin Yang
University of Utah
USA
yangzzzy@gmail.com

Chenfanfu Jiang
UCLA
USA
cffjiang@ucla.edu

Figure 1: Animal Crossing. Utilizing our system, individuals can engage in intuitive and interactive physics-based game-play with deformable virtual animals and realistic environments represented with 3D Gaussian Splatting.

ABSTRACT

As 3D content becomes increasingly prevalent, there's a growing focus on the development of engagements with 3D virtual content. Unfortunately, traditional techniques for creating, editing, and interacting with this content are fraught with difficulties. They tend to be not only engineering-intensive but also require extensive expertise, which adds to the frustration and inefficiency in virtual object manipulation. Our proposed VR-GS system represents a leap forward in human-centered 3D content interaction, offering a seamless and intuitive user experience. By developing a physical dynamics-aware interactive Gaussian Splatting (GS) in a Virtual Reality (VR) setting, and constructing a highly efficient two-level embedding strategy alongside deformable body simulations, VR-GS ensures real-time execution with highly realistic dynamic responses. The components of our system are designed for high efficiency and effectiveness, starting from detailed scene reconstruction and object segmentation, advancing through multi-view image in-painting, and extending to interactive physics-based editing. The system also incorporates real-time deformation embedding and dynamic shadow casting, ensuring a comprehensive and engaging virtual experience.

CCS CONCEPTS

• **Computing methodologies** → **Virtual reality**.

KEYWORDS

Gaussian Splatting, Neural Radiance Fields, Real-Time Interactions

* indicates equal contributions.
† This work is not related to F. Gao's position at Amazon.

SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07
https://doi.org/10.1145/3641519.3657448

ACM Reference Format:
Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, and Chenfanfu Jiang. 2024. VR-GS: A Physical Dynamics-Aware Interactive Gaussian Splatting System in Virtual Reality. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3641519.3657448

1 INTRODUCTION

As digital technology advances, the importance of 3D content is becoming increasingly prominent across various industries, from entertainment to education. This growth is driving the demand for high-fidelity 3D content within the Computer Graphics (CG) and Computer Vision (CV) communities. Despite their capabilities of being rendered efficiently, traditional 3D/4D content creation, reliant on 3D modeling tools and game engines, is time-intensive and complex, often beyond the reach of non-expert users. This accessibility barrier limits broader user engagement in high-quality content creation.

Recognizing the advantages and limitations of the traditional graphics pipeline, our goal is to find a modern variant. To enhance the visual quality and ease of creation for non-expert users, we move away from traditional 3D models in graphics pipelines and instead, adopt state-of-the-art radiance field techniques for rendering. In this context, Neural Radiance Fields (NeRF) emerge as a natural choice. Despite its versatility, NeRF's volume rendering falls short in efficiency for interactive applications, which demand high frame rates. Additionally, NeRF's approach to handling deformations—requiring bending of query rays via an inverse deformation map—is slow [Pumarola et al. 2021]. Fortunately, 3D Gaussian Splatting (GS) [Kerbl et al. 2023] has been recently introduced as an efficient and explicit alternative to NeRF. This method not only excels in rendering efficiency but also provides an explicit geometric representation that can be directly deformed or edited. The explicit nature of 3D GS can simplify the direct manipulation of geometry by solving a Partial Differentiable Equation (PDE) that governs the motions of Gaussian kernels. Furthermore, GS eliminates the need for high-fidelity meshes, UV maps, and textures, offering the potential for naturally photorealistic appearances. It is worth noting that many studies have already demonstrated the use of 3D GS for 4D dynamics [Park et al. 2021; Pumarola et al. 2021; Yang et al. 2023] and the integration of physics-based simulations for more realistic 4D content generation [Feng et al. 2024b; Xie et al. 2024], as well as for creating animatable avatars [Xu et al. 2023b; Zielonka et al. 2023].

In this paper, we introduce a physics-aware interactive system for immersive manipulation of 3D content represented with GS. To ensure an interactive experience, we utilize eXtended Position-based Dynamics (XPBD) [Macklin et al. 2016], a highly adaptable and unified physical simulator, for real-time deformation simulation. Direct incorporation of a simulator onto GS kernels presents challenges, as the simulation and rendering processes have distinct geometrical representations. To address this, we construct a tetrahedral cage for each segmented GS kernel group and embed these kernel groups into corresponding meshes. The deformed mesh, driven by XPBD, subsequently guides the deformation of the GS kernels. Noticing that simplistic embedding techniques can lead to undesirable, spiky deformations in GS kernels, we propose a novel two-level embedding approach. This method allows each Gaussian kernel to adapt to a smoothed average deformation of the surrounding tetrahedra. The intricate combination of GS and XPBD through our two-level embedding not only achieves real-time physics-based dynamics but also upholds high-quality, realistic rendering. In summary, our contributions include:

- *A Physics-Aware Interactive System:* Development of a system that enables interactive, physics-aware manipulation of 3D content represented with GS.
- *Two-Level Deformation Embedding:* Introduction of a novel two-level embedding approach that allows Gaussians to adapt smoothly to the mesh, enhancing the deformation realism and preventing undesirable spiky artifacts.

We deploy our system on VR devices, enriched by segmentation, inpainting, and shadow map, offering users a rich platform for 3D content manipulation.

2 RELATED WORK

2.1 Radiance Fields Rendering

A variety of 3D representations, such as mesh [Sorkine and Cohen-Or 2004], point clouds [Qi et al. 2017], signed distance fields [Wang et al. 2021], and grids [Zhu and Bridson 2005] are exploited in early work for visual computing tasks, including synthesis, estimation, manipulation, animation, reconstruction, and transmission of data about objects and scenes [Gao et al. 2023; Kerbl et al. 2023; Xie et al. 2022]. Since neural radiance fields (NeRF) [Mildenhall et al. 2021] was proposed for novel view synthesis with differentiable volume rendering, it has been applied to versatile computer graphics and computer vision tasks, such as scene reconstruction [Meng et al. 2023; Oechsle et al. 2021; Yariv et al. 2021], synthesis [Huang et al. 2023a; Sun et al. 2023], rendering [Lombardi et al. 2021; Wu et al. 2023], interactive games [Xia et al. 2024] and animation [Chen et al. 2021; Peng et al. 2021], simulation [Li et al. 2023a], etc. While Neural Radiance Fields (NeRF) have achieved remarkable image quality, they suffer from significant time and memory inefficiencies [Lindell et al. 2021]. To mitigate these issues, various methods have been introduced. Sparse representations [Liu et al. 2020], decomposed strategies [Niemeyer and Geiger 2021; Rebain et al. 2021], and multi-resolution encoding [Müller et al. 2022] have all been proposed to enhance volume rendering efficiency while maintaining high quality. However, NeRFs are implicit representations, making it challenging to detect and resolve collisions [Qiao et al. 2023]. Recently, 3D Gaussian splatting (GS) proposed by [Kerbl et al. 2023] utilizes an array of 3D Gaussian kernels to represent scenes explicitly, facilitating faster optimization and achieving state-of-the-art results.

2.2 Editing in Radiance-based Scene

A natural extension of NeRF is to support user-guided editing. Li and Pan [2023] made use of two proxy cages to offer interactive control of the shape deformation of NeRF. Besides geometry editing, text prompts [Wang et al. 2023] or a user-edited image [Bao et al. 2023] can also be adopted to conduct style transfer of NeRF. Besides,

Lin et al. [2023] put forward exploiting 2D sketches to edit high-quality facial NeRF. Concerning NeRF texture editing, Huang et al. [2023a] utilized a coarse-fine disentanglement representation and a patch-matching algorithm to synthesize textures of different shapes from multi-view images. De-Nerf [Wu et al. 2023] exploited a hybrid light representation that enables users to relight NeRF. In contrast, Gaussian Splatting is more appropriate for post-editing tasks thanks to its explicit nature and has inspired a series of follow-up works [Chen et al. 2023; Duisterhof et al. 2023; Fang et al. 2023; Huang et al. 2023b; Ye et al. 2023].

Another significant direction in NeRF research is the incorporation of dynamic components into static scenes. To achieve this, a popular strategy [Park et al. 2021; Pumarola et al. 2021] is to consider time as an additional input to the system, This method typically splits a dynamic NeRF into a canonical static field and a deformation field, with the latter mapping this canonical representation to a deformed one. More recently, Yang et al. [2023] drew inspiration from 3D GS to reconstruct a dynamic scene using 4D Gaussian primitives that change over time. However, the dynamics in these methods are generally confined to motions captured from input data, limiting their ability to synthesize unseen dynamics. To generate novel dynamics, Xie et al. [2024] and Feng et al. [2024b] have integrated physics-based simulations into 3D static representations, paving the way for generating physically plausible and novel dynamic scenes.

2.3 Real-time Neural Radiance Fields

Rendering NeRF is computationally expensive for real-time applications, such as VR, which causes high latency and low quality [Li et al. 2022b; Song et al. 2023]. To address these challenges and enable high-fidelity rendering with minimal latency on a single GPU, various techniques have been proposed. These include gaze-contingent 3D neural representations [Deng et al. 2022], variable rate shading [Rolff et al. 2023], and hybrid surface-volume representations [Turki et al. 2023], all aimed at accelerating NeRF. In contrast to single-GPU solutions, VR-NeRF [Xu et al. 2023a] leveraged multiple GPUs to achieve high-quality volumetric rendering of NeRF. Beyond software acceleration techniques, RT-NeRF [Li et al. 2022a] utilized an algorithm-hardware co-design framework to provide real-time NeRF solutions for immersive VR rendering. While these methods primarily focus on enhancing NeRF rendering, our work is dedicated to facilitating interactive editing.

Li et al. [2023c] integrated affine transformation with NeRF to enable interactive features, such as exocentric manipulation, editing, and VR tunneling effects. However, their approach is limited to filter-based edits such as edge blurring and fails to support the deformation of virtual objects' geometry. RealityGit [Li et al. 2023b] and Magic NeRF Lens [Li et al. 2023d] modified a NeRF model by erasing or revealing a portion of NeRF. Nevertheless, neither system permits deforming the geometrical structure of the model. In contrast to these works which concentrate on interactive editing of NeRF, our system enables users to interact with 3D GS in a physically realistic way in real time.

3 SYSTEM DESIGN

We propose a unified physics dynamics-aware interactive VR system for real-time interactions with Gaussian Splatting (GS).

Immersive and Realistic Generative Dynamics. Our targeted system aims to offer users an immersive and realistic experience, characterized by dynamic motions, 3D virtual shapes, and illuminations that closely mirror the real world [Kalawsky 1999]. This goal sets our system apart, especially in how it handles physics-based 3D deformations and dynamics, as opposed to those that rely solely on geometric deformation. The proposed system leverages physics-based simulation techniques to produce realistic dynamics and facilitate 3D shape editing. Unlike methods that reconstruct motion from time-dependent datasets or utilize generative AI to drive Gaussian models, our system adheres to physical principles. To create an immersive experience, we choose VR as the interaction platform.

Real-Time Interaction. For an interactive system, low latency is crucial to prevent disorientation or motion sickness [Deng et al. 2022; Sutcliffe et al. 2019]. To offer instantaneous output and feedback, most interactive systems support basic transformations including rotation, scaling, and translation. Our system additionally offers users real-time physics-based interaction. To our knowledge, VR-GS represents the first interactive physics-based GS editor. To achieve both real-time performance and physically realistic editing, we implement a reduced representation model and parallel-friendly algorithms within our frame budget. While the per-Gaussian-based discretization as done in PhysGaussian [Xie et al. 2024] offers detailed dynamics, its time cost is impractical for our system's needs. Instead, we utilize a tetrahedral mesh embedding reconstructed from Gaussian kernels to drive GS motion. We adopt XPBD with finite-element constraints to achieve real-time simulations.

Unified Framework. VR-GS is a framework that integrates rendering and simulation within a unified pipeline. Our design principles adhere to the goal of "what you see is what you simulate" [Müller et al. 2016]. Unlike Xie et al. [2024], our system employs a hybrid approach, combining cage mesh and Gaussian kernels, to achieve real-time performance. The deformation gradient field of Gaussian kernels is embedded in the cage mesh through piecewise constant interpolation. With each Gaussian kernel resembling a rotated ellipsoid, this simple embedding strategy leads to spiky artifacts during large deformation, a challenge highlighted in [Xie et al. 2024]. To mitigate this, we employ a two-level interpolation scheme: initially embedding each Gaussian within an individual bounding tetrahedron, followed by embedding these tetrahedra within the simulated cage mesh to elevate the smoothness of the deformation field.

Additional Components. Our framework is designed to construct VR-compatible interactive settings derived from real-world captures. Consequently, it introduces complexities associated with facilitating intuitive user interactions and delivering real-time feedback that meets user expectations and common sense perceptions [Gabbard et al. 1999; Hale and Stanney 2014; Stanney et al. 2003]. In particular, when users engage with objects within a scene, they anticipate natural movements and realistic appearances from both the selected objects and the surrounding environment. Moreover,

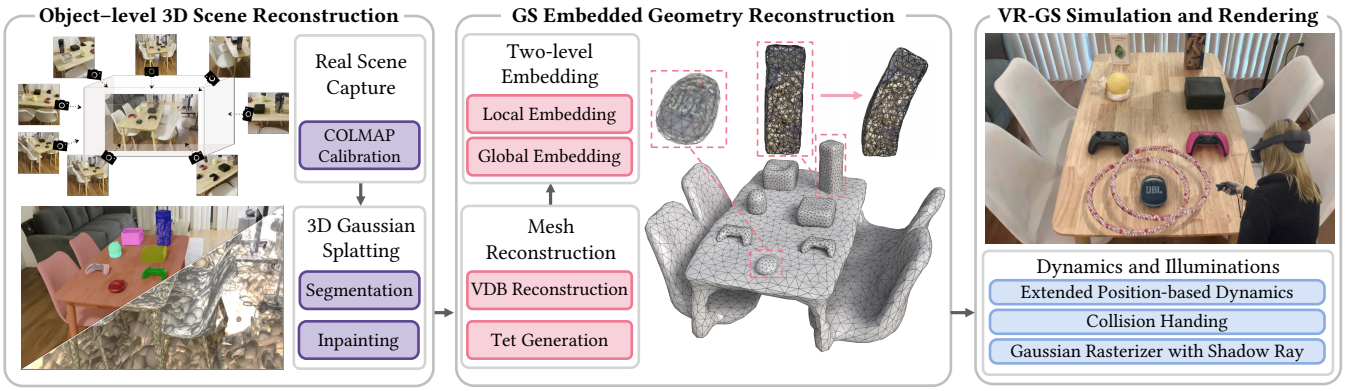

Figure 2: VR-GS is an interactive system designed to integrate 3D Gaussian Splatting (GS) and eXtended Position-based Dynamics (XPBD) for generating a real-time interactive experience. Beginning with multi-view images, the pipeline combines scene reconstruction, segmentation, and inpainting using Gaussian kernels. These kernels form the foundation for VR-GS's utilization of the sparse volumetric data structure VDB, facilitating bounding mesh reconstruction and subsequent tetrahedralization. VR-GS further harnesses a novel two-level Gaussian embedding, XPBD, collision detection, and shadow casting techniques, all converging to deliver a captivating and immersive user experience.

it is necessary to address voids when an object is removed from its supporting base. To manage these challenges, our system additionally integrates functionalities for object segmentation and scene inpainting.

4 METHOD

4.1 Gaussian Splatting

Gaussian splatting, proposed by Kerbl et al. [2023], is an explicit 3D representation to encapsulate 3D scene information using a set of 3D anisotropic Gaussian kernels, each with learnable mean μ, opacity σ, covariance matrix Σ, and spherical harmonic coefficients C. Spherical harmonics (SH) are expansions of the view-dependent color function defined on the unit sphere. To render a view, the 3D splats are projected to 2D screen space ordered by their z-depth. The color C of a pixel is computed by α-blending of these 2D Gaussians from near to far:

$$C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{1}$$

where c_i is the evaluated color by SHs viewed from the camera to the kernel's mean. α_i is the product of the kernel's opacity and 2D Gaussian weight evaluated at the pixel coordinate. Leveraging a differentiable implementation, the rendering loss towards the ground truth image can be backpropagated to Gaussians' parameters for optimization.

In contrast to traditional NeRFs based on implicit scene representations, GS provides an explicit representation that can be seamlessly integrated with post-processing manipulations, such as animating and editing. Moreover, the efficient rasterization and superior rendering quality of 3D Gaussians facilitate their integration with VR.

4.2 VR-GS Assets Preparation

Each 3D asset in our VR system consists of a high-fidelity 3D GS reconstruction and an enveloping simulatable tetrahedral mesh in a moderate resolution to enable real-time physics-aware dynamics. These preparations are conducted offline before immersive and interactive editings within our VR environment, which includes:

- *Segmented GS Generation*: we support interactions with individual objects in a large scene, achieved by segmented GS reconstructions.
- *Inpainting*: The occluded parts between the objects and their supporting planes usually have no texture. We inpaint 3D GS representations leveraging a 2D inpainting technique.
- *Mesh Generation*: A simulation-ready tetrahedral mesh is generated for each object.

4.2.1 Segmentation. The segmented GS is constructed during GS training. We first generate 2D masks on the multi-view RGB images by utilizing a 2D segmentation model [Cheng et al. 2023]. Each segmented part is assigned a different color that is consistent across different views. Subsequently, we enhance the scene representation by integrating three additional learnable RGB attributes into the 3D Gaussian kernels. During the reconstruction process, each 3D Gaussian kernel will automatically learn what object it belongs to utilizing a segmentation loss function L_{seg}:

$$L_{\text{seg}} = L_1(M_{2d}, I), \tag{2}$$

where M_{2d} represents colored 2D segmentation results and I denotes the rendering of 3D Gaussian kernels with colors replaced by the extended RGB attributes instead of evaluated from SHs. Thus, the total loss becomes

$$L_{\text{total}} = (1 - \lambda)L_1 + \lambda L_{\text{SSIM}} + \lambda_{\text{seg}} L_{\text{seg}}, \tag{3}$$

where L_1 and L_{SSIM} are computed between the normally rendered images and the multi-view ground truths. We use $\lambda = 0.2$ and $\lambda_{\text{seg}} = 0.1$ in all our experiments.

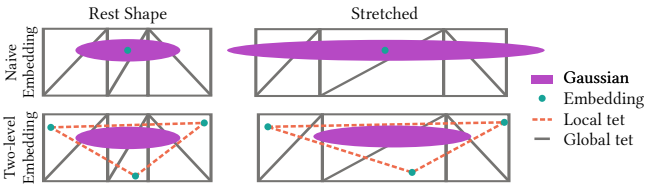

Figure 3: Our two-level embedding effectively resolves spiky artifacts. Each Gaussian kernel is embedded into a local tetrahedron. The vertices of the local tetrahedron are independently embedded into the global mesh.

4.2.2 Inpainting. After 3D GS segmentation, we extract all objects separately from the scene. This process of object removal, however, results in the emergence of holes within the regions that are previously occluded. To alleviate the issue, we utilize a 2D inpainting tool LaMa [Suvorov et al. 2022] to guide the 3D inpainting of Gaussian kernels. We freeze the Gaussian kernels located outside of the holes and then use an inpainting loss $L_{\text{inpaint}} = L_1(I_{\text{inpainted}}, I)$ to optimize a Gaussian kernel patch under the guidance of the 2D inpainted images $I_{\text{inpainted}}$ for the current 3D GS rendering I. The result of our inpainting is validated in Section 5.

4.2.3 Mesh Generation. Due to our design choice to use mesh-based simulation, we generate a tetrahedral mesh for each group of the segmented 3D GS kernels. These meshes will not be rendered during the interaction but only serve as the media of dynamics. Hence we can use a moderate mesh resolution which does not hinder performance. To construct a simulation mesh, we first use internal filling proposed by [Xie et al. 2024] to fill particles into the void internal region that is not reconstructed by GS. Then we treat centers of Gaussians as a point cloud and convert it to a voxelized VDB representation [Museth 2013]. A water-tight surface mesh is then extracted using marching cubes [Lorensen and Cline 1987] and tetrahedralized into a finite element mesh using TetGen [Si 2015].

4.3 Unified Framework for Simulation and Rendering

As shown in Xie et al. [2024], Gaussian kernels can be deformed by the simulated deformation field. We follow the Gaussian kinematics of this work but replace the simulator with XPBD [Macklin et al. 2016] to achieve real-time interactions. We employ the strain energy constraint as the elastic model and adopt the velocity-based damping model in the XPBD framework.

4.3.1 Physical Parameters. The physical parameters, such as densities and material stiffness, are tunable hyperparameters. Following PhysGaussian [Xie et al. 2024], we manually set physical parameters for all interactable objects, including Young's modules (E), Poisson ratio (ν), and density (ρ). We start from an initial configuration $E = 1000$ Pa, $\nu = 0.3$, $\rho = 1000$ kg for each object, then tune them to produce visually plausible dynamics. The Young's modulus governs the overall stiffness of an object and is manually adjusted by analyzing the severity of deformation of its free fall collision with the ground. Additionally, we adjust the relative density between two objects based on their deformations under collision. In practice we usually arrive at a finalized parameter setting that gives visually plausible results within 10 iterations of this manual process.

4.3.2 Embedding. In our mesh-based simulation, the deformation map is piece-wise linear, with the resulting deformation gradient piece-wise constant within each tetrahedron. Given a tetrahedron with the rest-shape configuration $\{\boldsymbol{x}_0^0, \boldsymbol{x}_1^0, \boldsymbol{x}_2^0, \boldsymbol{x}_3^0\}$ and the current configuration $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$, the deformation gradient is defined as

$$\boldsymbol{F} = \left[\boldsymbol{x}_1 - \boldsymbol{x}_0, \boldsymbol{x}_2 - \boldsymbol{x}_0, \boldsymbol{x}_3 - \boldsymbol{x}_0\right] \left[\boldsymbol{x}_1^0 - \boldsymbol{x}_0^0, \boldsymbol{x}_2^0 - \boldsymbol{x}_0^0, \boldsymbol{x}_3^0 - \boldsymbol{x}_0^0\right]^{-1}, \tag{4}$$

where the inverse of the rest-shape basis can be computed pre-simulation. The mean and the covariance matrix of the deformed Gaussian kernel inside this tetrahedron is given by

$$\boldsymbol{\mu} = \sum_i w_i \boldsymbol{x}_i, \quad \Sigma = \boldsymbol{F}\Sigma_0 \boldsymbol{F}^T, \tag{5}$$

where w_i is the barycentric coordinates of the initial center $\boldsymbol{\mu}_0$ in the rest-shape configuration, and Σ_0 is the initial covariance matrix [Xie et al. 2024]. The deformed Gaussian kernels can be directly rendered by the point splatting procedure. However, the direct embedding of Gaussian centers cannot guarantee that every ellipse shape is completely enveloped by some tetrahedron inside the simulation mesh. As shown in Figure 3, this can lead to spiky artifacts. Observe that kernels that are completely inside the tetrahedron will always be inside. This motivates us to propose a two-level embedding procedure:

(1) Local embedding: we independently envelope every Gaussian kernel by an as-tight-as-possible tetrahedron. There is no connectivity between these local tetrahedra.
(2) Global embedding: we embed the vertices of local tetrahedra into the global simulation mesh.

As the global mesh is deformed by the simulation, the vertices of local tetrahedra are kept inside the boundary, driving the kinetic evolution of Gaussian kernels. A local tetrahedron could overlap with multiple global tetrahedra. The deformation map on it can be understood as the average of the surrounding global tetrahedra, hence eliminating sharp, spiky artifacts, as validated in Section 5.

4.3.3 Shadow Map. While the original global illumination of the scene can be accurately learned and baked by the spherical harmonics on each Gaussian, the shadow will no longer be aligned with the object when it is moving or deforming. Bringing shadow map [Wanger et al. 1992] into the GS framework can enhance the immersive experience in VR. More importantly, it can guide human perception of the spatial relationships between objects: during manipulation, users will rely on the shadow to determine the distance between objects. The shadow map is a fast real-time algorithm and is well-aligned with the GS rasterization pipeline. We follow Equation (1) to estimate the depth map from the light source and test the visibility for each Gaussian using this depth map. The influence of shadow maps is studied in Section 5.

5 EVALUATION

In this section, we assess the essential components of our proposed system, including two-level GS embedding and shadow map.

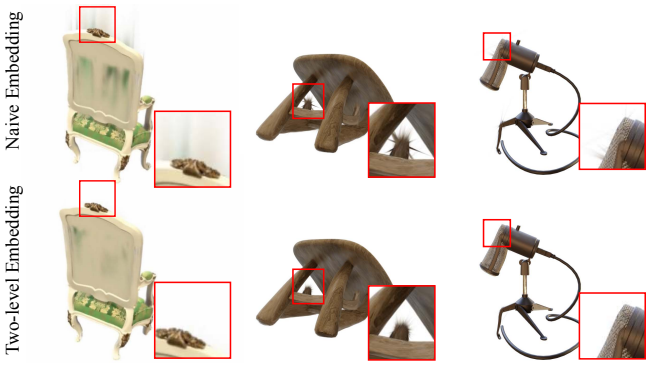

Figure 4: Two-level Embedding Evaluation. Our two-level embedding alleviates the commonly seen spiky artifacts for deformed GS kernels.

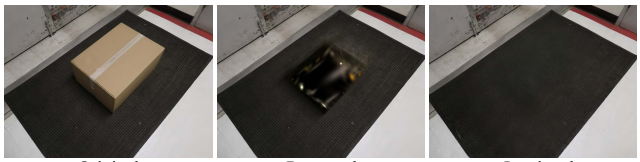

Figure 5: Inpainting Evaluation. GS struggles to reconstruct occluded surfaces. By leveraging LAMA [Suvorov et al. 2022], we produce 2D inpainted results that guide the 3D scene inpainting, enhancing the realism of the 3D representation.

Two-level Embedding. The two-level embedding is a crucial component in our physical dynamics-aware system, integrating the tetrahedra cage mesh with the embedded Gaussian. In Figure 4, we conduct an ablation study to validate the effectiveness of our two-level embedding approach. Under conditions of extreme stretching or twisting, the naive embedding method, which simply embeds the Gaussian kernel within the closest tetrahedron, leads to severe spiky artifacts. Our two-level embedding strategy addresses this by initially embedding each Gaussian into a localized, independent tetrahedron, followed by embedding the vertices of this tetrahedron into the cage mesh. The deformation of each Gaussian kernel is determined by averaging the deformations at these vertices, resulting in a smoother deformation gradient field than the naive approach and significantly reducing spiky artifacts.

Inpainting. With the guidance of 2D image segmentation results, our system achieves object-level 3D scene reconstruction, facilitating convenient post-physics-based manipulation. However, GS is limited to reconstructing surfaces visible in the provided multi-view training images. Consequently, some object and background areas, unseen in these images, remain unreconstructed by GS, leading to "black hole" like artifacts when foreground objects are moved. To address this, our system integrates the object segmentation mask with LAMA [Suvorov et al. 2022], a 2D inpainting model, to generate inpainted multi-view images. These images then guide the fine-tuning and inpainting of our 3D GS scene, yielding a more complete and realistic user experience, as evidenced in Figure 5.

Shadow Map. Furthermore, we take advantage of the shadow map to add extra time-dependent shadows into the GS scene, as shown in 6. Original GS represents shadows as textures and thus fails to provide dynamic shadows when objects move. VR-GS allows users to choose the parameters, e.g. position and direction, of the light source to reproduce the lighting setting of the original scene, leading to a more realistic VR environment.

6 EXPERIMENT

In this section, we benchmark our VR-GS system's simulation performance against other methods in GS and NeRF manipulation. Additionally, we showcase interactive demonstrations on VR devices. Our prototype, developed in Unity with a CUDA-implemented plugin, is tested using a Quest Pro Head-Mounted Display (HMD) and corresponding controllers, on an Intel Core i9-14900KF CPU with 32GB memory and an NVIDIA GeForce RTX 4090 GPU.

6.1 Performance

VR-GS is designed for real-time, physics-aware interactions by integrating Gaussian kernels within a cage mesh for simulation. This cage mesh is derived from the VDB representation of Gaussians, with adjustable mesh resolution to influence quality. We first conduct experiments to explore the trade-offs between mesh quality and system performance. As shown in Figure 12, using a coarse cage mesh facilitates higher frame rates but may lead to the loss of fine details. Conversely, a higher-resolution mesh inevitably increases the computational cost. Moreover, XPBD also requires much more iterations to achieve convergence for finer meshes. Without sufficient iterations, the simulated object may appear to be overly soft, yielding unrealistic dynamics. In practice, we constrain the number of mesh vertices to between 10K and 30K to maintain a balance between high frame rates and accurate physical dynamics.

We then compare VR-GS against two state-of-the-art NeRF/GS physics-based manipulation methods: PAC-NeRF [Li et al. 2023a] and PhysGaussian [Xie et al. 2024]. PAC-NeRF primarily concentrates on estimating material parameters from multi-view videos in reconstructed NeRF scenes. Although it offers novel dynamic generation capabilities, the resulting visuals often fall short in rendering quality. In contrast, PhysGaussian, leveraging GS, produces superior photorealistic dynamics. However, the Material Point Method (MPM) used by both systems can limit real-time performance in complex scenes. For a fair comparison, we standardize the frame time ($\Delta t_{\text{frame}} = 1/25$ sec) and the simulation step time ($\Delta t_{\text{step}} = 0.0001$ sec) across all methods. For VR-GS, we set the XPBD iteration per substep to 1, as the small Δt_{step} sufficiently resolves the dynamics. As depicted in Figure 7 and the inset table, VR-GS not only matches the visual quality of PhysGaussian but also outperforms PAC-NeRF in clarity and realism. Crucially, our XPBD-based simulation framework allows for significantly faster simulations, making VR-GS ideal for real-time physics-aware interactions.

Example	PAC-NeRF	Phys-Gaussian	VR-GS **(Ours)**
Stool	0.750	0.112	0.017
Chair	0.813	0.219	0.022
Materials	0.625	0.39	0.021

Figure 6: Shadow Map Evaluation. GS traditionally models shadows as static surface textures. Our approach, employing a shadow map, generates dynamic shadows for a more immersive experience.

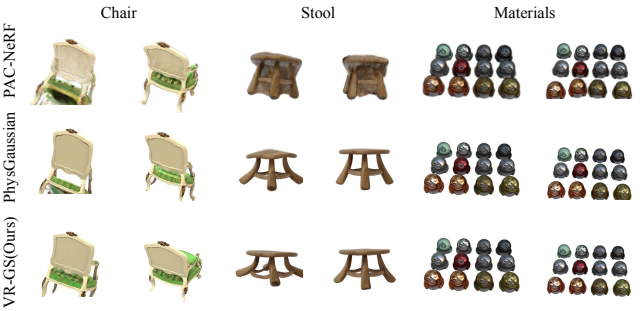

Figure 7: Visual Quality Comparison. Our method synthesizes competitive visual results compared to PhysGuassian [Xie et al. 2024] and significantly outperforms PAC-NeRF [Li et al. 2023a].

6.2 VR Demos

We then showcase VR-GS's capability to replicate real-world scenarios through several representative demos in VR. Detailed simulation setups and a timing breakdown for these demos are provided in Table 1 and Figure 13.

Fox, Bear, and Horse Manipulation. VR-GS empowers users to edit 3D Gaussian kernels intuitively and efficiently, utilizing our XPBD-based physics engine for dynamic manipulation. We demonstrate this through three examples as depicted in Figure 9: a fox, a bear, and a horse, reconstructed from the Instant-NGP dataset [Müller et al. 2022], the Instruct-NeRF2NeRF [Haque et al. 2023], and the Tanks and Temples dataset [Knapitsch et al. 2017], respectively.

Ring Toss and Table Brick Game. In this example, we demonstrate the system's ability to seamlessly integrate new *virtual objects* into existing scenes. We use a room scene reconstructed from real-world footage and virtual objects (rings and bricks modeled in Blender) to create ring tossing and table brick game (Figure 10). The dining table and objects on it serve as collision boundaries for the elastic ring and rigid bricks [Deul et al. 2016].

Table 1: Parameters and Timings of Demos in Section 6.2.

Example	#Gaussians	#Verts.	#Iter*	CD*	FPS
(Figure 9) Fox	304,875	28,205	1	/	161.2
(Figure 9) Bear	2,525,891	19,472	5	/	76.9
(Figure 9) Horse	1,295,223	10,611	5	/	115.4
(Figure 10) Ring Toss	1,456,209	9,002	10	10	37.7
(Figure 10) Table Brick Game	1,866,835	33,279	10	10	41.4
(Figure 11) Toy Collection	1,665,128	46,428	20	20	24.3
(Figure 5) Box Moving	1,769,412	1,434	10	/	73.8
(Figure 1) Animal Crossing	1,312,670	36,386	10	5	34.7
(Figure 8) Just Dance	1,059,054	13,936	50	/	33.9

*#Iter: XPBD iterations per substep; CD: collision detections per step.

Toy Collection and Animal Crossing. We reconstructed a living room scene and a set of plush toys (Figure 11) for users to interact with. The yellow spherical toy is generated using a text-to-3D generator LucidDreamer [Liang et al. 2023], while the others are reconstructed. Additionally, four animal plush toys were placed on chairs, allowing users to manipulate and deform them freely (Figure 1).

Just Dance. VR-GS also supports animation of reconstructed GS humans (Figure 8). Utilizing a reconstructed human character, we leverage Mixamo[1]'s auto-rig to generate motion sequences of its surface mesh. These sequences then serve as the boundary condition for the simulation, finally producing the GS human animation that can be viewed in our immersive system.

6.3 Modeling Details

We employ the *Toy Collection* (Figure 11) as a case study to demonstrate the time and labor required to create a fully interactive VR environment from scratch. The setup process is as follows:

(1) The initial stage involves setting up the real-world scene. For the *Toy Collection*, the indoor scene was captured directly as one scene, with all toys separately hung with strings as another scene. We completed it within 20 minutes.
(2) Next, we capture multi-view images for Gaussian Splatting by recording videos of the two scenes with a consumer-level camera and converting them into image sequences. Camera intrinsic and extrinsic parameters are collected using COLMAP, with the entire process taking around 60 minutes, of which COLMAP occupies approximately 40 minutes.
(3) We then proceed to image processing for segmentation and inpainting. Object classes are manually designated in the first video frame. Subsequent frames are automatically segmented using the method from Cheng et al. [2023], taking about 2 minutes. The inpainting of what's below the basket is fully automated [Suvorov et al. 2022] and requires 5 minutes.
(4) GS reconstruction is performed on the processed images, undergoing 30,000 training steps as specified by Kerbl et al. [2023] without additional treatment for the number of Gaussians, which takes about 30 minutes on a single RTX 4090.
(5) After generating 3D GS, we execute internal filling, VDB reconstruction, and TetGen, all within 10 minutes.

[1]https://www.mixamo.com/

Figure 8: Just Dance. VR-GS generates high-fidelity dance given a reconstructed human body.

(6) Lastly, object world positions and physical parameters are specified and estimated in Unity, as detailed in Section 4.3.1, taking 20 minutes.

In summary, the total preparation time for the *Toy Collection* is approximately 2 hours and 30 minutes. Manual tasks include scene staging, video recording, object class specification, and physical parameter selection. All other processes are automated. Once modeled, the scene is ready for a physics-aware interactive VR experience.

7 USER STUDY

We conducted a user study including two tasks with 10 participants: 2 professionals (P1–P2) with 5 and 7 years of experience in 3D VFX software (including Houdini and Blender), and 8 novices (P3–P10), with P2 also having 2 years of VR experience. Our study used the hardware specified in the experimental section and included the Fox, Bear, House, Ring Toss, and Toy Collection demos. Participants received a 10-minute video tutorial on our system before completing two tasks. After task 2, they answered a questionnaire covering usability (System Usability Scale [Bangor et al. 2009]) and subjective feedback on individual and overall system features, as shown in Section 7.2. Additionally, we conducted a 20-minute semi-structured interview for more in-depth feedback.

7.1 Tasks

Task 1: Goal-directed (30 minutes). The participants were required to play two VR games developed by our system: ring toss and toy collection (Figure 11) in two different settings (S1: physics-based and S2: transform-based interaction). Each game had a 5-minute duration under each configuration. A 5-minute break was allowed between sessions. The arrangement of sessions and tasks used a Latin square design to counter potential learning effects. We requested users to rate the immersive and realistic experience in VR using a 7-point Likert scale.

Task 2: Open-ended (30 minutes). The participants are asked to edit the scene and generate dynamics freely in aforementioned scenes (shown in Figure 9), Figure 10 and Figure 11) with our prototype system, which includes geometry-based editing, physics-based editing, transform, rotation, duplicating, undoing, rescaling, etc. Task 2 is designed to evaluate the detailed potential factors impacting VR immersive experience.

7.2 Results and Discussion

According to a paired t-test conducted on the score collected from task 1, it could be noticed that physics-based interaction significantly enhanced user immersion and realism compared to transform-based interaction (physics-based: 6.1 vs transform-based: 4.8 on average, $p = .0227$). As shown in Figure 14, our system has received overall positive opinions. Users spoke highly of the placement of virtual content. Transforming objects without physics rules resulted in floating objects, while in our system, all virtual content placements, such as putting toys onto a sofa, are followed by physics rules and are consistent with the user's understanding. P7 commented, *"I love this a lot. I have a dog. Petting the fox is really like what I did to my dog at home. I felt so real. Moreover, the placement of the toy is awesome. After they all fell to the basket, the basket even shook for a while."* The VR-GS system's realistic lighting and physics-based dynamics create an immersive experience. As a professional user, P2 spoke highly of high-fidelity generative dynamics and illumination. *"Those motions of virtual content are just like what I see in the physical world. I can't wait to take photos of my own house and then put them in VR with my video game character."* Users rated the system highly on ease of use (4.6/5) and overall satisfaction (4.8/5). With a System Usability Scale (SUS) score of 83.5, VR-GS is classified as "excellent" according to [Bangor et al. 2009].

8 CONCLUSION AND FUTURE WORK

We presented a physical dynamics-aware interactive Gaussian Splatting system for addressing challenges in editing real-time high-fidelity virtual content. By leveraging the advancements in Gaussian Splatting, VR-GS bridges the quality gap traditionally observed between machine-generated and manually created 3D content. Our system not only enhances the realism and immersion via physically-based dynamics, but also provides fine-grained interaction and manipulation controllability.

Although all the study participants appreciated the efficiency and effectiveness of our system, it still remains to be improved. Firstly, rendering high-fidelity Gaussian kernels in VR is computationally demanding. As a result, rendering generative dynamics in a large scene with 2K resolution might lead to potential latency issues in our system. Secondly, the physical parameters in our system are manually defined. Estimating parameters from videos like PAC-NeRF [Li et al. 2023a], or leveraging large-vision models, would be interesting directions to automate this process. In future work, we aim to incorporate a broader range of materials, such as fluid [Feng et al. 2024a] and cloth, to enhance the system's capabilities. Furthermore, it is also interesting to explore how to utilize large multimodal models to assess the fidelity of the generated dynamics.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback. We acknowledge support from NSF (2301040, 2008915, 2244651, 2008564, 2153851, 2023780), UC-MRPI, Sony, Amazon, and TRI.

REFERENCES

Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of usability studies* 4, 3 (2009), 114–123.

Chong Bao, Yinda Zhang, Bangbang Yang, Tianxing Fan, Zesong Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. 2023. Sine: Semantic-driven image-based nerf editing with prior-guided editing field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20919–20929.

Jianchuan Chen, Ying Zhang, Di Kang, Xuefei Zhe, Linchao Bao, Xu Jia, and Huchuan Lu. 2021. Animatable neural radiance fields from monocular rgb videos. *arXiv preprint arXiv:2106.13629* (2021).

Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. 2023. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. *arXiv preprint arXiv:2311.14521* (2023).

Ho Kei Cheng, Seoung Wug Oh, Brian Price, Joon-Young Lee, and Alexander Schwing. 2023. Putting the Object Back into Video Object Segmentation. *arXiv preprint arXiv:2310.12982* (2023).

Nianchen Deng, Zhenyi He, Jiannan Ye, Budmonde Duinkharjav, Praneeth Chakravarthula, Xubo Yang, and Qi Sun. 2022. Fov-nerf: Foveated neural radiance fields for virtual reality. *IEEE Transactions on Visualization and Computer Graphics* 28, 11 (2022), 3854–3864.

Crispin Deul, Patrick Charrier, and Jan Bender. 2016. Position-based rigid-body dynamics. *Computer Animation and Virtual Worlds* 27, 2 (2016), 103–112.

Bardienus P Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Mike Zheng Shou, Shuran Song, and Jeffrey Ichnowski. 2023. MD-Splatting: Learning Metric Deformation from 4D Gaussians in Highly Deformable Scenes. *arXiv preprint arXiv:2312.00583* (2023).

Jiemin Fang, Junjie Wang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. 2023. Gaussianeditor: Editing 3d gaussians delicately with text instructions. *arXiv preprint arXiv:2311.16037* (2023).

Yutao Feng, Xiang Feng, Yintong Shang, Ying Jiang, Chang Yu, Zeshun Zong, Tianjia Shao, Hongzhi Wu, Kun Zhou, Chenfanfu Jiang, et al. 2024a. Gaussian Splashing: Dynamic Fluid Synthesis with Gaussian Splatting. *arXiv preprint arXiv:2401.15318* (2024).

Yutao Feng, Yintong Shang, Xuan Li, Tianjia Shao, Chenfanfu Jiang, and Yin Yang. 2024b. PIE-NeRF: Physics-based Interactive Elastodynamics with NeRF. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Joseph L Gabbard, Deborah Hix, and J Edward Swan. 1999. User-centered design and evaluation of virtual environments. *IEEE computer Graphics and Applications* 19, 6 (1999), 51–59.

Lin Gao, Jia-Mu Sun, Kaichun Mo, Yu-Kun Lai, Leonidas J Guibas, and Jie Yang. 2023. SceneHGN: Hierarchical Graph Networks for 3D Indoor Scene Generation with Fine-Grained Geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

Kelly S Hale and Kay M Stanney. 2014. *Handbook of virtual environments: Design, implementation, and applications*. CRC Press.

Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. 2023. Instruct-nerf2nerf: Editing 3d scenes with instructions. *arXiv preprint arXiv:2303.12789*.

Yi-Hua Huang, Yan-Pei Cao, Yu-Kun Lai, Ying Shan, and Lin Gao. 2023a. NeRF-texture: Texture synthesis with neural radiance fields. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–10.

Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. 2023b. SC-GS: Sparse-Controlled Gaussian Splatting for Editable Dynamic Scenes. *arXiv preprint arXiv:2312.14937* (2023).

Roy S Kalawsky. 1999. VRUSE—a computerised diagnostic tool: for usability evaluation of virtual/synthetic environment systems. *Applied ergonomics* 30, 1 (1999), 11–25.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (2023).

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)* 36, 4, 1–13.

Chaojian Li, Sixu Li, Yang Zhao, Wenbo Zhu, and Yingyan Lin. 2022a. RT-NeRF: Real-Time On-Device Neural Radiance Fields Towards Immersive AR/VR Rendering. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

Ke Li, Tim Rolff, Reinhard Bacher, and Frank Steinicke. 2023b. RealityGit: Cross Reality Version Control of R&D Optical Workbench. In *2023 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE, 807–808.

Ke Li, Tim Rolff, Susanne Schmidt, Reinhard Bacher, Simone Frintrop, Wim Leemans, and Frank Steinicke. 2022b. Immersive Neural Graphics Primitives. *arXiv preprint arXiv:2211.13494* (2022).

Ke Li, Tim Rolff, Susanne Schmidt, Reinhard Bacher, Wim Leemans, and Frank Steinicke. 2023c. Interacting with Neural Radiance Fields in Immersive Virtual Reality. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–4.

Ke Li, Susanne Schmidt, Tim Rolff, Reinhard Bacher, Wim Leemans, and Frank Steinicke. 2023d. Magic nerf lens: Interactive fusion of neural radiance fields for virtual facility inspection. *arXiv preprint arXiv:2307.09860* (2023).

Shaoxu Li and Ye Pan. 2023. Interactive Geometry Editing of Neural Radiance Fields. *arXiv preprint arXiv:2303.11537* (2023).

Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. 2023a. PAC-neRF: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *arXiv preprint arXiv:2303.05512* (2023).

Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. 2023. LucidDreamer: Towards High-Fidelity Text-to-3D Generation via Interval Score Matching. *arXiv preprint arXiv:2311.11284* (2023).

Gao Lin, Liu Feng-Lin, Chen Shu-Yu, Jiang Kaiwen, Li Chunpeng, Yukun Lai, and Fu Hongbo. 2023. SketchFaceNeRF: Sketch-based facial generation and editing in neural radiance fields. *ACM Transactions on Graphics* (2023).

David B Lindell, Julien NP Martel, and Gordon Wetzstein. 2021. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14556–14565.

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural sparse voxel fields. *Advances in Neural Information Processing Systems* 33 (2020), 15651–15663.

Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. 2021. Mixture of volumetric primitives for efficient neural rendering. *ACM Transactions on Graphics (ToG)* 40, 4 (2021), 1–13.

William E. Lorensen and Harvey E. Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Comput. Graph.* 21, 4 (aug 1987), 163–169.

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (Burlingame, California) *(MIG '16)*. Association for Computing Machinery, New York, NY, USA, 49–54.

Xiaoxu Meng, Weikai Chen, and Bo Yang. 2023. NeAT: Learning Neural Implicit Surfaces with Arbitrary Topologies from Multi-view Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 248–258.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Matthias Müller, Nuttapong Chentanez, and Miles Macklin. 2016. Simulating visual geometry. In *Proceedings of the 9th International Conference on Motion in Games* (Burlingame, California) *(MIG '16)*. Association for Computing Machinery, New York, NY, USA, 31–38.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)* 41, 4 (2022), 1–15.

Ken Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph.* 32, 3, Article 27 (jul 2013), 22 pages.

Michael Niemeyer and Andreas Geiger. 2021. Campari: Camera-aware decomposed generative neural radiance fields. In *2021 International Conference on 3D Vision (3DV)*. IEEE, 951–961.

Michael Oechsle, Songyou Peng, and Andreas Geiger. 2021. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5589–5599.

Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. 2021. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5865–5874.

Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. 2021. Animatable neural radiance fields for modeling dynamic human bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14314–14323.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10318–10327.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.

Yi-Ling Qiao, Alexander Gao, Yiran Xu, Yue Feng, Jia-Bin Huang, and Ming C Lin. 2023. Dynamic mesh-aware radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 385–396.

Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. 2021. Derf: Decomposed radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14153–14161.

Tim Rolff, Susanne Schmidt, Ke Li, Frank Steinicke, and Simone Frintrop. 2023. VRS-NeRF: Accelerating Neural Radiance Field Rendering with Variable Rate Shading. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 243–252.

Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2, Article 11 (feb 2015), 36 pages.

Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. 2023. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics* 29, 5 (2023), 2732–2742.

Olga Sorkine and Daniel Cohen-Or. 2004. Least-squares meshes. In *Proceedings Shape Modeling Applications, 2004.* IEEE, 191–199.

Kay M Stanney, Mansooreh Mollaghasemi, Leah Reeves, Robert Breaux, and David A Graeber. 2003. Usability engineering of virtual environments (VEs): identifying multiple criteria that drive effective VE system design. *International Journal of Human-Computer Studies* 58, 4 (2003), 447–481.

Jia-Mu Sun, Tong Wu, Yong-Liang Yang, Yu-Kun Lai, and Lin Gao. 2023. SOL-NeRF: Sunlight modeling for outdoor scene decomposition and relighting. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11.

Alistair G Sutcliffe, Charalambos Poullis, Andreas Gregoriades, Irene Katsouri, Aimilia Tzanavari, and Kyriakos Herakleous. 2019. Reflecting on the design process for virtual reality applications. *International Journal of Human–Computer Interaction* 35, 2 (2019), 168–179.

Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. 2022. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2149–2159.

Haithem Turki, Vasu Agrawal, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Deva Ramanan, Michael Zollhöfer, and Christian Richardt. 2023. HybridNeRF: Efficient Neural Rendering via Adaptive Volumetric Surfaces. *arXiv preprint arXiv:2312.03160* (2023).

Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. 2023. Nerf-art: Text-driven neural radiance fields stylization. *IEEE Transactions on Visualization and Computer Graphics* (2023).

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689* (2021).

L.R. Wanger, J.A. Ferwerda, and D.P. Greenberg. 1992. Perceiving spatial relationships in computer-generated images. *IEEE Computer Graphics and Applications* 12, 3, 44–58.

Tong Wu, Jia-Mu Sun, Yu-Kun Lai, and Lin Gao. 2023. De-nerf: Decoupled neural radiance fields for view-consistent appearance editing and high-frequency environmental relighting. In *ACM SIGGRAPH 2023 conference proceedings*. 1–11.

Hongchi Xia, Zhi-Hao Lin, Wei-Chiu Ma, and Shenlong Wang. 2024. Video2Game: Real-time, Interactive, Realistic and Browser-Compatible Environment from a Single Video. In *CVPR*.

Tianyi Xie, Zeshun Zong, Yuxin Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. 2024. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. 2022. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 641–676.

Linning Xu, Vasu Agrawal, William Laney, Tony Garcia, Aayush Bansal, Changil Kim, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Aljaž Božič, et al. 2023a. VR-NeRF: High-fidelity virtualized walkable spaces. In *SIGGRAPH Asia 2023 Conference Papers*. 1–12.

Yuelang Xu, Benwang Chen, Zhe Li, Hongwen Zhang, Lizhen Wang, Zerong Zheng, and Yebin Liu. 2023b. Gaussian Head Avatar: Ultra High-fidelity Head Avatar via Dynamic Gaussians. *arXiv preprint arXiv:2312.03029* (2023).

Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. 2023. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642* (2023).

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. 2021. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems* 34 (2021), 4805–4815.

Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. 2023. Gaussian grouping: Segment and edit anything in 3d scenes. *arXiv preprint arXiv:2312.00732* (2023).

Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 965–972.

Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. 2023. Drivable 3d gaussian avatars. *arXiv preprint arXiv:2311.08581* (2023).

Figure 9: Fox, Bear, and Horse. VR-GS allows users to manipulate 3D GS at a real-time rate with physically plausible responses.

Figure 10: Ring Toss and Table Brick Game. Participants can engage in interactive ring toss and table brick breakout games simulated within authentic environments.

Figure 11: Toy Collection. We place all the toys on the ground first, and then move them into the basket.

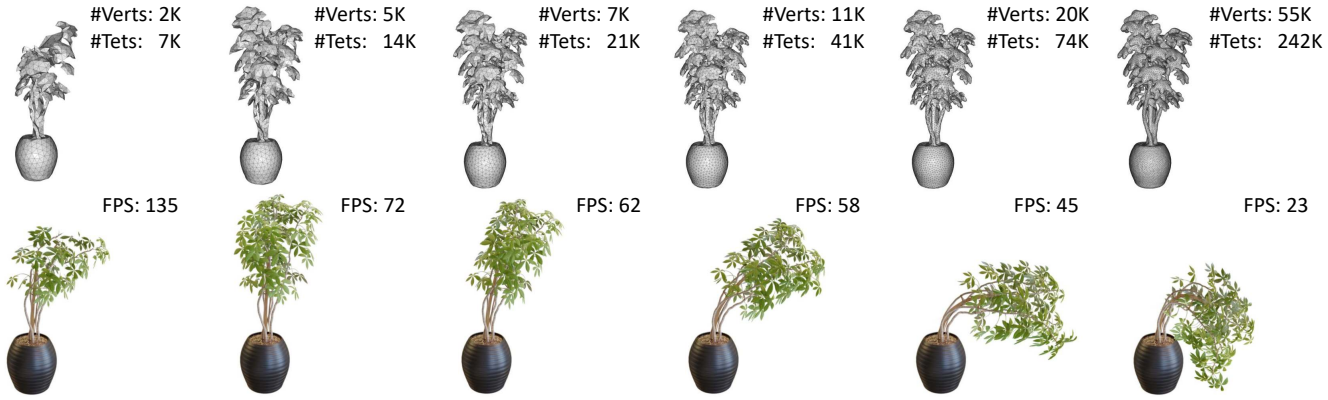

Figure 12: Trade-offs between Quality and Performance. The top row displays cage meshes at varying resolutions, and the bottom row illustrates the corresponding simulation dynamics. Low-resolution meshes fail to capture fine dynamic details, whereas high-resolution meshes compromise real-time performance and can result in overly soft artifacts in the simulated object due to non-converging simulations. We employ mid-resolution meshes in practice to achieve an optimal balance between high frame rates and realistic physical dynamics.

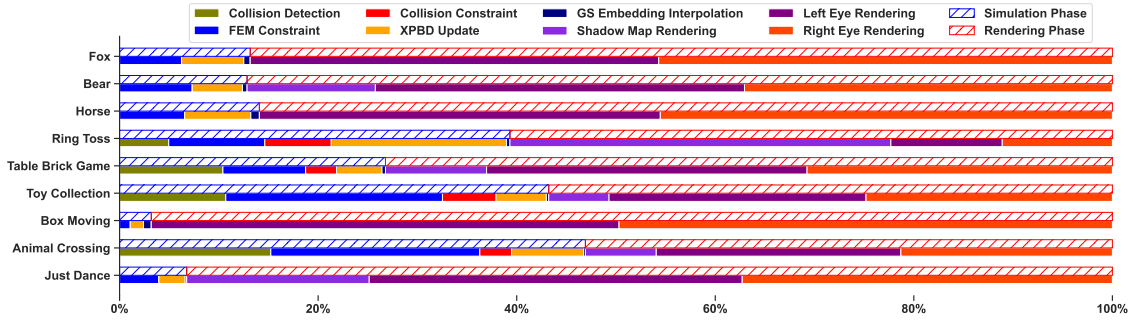

Figure 13: Timing Breakdown of Demos in Section 6.2.

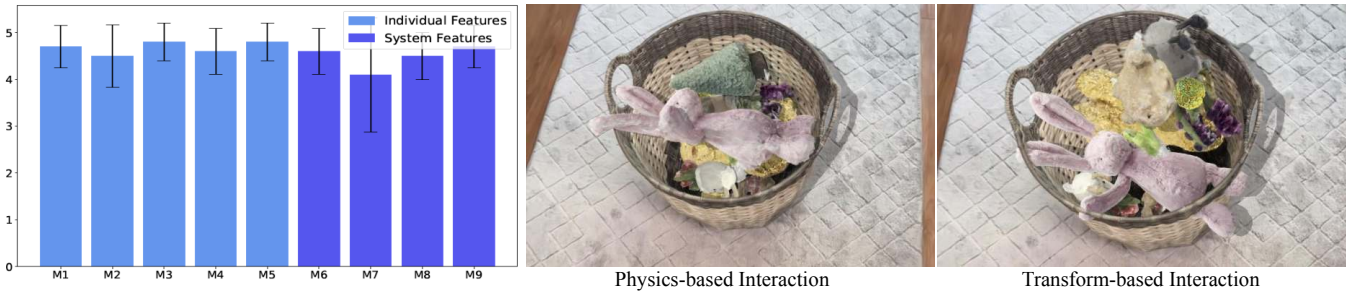

Figure 14: Study Results. The left chart summarizes individual (M1-M5: object manipulation, scene inpainting, illumination, dynamics, physics placement) and system feedback (M6-M9: ease of use, latency, functionality, satisfaction). The right two figures show that physics-based interaction enhances immersion and realism in editing, whereas transform-based interaction yields less authentic outcomes, e.g. an undeformed and penetrated toys.