# EnliveningGS: Active Locomotion of 3DGS

Siyuan Shen[1] Tianjia Shao[1*] Kun Zhou[1] Chenfanfu Jiang [2] Yin Yang[3]
[1] State Key Laboratory of CAD&CG, Zhejiang University, [2] UCLA, [3] University of Utah

## Abstract

*This paper presents a novel pipeline named* EnliveningGS*, which enables active locomotion of 3D models represented with 3D Gaussian splatting (3DGS). We are inspired by the fact that real-world lives pose their bodies in a natural and physically meaningful manner by compressing or elongating muscle fibers embedded in the body. EnliveningGS aims to replicate the similar functionality of 3DGS models so that the object within a 3DGS scene acts like a living creature rather than a static shape — they walk, jump, and twist in the scene under provided motion trajectories driven by muscle activations. While the concept is straightforward, many challenging technical difficulties need to be taken care of. Synthesizing realistic locomotion of a 3DGS model embodies an inverse physics problem of very high dimensions. The core challenge is how to efficiently and robustly model frictional contacts between an "enlivened model" and the environment, as it is the composition of contact/collision/friction forces triggered by muscle activation that generates the final movement of the object. We propose a hybrid numerical method mixing LCP and penalty method to tackle this NP-hard problem robustly. Our pipeline also addresses the limitation of existing 3DGS deformation algorithms and inpainting the missing information when models move around.*

## 1. Introduction

The advent of learning-based radiance field rendering techniques like NeRF [34] and 3D Gaussian splitting (3DGS) [26] have rebranded the pipeline of novel view synthesis. From synthesized RGB information, users can further alter and edit the illumination, appearance, and geometry. It is also possible to integrate those neural scene representations with physics to synthesize plausible physics-based dynamics. For instance, PIE-NeRF uses a quadratic moving least square interpolation to approximate the nonlinear elastodynamics of a NeRF model [11]. PhysGaussian [45] integrates 3DGS with the material point method (MPM) [39]. GaussianSplashing leverages a universal particle-based discretization to animate 3DGS scenes with fluids [10]. Those frameworks allow deeper integrations of 3DGS with relevant downstream graphics/vision tasks, enabling a wide range of exciting applications.

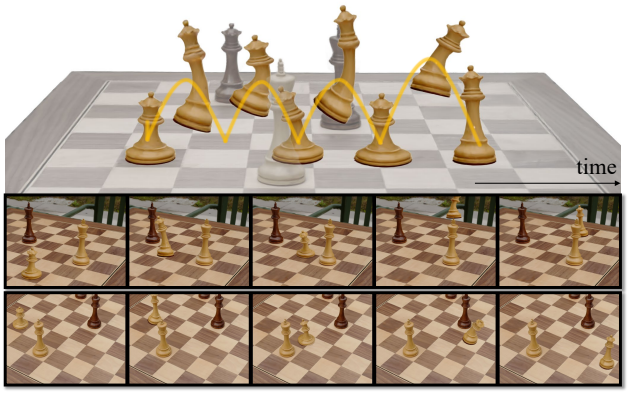


Figure 1. **Jumping Chess.** We use static 3DGS as input and generates physics-aware 4D animations. The chess piece contracts and expands its body, creating jumping motions through collisions and friction with the chessboard, fitting the user's input trajectory.

It is a common paradigm, when combining physics and 3DGS, that the physical effects of the neural scene are *passively* triggered by external forces, which are normally specified by the user or pre-scripted (e.g., the gravity force). This makes sense for inanimate objects — they are not changing their kinematic states unless forced by external excitements. However, the locomotion of a living being follows a distinct modality. Rather than relying on external forces passively, a living entity *actively* adjusts its poses to achieve a desired movement such as walking, bending, or jumping. The enabler of such locomotion is muscle activations that contract or elongate different parts of the body, which interact with the environment e.g., the floor. The resultant contact and friction forces, in turn, provide the necessary momentum of the motion. This procedure is beyond the capability of existing physics-based 3DGS techniques as the muscle activations can hardly be measured or synthesized with a passive physics simulator. It is unfortunate that an interesting 3DGS scene can only house lifeless objects.

In this paper, we show a novel framework to further en-

* Corresponding author.

hance the 3DGS expressivity. We name our framework *EnliveningGS*, which inversely computes muscle activations embedded in a 3DGS model that lead to user-specific motion. As shown in Figure 1, the user only needs to provide a desired trajectory of the object (or other high-level descriptions of the kinematics). Given a set of embedded muscle fibers (approximated by piece-wise linear springs), EnliveningGS segments the object from the scene, estimates the possible contact/friction forces between the object and its surrounding environment, and calculates the optimal muscle activations to drive the corresponding body motion so that the object behaves like a living creature and moves following the prescribed trajectories as much as possible. As contact modeling is known to be an NP-hard problem and quickly becomes unsolvable when there are a moderate number of contact incidents, EnliveningGS introduces a two-stage locomotion procedure to reduce the solution space and efficiently handle frictional contacts between the 3DGS model and the scene. To summarize, our contributions include:

- **Active Animatable 3DGS Pipeline:** We present EnliveningGS pipeline that brings any household object to life. When they walk or jump, occluded areas become visible again. To overcome such visual artifacts, we propose an improved inpainting algorithm to fill colors and textures back to those exposed areas based on the known prior of the environment. We also present a self-splitting machanism for Gaussian kernels to capture and reduce spiky artifacts under significant deformation.
- **Hybrid Contact Modeling Framework for 3DGS:** We design a novel hybrid numerical procedure combing the penalty method, which softens the hard contact constraint, and the classic Linear Complementarity Programming (LCP) [38] so that the combinatorial space of the active set can be significantly reduced, and the two-way coupling with the underlying nonlinear physics simulation becomes feasible.

## 2. Related Work

**3D Gaussian editing.** Recently, 3D Gaussian splatting (GS) [25] utilizes a set of 3D Gaussian kernels to represent static scenes explicitly and achieve state-of-the-art results. Subsequently, editing on GS has become a popular topic[7, 13, 30, 31, 47, 48, 50]. To make static 3D scenes deformable, some works extract explicit high-quality meshes from 3DGS representations and align GS kernels with mesh surfaces for guidance when deformed [12, 14, 44] . Their reliance on strong topological information and high-quality meshes makes it difficult to balance quality and interactivity. SC-GS [20] learns sparse control points for 3D scene dynamics but faces challenges with large movements and complex deformation.

**Physics-aware dynamics on 3D GS.** To combine a static GS scene with a physics engine to generate dynamics on GS, PhysGaussian[45] leverages the first-order information from the displacement map (deformation gradient) to assist dynamic simulations. VR-GS [23] constructed a tetrahedral cage for each segmented Gaussian kernel group and embedded these kernel groups into corresponding meshes, allowing the deformed mesh driven by eXtended Position-based Dynamics (XPBD) [32] to guide the deformation of the Gaussian kernels. GASP[6] treats the simulation process as a black box and integrates Newtonian dynamics into the GaMeS[44] framework. PhysDreamer [49] distills dynamics priors learned by video generation models. It is worth mentioning that existing methods consider the simulation process as a forward process, leaving the inverse solving and control of physical parameters of objects on 3D GS as an unexplored area.

**Contacts in locomotion control** Friction plays a crucial role in soft body locomotion control. Stewart and Trinkle linearize the 3D friction model using a polyhedral cone and propose an LCP formulation to approximate Coulomb's friction cone conditions [38] and many improve upon it [2, 3, 24, 35, 43]. LCP-based methods offer exact solutions and are efficient for small to medium-sized problems but can be computationally expensive for optimization problems. Due to this drawback, many previous methods explicitly assumed that the contacts remain static [21, 27] while optimizing control forces subject to equations of motion. A few studies [41] exploit the physical meaning of complementarity constraints as heuristics to improve the solution and performance of the solver but still often fall back into exhaustive searching due to a lack of good initial state. The penalty method models contacts and frictions as spring forces and evaluates them at the instant [16–18, 42, 46] but can introduce numerical challenges such as stiffness and instability [1, 36]. The combination of the penalty methods and LCP-based methods has been less explored.

## 3. Method

### 3.1. Overview

A 3DGS scene consists of a collection of 3D Gaussians $\mathcal{G}_{scene}$. Each Gaussian kernel is defined by a center position $\mu \in \mathbb{R}^3$, a covariance matrix $\Sigma$, the opacity $\sigma \in [0, 1)$ and a set of spherical harmonics coordinates $c \in \mathbb{R}^k$ describing the colors emitted by the Gaussian in all directions. The covariance matrix $\Sigma$ can be factorized into a rotation matrix $\mathbf{R}$ expressed as a quaternion $\mathbf{q} \in \mathbb{R}^4$ and a scaling matrix $\mathbf{S}$ of its diagonal vector $\mathbf{s} \in \mathbb{R}^3$: $\Sigma = \mathbf{RSS}^T\mathbf{R}^T$.

Vanilla 3DGS represents a static environment where users can switch their viewpoints within the scene freely. It is also possible to include dynamics to the scene by adding

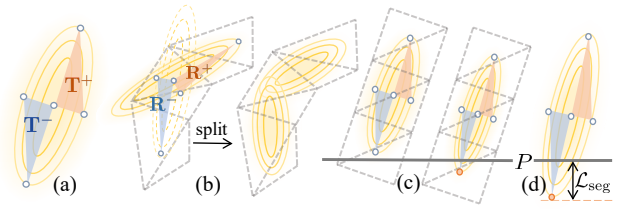


Figure 2. **Bidirectional local embedding**. Our bidirectional local embedding (a) serves as guidance for (b) self-splitting GS deformation, (c) accurate collision handling, and (d) non-penetration regularization.

external forces e.g., as in PhysGaussian [45] so that 3DGS models move or deform under user-specified forces *passively*. Nevertheless, real-world living creatures are self-driven — their movements are enabled by embedded muscles given a desired kinematic motion target. Such an *active* movement mechanism is much more difficult to obtain as it inevitably involves frictional contacts between the model and the environment. To this end, we incorporate additional physical expressions into the Gaussian elements, enabling them to be self-driven in a physics-based way. These elements can change their appearance at each time step and approximate user-specified motion targets interacting with the environment.

### 3.2. Hybrid Mesh-Gaussian Representation

Let $\mathcal{G}_o$ denote the Gaussian elements of an object of interest, which is segmented from the scene $\mathcal{G}_{scene}$. We first create an encapsulating tetrahedral mesh $\mathcal{T} = (\mathcal{V}, \mathcal{F})$ to approximate the object's overall volume, mass, and deformation. Here, $\mathcal{V}$ and $\mathcal{F}$ refer to the sets of nodal points and faces in $\mathcal{T}$. To achieve this segmentation, we approximate the nearby environment with planes (more in Secs. 3.5 and 3.6). The plane $P$ expressed as a unit normal vector $\mathbf{n}$ and its distance $d$ from the origin:

$$P(\mathbf{n}, d) = \{\mathbf{x} \in \mathbb{R}^3 \mid \mathbf{N}^T\mathbf{x} + d = 0\}, \tag{1}$$

is used to "cut out" irrelevant Gaussians, reduce the blurred regions at the cut, and serve as a prior for inpainting the missing parts after segmentation. Additionally, it is utilized for collision detection to determine the collision forces acting on $\mathcal{G}_o$.

$\mathcal{T}$ determines the states of Gaussian kernels in $\mathcal{G}_o$ after deformation. The deformed position of a point $\mathbf{x}'$ originally at $\mathbf{x}$ within a tetrahedron is represented by the barycentric coordinates of the four vertices of the tetrahedron $\mathbf{v}'$:

$$\mathbf{x}' = \sum_i \lambda_i(\mathbf{x})\mathbf{v}'_i. \tag{2}$$

After deformation, $\mathcal{T}$ provides a global displacement map for all Gaussians in $\mathcal{G}_o$. However, the internal rotation and scaling of the Gaussians cannot be accurately described by position-based interpolation, resulting in spiky artifacts. Additionally, the tetrahedral mesh is a coarse mesh. Its faces have significant gaps from the boundaries of the object, and it does not accurately describe the collision between $\mathcal{G}_o$ and the plane. To address these issues, we establish a *bidirectional local embedding* within the Gaussian around the mean $\mathbf{m}$ to precisely describe its influence range. This is determined by the endpoints of the ellipsoid corresponding to the Gaussian $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$:

$$\begin{aligned} \mathbf{T}^{\pm} &= [\mathbf{m}, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3] \\ &= [\mathbf{m}, \mathbf{m} \pm \gamma s_1 \mathbf{r}_1, \mathbf{m} \pm \gamma s_2 \mathbf{r}_2, \mathbf{m} \pm \gamma s_3 \mathbf{r}_3], \end{aligned} \tag{3}$$

where the relevant parameters can be obtained from the scaling matrix $\mathbf{S} = \mathsf{diag}(s_1, s_2, s_3)$ and the rotation Matrix $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$. $\gamma$ is a parameter that describes the range of influence, and we set its default value as 3.0. Each Gaussian has two directional embeddings, $\mathbf{T}^+$ and $\mathbf{T}^-$, and each contains four points.

We design bidirectional local embeddings for several purposes. For collision processing, it is used to detect the extent to which an object collides with the environmental plane; and for segmentation, it detects whether the Gaussian is cut by the plane and establishes a loss term to penalize Gaussian kernels that intersect the plane under initial setting; For deformation, it is used to accurately describe the scaling and rotation of the deformed Gaussian kernel. Finally, based on the angle between two directions, it determines whether the deformed Gaussian kernel is sufficient to cover the radiance field.

### 3.3. Soft Body Locomotion on Plane

The core of our pipeline is an active locomotion solver for $\mathcal{T}$ and $\mathcal{G}_o$. This is in the form of an inverse problem involving high-dimension inequalities constraints for processing frictional contacts between a 3DGS object and the environment.

#### 3.3.1. Muscle-Driven Dynamics

We embed muscle fibers within $\mathcal{T}$ to serve as the primary drivers of motion. At each time step, the nodal positions of $N$ nodes in $\mathcal{V}$, denoted as $\mathbf{p}$, is determined through numerical integration of the dynamic equations of motion:

$$\mathbf{M}\ddot{\mathbf{p}} = \mathbf{f}_{ext} + \mathbf{f}_{int} + \mathbf{f}_d + \mathbf{f}_m + \mathbf{f}_c, \tag{4}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{f}_{ext}$, $\mathbf{f}_{int}$, $\mathbf{f}_d$, $\mathbf{f}_c$, $\mathbf{f}_m$ represent external, internal, damping, muscle, and contact forces, respectively. We use $\dot{(\cdot)}$ for velocity and $\ddot{(\cdot)}$ for acceleration. The external force $\mathbf{f}_{ext}$ includes e.g., the gravity force. The internal force $\mathbf{f}_{int}$, also known as the elastic force, is the resultant force generated by the material's elastic properties that resist deformation. In our implementation, this force

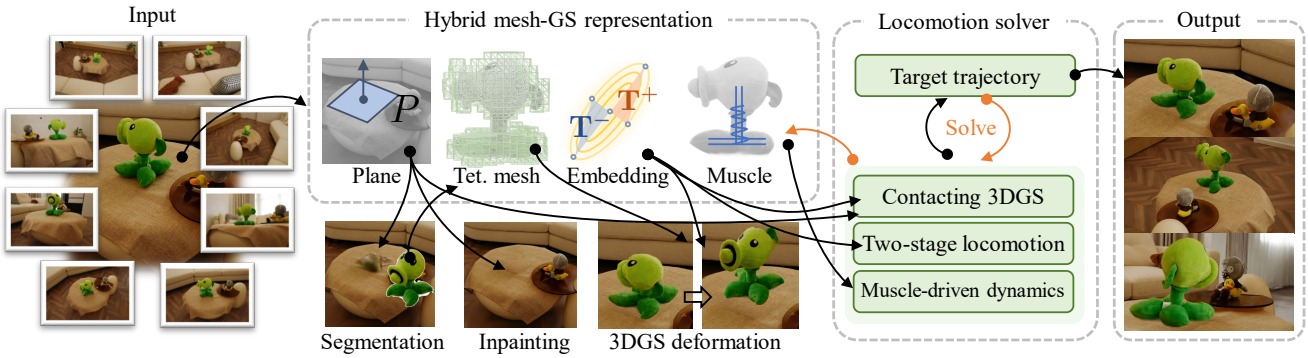


Figure 3. **An overview of EnliveningGS pipeline.** Our hybrid mesh-Gaussian representation plays an important role in all tasks within the pipeline. Our locomotion solver inversely computes the muscle activation embedded in the segmented 3DGS object that leads to user-specific motion.

is determined as the negative gradient of the Neo-Hookean energy [5]:

$$\Psi = \frac{\mu}{2}(I_C - 3) - \mu \ln J + \frac{\lambda}{2}(\ln J)^2 . \tag{5}$$

Here, $\mu$ and $\lambda$ are the Lamé parameters. $I_C = \mathrm{tr}(\mathbf{F}^T\mathbf{F})$ and $J = \det(\mathbf{F})$ are computed based on the deformation gradient $\mathbf{F}$. We use the Rayleigh damping for the damping force:

$$\mathbf{f}_d = (\alpha \mathbf{M} + \beta \mathbf{K}_0)\,\dot{\mathbf{p}}, \tag{6}$$

where $\mathbf{K}_0$ is the rest-shape stiffness matrix.

We consider muscle fibers as polygonal curves with $M$ segments. The spring force caused by each segment is computed as: $f = ka$ where $k$ is the stiffness of the muscle fiber and $a$ denotes the change in the segment's length. When a muscle segment activates, the spring force will shorten or bend the body around it. Muscle force on the whole soft body is encoded in a pose-dependent activation matrix $\mathbf{A} \in \mathbb{R}^{3N\times M}$:

$$\mathbf{f}_m = \mathbf{A}(\mathbf{p})\mathbf{a}\,, \tag{7}$$

where $\mathbf{a} \in \mathbb{R}^M$ is the vector of activations of all the muscle segments. Please refer to the supplementary materials for a complete description of the calculations involving activation matrix $\mathbf{A}$.

We provide an automatic muscle generation tool given $\mathcal{T}$, which uses the medial axis transform [8, 29] of the model. Four longitudinal muscles are placed parallel near the medial axis at a given radius, and radial muscles are placed at given intervals. Users can also customize the muscle setup with preferred arrangements inside the object to achieve different movements. For instance, by implanting several longitudinal muscle fibers that run the length of the body, the body can be shortened when these muscles contract. Bending the body can be achieved by contracting these muscles unevenly on one side. Additionally, circular muscles wrap around the body, enabling it to twist, while radial muscles extend across the body's cross-section, allowing the body to elongate while preserving its volume.

### 3.3.2. Contacting 3DGS

In this section we present a novel contact model for 3DGS. For Gaussian kernels in $\mathcal{G}_o$, we first define $\mathbf{u}$ as the position of points in bidirectional local embedding $\mathbf{T}^{\pm}$ below the plane as $\mathbf{N}^T\mathbf{u} + d \le 0$. Non-interpenetration constraints on 3DGS is thus defined as inequalities:

$$C_n(\mathbf{u}) \equiv \mathbf{N}^T\dot{\mathbf{u}} \ge \mathbf{0}. \tag{8}$$

We decompose contact forces on the Gaussian as $\mathbf{f}_g = \mathbf{N}f_\perp + \mathbf{D}\mathbf{f}_\parallel$, where $\mathbf{D}$ is a set of directions at the contact point approximating the friction cone, and they are perpendicular to the plane normal $\mathbf{N}$. $f_\perp$ and $\mathbf{f}_\parallel$ are the magnitudes of normal and tangent forces.

LCP formulation [2] that regulates the contact velocity and contact force is:

$$\mathbf{0} \le \begin{bmatrix} f_\perp \\ \mathbf{f}_\parallel \\ \lambda \end{bmatrix} \perp \begin{bmatrix} \mathbf{N}^T\dot{\mathbf{u}} \\ \mathbf{D}^T\dot{\mathbf{u}} + \mathbf{E}\lambda \\ \mu f_\perp - \mathbf{E}^T\mathbf{f}_\parallel \end{bmatrix} \ge \mathbf{0}, \tag{9}$$

where $\mu$ is the friction coefficient, and $\lambda$ is an auxiliary variable (e.g., the Lagrange multiplier) that relates the tangent velocity to a sliding contact. $\mathbf{E}$ is a block-diagonal matrix of $\mathbf{e}$. LCP condition notation $\mathbf{x} \perp \mathbf{y}$ indicates $\mathbf{x}^T\mathbf{y} = 0$. The first line of Equation (9) is a complementarity formulation of contact that enforces the non-penetration condition, and the following two lines encode Coulomb's friction law.

Like [33], contact can be viewed as a regularization of the complementarity form of the non-penetration constraint, which gives penalty forms of contact as spring force $\mathbf{f}_g^{spring}$:

$$\mathbf{f}_\perp^{spring} \equiv -k_n \min(0, C_n(\mathbf{u})), \tag{10}$$

$$\mathbf{f}_\parallel^{spring} \equiv -\min\left(k_f, \mu\frac{\|\mathbf{f}^{spring}\|}{\|\mathbf{D}^T\dot{\mathbf{u}}\|}\right)\mathbf{D}^T\dot{\mathbf{u}}, \tag{11}$$

where $k_n$ and $k_f$ control the stiffness of the contact. Forces on local embeddings can be converted back onto points on $\mathcal{T}$ using the same weights in Equation (2), giving us $\mathbf{f}_c$ or $\mathbf{f}_c^{spring}$.

#### 3.3.3. Two-stage Locomotion

Unlike force-based passive dynamics, we achieve the desired locomotion mainly by intuitively prescribing the motion trajectory of the object. For the position of a target point $\mathbf{p}^\star$, the objective function that constrains the position can be written as:

$$G_{position} = \|f(\mathbf{p}) - \mathbf{p}^\star\|^2, \tag{12}$$

where $f$ can be functions that compute the positional target of the object. For instance, a commonly used setup is $f_{COM}$, which calculates the COM (center of mass) $\mathbf{c} = f_{COM}(\mathbf{p})$ of the body, of local COM coordinates (foot, head, base, etc.). Similar objective functions can be used to track the velocity, acceleration, angular momentum, base area, etc.

We first solve Equation (4) by substituting $\mathbf{f}_c$ with $\mathbf{f}_c^{spring}$ and temporarily keeping muscle activation $\mathbf{a}$ same as the last time step. Based on the resulting $\dot{\mathbf{p}}$, we convert the state of contact points back into the inequalities in Equation (9), and solve a Quadratic Programming (QP) problem to find the best $\mathbf{a}$. For example, if a contact point is in static contact, the corresponding QP is:

$$\begin{aligned} &\min_{\mathbf{a}, f_\perp, \mathbf{f}_\parallel, \lambda} G\left(\mathbf{a}, f_\perp, \mathbf{f}_\parallel\right) \\ &\text{subject to} \\ &\mathbf{0} \le f_\perp, \quad \mathbf{N}^T \dot{\mathbf{u}}^{n+1} = \mathbf{0} \\ &\mathbf{0} \le \mathbf{f}_\parallel, \quad \mathbf{D}^T \dot{\mathbf{u}}^{n+1} + \mathbf{E}\lambda = \mathbf{0} \\ &\mathbf{0} = \lambda, \quad \mu f_\perp - \mathbf{E}^T \mathbf{f}_\parallel \ge \mathbf{0}. \end{aligned} \tag{13}$$

Note that all contact points may have their own states. If the results are not satisfactory, we will use the pivoting method as in [41] to find the appropriate contact states.

### 3.4. Gaussian Deformation in Tetrahedral Mesh

The barycentric coordinates of the points in $\mathbf{T}^+$ are calculated in the reference configuration, and the deformed coordinates in the current configuration $\mathbf{T}_s^+ = [\mathbf{m}', \mathbf{v}_1', \mathbf{v}_2', \mathbf{v}_3']$ are then calculated at each timestep to describe the deformation.

After that, we construct two matrices $\mathbf{D}_m$ and $\mathbf{D}_s$, representing the differences in vertex coordinates in the reference and current configurations, respectively:

$$\begin{aligned} \mathbf{D}_m &= [\mathbf{v}_1^0 - \mathbf{m}, \mathbf{v}_2^0 - \mathbf{m}, \mathbf{v}_3^0 - \mathbf{m}] \\ \mathbf{D}_s &= [\mathbf{v}_1' - \mathbf{m}', \mathbf{v}_2' - \mathbf{m}', \mathbf{v}_3' - \mathbf{m}']. \end{aligned} \tag{14}$$

The linear transformation $\mathbf{F}$ that maps the reference configuration to the current configuration can be calculated as:

$$\mathbf{F} = \mathbf{D}_s \cdot \mathbf{D}_m^{-1}.$$

The deformed covariance becomes: $\Sigma' = \mathbf{F}\Sigma\mathbf{F}^T$ and is converted into scaling and rotation by performing singular value decomposition (SVD) on $\Sigma'$:

$$\begin{aligned} \mathbf{S}^+ &= \mathsf{diag}(\sqrt{\mathsf{SVD}_\Sigma(\Sigma')}) \\ \mathbf{R}^+ &= \mathsf{SVD}_U(\Sigma'). \end{aligned} \tag{15}$$

Note that if $\det(\mathbf{R}^+) = -1$, one needs to negate its last column to ensure it is a valid rotation matrix in $SO(3)$. It is also necessary to adjust the spherical harmonics' orientation by applying rotation $R_{SH} = \mathsf{SVD}_U(\Sigma')\mathsf{SVD}_V(\Sigma')^T$ to display the correct colors from various angles.

The points in local embeddings represent the shape of the deformed Gaussian kernel but they do not necessarily belong to the same tetrahedron. For some larger Gaussian kernels, significant rotations may cause them to extend beyond the original radiance field's boundaries. To avoid this situation, we use the reverse embedding $\mathbf{T}^-$ and use the aforementioned method to obtain its corresponding rotation matrix $\mathbf{R}^-$. The angle between $\mathbf{R}^+$ and $\mathbf{R}^-$ can be expressed as:

$$\theta = \arccos\left(\frac{\mathsf{tr}(\mathbf{R}^{+T}\mathbf{R}^-) - 1}{2}\right). \tag{16}$$

If $\theta$ exceeds the threshold $\theta > \eta$, it indicates that the region covered by the Gaussian has transformed from a plane to a curved surface. A single kernel can no longer represent the deformed area accurately; thus, this Gaussian will be split into two from the center. Figure 2 (b) provides a schematic of our splitting process, while Figure 8 presents the corresponding experimental results.

Inspired by [9, 15], the splitting of a Gaussian $(\alpha_0, \mu_0, \Sigma_0)$ into two new Gaussian kernels $(\alpha_l, \mu_l, \Sigma_l)$ and $(\alpha_r, \mu_r, \Sigma_r)$ at the center is calculated by:

$$\begin{aligned} &\alpha_l = \alpha_r = \frac{\alpha_0}{2}, \\ &\mu_l = \mu_0 - \frac{\mathbf{v}_k - \mu_0}{\kappa}, \quad \mu_r = \mu_0 + \frac{\mathbf{v}_k - \mu_0}{\kappa}, \\ &\mathbf{\Sigma}_l = \mathbf{\Sigma}_r = \mathbf{\Sigma}_0 - \frac{(\mathbf{v}_k - \mu_0)(\mathbf{v}_k - \mu_0)^T}{\kappa^2}. \end{aligned} \tag{17}$$

In the above, $\mathbf{v}_k \in \mathbf{T}^+$ stands for the endpoint on main principle axis with the largest scaling factor $s_k$. Factor $\kappa = \frac{2\gamma}{\sqrt{2\pi}}$ is derived by maximizing the similarity between each new Gaussian and their own part of the original. We refer readers to supplementary material for the complete derivation.

### 3.5. Plane Estimation

To find the parameters $\mathbf{N}$ and $d$ of a plane in a 3DGS scene, we first construct a depth map on the plane:

$$\mathcal{D}_p(u, v) = -\frac{\mathbf{N} \cdot \mathbf{t}_c + d}{\mathbf{n}^T \mathbf{R}_c^T \mathbf{K}^{-1} \mathbf{p}(u, v)}, \tag{18}$$

where $\mathbf{p}(u, v)$ represents the pixel coordinates in homogeneous form $[u, v, 1]^T$, $\mathbf{R}_c$ and $\mathbf{t}_c$ are the rotation and translation extracted from the camera's extrinsic matrix, and $\mathbf{K}$ denotes the camera's intrinsic matrix. Initially, $\mathbf{N}$ is set to camera's look-at direction and $d$ is set to zero to ensure solvability.

By minimizing the error between the depth map within the region and the GS depth map $\mathcal{D}_g$, the plane parameters can be optimized by:

$$\arg\min_{\mathbf{N},d} \sum_{(u,v)\in\mathcal{M}} \|\mathcal{D}_p(u, v) - \mathcal{D}_g(u, v)\|, \tag{19}$$

where $\mathcal{M}$ is a mask of the planar region which can be obtained through image-based segmentation algorithms [28] or roughly drawn by the user.

The user can repeat this step on multiple images and optimize simultaneously to obtain fine-tuned plane parameters. In practice, a single viewpoint is usually sufficient to estimate an accurate plane.

### 3.6. Object Segmentation with Plane Priors

We categorize the Gaussian scene into two types: dynamic splitting and static splitting. Equation (17) showcases dynamic online splitting to enhance the quality during the deformation process and ensure the speed of splitting on animating objects. For a Gaussian model, since the segmentation plane of the object and environment is already determined, it is more suitable to incorporate the plane information during training to regulate the Gaussian parameters at the cutting plane.

To perform static splitting on the training stage, we first determine whether the Gaussian intersects with the plane. This requires checking if all the endpoints in $\mathbf{T}^{\pm}$ are on the same side of the plane. We project center-to-endpoints distance onto the plane normal:

$$\eta_{seg} = \max_{\mathbf{v}_i\in\mathbf{T}^{\pm}} |\mathbf{N}^T(\mathbf{v}_i - \mathbf{m})|. \tag{20}$$

If the distance from Gaussian kernel to the plane $|d_0| = |P(\mathbf{n}, d)(\mathbf{m})| < \eta_{seg}$, we will constrain the Gaussian's parameter or split the Gaussian with the plane.

We fine-tune the trained Gaussian model by introducing the following non-penetration regularization term to ensure the model's influence range does not cross the plane:

$$\mathcal{L}_{seg} = ||\eta_{seg} - |d_0|||_2 \tag{21}$$

### 3.7. Inpainting 3D Object on GS

After extracting the object $\mathcal{G}_o$, a hole is left in the background $\mathcal{G}_{scene}$. To address this issue, we introduce the depth map of the plane in Equation (18) into the depth-based inpainting method [30] and use a 2D inpainting tool [40] to obtain the color of the missing parts from the current viewpoint. Subsequently, we fine-tune the filled patch using the inpainted single-view image.

## 4. Experiments

We implemented our pipeline on a workstation with a single NVIDIA RTX4090 GPU and Intel 14900K CPU. We used PyTorch for storing and editing 3DGS on GPU, and OSQP [37] for solving QP in Equation (13). Our method supports the original 3DGS implementation [25]. For better visual quality on the planar surface, we set the minimum scaling factor on Gaussian kernels to $\epsilon = e^{-10}$, similar to 2DGS[19]. More benchmarks for the locomotion examples are reported in the supplementary document.

### 4.1. Locomotion Design

**Walking.** We achieve walking by controlling the trajectories of all four legs. The diagonal support legs simultaneously lift the body, while the remaining two legs swing forward with the body's movement. As shown in Figure 4, the wooden stool moves one position to the right by taking four steps.

**Jumping.** Jumping soft-bodied characters are more commonly seen in cartoons and animations. Jumping is a complex action because the contact state between the object and the ground changes rapidly and significantly. To achieve continuous jumping, it is necessary to precisely control the position of the COM and the relative velocity of the center of the base to jump the appropriate distance and maintain balance. We have designed an animation of a game of chess, where the soft-bodied chess pieces move across the board by continuously jumping, as shown in Figure 1.

**Twisting.** Figure 5 illustrates an example of the Peashooter doll twisting, jumping, and rotating 180 degrees. Based on the jumping motion, we additionally control the change in angular momentum along the central axis, allowing the Peashooter to rotate its body while airborne and face the zombie upon landing.

### 4.2. Comparison I. Deformation on GS

We compare our method with the current state-of-the-art physic-aware GS methods. Figure 6 shows a qualitative comparison, where our method demonstrates better results under large deformation. Table 1 presents the related quantitative evaluation: we apply bending, twisting, and falling on the ground deformations to the ficus, chair, and mic examples from the NeRF synthetic dataset [34], and compare them with the rendered images of the source 3D mesh subjected to the same deformations as the ground truth.

### 4.3. Comparison II. Two-stage Locomotion

To evaluate our two-stage locomotion solver, we conducted frame-by-frame tests on an animation sequence with given muscle activation. This animation sequence includes 50 frames, 60 variables, and 50 pairs of linear complementarity constraints. We compared our solution with a QP

Figure 4. **Walking stool.** In the bedroom scene, a wooden stool walks like a quadruped.

Figure 5. **Peashooter vs. Zombie.** On a cloth-covered coffee table, a Peashooter figurine contracts, jumps, and twists its body, landing after a 180-degree spin to face the zombie.

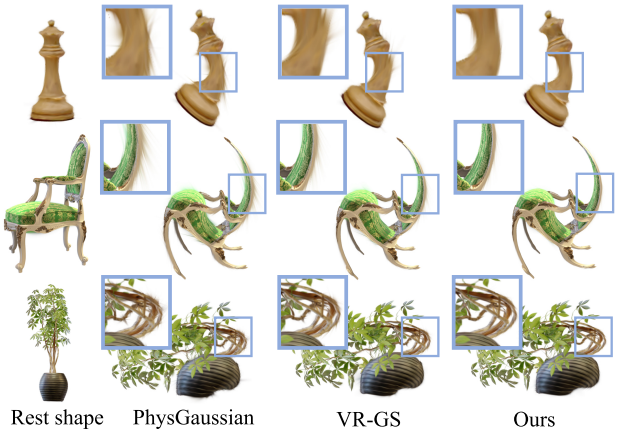


Figure 6. **Visual quality comparison on deforming objects**. Compared to PhysGaussian [45] and VR-GS [23], our method can still suppress spiky artifacts under large deformations.

solver based directly on the static contact assumption and the QPCC solver proposed by [41], and reported the average best and worst results across all frames in Table 2. Our locomotion solver uses the results predicted by the penalty force as guidance, reducing the number of iterations and lowering the objective value.

### 4.4. Comparison III. Inpainting

As shown in Figure 7, we demonstrate the inpainting effect on a plane using the garden example from Mip-NeRF360 dataset [4] and the truck example from DTU dataset [22], comparing it with the existing 3D object inpainting method on GS. In both examples, the removed objects occupy a significant portion of the photo, leaving a large hole on the ground after removal. Existing methods, due to the lack of relevant depth information, struggle to fill such a large missing area even after extensive finetuning of the inpainted 2D image (approximately 10,000 iterations). In contrast,

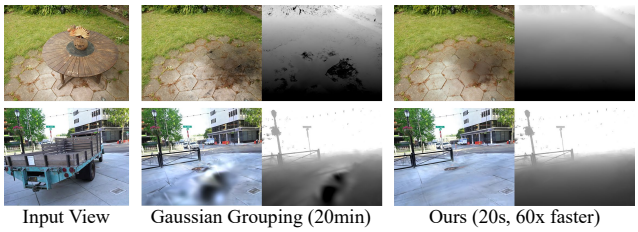


Figure 7. **Large 3D object inpainting**. Comparison on large 3D object inpainting cases on the ground, where Gaussian Grouping [48] requires 20 minutes fine-tuning but still fails to fill the large hole on the ground while our method with better inpainting quality only needs 20 seconds fine-tuning. We show the render result as well as the depth.

our method leverages the plane as a prior, placing a dense and reasonably positioned point cloud in the missing area. This approach requires only about 100 iterations of finetuning, speeding up the inpainting process by several orders of magnitude and remaining unaffected by the size of the missing region.

### 4.5. Ablation Studies

**Bidirectional local embedding** We first use a checkered bar to showcase the effect of bidirectional local embedding as in Figure 8. The checkered bar twists increasingly from right to left, and the sparky artifacts become progressively more pronounced. Without the need to detect whether the Gaussian kernel exceeds the object's surface, our bidirectional local embedding automatically identifies the regions causing sparky artifacts based on the information within the Gaussian kernel. Table 1 also reports quantitative results.

**Non-penetration regularization** Since the segmentation plane under $\mathcal{G}_o$ is occluded from any viewpoint, directly masking out the object from the environment results in significant blurring and penetration at the segmentation plane.

| | Bending | | | Twisting | | | Falling | | |
|---|---|---|---|---|---|---|---|---|---|
| | PSNR ↑ | SSIM ↑ | LPIPS ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
| SuGaR [14] | 28.04 | 0.890 | 0.046 | 26.37 | 0.823 | 0.101 | 25.01 | 0.913 | 0.053 |
| PhysGaussian [45] | 28.14 | 0.907 | 0.043 | 26.53 | 0.856 | 0.089 | 25.10 | 0.923 | 0.048 |
| VR-GS [23] | 28.63 | 0.917 | 0.038 | 26.53 | 0.893 | 0.066 | 25.31 | 0.938 | 0.044 |
| Ours (w/o split) | 28.62 | 0.918 | 0.038 | 26.47 | 0.891 | 0.062 | 25.34 | 0.939 | 0.042 |
| Ours | **28.79** | **0.923** | **0.031** | **26.89** | **0.910** | **0.041** | **25.67** | **0.951** | **0.032** |

Table 1. **Quantitative evaluation of rendering quality after deformation on the NeRF synthetic dataset.**

| | Iterations ↓ | | | Object value ↓ | | |
|---|---|---|---|---|---|---|
| | Avg. | Best | Worst | Avg. | Best | Worst |
| QP | - | - | - | 9.43 | 0.16 | 23.86 |
| QPCC [41] | 46.19 | 1 | 510 | 1.29 | 0.14 | 12.11 |
| Ours | **17.01** | 1 | **382** | **0.83** | **0.13** | **7.89** |

Table 2. **Numerical experiments of locomotion solvers**.

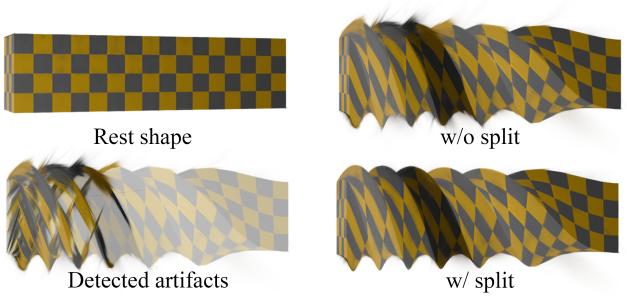


Figure 8. **Ablation study on bidirectional local embedding.** Our method captures the regions that produce spiky artifacts and removes them by splitting the Gaussian kernels to fit the radiance field.

| | PSNR ↑ | SSIM ↑ | LPIPS ↓ | Residual ↓ |
|---|---|---|---|---|
| Mask (w/o $\mathcal{L}_{seg}$) | - | - | - | 0.058 |
| Remove | 25.917 | 0.827 | 0.211 | - |
| w/ $\mathcal{L}_{seg}$ | **35.291** | **0.945** | **0.034** | **0.002** |

Table 3. **Ablation study on non-penetration regularization: quantitative results.**

The Gaussian kernel that penetrates the plane contributes to the object's appearance, so a direct *Remove* operation would create holes in the object. Figure 9 shows the relevant qualitative comparison results. Our non-penetration regularization restricts the Gaussian from penetrating the plane, maintaining the integrity of the object's surface. In the quantitative experiments shown in Table 3, we placed the camera on the segmentation plane, compared the image quality of the object's upper half with full mask-out, and evaluated the residual of the lower half. The residuals are expressed as the ratio of the sum of pixel values in the lower half to those in the upper half.

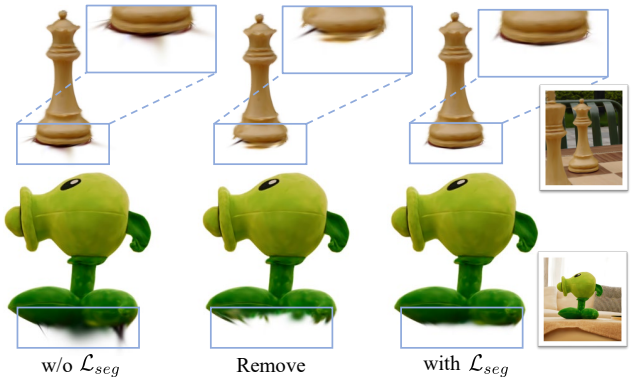


Figure 9. **Ablation study on non-penetration regularization: qualitative results.** With $L_{seg}$ applied to Gaussian kernels penetrating the plane, the object retains its complete appearance near the cutting plane while removing the parts below the cutting plane.

## 5. Conclusion

In this paper, we present EnliveningGS, a novel pipeline that enables active locomotion for 3D models represented by 3DGS. We use a hybrid mesh embedding of 3DGS to better estimate the contact between a 3DGS model and the environment. Given a high-level motion goal, we find the optimal muscle activations driving the movement of the object's body. An LCP-penalty mixed optimizer is used to efficiently calculate the contact and friction force.

**Limitation** Currently, our method relies on plane-based simplification of the environment. When the surroundings consist of more complicated geometries, users may need to use more sophisticated approximation strategies to estimate the contact plane. It is also possible to use triangulated surfaces. Doint so could impose more computational overhead for the contact solver.

**Acknowledgements** The authors would like to thank the reviewers for their insightful comments. Yin Yang is partially supported by NSF 2301040. This work is also supported by NSF China (No. 62322209 and No. 62421003), the gift from Adobe Research, the XPLORER PRIZE, and the 100 Talents Program of Zhejiang University. The source code and data are available at https://gapszju.github.io/EnliveningGS.

## References


[1] Sheldon Andrews, Kenny Erleben, and Zachary Ferguson. Contact and friction simulation for computer graphics. In *ACM SIGGRAPH 2022 Courses*, pages 1–172, Vancouver British Columbia Canada, 2022. ACM. 2

[2] Mihai Anitescu and Florian A Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14: 231–247, 1997. 2, 4

[3] Lijie Bai, John E Mitchell, and Jong-Shi Pang. On convex quadratic programs with linear complementarity constraints. *Computational Optimization and Applications*, 54(3):517–554, 2013. 2

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 7

[5] Javier Bonet and Richard D Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press, 1997. 4

[6] Piotr Borycki, Weronika Smolak, Joanna Waczyńska, Marcin Mazur, Sławomir Tadeja, and Przemysław Spurek. Gasp: Gaussian splatting for physic-based simulations. *arXiv preprint arXiv:2409.05819*, 2024. 2

[7] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21476–21485, 2024. 2

[8] Hyeong In Choi, Sung Woo Choi, and Hwan Pyo Moon. Mathematical theory of medial axis transform. *pacific journal of mathematics*, 181(1):57–88, 1997. 4

[9] Qiyuan Feng, Gengchen Cao, Haoxiang Chen, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. A new split algorithm for 3d gaussian splatting. *arXiv preprint arXiv:2403.09143*, 2024. 5

[10] Yutao Feng, Xiang Feng, Yintong Shang, Ying Jiang, Chang Yu, Zeshun Zong, Tianjia Shao, Hongzhi Wu, Kun Zhou, Chenfanfu Jiang, et al. Gaussian splashing: Dynamic fluid synthesis with gaussian splatting. *arXiv preprint arXiv:2401.15318*, 2024. 1

[11] Yutao Feng, Yintong Shang, Xuan Li, Tianjia Shao, Chenfanfu Jiang, and Yin Yang. Pie-nerf: Physics-based interactive elastodynamics with nerf. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4450–4461, 2024. 1

[12] Lin Gao, Jie Yang, Bo-Tao Zhang, Jia-Mu Sun, Yu-Jie Yuan, Hongbo Fu, and Yu-Kun Lai. Mesh-based Gaussian Splatting for Real-time Large-scale Deformation. *arXiv preprint arXiv:2402.04796*, 2024. 2

[13] Antoine Guédon and Vincent Lepetit. Gaussian Frosting: Editable Complex Radiance Fields with Real-Time Rendering. *arXiv preprint arXiv:2403.14554*, 2024. 2

[14] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2024. 2, 8

[15] Peter Hall, David Marshall, and Ralph Martin. Adding and subtracting eigenspaces with eigenvalue decomposition and singular value decomposition. *Image and vision computing*, 20(13-14):1009–1016, 2002. 5

[16] Fursan Hamad, Shreyas Giridharan, and Christian Moormann. A penalty function method for modelling frictional contact in mpm. *Procedia Engineering*, 175:116–123, 2017. 2

[17] David Harmon, Etienne Vouga, Breannan Smith, Rasmus Tamstorf, and Eitan Grinspun. Asynchronous contact mechanics. In *ACM SIGGRAPH 2009 Papers*, New York, NY, USA, 2009. Association for Computing Machinery.

[18] Shoichi Hasegawa and Nobuaki Fujii. Real-time rigid body simulation based on volumetric penalty method. In *11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings.*, pages 326–332. IEEE, 2003. 2

[19] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24*, pages 1–11, 2024. 6

[20] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4220–4230, 2024. 2

[21] Sumit Jain, Yuting Ye, and C Karen Liu. Optimization-based interactive motion synthesis. *ACM Transactions on Graphics (TOG)*, 28(1):1–12, 2009. 2

[22] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanaes. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 7

[23] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, and Chenfanfu Jiang. VR-GS: A Physical Dynamics-Aware Interactive Gaussian Splatting System in Virtual Reality. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–1, 2024. 2, 7, 8

[24] Joaquim J Júdice and Fernanda M Pires. A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems. *Computers & operations research*, 21(5):587–596, 1994. 2

[25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023. 2, 6

[26] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1

[27] Junggon Kim and Nancy S Pollard. Direct control of simulated nonhuman characters. *IEEE Computer Graphics and Applications*, 31(4):56–65, 2011. 2

[28] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 6

[29] Lei Lan, Ran Luo, Marco Fratarcangeli, Weiwei Xu, Huamin Wang, Xiaohu Guo, Junfeng Yao, and Yin Yang. Medial Elastics: Efficient and Collision-Ready Deformation via Medial Axis Transform. *ACM Transactions on Graphics*, 39(3): 1–17, 2020. 4

[30] Zhiheng Liu, Hao Ouyang, Qiuyu Wang, Ka Leong Cheng, Jie Xiao, Kai Zhu, Nan Xue, Yu Liu, Yujun Shen, and Yang Cao. InFusion: Inpainting 3D Gaussians via Learning Depth Completion from Diffusion Prior. *arXiv preprint arXiv:2404.11613*, 2024. 2, 6

[31] Zhicheng Lu, Xiang Guo, Le Hui, Tianrui Chen, Min Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 3d geometry-aware deformable gaussian splatting for dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8900–8910, 2024. 2

[32] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. XPBD: Position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, pages 49–54, New York, NY, USA, 2016. Association for Computing Machinery. 2

[33] Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Tae-Yong Kim. Primal/dual descent methods for dynamics. In *Computer Graphics Forum*, pages 89–100. Wiley Online Library, 2020. 4

[34] B Mildenhall, PP Srinivasan, M Tancik, JT Barron, R Ramamoorthi, and R Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, 2020. 1, 6

[35] Todd S Munson, Francisco Facchinei, Michael C Ferris, Andreas Fischer, and Christian Kanzow. The semismooth algorithm for large scale complementarity problems. *INFORMS Journal on Computing*, 13(4):294–311, 2001. 2

[36] Andrei-Ionut Stefancu, Silviu-Cristian Melenciuc, and Mihai Budescu. Penalty based algorithms for frictional contact problems. *Buletinul Institutului Politehnic din Iasi. Sectia Constructii, Arhitectura*, 57(3):119, 2011. 2

[37] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4): 637–672, 2020. 6

[38] D. E. Stewart and J. C. Trinkle. An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Inelastic Collisions and Coulomb Friction. *International Journal for Numerical Methods in Engineering*, 39(15):2673–2691, 1996. 2

[39] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)*, 32(4): 1–10, 2013. 1

[40] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. *arXiv preprint arXiv:2109.07161*, 2021. 6

[41] Jie Tan, Greg Turk, and C. Karen Liu. Soft body locomotion. *ACM Transactions on Graphics*, 31(4):26:1–26:11, 2012. 2, 5, 7, 8

[42] Min Tang, Dinesh Manocha, Miguel A Otaduy, and Ruofeng Tong. Continuous penalty forces. *ACM Transactions on Graphics (TOG)*, 31(4):1–9, 2012. 2

[43] Alessandro Tasora, Dario Mangoni, Simone Benatti, and Rinaldo Garziera. Solving variational inequalities and cone complementarity problems in nonsmooth dynamics using the alternating direction method of multipliers. *International Journal for Numerical Methods in Engineering*, 122(16): 4093–4113, 2021. 2

[44] Joanna Waczyńska, Piotr Borycki, Sławomir Tadeja, Jacek Tabor, and Przemysław Spurek. GaMeS: Mesh-Based Adapting and Modification of Gaussian Splatting. *arXiv preprint arXiv:2402.01459*, 2024. 2

[45] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4389–4398, 2024. 1, 2, 3, 7, 8

[46] Hongyi Xu, Yili Zhao, and Jernej Barbič. Implicit multibody penalty-baseddistributed contact. *IEEE transactions on visualization and computer graphics*, 20(9):1266–1279, 2014. 2

[47] Shuojue Yang, Qian Li, Daiyun Shen, Bingchen Gong, Qi Dou, and Yueming Jin. Deform3dgs: Flexible deformation for fast surgical scene reconstruction with gaussian splatting. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 132–142. Springer, 2024. 2

[48] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian Grouping: Segment and Edit Anything in 3D Scenes. *arXiv preprint arXiv:2312.00732*, 2024. 2, 7

[49] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y. Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T. Freeman. PhysDreamer: Physics-based interaction with 3d objects via video generation. In *European Conference on Computer Vision*. Springer, 2024. 2

[50] Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. Drivable 3d gaussian avatars. *arXiv preprint arXiv:2311.08581*, 2023. 2