PIE-NeRF: Physics-based Interactive Elastodynamics with NeRF

Yutao Feng[1,2*]   Yintong Shang[2*]   Xuan Li[3]   Tianjia Shao[1†]   Chenfanfu Jiang[3]   Yin Yang[2]

[1]State Key Laboratory of CAD&CG, Zhejiang University
[2]University of Utah
[3]University of California, Los Angeles

fytal0n@gmail.com   yintong.shang@utah.edu   xuanli1@math.ucla.edu
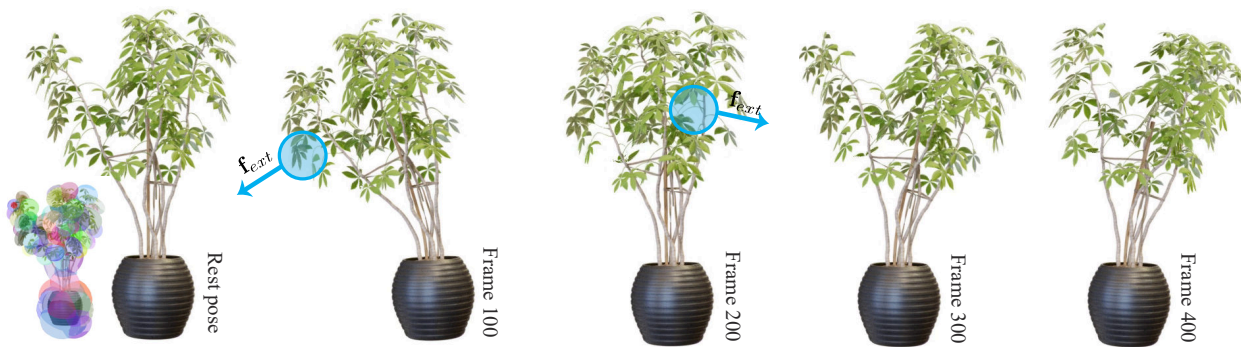tjshao@zju.edu.cn   chenfanfu.jiang@gmail.com   yangzzzy@gmail.com

Figure 1. **Swaying plant.** PIE-NeRF is an efficient and versatile pipeline that synthesizes physics-based novel motions of complex NeRF models interactively. In this example, the user interactively manipulates the plant by applying external forces with the mouse. The geometry of the plant is sampled in a meshless way, and a spatial model reduction is followed. We use 78 Q-GMLS kernels to capture the nonlinear dynamics of the plant in real-time. PIE-NeRF generates novel poses of the model from novel views in a physics-grounded way.

Abstract

We show that physics-based simulations can be seamlessly integrated with NeRF to generate high-quality elastodynamics of real-world objects. Unlike existing methods, we discretize nonlinear hyperelasticity in a meshless way, obviating the necessity for intermediate auxiliary shape proxies like a tetrahedral mesh or voxel grid. A quadratic generalized moving least square is employed to capture nonlinear dynamics and large deformation on the implicit model. Such meshless integration enables versatile simulations of complex and codimensional shapes. We adaptively place the least-square kernels according to the NeRF density field to significantly reduce the complexity of the nonlinear simulation. As a result, physically realistic animations can be conveniently synthesized using our method for a wide range of hyperelastic materials at an interactive rate. For more information, please visit our project page.

1. Introduction

Neural radiance field or NeRF [46] offers a new perspective to 3D reconstruction and representation. NeRF encodes the color, texture, and geometry information of a 3D scene with an MLP net implicitly from multi-view input photos. Its superior convenience and efficacy inspired numerous follow-up research for improved visual quality [41], faster performance [19, 85], and sparser inputs [29, 86]. The target application has also been generalized from novel view synthesis to moving scene reconstruction or shape editing [22, 59, 81, 87]. Nevertheless, complex, nonlinear, and time-coherent elastodynamic motion synthesis that is grounded on real-world physics remains less explored with the current NeRF ecosystem.

This is probably because a physical procedure is innately incompatible with implicit representations. For dynamic models (i.e., with accelerated trajectories), spatial partial differential equations (PDEs) of stress equilibriums are coupled with an ordinary differential equation (ODE) to enforce Newtonian laws of motion. One needs a good discretization for existing simulation methods e.g., the finite element

---
*Both authors contributed equally to this work
†Corresponding author

method (FEM) [92], and polygonal meshes remain the most popular choice in this regard. As a result, a dedicated meshing step is often needed [42, 87]. The computation cost is another concern. Dynamic simulation normally leads to a large sparse nonlinear system at each time step, and the simulation becomes expensive and has to be offline [42].

We propose PIE-NeRF, a NeRF-based framework that allows users to interact with the scene in a physically meaningful way, and thus generate novel deformed poses dynamically. PIE-NeRF uses a meshless discretization scheme by adaptively sampling the density field encoded with NeRF based on the magnitude of the density gradient. The efficiency of our computation comes from the meshless spatial reduction that makes the simulation independent of the sampling resolution. Specifically, PIE-NeRF employs a generalized quadratic moving least square (Q-GMLS) [44] to drive dynamics robustly even for codimensional shapes. The prior of the quadratic displacement also allows us to design a better ray-warping algorithm so that the color/texture information of the deformed model can be accurately retrieved. PIE-NeRF uses instant neural graphics primitives (NGP) [52] for faster rendering. Some noteworthy features of PIE-NeRF include:

Meshless Lagrangian dynamics in NeRF. We show the feasibility of integrating classic Lagrangian dynamics with NeRF in a meshless way. Honestly, we do not completely avoid the conversion from the implicit deep representation to an explicit form but a meshless shape proxy enhances the flexibility and simplifies the pipeline of simulation in NeRF.

Robust Q-GMLS for meshless model reduction. We carefully design PIE-NeRF by accommodating the robustness, expressivity and computation cost simultaneously. With spatial reduction based on the Voronoi partition, we enhance the expressivity of our reduced model using quadratic displacement interpolation, which captures the nonlinear deformation of the model without locking artifacts. The quadratic field also contributes an improved ray-warping algorithm during the view synthesis.

Versatile simulation at an interactive rate. Being a full physics-based pipeline, PIE-NeRF can faithfully designate material parameters such as Young's modulus and Poisson's ratio to NeRF models. It is also efficient, allowing an interactive interaction between the virtual NeRF scene and end users. Thanks to our high-order interpolation, codimensional models can also be well handled with PIE-NeRF.

2. Related work

NeRF editing. Since the advent of NeRF, many techniques tailored for implicit representation have emerged i.e., discretely estimating the deformation/displacement field for each frame [55, 56, 72, 77] or estimating the time-continuous 3D motion field [15, 17, 23, 38, 40, 61, 79].

Recently, Cao and colleagues have matched space-time features to hexplanes for improved NeRF training [11].

There also exists a wide range of NeRF editing methods for various purposes. These include semantic-driven editing [3, 13, 24, 45, 66, 73], shading-driven adjustments (like relighting and texturing) [21, 43, 64, 68, 78, 84], scene modifications (such as object addition or removal) [35, 36, 76, 83, 90], face editing [27, 31, 70, 89], physics based editing from video[25, 62], and multi-purpose editing [30, 75, 82].

Geometry editing with NeRF has also been widely investigated [32, 86, 88, 91]. They normally concern static shapes only, where the as-rigid-as-possible (ARAP) energy suffices [67] in most situations. The ARAP energy is often computed by converting neural implicit representation to some explicit forms like grid or mesh [18]. To reduce the computational overhead, some opt for coarser meshes, utilizing cage-based deformation techniques [30, 59, 81].

Point-based shape editing is a viable alternative. Chen and colleagues [12] proposed a more general editing framework with the optimized points inherent in a point-based variant of NeRF [80]. More recently, Prokudin and colleagues exploited a point-based surface derived from an implicit volumetric representation [60].

Physics-based deformable model. The concept of deformable model dates back to 1980s [71], primarily associated with physics-based models [54]. Typically, an explicit discretization is needed such as mass-spring systems [4] that were widely used in early graphics applications. FEM has become the standard for physics-based simulation [10, 34, 48, 65], wherein the deformation is usually measured by integrating over each tetrahedral or hexahedral unit, that is, the element [92]. Physics-based modeling is known to be expensive, which inspires a series of research for accelerated simulation such as model reduction [5] or GPU parallelization [74].

Meshless simulation. Meshless methods use unstructured vertices in lieu of a predefined mesh [7, 16, 39]. This modality is quite effective when the simulation domain has varying topology, such as fluids, gases, fracturing or melting. Meshless deformation has evolved to handle continuum mechanics [51]. A notable example is the shape matching [50]. With the core idea of constraining vertices' positions, shape matching paves the way to the position-based methods [47, 49, 69]. Similar to shape functions in FEM, meshless methods also need well-designed interpolation schemes [16], such as moving least squares (MLS) [51, 58] or smoothed-particle hydrodynamics SPH [1].

Due to the large volume of relevant work, we can only discuss a small fraction of excellent prior arts in this section. Nevertheless, we note that synthesizing novel dynamic motions of a NeRF scene in a physically grounded way remains less explored. This gap inspires us to develop PIE-NeRF. PIE-NeRF is a physics-based, meshless, and effi-

cient framework allowing users to interactively manipulate the NeRF scene.

3. Preliminary

To make the paper self-contained, we start with a brief review of some core techniques on which our pipeline is built. More details of our system are elaborated in § 4.

3.1. Neural radiance field

NeRF implicitly represents the geometry and appearance information of a 3D scene via a multi-layer perceptron (MLP) net. Given the camera parameters, a pixel's color on the image plane is obtained via integrating the density and color along the ray. A spatial coordinate $\mathbf{p}$ and a ray direction $\mathbf{d}$ are often encoded as a feature vector $\psi_{\mathbf{p}}, \psi_{\mathbf{d}}$ before being fed to the MLP for the prediction of density (σ) and color ($\mathbf{c}$). For instance, the vanilla NeRF [46] uses positional encoding to better tackle high-frequency information with MLPs. Our pipeline uses the instant neural graphics primitives (NGP) [52]. NGP adopts a multi-level hash-based encoding scheme and has demonstrated a strong performance in terms of both efficiency and quality.

3.2. Nonlinear elastodynamic

Following the classic Lagrangian mechanics [53], the dynamic equilibrium of a 3D model is characterized as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial L}{\partial \mathbf{q}} = \mathbf{f}_q, \qquad (1)$$

where $L = T - U$ is *Lagrangian* i.e., the difference between the kinematic energy (T) and the potential energy (U) of the system. $\mathbf{q}$ and $\dot{\mathbf{q}}$ are generalized coordinate and velocity. $\mathbf{f}_q$ is the generalized external force. Given a time integration scheme such as implicit Euler: $\mathbf{q}_{n+1} = \mathbf{q}_n + h\dot{\mathbf{q}}_{n+1}$, $\dot{\mathbf{q}}_{n+1} = \dot{\mathbf{q}}_n + h\ddot{\mathbf{q}}_{n+1}$, Eq. (1) can be reformulated a set of nonlinear equations to be solved at each time step:

$$\mathbf{M}(\mathbf{q}_{n+1} - \mathbf{q}_n - h\dot{\mathbf{q}}) = h^2\left(-\frac{\partial U}{\partial \mathbf{q}} + \mathbf{f}_q\right). \qquad (2)$$

Here, the subscript indicates the time step index, and h is the time step size. $\mathbf{q}_{n+1}$ is the unknown system coordinate to be solved, while all the kinematic variables of the previous time step such as $\mathbf{q}_n$ or $\dot{\mathbf{q}}_n$ are considered known. $-\partial U/\partial \mathbf{q}$ is the negative gradient of the potential, which embodies the internal force.

4. Our method

As shown in Fig. 2, the input of our system is a collection of images of a given 3D scene. We use NGP to encode positional and texture information and train the corresponding NeRF. Afterwards, we disperse particles into the scene.

Those particles form an unstructured point-cloud-like proxy of the 3D model of interest. They are then grouped under a Voronoi partition, and the centers of Voronoi cells house the generalized coordinate of the system ($\mathbf{q}$ in Eq. (2)). We further assign multiple integrator points (IPs) to facilitate energy integration. A quadratic generalized moving least square (Q-GMLS) strategy is used to discretize the Lagrangian equation of Eq. (1). With the help of GPU, the simulation can be done at an interactive rate or even in real-time. We leverage the deformation information at IPs to infer the rest-pose position during NGP-based NeRF rendering. Thanks to NGP, this procedure is also in real-time. Our pipeline allows users to interact with a NeRF scene by applying external forces, position constraints etc., leading to novel and physics-grounded dynamic effects. Next, we give detailed expositions of each major step of the pipeline.

4.1. Augmented Poisson disk sampling

After the NGP-NeRF is trained, we choose a meshless way to model the geometry of the 3D shape. While the underlying goal of this step is similar to other static NeRF editing systems [59, 81, 88], being mesh-free makes our pipeline more flexible and versatile. In theory, any sampling method should work as long as the sampling particles sufficiently capture the boundary of the model. For instance, one can distribute particles by simply following the evenly-spaced grid (Fig. 3, right). Doing so is similar to using a grid-based cage to approximate the shape of the model [22].

Alternatively, we design an augmented Poisson disk sampling (PDS) strategy. The original PDS requires that the distance between any two particles be larger than a threshold $\bar{r}$. Starting from an initial point, PDS then tries to fill a banded ring between $\bar{r}$ and $2\bar{r}$ with new samples i.e., see [9]. Our observation is that more particles are needed at the boundary of the shape, which coincides with a sharp density variation. To this end, we adaptively adjust the sample radius r based on the norm of the density gradient of NGP-NeRF $\|\nabla\sigma\|$ such that:

$$r = \min\left\{\bar{r}, \kappa\frac{\bar{r}}{\sqrt{\|\nabla\sigma\| + \alpha}}\right\}, \qquad (3)$$

where $\alpha = 10^{-3}$ is a small number avoiding the division-by-zero error. Eq. (3) suggests that the actual sample radius r decreases when $\|\nabla\sigma\|$ is a large quantity. The density gradient $\|\nabla\sigma\|$ can be conveniently computed by differentiating the NPG-NeRF using `AutoDiff` [57]. We discard PDS particles whose density values are less than $\epsilon = 10^{-2}$, which are visualized as pink dots in Fig. 3.

Our sampling ensures that the distance between a PDS particle at $\mathbf{x}$ and its nearest neighbor is at least $r(\mathbf{x})$, and we assign a volume of the PDS particle as:

$$V(\mathbf{x}) = \frac{4}{3}\pi r^3(\mathbf{x}). \qquad (4)$$
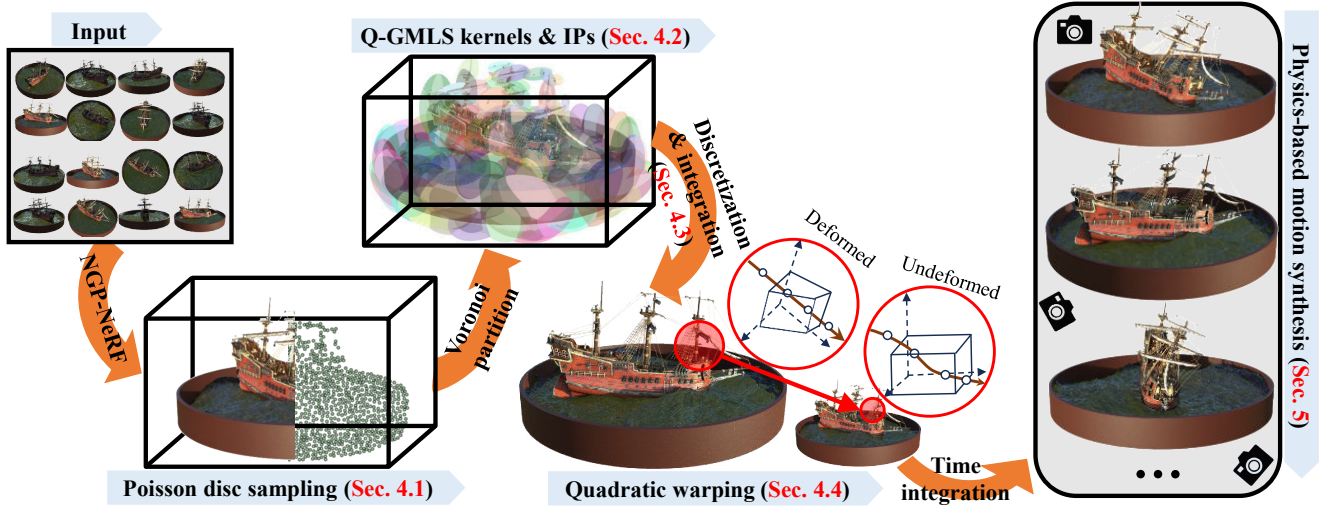
Figure 2. **Pipeline overview.** The input of PIE-NeRF is the same as other NeRF-based frameworks, which consists of a collection of images of a static scene. An adaptive Poisson disk sampling is followed to query the 3D geometry of the model, which are sparsified into n Q-GMLS kernels. Integrator points are placed over the model, including centers of Q-GMLS kernels (i.e., kernel IPs). Discretization at kernels and numerical integration at IPs enable efficient synthesis of novel and physics-based elastodynamic motions. The quadratic warping scheme helps to better retrieve the color/texture of a deformed spatial position to render the final result.
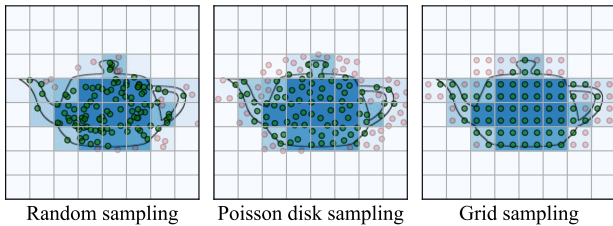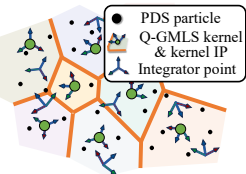

Figure 3. **Particle sampling.** Our method is compatible with most sampling algorithms – as long as particles cover the shape of the 3D model sufficiently well. In our implementation, we design a novel augmented Poisson disk sampling scheme that is fast and well captures the boundary of the model by default.

4.2. Q-GMLS kernels and integrator points

We perform a Voronoi partition [2] over PDS particles (see the inset) and use each Voronoi cell as a GMLS kernel for the body-wise displacement interpolation to reduce the computation overhead. Let Ω be the body of a 3D model in NeRF sampled by PDS particles with n GMLS kernels. The classic MLS assumes a kernel possesses an affine displacement filed: $\mathbf{A}_i \mathbf{p}(\mathbf{x}_i)$, where $\mathbf{x}_i$ is the center of the i-th kernel at the rest pose (green dots in the inset); $\mathbf{p}(\mathbf{x}) = [1, \mathbf{x}^\top]^\top$. By minimizing a displacement-based tar-

get of: $\sum_{i=1}^{n} w(\mathbf{x} - \mathbf{x}_i) \|\mathbf{A}\mathbf{p}(\mathbf{x}_i) - \mathbf{u}_i\|^2$ one can obtain:

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{u}_i N_i(\mathbf{x}). \tag{5}$$

Here $\mathbf{u}_i = \mathbf{u}(\mathbf{x}_i)$ is the displacement of i-the kernel center; $N_i(\mathbf{x}) = \mathbf{p}(\mathbf{x})^\top \mathbf{G}^{-1}(\mathbf{x})\mathbf{p}(\mathbf{x}_i)w(\mathbf{x} - \mathbf{x_i})$, for $\mathbf{G}(\mathbf{x}) = \sum_{i=1}^{n} w(\mathbf{x} - \mathbf{x}_i)\mathbf{p}(\mathbf{x}_i)\mathbf{p}(\mathbf{x}_i)^\top$, is a shape-function-like trial function; and $w(\mathbf{d}) = (1 - \|\mathbf{d}\|^2)^3$ is a MLS weighting function based on the distance between $\mathbf{x}$ and $\mathbf{x}_i$.

As the complexity of the simulation is up to n, we are in favor of using fewer kernels for faster computation. Doing so is likely to have $\mathbf{x}_i$ be colinear/coplanar, and $\mathbf{G}$ becomes singular. To improve the robustness of the kinematic interpolation of Eq. (5), GMLS takes the local deformation gradient information into account, which seeks the optimal $\mathbf{A}_i$ to minimize $\sum_{i=1}^{n} w(\mathbf{x} - \mathbf{x}_i) \|\mathbf{A}_i\mathbf{p}(\mathbf{x}_i) - \mathbf{u}_i\|^2 + \sum_{i=1}^{n} \sum_{j=1}^{3} w(\mathbf{x}_0 - \mathbf{x}_i) \|\mathbf{A}_i\mathbf{p}_{,j}(\mathbf{x}_i) - \mathbf{u}_{i,j}\|^2$. The comma here denotes the partial differentiation such that $\mathbf{u}_{i,1} = \partial\mathbf{u}_i/\partial x$, $\mathbf{u}_{i,2} = \partial\mathbf{u}_i/\partial y$, and $\mathbf{u}_{i,3} = \partial\mathbf{u}_i/\partial z$.

For thin and codimensional shapes, affine GMLS suffers from locking issues, wherein linearized shearing energy becomes orders stronger than nonlinear bending/twisting due to the interpolation error. This problem gets more serious with fewer kernels. To this end, we elevate the interpolation order, leading to quadratic GMLS or Q-GMLS, which assumes the per-kernel displacement field is quadratic. Namely, each x, y, or z component of the displacement (i.e., for $j = 1, 2, 3$ respectively) is fit by: $u_{ij} = \mathbf{x}_i^\top \mathbf{Q}_i^j \mathbf{x}_i + \mathbf{a}_i^{j\top} \mathbf{p}(\mathbf{x}_i)$. Here $\mathbf{Q}_i^j$ is a symmetric tensor, and $\mathbf{a}_i^j \in \mathbb{R}^4$ is

j-th row of $\mathbf{A}_i$. The Q-GMLS displacement interpolation can then be derived as:

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{n} \left[\mathbf{u}_i N_i + \sum_{j} \mathbf{u}_{i,j} N_i^j + \sum_{j,k} \mathbf{u}_{i,jk} N_i^{jk} \right], \quad (6)$$

for $j, k = 1, 2, 3$. Here

$$\begin{aligned} N_i(\mathbf{x}) &= \mathbf{p}^\top(\mathbf{x}) \mathbf{G}^{-1}(\mathbf{x}) \mathbf{p}(\mathbf{x}_i) w(\mathbf{x} - \mathbf{x}_i), \\ N_i^j(\mathbf{x}) &= \mathbf{p}^\top(\mathbf{x}) \mathbf{G}^{-1}(\mathbf{x}) \mathbf{p}_{,j}(\mathbf{x}_i) w(\mathbf{x} - \mathbf{x}_i), \quad (7) \\ N_i^{jk}(\mathbf{x}) &= \mathbf{p}^\top(\mathbf{x}) \mathbf{G}^{-1}(\mathbf{x}) \mathbf{p}_{,jk}(\mathbf{x}_i) w(\mathbf{x} - \mathbf{x}_i) \end{aligned}$$

only depend on the rest-shape position $\mathbf{x}$, and

$$\begin{aligned} \mathbf{G}(\mathbf{x}) = \sum_{i=1}^{n} w(\mathbf{x} - \mathbf{x}_i) \Big[&\mathbf{p}(\mathbf{x}_i) \mathbf{p}^\top(\mathbf{x}_i) \\ &+ \sum_{j} \mathbf{p}_{,j}(\mathbf{x}_i) \mathbf{p}_{,j}^\top(\mathbf{x}_i) + \sum_{j,k} \mathbf{p}_{,jk}(\mathbf{x}_i) \mathbf{p}_{,jk}^\top(\mathbf{x}_i) \Big]. \quad (8) \end{aligned}$$

It is convenient to re-organize Eq. (6) as:

$$\mathbf{u}(\mathbf{x}) = \mathbf{J}(\mathbf{x}) \mathbf{q}, \quad (9)$$

such that $\mathbf{J} = [N_1 \mathbf{I}, N_1^1 \mathbf{I}, N_1^2 \mathbf{I}, ..., N_1^{11} \mathbf{I}, ...] \in \mathbb{R}^{3 \times 30n}$ and $\mathbf{q} = [\mathbf{u}_1^\top, \mathbf{u}_{1,1}^\top, \mathbf{u}_{1,2}^\top, ..., \mathbf{u}_{1,11}^\top, ...]^\top \in \mathbb{R}^{30n}$ are the Jacobi matrix and generalized coordinate (i.e., in Eq. (1)). Thus the generalized external force is computed via: $\mathbf{f}_q = \mathbf{J}^\top \mathbf{f}_{ext}$.

4.3. Energy integration

The total kinematic and potential energies of the model are:

$$T = \frac{1}{2} \int_\Omega \rho(\mathbf{x}) \dot{\mathbf{x}}^\top \dot{\mathbf{x}} d\Omega, \text{ and } U = \int_\Omega \Psi(\mathbf{x}) d\Omega. \quad (10)$$

We want to avoid integrating over all the PDS particles. Therefore, our system includes another set of integrator points or IPs. Conceptually, IPs are similar to the quadrature points used in numerical integration [20], which allows us to substantially reduce the computational cost of full integrals in Eq. (10). In addition to the centers of Q-GMLS kernels i.e., kernel IPs, we add more IPs aiming to approximate Eq. (10) with high accuracy. Specifically, we initialize new IPs at the PDS particle which is the most distant from existing IPs to sample remote T and V values. This strategy however tends to favor PDS particles at the model's boundary. As a result, we apply a few Lloyd relaxations [14] to new IPs while keeping kernel IPs fixed. The total number of IPs is bigger than the number of Q-GMLS kernels but they are of the same order, and we use $\mathcal{I}$ to denote the set of all the IPs.

4.4. Per-IP integration
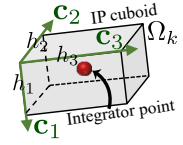
We envision each IP as a small elastic cuboid Ω_k (see the inset) with three edges being $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$ whose lengths are h_1, h_2, h_3 respectively. Its covariance matrix can be computed as:

$\mathbf{C} = \sum_j V(\mathbf{x}_j) \mathbf{x}_j \mathbf{x}_j^\top$, where the summation carries over K-nearest PDS particles whose rest positions are $\mathbf{x}_j$. Being a symmetric matrix, $\mathbf{C}$ always has three real non-negative eigen values namely, λ_1, λ_2, and λ_3. Note that coplanar geometry around an IP can make $\mathbf{C}$ singular. It is fine for numerical integration, suggesting the strains along certain directions are zero. We then set the ratio among h_i to be the same as $\sqrt{\lambda_i}$ (i.e., $h_1 : h_2 : h_3$ equals $\sqrt{\lambda_1} : \sqrt{\lambda_2} : \sqrt{\lambda_3}$) while requiring $\Pi_i h_i = \sum_j V(\mathbf{x}_j)$. Those two constraints allow us to compute h_1, h_2, and h_3 while $\mathbf{c}_i$ are the corresponding eigen vectors.

The total kinematic energy can now be approximated as:

$$T = \frac{1}{2} \int_\Omega \rho(\mathbf{x}) \dot{\mathbf{x}}^\top \dot{\mathbf{x}} d\Omega = \frac{1}{2} \dot{\mathbf{q}}^\top \left(\int_\Omega \rho \mathbf{J}^\top \mathbf{J} d\Omega \right) \dot{\mathbf{q}}$$

$$\approx \frac{1}{2} \dot{\mathbf{q}}^\top \left[\sum_{\mathbf{x}_k \in \mathcal{I}} \rho V_k \mathbf{J}^\top(\mathbf{x}_k) \mathbf{J}(\mathbf{x}_k) \right] \dot{\mathbf{q}}, \quad (11)$$

and $\mathbf{M} = \sum_{\mathbf{x}_k \in \mathcal{I}} \rho V_k \mathbf{J}^\top(\mathbf{x}_k) \mathbf{J}(\mathbf{x}_k)$ is the mass matrix. Note that $V_k = h_1 h_2 h_3$ is the estimated volume of the IP cuboid, not the volume of the PDS particle. ρ is the density of the 3D model, which should not be confused with σ.

Integrating the potential energy U is handled in a similar way. Under the assumption of hyperelasticity, the energy density $\Psi(\mathbf{x})$ depends on the deformation gradient at $\mathbf{x}$: $\mathbf{F} = \nabla \mathbf{u}(\mathbf{x}) + \mathbf{I} \in \mathbb{R}^{3 \times 3}$. According to Eq. (9), the deformation gradient at the k-th IP is:

$$\mathbf{F}(\mathbf{x}_k) = \mathbf{F}_k = \mathbf{q} \cdot \nabla \mathbf{J}^\top(\mathbf{x}_k) + \mathbf{I} = \mathbf{q} \cdot \nabla \mathbf{J}_k^\top + \mathbf{I}. \quad (12)$$

$\mathbf{J}_k \in \mathbb{R}^{3 \times 30n}$ is the Jacobi corresponding to the IP, and $\nabla \mathbf{J}_k^\top \in \mathbb{R}^{30n \times 3 \times 3}$ is a third tensor. The potential accumulated at the IP is estimated by integrating over its cuboid (Ω_k) assuming the IP lies at the center:

$$U_k = \int_{\Omega_k} \Psi(\mathbf{F}(\mathbf{h})) = \int_{-\frac{h_1}{2}}^{\frac{h_1}{2}} \int_{-\frac{h_2}{2}}^{\frac{h_2}{2}} \int_{-\frac{h_3}{2}}^{\frac{h_3}{2}} \Psi(\mathbf{F}(\mathbf{h})). \quad (13)$$

Here, $\mathbf{h}$ is the local coordinate spanning Ω_k, and $\mathbf{x}_k$ aligns with $\mathbf{h} = 0$. When $\mathbf{h} \neq 0$, we first-order approximate $\mathbf{F}$ as:

$$\mathbf{F}(\mathbf{h}) \approx \mathbf{F}(0) + \nabla \mathbf{F}(0) \cdot \mathbf{h} = \mathbf{F}(\mathbf{x}_k) + \nabla \mathbf{F}(\mathbf{x}_k) \cdot \mathbf{h}. \quad (14)$$

This makes sense because Q-GMLS assumes $\mathbf{u}$ is quadratic, which has a linearly-vary deformation gradient. Therefore, the approximate in Eq. (14) should be exact. $\nabla \mathbf{F}$ can be computed by differentiating Eq. (12):

$$\nabla \mathbf{F}(\mathbf{x}_k) = \mathbf{q} \cdot \nabla^2 \mathbf{J}_k^\top = \mathbf{H}_k \cdot \mathbf{q}. \quad (15)$$
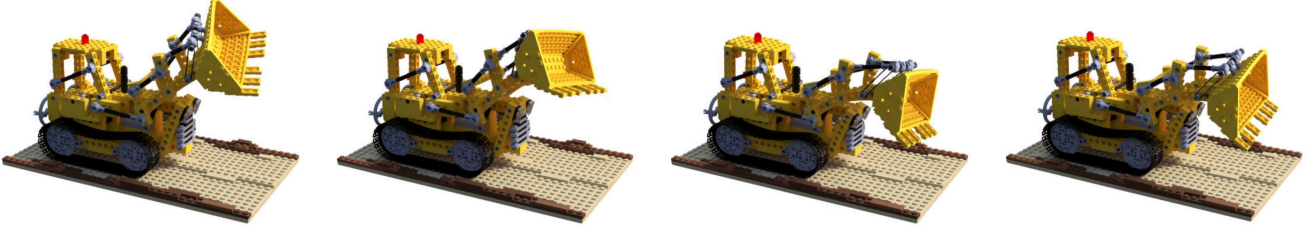
Figure 4. **Elastically deforming excavator.** The excavator is a standard benchmark for NeRF-based frameworks. We use this classic model to showcase the capability of PIE-NeRF, which generates interesting and novel dynamic effects in real time.

Here $\mathbf{H} = \partial \nabla \mathbf{J}^\top / \partial \mathbf{x}$ is a fourth tensor. This computation boils down to evaluating first- and second-derivatives of N_i, N_i^j, and N_i^{jk}, and can be pre-computed per Eq. (7).

Given an elastic material model $\Psi(\mathbf{F})$, the total potential can then be computed via:

$$U \approx \sum_{\mathbf{x}_k \in \mathcal{I}} \int_{\Omega_k} \Psi\left(\mathbf{F}(0) + \mathbf{H}_k : (\mathbf{q}\mathbf{h}^\top)\right) d\Omega_k. \quad (16)$$

The actual integration computation relies on the specific formulation of $\Psi(\mathbf{F})$. Please refer to the supplementary document for detailed derivations of some commonly-used material models such as ARAP and Neo-Hookean.

4.5. System assembly and solve

With energy integrals, we can assemble Eq. (2). The generalized internal force is $\mathbf{f}_{int} = -\partial U / \partial \mathbf{q}$, and it can be conveniently computed by the chain rule:

$$\mathbf{f}_{int} = -\frac{\partial \Psi}{\partial \mathbf{F}} : \frac{\partial \mathbf{F}}{\partial \mathbf{q}} = -\mathbf{P} : \nabla \mathbf{J}, \quad (17)$$

where $\mathbf{P}$ is the first Piola-Kirchhoff stress. This is a $30n$-dimension dense system as all Q-GMLS kernels have global influences. We use Newton's method to solve this system iteratively. Each Newton iteration solves a linearized problem for an incremental improvement $\Delta \mathbf{q}$:

$$\left(\mathbf{M} + h^2 \frac{\partial \mathbf{f}_{int}}{\partial \mathbf{q}}\right) \Delta \mathbf{q} = \mathbf{M}(\mathbf{q}_n + h\dot{\mathbf{q}}_n) + h^2 \mathbf{J}^\top \mathbf{f}_{ext}, \quad (18)$$

where $\partial \mathbf{f}_{int} / \partial \mathbf{q}$ is the second differentiation of the total potential U known as the tangent stiffness matrix. $\mathbf{f}_{ext}$ is the external forces applied to the model. It is projected to the Q-GMLS kinematic space by left multiplying $\mathbf{J}^\top$.

4.6. NeRF rendering using quadratic warping

After the deformed model geometry is computed, we leverage the NGP-NeRF that is built for its rest shape to synthesize *both novel views and novel deformations*. Whenever we query the NGP-NeRF for a deformed location $\tilde{\mathbf{x}}$ along a ray, we warp this position to its rest configuration $\mathbf{x}$, ideally through $\mathbf{x} = \tilde{\mathbf{x}} - \mathbf{u}(\mathbf{x})$. Unfortunately as $\mathbf{x}$ is unknown

here, we cannot obtain $\mathbf{u}(\mathbf{x})$ directly with Eq. (9). Instead, we approximate $\mathbf{u}(\mathbf{x})$ based on the displacements at nearby (deformed) IPs. The general rationale is that if $\tilde{\mathbf{x}}$ is sufficiently close to an IP, we can Taylor expand the IP's displacement to estimate $\mathbf{u}(\mathbf{x})$. As IPs are sparse, it is possible that $\tilde{\mathbf{x}}$ is not particularly close to one IP. In this case, we find three nearest IPs and average Taylor expansions at those IPs based on the inverse distance weight.

For the IP at $\mathbf{x}_k$, we have:

$$\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_k = \mathbf{u}(\mathbf{x}) - \mathbf{u}(\mathbf{x}_k) \approx \nabla \mathbf{u}(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k)$$
$$+ \frac{1}{2}\left(\nabla \mathbf{F}(\mathbf{x}_k) \cdot (\mathbf{x} - \mathbf{x}_k)\right) \cdot (\mathbf{x} - \mathbf{x}_k). \quad (19)$$

We can then compute $\mathbf{x}$ via solving a nonlinear system of:

$$\mathbf{A}(\mathbf{x})(\mathbf{x} - \mathbf{x}_k) = \mathbf{b}, \quad (20)$$

where

$$\mathbf{A}(\mathbf{x}) = \nabla \mathbf{u}(\mathbf{x}_k) + \frac{1}{2}\nabla \mathbf{F}(\mathbf{x}_k) \cdot (\mathbf{x} - \mathbf{x}_k),$$
$$\mathbf{b} = \tilde{\mathbf{x}} - \tilde{\mathbf{x}}_k. \quad (21)$$

While the analytic solution of Eq. (20) can be derived, we find Newton's method starting from the guess of $\mathbf{x} = \mathbf{x}_k$ is effective. The system converges within tens of iterations, and each iteration only solves a 3 by 3 linear system.

This strategy of quadratic warping fully exploits the prior of $\mathbf{u}$ being the quadratic displacement field. If we only use the first-order Taylor expansion to estimate the undeformed position of $\tilde{\mathbf{x}}$, as chosen in most existing NeRF editing systems [59, 81], visual artifacts can be observed under large deformations. Examples of such failure cases are provided in the supplementary materials.

5. Experiments

We implemented PIE-NeRF pipeline using `Python` and `C++`. The simulation module was based on `CUDA`. In addition, we used `PyTorch` [28] and `Taichi` [26] to implement a modified instant-NGP [52] for ray warping (§ 4.6). Our hardware platform is a desktop computer equipped with an `Intel i7-12700F` CPU and an `NVIDIA 3090` GPU.

Figure 5. **Interactive NeRF deformation.** We developed an intuitive UI for users to interact with NeRF scenes like applying external forces and position constraints. Q-GMLS kernels can also be set adaptively to capture local dynamics as highlighted.

Datasets We evaluate PIE-NeRF with several NeRF scenes. In addition to original NeRF datasets, we utilize BlenderNeRF [63] to synthesize additional scenes including codimensional objects. We used 100 multi-view images for each scene as inputs for our NGP-NeRF training.

5.1. Interactive and dynamic NeRF simulation

PIE-NeRF formulates nonlinear dynamics of NeRF models with the generalized coordinate and Lagrangian equations (i.e., Eq. (1)), which makes the computation independent of the PDS sampling resolution. We find that a few dozen Q-GMLS kernels are often sufficient to model complex models. The corresponding computation is lightweight and can be processed in real-time on the GPU (see Fig. 4). Therefore, interactive physics-based manipulation of the NeRF scene becomes possible. To this end, we also implemented a user-friendly interface as shown in Fig. 5, left. With the interface, users can intuitively apply external forces to the model, and observe the resulting novel motions interactively. The users have full control over the trade-off between visual richness and the efficiency of the simulation. For instance, one can create a dedicated kernel to capture local dynamics at specific foliage of the plant (Fig. 5, right).

5.2. Physics-grounded pose synthesis

Being a physics-based framework, PIE-NeRF is able to model any nonlinear hyperelastic materials to match real-world observations. This enhances existing NeRF editing systems, which are mostly based on geometry-based heuristic energy models like ARAP. Fig. 6 reports a comparison using the Neo-Hookean material [8] and ARAP to compress chocolate jelly with NeRF. The energy density of Neo-Hookean material is:

$$\Psi = \frac{\mu}{2} \left(I_C - 3 \right) - \mu \log J + \frac{\lambda}{2} \log^2 J, \qquad (22)$$

where μ, λ are material parameters (a.k.a Lamé coefficients); $I_C = \mathrm{tr}(\mathbf{F}^\top \mathbf{F})$; and $J = \det(\mathbf{F})$. The log barrier
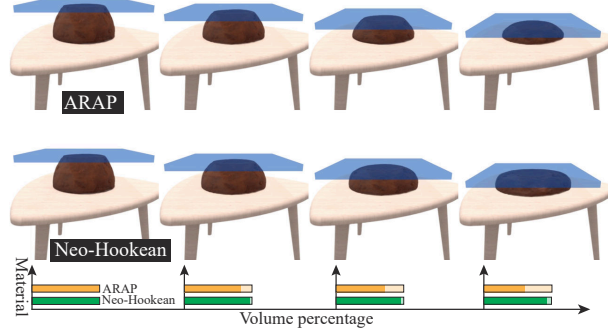

Figure 6. **Volume preservation test.** PIE-NeRF is capable of incorporating any real-world material models. In this example, we apply a position constraint to compress a chocolate jelly. ARAP energy, widely used in exiting NeRF editing systems [87], collapses unnaturally. With Neo-Hookean energy, PIE-NeRF can better synthesize this procedure.
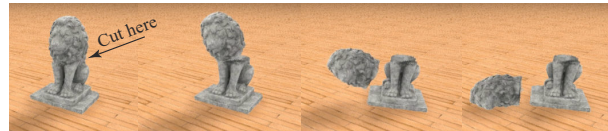

Figure 7. **Topology change and shadows.** We edit kernel weights to cut the sculpture, which then falls on the floor. We then compute the depth map from NGP-NeRF and generate the moving shadow using shadow maps.

$\log J$ in Neo-Hookean energy strongly preserves the volume of the object. This feature is clearly demonstrated in Fig. 6. As the compression rate increases, ARAP jelly (on the top) loses nearly 40% of the original volume (visualized as bar graphs at the bottom in Fig. 6).

Handle topology change. Being meshless makes PIE-NeRF less sensitive to topology changes. As shown in Fig. 7, we cut the NeRF sculpture by modifying Q-GMLS weight functions. Besides that, there is no need for extra safeguards for dealing with the change of the mesh connectivity and resolution at the cutting area. In this example, we extract the depth image with NGP-NeRF, from which a shadow map can be generated for shadow synthesis.

Comparison with ground truth. We employ FEM as the ground truth for comparison with our method. We generate multi-view images from the mesh rendered with same rendering settings to ground truth as the dataset for PIE-NeRF. Subsequently, dynamic results are produced using the same boundary conditions, physical parameters, and external forces. Our implementation of FEM involves tetrahedralizing the mesh and applying Newton's method to solve the dynamic system on tetrahedral mesh. The comparison results, as illustrated in Fig. 8, demonstrate that our method is nearly same to the ground truth. Thanks to the Q-GMLS, the number of kernels in our approach is significantly lower than the number of vertices in FEM, and the number of IPs
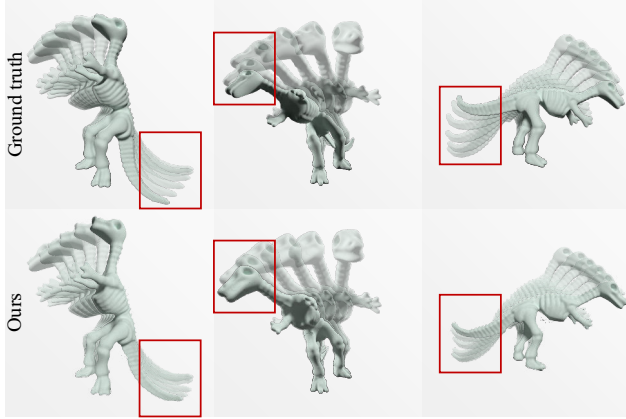
Figure 8. **Comparison with ground truth.** We generate ground-truth results by simulating the tetrahedral mesh with FEM. We can see that our results are quite similar to the ground truth, despite minor differences highlighted.
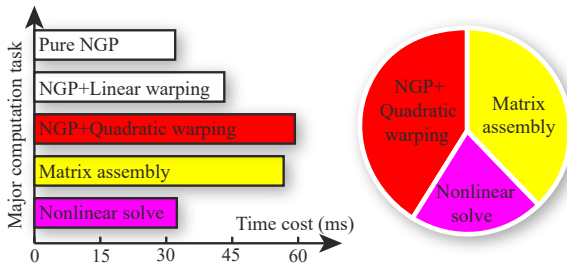

Figure 9. **Time breakdown.** There are three major steps for PIE-NeRF, namely matrix assembly, solve, and warping. Thanks to Q-GMLS-based reduction, they are all manageable making the simulation interactive. The percentage of each step is visualized as a pie diagram on the right.

is also less than the number of tetrahedra in FEM.

Comparison with PAC-NeRF. PAC-NeRF [37] is a recent contribution also aiming to combine physical models with NeRF-based representations. The underlying numerical solver, on the other hand, is based on the material point method (MPM) [6], a hybrid method that uses both particles and grids. While MPM excels in handling complicated, multi-phase physics, it does not synergize well with NeRF-based rendering. Specifically, under large deformation, PAC-NeRF fails to map material points back to their rest-pose positions accurately due to excessive interpolation smoothing between particles and grid cells, which leads to over-blurred results at local fine shapes. We show a side-by-side comparison in Fig. 10.

5.3. Time performance

Fig. 9 reports a breakdown of the run time performance of our PIE-NeRF pipeline for Fig. 5. Three major tasks for the simulation at the runtime are matrix assembly, nonlinear solve, and quadratic warping. The matrix assembly re-

Figure 10. **PIE-NeRF vs PAC-NeRF.** PAC-NeRF [37] is a closely relevant competitor. A major limitation of PAC-NeRF is the rendering. It is less intuitive to infer the right color/texture information under large deformation using MPM. PIE-NeRF overcomes this limitation with ease. Using implicit time integration, PIE-NeRF runs much faster than PAC-NeRF (two orders in these examples).

quires an integral over all the IPs, leading to a dense $30n$ by $30n$ system. We use Cholesky factorization to solve the resulting Newton system (Eq. (18)). In general, a couple of iterations will converge the system so that we forward to the next time step. As shown in the figure, quadratic warping used in PIE-NeRF is slightly more expensive than linear warping. However, these additional expenses yield significantly improved visual results in general, as detailed in the supplementary material.

6. Conclusion

PIE-NeRF🍕 is a physics-NeRF simulation pipeline. It is directly based on PDS particles sampled over the NeRF scene and applies a Q-GMLS model reduction to lower the computational overhead of the simulation. As a result, PIE-NeRF faithfully characterizes various real-world material models. Its meshless representation makes the simulation flexible, and topology changes can be well accommodated. The quadratic interpolation scheme is not only helpful in tackling thin-geometry models but also leads to better image synthesis with NGP-NeRF. We hope PIE-NeRF could contribute new ingredients to the existing NeRF ecosystem. Based on PIE-NerF, it is possible to integrate more (better and faster) simulation and graphics techniques to deep 3D vision applications to imbue vivid, realistic, real-time physics into static or dynamic environments. Along this exciting endeavor, we will also explore other opportunities, such as Gaussian splatting-based techniques [33], and ultimately reach *what you see is what you simulate, WS²*.

References

[1] Carla Antoci, Mario Gallati, and Stefano Sibilla. Numerical simulation of fluid–structure interaction by sph. *Computers & structures*, 85(11-14):879–890, 2007. 2

[2] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991. 4

[3] Chong Bao, Yinda Zhang, Bangbang Yang, Tianxing Fan, Zesong Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. 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*, pages 20919–20929, 2023. 2

[4] D BARAFF. Large steps in cloth simulation. In *SIGGRAPH'98 Proceedings*, pages 43–54, 1998. 2

[5] Jernej Barbič and Doug L James. Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM transactions on graphics (TOG)*, 24(3):982–990, 2005. 2

[6] S. Bardenhagen and Edward Kober. The generalized interpolation material point method. *CMES - Computer Modeling in Engineering and Sciences*, 5, 06 2004. 8

[7] Ted Belytschko, Yury Krongauz, Daniel Organ, Mark Fleming, and Petr Krysl. Meshless methods: an overview and recent developments. *Computer methods in applied mechanics and engineering*, 139(1-4):3–47, 1996. 2

[8] Javier Bonet and Richard D Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press, 1997. 7

[9] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*, SIGGRAPH '07, page 22–es, New York, NY, USA, 2007. Association for Computing Machinery. 3

[10] Morten Bro-Nielsen and Stephane Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Computer graphics forum*, volume 15, pages 57–66. Wiley Online Library, 1996. 2

[11] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 2

[12] Jun-Kun Chen, Jipeng Lyu, and Yu-Xiong Wang. Neuraleditor: Editing neural radiance fields via manipulating point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12439–12448, 2023. 2

[13] Jiahua Dong and Yu-Xiong Wang. Vica-nerf: View-consistency-aware 3d editing of neural radiance fields. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 2

[14] Qiang Du, Maria Emelianenko, and Lili Ju. Convergence of the lloyd algorithm for computing centroidal voronoi tessellations. *SIAM journal on numerical analysis*, 44(1):102–119, 2006. 5

[15] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14304–14314. IEEE Computer Society, 2021. 2

[16] Thomas-Peter Fries, Hermann G Matthies, et al. Classification and overview of meshfree methods. *Department of Mathematics and Computer Science, Technical University of Braunschweig*, 2003. 2

[17] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021. 2

[18] Stephan J Garbin, Marek Kowalski, Virginia Estellers, Stanislaw Szymanowicz, Shideh Rezaeifar, Jingjing Shen, Matthew Johnson, and Julien Valentin. Voltemorph: Real-time, controllable and generalisable animation of volumetric representations. *arXiv preprint arXiv:2208.00949*, 2022. 2

[19] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14346–14355, 2021. 1

[20] Thomas Gerstner and Michael Griebel. Numerical integration using sparse grids. *Numerical algorithms*, 18(3-4):209–232, 1998. 5

[21] Bingchen Gong, Yuehao Wang, Xiaoguang Han, and Qi Dou. Recolornerf: Layer decomposed radiance field for efficient color editing of 3d scenes. *arXiv preprint arXiv:2301.07958*, 2023. 2

[22] Xiang Guo, Guanying Chen, Yuchao Dai, Xiaoqing Ye, Jiadai Sun, Xiao Tan, and Errui Ding. Neural deformable voxel grid for fast optimization of dynamic view synthesis. In *Proceedings of the Asian Conference on Computer Vision*, pages 3757–3775, 2022. 1, 3

[23] Xiang Guo, Jiadai Sun, Yuchao Dai, Guanying Chen, Xiaoqing Ye, Xiao Tan, Errui Ding, Yumeng Zhang, and Jingdong Wang. Forward flow for novel view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16022–16033, 2023. 2

[24] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. *arXiv preprint arXiv:2303.12789*, 2023. 2

[25] Florian Hofherr, Lukas Koestler, Florian Bernard, and Daniel Cremers. Neural implicit representations for physical parameter inference from a single video. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2093–2103, 2023. 2

[26] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019. 6

[27] Sungwon Hwang, Junha Hyung, Daejin Kim, Min-Jung Kim, and Jaegul Choo. Faceclipnerf: Text-driven 3d face manipulation using deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3469–3479, 2023. 2

[28] Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. Pytorch. *Programming with TensorFlow: Solution for Edge Computing Applications*, pages 87–104, 2021. 6

[29] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis.

In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5885–5894, 2021. 1

[30] Clément Jambon, Bernhard Kerbl, Georgios Kopanas, Stavros Diolatzis, George Drettakis, and Thomas Leimkühler. Nerfshop: Interactive editing of neural radiance fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(1), 2023. 2

[31] Kaiwen Jiang, Shu-Yu Chen, Feng-Lin Liu, Hongbo Fu, and Lin Gao. Nerffaceediting: Disentangled face editing in neural radiance fields. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 2

[32] Kacper Kania, Kwang Moo Yi, Marek Kowalski, Tomasz Trzciński, and Andrea Tagliasacchi. Conerf: Controllable neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18623–18632, 2022. 2

[33] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023. 8

[34] Theodore Kim and David Eberle. Dynamic deformables: implementation and production practicalities (now with code!). In *ACM SIGGRAPH 2022 Courses*, pages 1–259. 2022. 2

[35] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *Advances in Neural Information Processing Systems*, 35:23311–23330, 2022. 2

[36] Verica Lazova, Vladimir Guzov, Kyle Olszewski, Sergey Tulyakov, and Gerard Pons-Moll. Control-nerf: Editable feature volumes for scene rendering and manipulation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4340–4350, 2023. 2

[37] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *arXiv preprint arXiv:2303.05512*, 2023. 8

[38] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 2

[39] Gui-Rong Liu and D Karamanlidis. Mesh free methods: moving beyond the finite element method. *Appl. Mech. Rev.*, 56(2):B17–B18, 2003. 2

[40] Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Devrf: Fast deformable voxel radiance fields for dynamic scenes. *Advances in Neural Information Processing Systems*, 35:36762–36775, 2022. 2

[41] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 1

[42] Ruiyang Liu, Jinxu Xiang, Bowen Zhao, Ran Zhang, Jingyi Yu, and Changxi Zheng. Neural impostor: Editing neural radiance fields with explicit shape manipulation. *arXiv preprint arXiv:2310.05391*, 2023. 2

[43] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5773–5783, 2021. 2

[44] Sebastian Martin, Peter Kaufmann, Mario Botsch, Eitan Grinspun, and Markus Gross. Unified simulation of elastic rods, shells, and solids. *ACM Transactions on Graphics (TOG)*, 29(4):1–10, 2010. 2

[45] Aryan Mikaeili, Or Perel, Mehdi Safaee, Daniel Cohen-Or, and Ali Mahdavi-Amiri. Sked: Sketch-guided text-based 3d editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14607–14619, 2023. 2

[46] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *The European Conference on Computer Vision (ECCV)*, 2020. 1, 3

[47] Matthias Müller and Nuttapong Chentanez. Solid simulation with oriented particles. In *ACM SIGGRAPH 2011 papers*, pages 1–10. 2011. 2

[48] Matthias Müller and Markus H Gross. Interactive virtual materials. In *Graphics interface*, volume 2004, pages 239–246, 2004. 2

[49] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007. 2

[50] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless deformations based on shape matching. *ACM transactions on graphics (TOG)*, 24(3):471–478, 2005. 2

[51] Matthias Müller, Richard Keiser, Andrew Nealen, Mark Pauly, Markus Gross, and Marc Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151, 2004. 2

[52] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4), jul 2022. 2, 3, 6

[53] Richard M Murray. Nonlinear control of mechanical systems: A lagrangian perspective. *Annual Reviews in Control*, 21:31–42, 1997. 3

[54] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. In *Computer graphics forum*, volume 25, pages 809–836. Wiley Online Library, 2006. 2

[55] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 2

[56] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021. 2

[57] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Al-

ban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 3

[58] Mark Pauly, Richard Keiser, Leif P Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics (TOG)*, 22(3):641–650, 2003. 2

[59] Yicong Peng, Yichao Yan, Shengqi Liu, Yuhao Cheng, Shanyan Guan, Bowen Pan, Guangtao Zhai, and Xiaokang Yang. Cagenerf: Cage-based neural radiance field for generalized 3d deformation and animation. *Advances in Neural Information Processing Systems*, 35:31402–31415, 2022. 1, 2, 3, 6

[60] Sergey Prokudin, Qianli Ma, Maxime Raafat, Julien Valentin, and Siyu Tang. Dynamic point fields. *arXiv preprint arXiv:2304.02626*, 2023. 2

[61] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 2

[62] Yi-Ling Qiao, Alexander Gao, and Ming C. Lin. Neuphysics: Editable neural geometry and physics from monocular videos. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 2

[63] Maxime Raafat. BlenderNeRF, May 2023. 7

[64] Viktor Rudnev, Mohamed Elgharib, William Smith, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. Nerf for outdoor scene relighting. In *European Conference on Computer Vision*, pages 615–631. Springer, 2022. 2

[65] Eftychios Sifakis and Jernej Barbic. Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction. In *Acm siggraph 2012 courses*, pages 1–50. 2012. 2

[66] Hyeonseop Song, Seokhun Choi, Hoseok Do, Chul Lee, and Taehyeong Kim. Blending-nerf: Text-driven localized editing in neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14383–14393, 2023. 2

[67] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116. Citeseer, 2007. 2

[68] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7495–7504, 2021. 2

[69] Denis Steinemann, Miguel A Otaduy, and Markus Gross. Fast adaptive shape matching deformations. In *Proceedings of the 2008 ACM SIGGRAPH/eurographics symposium on computer animation*, pages 87–94, 2008. 2

[70] Jingxiang Sun, Xuan Wang, Yong Zhang, Xiaoyu Li, Qi Zhang, Yebin Liu, and Jue Wang. Fenerf: Face editing in neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7672–7682, 2022. 2

[71] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, 1987. 2

[72] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Nonrigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12959–12970, 2021. 2

[73] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 2

[74] Huamin Wang and Yin Yang. Descent methods for elastic body simulation on the gpu. *ACM Transactions on Graphics (TOG)*, 35(6):1–10, 2016. 2

[75] Xiangyu Wang, Jingsen Zhu, Qi Ye, Yuchi Huo, Yunlong Ran, Zhihua Zhong, and Jiming Chen. Seal-3d: Interactive pixel-level editing for neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17683–17693, 2023. 2

[76] Silvan Weder, Guillermo Garcia-Hernando, Aron Monszpart, Marc Pollefeys, Gabriel J Brostow, Michael Firman, and Sara Vicente. Removing objects from neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16528–16538, 2023. 2

[77] Chung-Yi Weng, Brian Curless, Pratul P Srinivasan, Jonathan T Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern Recognition*, pages 16210–16220, 2022. 2

[78] Qiling Wu, Jianchao Tan, and Kun Xu. Palettenerf: Palette-based color editing for nerfs. *arXiv preprint arXiv:2212.12871*, 2022. 2

[79] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9421–9431, 2021. 2

[80] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 2

[81] Tianhan Xu and Tatsuya Harada. Deforming radiance fields with cages. In *European Conference on Computer Vision*, pages 159–175. Springer, 2022. 1, 2, 3, 6

[82] Bangbang Yang, Chong Bao, Junyi Zeng, Hujun Bao, Yinda Zhang, Zhaopeng Cui, and Guofeng Zhang. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *European Conference on Computer Vision*, pages 597–614. Springer, 2022. 2

[83] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788, 2021. 2

[84] Weicai Ye, Shuo Chen, Chong Bao, Hujun Bao, Marc Polle-

feys, Zhaopeng Cui, and Guofeng Zhang. Intrinsicnerf: Learning intrinsic neural radiance fields for editable novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 339–351, 2023. 2

[85] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 1

[86] Yu-Jie Yuan, Yu-Kun Lai, Yi-Hua Huang, Leif Kobbelt, and Lin Gao. Neural radiance fields from sparse rgb-d images for high-quality view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1, 2

[87] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18353–18364, 2022. 1, 2, 7

[88] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, Leif Kobbelt, and Lin Gao. Interactive nerf geometry editing with shape priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 2, 3

[89] Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. Fdnerf: Few-shot dynamic neural radiance fields for face reconstruction and expression editing. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 2

[90] Jiakai Zhang, Xinhang Liu, Xinyi Ye, Fuqiang Zhao, Yanshun Zhang, Minye Wu, Yingliang Zhang, Lan Xu, and Jingyi Yu. Editable free-viewpoint video using a layered neural representation. *ACM Transactions on Graphics (TOG)*, 40(4):1–18, 2021. 2

[91] Chengwei Zheng, Wenbin Lin, and Feng Xu. Editablenerf: Editing topologically varying neural radiance fields by key points. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8317–8327, 2023. 2

[92] Olek C Zienkiewicz, Robert L Taylor, and Jian Z Zhu. *The finite element method: its basis and fundamentals*. Elsevier, 2005. 2