

ElastoGen: 4D Generative Elastodynamics

Yutao Feng^{1,2*}, Yintong Shang^{2*}, Xiang Feng^{1,2*}, Lei Lan^{1,2}, Shandian Zhe², Tianjia Shao^{1†},
Hongzhi Wu¹, Kun Zhou¹, Chenfanfu Jiang³, Yin Yang²

¹State Key Laboratory of CAD&CG, Zhejiang University

²University of Utah

³UCLA

Abstract

We present ElastoGen, a *knowledge-driven* AI model that generates physically accurate 4D elastodynamics. Unlike deep models that learn from video- or image-based observations, ElastoGen leverages the principles of physics and learns from established mathematical and optimization procedures. The core idea of ElastoGen is converting the differential equation, corresponding to the nonlinear force equilibrium, into a series of iterative local convolution-like operations, which naturally fit deep architectures. We carefully build our network module following this overarching design philosophy. ElastoGen is much more lightweight in terms of both training requirements and network scale than deep generative models. Because of its alignment with actual physical procedures, ElastoGen efficiently generates accurate dynamics for a wide range of hyperelastic materials and can be easily integrated with upstream and downstream deep modules to enable end-to-end 4D generation.

Introduction

Recent advancements in generative models have enhanced the ability to produce high-quality digital contents across diverse media formats (e.g. images, videos, 3D models, 4D data). In particular, the generation of 4D data, including both spatial and temporal dimensions, has seen notable progress (Singer et al. 2023; Shen et al. 2023; Xu et al. 2024; Ling et al. 2023; Bahmani et al. 2024a; Yin et al. 2023; Bahmani et al. 2024b).

On the other hand, learning physical dynamics that exhibit temporal consistency and adhere to physical laws from observable data remains a difficult problem. Data are in the wild and noisy. Their underlying coherence is agnostic to the user. As a result, existing deep models have to assume some distributions of the data, which may not be the case in reality. In theory, the network would extract any knowledge provided sufficient data. In practice however, such data-based learning becomes more and more cumbersome with increased dimensionality of generated contents – it is unintuitive to define the right network structure to guide a physically meaningful generation; it requires terabyte- or

petabyte-scale high-quality training data, and center-level computing resource to facilitate the training. Those theoretical and practical obstacles combined impose significant challenges.

We explore a new way to establish physics-in-the-loop generative models. Our argument is that *learning from knowledge* instead of from raw data is more effective for generative models. Physical laws and principles are often in the form of partial differential equations (PDEs) and numerically solved with discretized differential operators. We note that those operators hold a similar structure as a convolution kernel on the problem domain, where the values of those convolution kernels depend on the specific problem setting. Inspired by those observations, we propose ElastoGen, a knowledge-driven neural model that generates physically accurate and coherent 4D elastodynamics. ElastoGen can be easily coupled and integrated with upstream and downstream neural modules to enable end-to-end 4D generation. The core idea of ElastoGen is converting the global differential operator, corresponding to the nonlinear force equilibrium, to iterative local convolution-like procedures. Such knowledge-level priors allow us to design dedicated network modules for ElastoGen, where each network module has a well-defined purpose of relaxing locally concentrated strain rather than being treated as a piece of a black box. Compared with other data-learning-based generative models, ElastoGen is lightweight – in terms of both training requirements and the network scales. Furthermore, due to its consistency with physics procedure, ElastoGen generates physically accurate dynamics for a wide range of hyperelastic materials. Specifically, we summarize some features of ElastoGen as follows:

Compact generative network inspired by physics principles The network architecture of ElastoGen is strongly inspired by our prior knowledge of physics and corresponding numerical procedures. This allows a compact and effective generative framework in the form of deep neural networks. The training efforts for such a carefully tailored deep model become lightweight as well.

NeuralMTL with diffusion parameterization ElastoGen features a neural material module, *NeuralMTL*, to encode the underlying constitutive relations for real-world hyperelastic materials such as Neo-Hookean and or Saint Venant-Kirchhoff (StVK). We leverage a lightweight conditional

*These authors contributed equally.

†Corresponding author

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

diffusion model to predict its network parameters to isolate our training efforts.

Nested RNN with low-frequency encoding ElastoGen constitutes a two-level recurrent neural network (RNN) architecture. An encoder extracts low-frequency deformations so that the inner RNN relaxation only takes care of the local high-frequency strains. This design makes ElastoGen more efficient for stiff materials.

Related Work

4D generation with diffusion models The primary objective of generative models is to produce new, high-quality samples from vast datasets. These models are designed to learn and understand the distribution of data, thereby generating samples that meet specific criteria. Recently, diffusion models have emerged as a powerful technique, achieving state-of-the-art results in generating high-fidelity images (Sohl-Dickstein et al. 2015; Ho, Jain, and Abbeel 2020; Rombach et al. 2022), beating the competitors that are based on generative adversarial networks (GANs) (Goodfellow et al. 2014; Zhu et al. 2019; Chan et al. 2021) or variational autoencoders (VAEs) (Kingma and Welling 2014; Child 2021; Razavi, Van den Oord, and Vinyals 2019). This sets the stage for further explorations in more complex applications, such as the generation of 3D content (Jain et al. 2022; Lin et al. 2023; Metzger et al. 2023; Poole et al. 2022; Wang et al. 2024; Liu et al. 2023, 2024; Feng et al. 2024a), video content (Blattmann et al. 2023; Harvey et al. 2022; Ho et al. 2022b,a; Karras et al. 2023; Ni et al. 2023), and 3D videos or called 4D dynamic scenes (3D objects with shape change throughout time) (Singer et al. 2023; Shen et al. 2023; Xu et al. 2024; Ling et al. 2023; Bahmani et al. 2024a; Yin et al. 2023; Bahmani et al. 2024b). These advanced applications demonstrate the versatility and expanding potential of diffusion models across diverse media formats. However, existing 4D generation techniques struggle to ensure temporal consistency and require substantial training data, underscoring the challenges of capturing and replicating the dynamic and interconnected behaviors present in real-world scenarios within a generative model framework.

Neural physical dynamics Physical dynamics has traditional numerical solutions, such as the finite element method (FEM) (Zienkiewicz and Morice 1971; Zienkiewicz, Taylor, and Zhu 2005) and mass-spring systems (Liu et al. 2013). Each approach offers distinct advantages and limitations. For example, Position-Based Dynamics (PBD) (Müller et al. 2007) and Projective Dynamics (PD) (Bouaziz et al. 2014) offer simplified implementation and faster convergence but can struggle with complex material behaviors and do not always guarantee consistent convergence rates. Recently, neural physics solvers, which integrate neural networks with traditional solvers, aim to accelerate and simplify the computation process. The pioneering works (Chang et al. 2017; Battaglia et al. 2016) directly utilized neural networks to predict dynamics, achieving promising results in simple particle systems. Subsequent studies (Sanchez-Gonzalez et al. 2018; Kipf et al. 2018; Ajay et al. 2018; Li et al. 2019c,a,b) adopted network architectures to the specific features of the systems, thereby enhancing performance. Differentiable

simulators are also proposed (Degraeve et al. 2019; de Avila Belbute-Peres et al. 2018; Hu et al. 2019), enabling end-to-end training of the physics-based behavior from ground-truth data. The advent of Physics Informed Neural Networks (PINNs) (Raissi, Perdikaris, and Karniadakis 2019; Pakravan et al. 2021) marks a leap forward. These networks incorporate extensive physical information to constrain and guide the learning process, ensuring that predictions adhere more closely to physical laws and has succeeded in domains such as cloths (Geng, Johnson, and Fedkiw 2020) and fluids (Um et al. 2020; Gibou, Hyde, and Fedkiw 2019; Chu et al. 2022). Some work (Yang, He, and Zhu 2020) shifts away from end-to-end structures and use neural networks to optimize part of the simulation. Another line of research generates dynamics through physics-based simulators, where network learns static information while physical laws govern the generation of dynamics (Li et al. 2023; Feng et al. 2023; Xie et al. 2023; Feng et al. 2024b; Jiang et al. 2024), giving physical meanings to Neural Radiance Fields (NeRF) (Mildenhall et al. 2020; Kerbl et al. 2023a). These methods demonstrate the benefits of embedding human knowledge into networks to reduce the learning burden.

Background

To make the paper self-contained, we start with a brief review of the preliminaries of elastodynamics and the diffusion model.

Variational optimization of elastodynamics

The dynamic equilibrium of a 3D model can be characterized by $\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\mathbf{q}}} \right) - \frac{\partial L}{\partial \mathbf{q}} = \mathbf{f}_q$, where $L = T - U$ is system *Lagrangian i.e.*, the difference between the kinematic energy (T) and the potential energy (U). \mathbf{q} and $\dot{\mathbf{q}}$ are generalized coordinate and velocity. \mathbf{f}_q is the generalized external force. With the implicit Euler time integration scheme: $\mathbf{q}^{n+1} = \mathbf{q}^n + h\dot{\mathbf{q}}^{n+1}$, $\dot{\mathbf{q}}^{n+1} = \dot{\mathbf{q}}^n + h\ddot{\mathbf{q}}^{n+1}$, it can be reformulated as a nonlinear optimization to be solved at each time step:

$$\mathbf{q}^{n+1} = \operatorname{argmin}_{\mathbf{q}} \left\{ \frac{1}{2h^2} \|\mathbf{q} - \hat{\mathbf{q}}\|_{\mathbf{M}}^2 + U(\mathbf{q}) \right\}, \quad (1)$$

where $\hat{\mathbf{q}} = \mathbf{q}^n - h\dot{\mathbf{q}}^n - h^2\mathbf{M}^{-1}\mathbf{f}_q$, the superscript n and $n + 1$ indicates the time step, h is the time step size, and \mathbf{M} is the mass matrix.

Diffusion model

A diffusion model transforms a probability from the real data distribution $\mathcal{P}_{\text{real}}$ to a target distribution $\mathcal{P}_{\text{target}}$ through diffusion and denoising.

Diffusion. The diffusion process incrementally adds Gaussian noise to the initial data $\mathbf{x}_0 \sim \mathcal{P}_{\text{target}}$, gradually transforming it into a sequence $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T$, where \mathbf{x}_T approximates the real distribution $\mathcal{P}_{\text{real}}$. The aim is to learn a noise prediction model $\epsilon_\theta(\mathbf{x}_t, t)$, estimating the noise at each iteration t to facilitate data recovery in the denoising phase. The noise learning objective is formulated as:

$$L = \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{P}_{\text{target}}, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \text{Uniform}(\{1, \dots, T\})} [\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2]. \quad (2)$$

Denoising. Denoising iteratively removes noise from $\mathbf{x}_T \sim \mathcal{P}_{\text{real}}$, recovering the original data \mathbf{x}_0 by adjusting the noisy data at each iteration t as:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}, \mathbf{z} \sim N(\mathbf{0}, \mathbf{I}), \quad (3)$$

where $1 - \alpha_t = \beta_t$ is a scheduled variance, and σ_t is typically set to $\sigma_t = \sqrt{\beta_t}$. $N(\mathbf{0}, \mathbf{I})$ is standard normal distribution. Diffusion and denoising processes allow for effective modeling of the transition between distributions, using learned Gaussian transitions for noise prediction and reduction.

Methodology

ElastoGen, as illustrated in Fig.1, is a lightweight generative deep model that produces physically grounded 4D content from general object descriptions, such as stiffness or mass. ElastoGen rasterizes the input shape and uses a nested two-level recurrent neural network (RNN) to predict its trajectory sequentially. Each prediction undergoes an accuracy check to ensure physical validity. The network design follows a numerical procedure based on variational optimization as in Eq.(1), ensuring that ElastoGen avoids redundant components that could lead to overfitting. The following sections detail each major module of the pipeline.

Method overview: piece-wise local quadratic approximation

Our elastodynamics generation mimics numerical optimization procedures that minimize the variational energy as in Eq. (1). It is possible to tackle this problem at the global level, *i.e.*, optimizing all the degrees of freedom (DoFs) of the system at once *e.g.*, using Newton’s method. Such a brute-force scheme requires to learn dense inter-correlations among features at all DoFs, which inevitably leads to complex and large-scale network architectures with numerous parameters to be learned.

Alternatively, we opt for a divide-and-conquer way to approach Eq. (1). We consider the total potential energy U as the summation of multiple energies of quadratic form: $U(\mathbf{q}) \approx \sum_i E_i(\mathbf{q}_i)$, where $E_i(\mathbf{q}_i) = \min_{\mathbf{p}_i \in \mathcal{M}_i} \frac{\omega_i}{2} \|\mathcal{G}_i[\mathbf{q}_i] - \mathbf{p}_i\|^2$. Here, i indicates the i -th sub-volume of the object. For instance, one may discretize the object into a tetrahedral mesh, and E_i then represents the elastic potential stored at the i -th element. \mathcal{G}_i denotes a *discrete differential operator*, which converts positional features \mathbf{q}_i to strain-level features. To this end, we build \mathcal{G}_i such that $\mathcal{G}_i[\mathbf{q}_i] = \text{vec}(\mathbf{F}_i)$, *i.e.*, the vectorized deformation gradient ($\mathbf{F} \in \mathbb{R}^{3 \times 3}$) of the sub-volume, which gives the local first-order approximation of the displacement field. The constraint manifold \mathcal{M}_i denotes the zero level set of E_i . In other words, we consider E_i as a quadratic energy based on how far local displacement \mathbf{q}_i is from its closest energy-free configuration (\mathbf{p}_i), given the local material stiffness ω_i .

Provided the current deformed shape \mathbf{q}_i , we can find $\arg \min_{\mathbf{p}_i} \frac{\omega_i}{2} \|\mathcal{G}_i[\mathbf{q}_i] - \mathbf{p}_i\|^2$, which suggests a locally optimal descent direction to reduce U . The global displacement can then be obtained by minimizing \mathbf{q} over E_i at all

the sub-volumes. While this is a global operation, it acts as a Laplacian-like smoothing operator, which can be approximated through repeated local smoothing. This procedure resembles shape matching methods (Müller et al. 2005) and PD (Bouaziz et al. 2014), offering a piecewise sequential quadratic programming (SQP) approach (Boggs and Tolle 1995) to approximate U locally. ElastoGen functions as a neural version of the aforementioned procedure, with a nested RNN structure that handles local strain relaxation through volume convolutions, ensuring the network remains compact and lightweight.

Unfortunately, real-world materials are more than a collection of quadratic forms. The elastic energy of nonlinear materials cannot be well approximated by a quadratic form of the deformation gradient. This limitation means that shape matching or PD can only handle simplified material behavior. To this end, we augment ElastoGen with a NeuralMTL module to ensure that each local SQP closely matches actual materials.

NeuralMTL & neural projection

The goal of NeuralMTL is to correct local quadratic approximations of U so that ElastoGen faithfully generates physically accurate results for any real-world hyperelastic material. Specifically with NeuralMTL (\mathcal{N}), E_i becomes:

$$E_i(\mathbf{q}_i) = \arg \min_{\mathbf{P}_i \in \mathcal{SO}(3)} \frac{\omega_i}{2} \|\mathbf{F}_i \cdot \mathcal{N}(\mathcal{G}_i[\mathbf{q}_i]) - \mathbf{P}_i\|_F^2. \quad (4)$$

We set ω_i as $\omega_i = V_i e$ based on real-world material parameters: Young’s modulus e and the size of the sub-volume V_i . \mathcal{G}_i extracts the deformation gradient $\text{vec}(\mathbf{F}_i)$ and feeds it to NeuralMTL, \mathcal{N} . As the name suggests, \mathcal{N} predicts a *neural strain* based on the information of local deformation \mathbf{F}_i . Given the material model and parameters, \mathcal{N} is used for all E_i , and we do not put a subscript on \mathcal{N} . $\|\cdot\|_F$ denotes the Frobenius norm. \mathcal{N} predicts a material-space strain prediction, which is then converted to world space by \mathbf{F}_i . $\mathbf{P}_i \in \mathbb{R}^{3 \times 3}$ is a rotation matrix *i.e.*, $\mathbf{P}_i \in \mathcal{SO}(3)$. Intuitively, as shown in Fig. 2 (a), NeuralMTL warps \mathbf{F}_i to a different configuration of $\mathbf{F}_i \cdot \mathcal{N}(\mathbf{F}_i)$ so that the new distance to \mathbf{P}_i correctly reflects the local energy landscape of E_i as visualized in the inset.

For isotropic elastic materials, we add a nonlinear singular value decomposition (SVD) activation to the operator \mathcal{G}_i such that $\mathbf{F}_i = \mathbf{U}_i \mathbf{S}_i \mathbf{V}_i^T$. \mathbf{S}_i is a diagonal matrix with singular values arranged in descending order, which correspond to the local principal strains. This activation converts E_i to:

$$\begin{aligned} E_i(\mathbf{q}_i) &= \frac{\omega_i}{2} \|\mathbf{U}_i \mathbf{S}_i \mathbf{V}_i^T \cdot \mathcal{N}(\mathcal{G}_i[\mathbf{q}_i]) - \mathbf{U}_i \mathbf{V}_i^T\|_F^2 \\ &= \frac{\omega_i}{2} \text{tr} (\mathbf{S}_i \mathbf{S}_i \mathbf{V}_i^T \cdot \mathcal{N}(\mathcal{G}_i[\mathbf{q}_i]) \cdot \mathcal{N}^T(\mathcal{G}_i[\mathbf{q}_i]) \mathbf{V}_i \\ &\quad + \mathbf{I} - 2\mathbf{V}_i \mathbf{S}_i \mathbf{V}_i^T \cdot \mathcal{N}(\mathcal{G}_i[\mathbf{q}_i])). \end{aligned} \quad (5)$$

We further require this learning-based strain measure that 1) NeuralMTL predicts a symmetric strain; and 2) the adjusted energy remains invariant to rotation and merely depends on \mathbf{S}_i . Let $\mathbf{N}_i = \mathcal{N}(\mathcal{G}_i[\mathbf{q}_i]) \in \mathbb{R}^{3 \times 3}$ be the raw output of NeuralMTL. Instead of directly imposing those restrictions during the training, we append a network module to

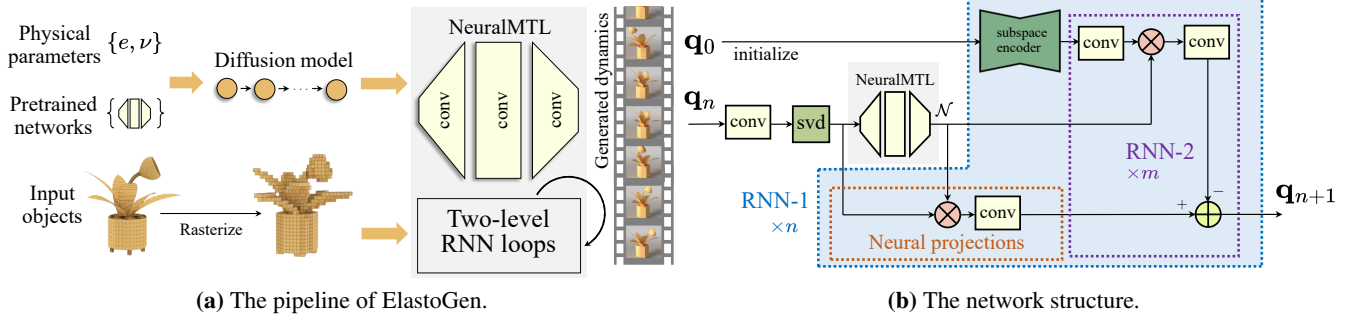


Figure 1: **Pipeline overview.** (a) ElastoGen rasterizes an input 3D model (with boundary conditions) and generates parameters filling our NeuralMTL module. Conceptually, NeuralMTL predicts locally concentrated strain of the object, which is relaxed by a nested RNN loop. (b) The RNN predicts the future trajectory of the object. There are two sub RNN modules. RNN-1 repeatedly relaxes the local stress in a 3D convolution manner. Those relaxed strains are converted to positional signals, and RNN-2 merges local deformation into a displacement field of the object. ElastoGen automatically checks the accuracy of the prediction of both RNN loops, and outputs the final prediction of \mathbf{q}_{n+1} once the prediction error reaches the prescribed threshold.

nonlinearly activate the raw output of \mathcal{N} as:

$$\mathcal{N}(\mathcal{G}_i[\mathbf{q}_i]) \leftarrow \mathbf{V}_i (\mathbf{N}_i + \mathbf{N}_i^\top) \mathbf{V}_i^\top, \quad (6)$$

which further simplifies E_i to:

$$E_i = \frac{\omega_i}{2} \text{tr}(\mathbf{Q}_i \mathbf{Q}_i^\top) + \frac{3\omega_i}{2} - \omega_i \text{tr}(\mathbf{Q}_i), \quad (7)$$

$$\mathbf{Q}_i(\mathbf{S}_i) = \mathbf{S}_i (\mathbf{N}_i + \mathbf{N}_i^\top).$$

Intuitively, this activation escalates the order of the neural strain predicted by \mathcal{N} , pushing it to become a nonlinear strain estimation with a prescribed format — just like upgrading an infinitesimal strain to Green’s strain to better measure large rotational deformation. As a result, the neural projection corresponding to our NeuralMTL can be easily obtained as $\mathbf{P}_i = \mathbf{U}_i \mathbf{V}_i^\top$, *i.e.*, the rotational component from \mathbf{F}_i . This is an important property of NeuralMTL — if we choose to employ the network to learn an adjustment of \mathbf{P}_i (which is also technically feasible), the local relaxation that predicts \mathbf{P}_i becomes complicated, and the generation is less robust.

Given an input 3D object, ElastoGen rasterizes it into a set of sub-volumes. For a user-specified sub-volume, which in our implementation is a voxel that intersects with the object, \mathcal{G}_i operator extracts the local covariance matrix of the displacement field over this sub-volume. Let $\mathbf{A}_i = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k] \in \mathbb{R}^{3 \times k}$ and $\bar{\mathbf{A}}_i = [\bar{\mathbf{q}}_1, \bar{\mathbf{q}}_2, \dots, \bar{\mathbf{q}}_k]$ be deformed and rest-shape position of vertices of a sub-volume with k vertices ($k = 8$ for a cubic volume). \mathcal{G}_i has an analytic format of:

$$\mathcal{G}_i[\mathbf{q}_i] = \left[(\bar{\mathbf{A}} \bar{\mathbf{A}}^\top)^{-1} \bar{\mathbf{A}} \otimes \mathbf{I} \right] \mathbf{q}_i, \quad (8)$$

which is a convolutional neural network (CNN) whose weights can be pre-computed given the rasterized object. The output of \mathcal{G}_i is then activated via a SVD module, which outputs \mathbf{U}_i , \mathbf{V}_i , and \mathbf{S}_i .

And the NeuralMTL \mathcal{N} can also be implemented as a CNN whose weights are predicted by a generative diffu-

sion model given the material type and parameters such as Young’s modulus e and Poisson’s ratio ν .

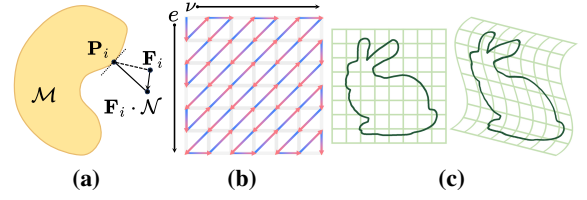


Figure 2: (a). NeuralMTL learns a mapping \mathcal{N} to warp \mathbf{F}_i , enabling the quadratic strain energy to work for hyperelastic materials (b). Sampling topology over 2D input space to ensure smooth output variation. (c) Deforming object with rasterization grid.

Decoupling NeuralMTL from deformation and material parameters

NeuralMTL takes \mathbf{F}_i as input and outputs the neural strain measure $\mathcal{N}(\mathbf{F}_i)$. This strain is then evaluated against Eq. (4) to check if ElastoGen reasonably minimizes Eq. (1) and is ready for the next time step. NeuralMTL is designed to accommodate material nonlinearity, meaning different material parameters $\{e, \nu\}$ result in different outputs even for the same \mathbf{F}_i .

A direct approach would be training NeuralMTL $\mathcal{N}(\mathbf{F}_i, e, \nu)$ on both \mathbf{F}_i and $\{e, \nu\}$. However, since NeuralMTL needs to be evaluated frequently during deformation, we decouple the influences of \mathbf{F}_i and $\{e, \nu\}$ to maintain a compact network. Following the approach in (Zhang et al. 2024a), we generate $\mathcal{N}(\mathbf{F}_i)$ using a diffusion model $\mathcal{D}(e, \nu)$, where $\mathbf{W} = \mathcal{D}(e, \nu)$ is the parameters for the network $\mathcal{N}(\mathbf{F}_i)$.

To train the model \mathcal{D} , we prepare a dataset of paired $\{e, \nu\}$ and \mathbf{W} . To this end, we first uniformly sample both e and ν at fixed intervals and then establish a topological order,

as shown in Fig. 2 (b). A target elastic energy $\Psi(e, \nu)$ can be easily computed for each sampled $\{e, \nu\}$. \mathbf{W} is then obtained via the following optimization:

$$\mathbf{W} = \underset{\mathbf{W}}{\operatorname{argmin}} \|\log(\Psi_{\mathcal{N}} + 1) - \log(\Psi + 1)\|^2, \quad (9)$$

where $\Psi_{\mathcal{N}} = \frac{\omega_i}{2} \|\mathbf{F}_i \cdot \mathcal{N}(\mathbf{W}, \mathbf{F}_i) - \mathbf{U}_i \mathbf{V}_i^T\|^2$, and $\mathcal{N}(\mathbf{W}, \mathbf{F}_i)$ suggests parameters of \mathcal{N} are prescribed by \mathbf{W} . The logarithmic function \log is used to penalize energy deviations under the same deformation, ensuring non-negative energy. As the energy function varies smoothly with $\{e, \nu\}$, our pre-defined topological ordering of $\{e, \nu\}$ samples facilitates efficient training. \mathbf{W} can converge within a few hundred gradient descent iterations when initialized from the previous \mathbf{W} . During inference, after \mathcal{D} predicts \mathbf{W} , we perform a few additional gradient descent iterations to fine-tune the weights, ensuring \mathcal{N} accurately fits the desired elastic energy function. This two-step process enables smooth variation of the energy function with respect to $\{e, \nu\}$, ensuring efficient and precise network parameter generation.

Subspace encoding

If the quadratic approximation of Eq. (1) is exact, NeuralMTL, \mathcal{N} , is not needed. And after obtaining \mathbf{Q}_i for all voxels, we set its derivative to zero leading to:

$$\left(\frac{\mathbf{M}}{h^2} + \sum_i \mathbf{L}_i \right) \mathbf{q}^{n+1} = \mathbf{f}_q + \frac{\mathbf{M}}{h^2} (\mathbf{q}^n + h\dot{\mathbf{q}}^n) + \sum_i \mathbf{b}_i, \quad (10)$$

where $\mathbf{b}_i = \mathbf{L}_i \mathbf{q}_n - \frac{\partial E_i}{\partial \mathbf{q}}$. The term $\frac{\mathbf{M}}{h^2} + \sum_i \mathbf{L}_i$ as the *global matrix*, which remains constant in this case. This allows for pre-factorization, converting the global matrix into lower and upper triangular forms to facilitate efficient solving of the linear system. However, the use of NeuralMTL introduces nonlinearities in the energy landscape, making $\mathbf{L}_i(\mathbf{q})$ dependent on the current deformed pose \mathbf{q} . A fully implicit evaluation would require $\nabla_{\mathbf{q}} \mathbf{L}_i$ and $\nabla_{\mathbf{q}} \mathcal{N}$, which is computationally expensive and less stable due to the need for additional training constraints (e.g., penalizing $|\nabla \mathcal{N}|$ to prevent overfitting). To address this, we adopt a lagged approach, computing \mathbf{L}_i by using \mathbf{q} from the most recent update.

Directly inverting the global matrix is computationally expensive due to the dense correlations between vertices. As an alternative, traditional iterative methods update each node by incorporating information from its neighbors, using results from the previous iteration, and continue relaxing locally until convergence is achieved. Inspired by these methods, we decompose the global solve for \mathbf{q} by perform multiple local operations at \mathbf{q}_i . Additionally, since the objects are rasterized into a grid, these repetitive local operators can be efficiently implemented using a recurrent CNN. This CNN takes \mathbf{q}_i as input and outputs the relaxed result after one step. This CNN collects information from the vertices to the voxels via a convolution, then relaxes it back to the vertices using a transposed convolution. Additional implementation details can be found in our supplementary material (Feng et al. 2025). Conceptually, this approach functions as a matrix-free method for solving the global system,



Figure 3: **ElastoGen with implicit models.** ElastoGen supports both explicit and implicit models. We dense-sample the implicit neural field, directly generating physically accurate dynamics without a simulator, enabling image-to-image generation from novel camera poses.

ensuring that ElastoGen generates a physically accurate trajectory.

ElastoGen is therefore built as a two-level RNN network. The outer RNN (RNN-1, see Fig. 1) applies local NeuralMTL adjustments over \mathcal{G}_i at each voxel region and the neural projection for \mathbf{P}_i . The inner RNN (RNN-2) handles global solve by iterating a CNN, approximating the local relaxation effect through repeated local operations. Each local operator relaxes the concentrated strain predicted by NeuralMTL \mathcal{N} and is propagated across the object. The process iterates until the difference between the results of two consecutive iterations is below a specified threshold.

However, this approach requires many RNN loops to achieve effective global relaxation. This is because local operations are more effective in processing locally concentrated strains, while object-wise global deformation can only be progressively approximated by information exchange across voxels. This is also a well-known limitation in numerical computation — Gauss-Seidel- or Jacobi-style iterative methods are less effective in relaxing low-frequency residual errors, which are often paired with a multigrid solver for large-scale problems.

To address this issue, we apply SVD to encode Eq. (10) into a low-frequency latent space and directly solve the system within this space. The result is then decoded back to the original space, serving as the initialization for RNN-2. However, since the global matrix varies across objects and timesteps, performing frequent SVD computations on the global matrix is inefficient. Given that only a prediction of the overall low-frequency deformation is required, we approximate the object’s global matrix using the global matrix of a uniform rasterized grid. This projection can be precomputed, eliminating the need for recalculation at each step or for each object. With this approach, each latent mode takes the form of a smooth sine wave, as illustrated in Fig. 2 (c).

Experiments

We implement ElastoGen using Python. Specifically, we use PyTorch (Imambi, Prakash, and Kanagachidambaresan 2021) to implement the network and a simulator for training data generation. Our hardware is a desktop computer equipped with an Intel i7-12700F CPU and an NVIDIA 3090 GPU. Detailed statistics of the settings, models, and fitting errors are reported in Tab. 1. *All the experiments are also available in the supplemental video.*

| Scene | Grid resolution | #DoFs | #latent | Δt | #R1 | #R2 | EM | Fitting error | t/frame |
|----------------------------|--------------------------|-------|---------|------------|-----|-----|------|-----------------------|---------|
| Cantilever (Fig. 4) | $16 \times 3 \times 3$ | 432 | 18 | 0.001 | 5 | 108 | All | 4.11×10^{-4} | 0.01 |
| Lego (Fig. 3) | $26 \times 46 \times 30$ | 11K | 54 | 0.005 | 15 | 320 | NH | 2.34×10^{-4} | 0.44 |
| Drums (Fig. 3) | $28 \times 22 \times 34$ | 4K | 54 | 0.005 | 15 | 320 | CR | 7.63×10^{-5} | 0.21 |
| Bridge (Fig. 5) | $66 \times 13 \times 27$ | 7K | 81 | 0.003 | 5 | 96 | StVK | 5.78×10^{-4} | 0.92 |
| Ship (Fig. 5) | $53 \times 33 \times 16$ | 14K | 81 | 0.001 | 5 | 100 | NH | 2.34×10^{-4} | 1.20 |
| monster (Fig. 6) | $32 \times 30 \times 22$ | 20K | 36 | 0.001 | 5 | 100 | NH | 2.34×10^{-4} | 0.78 |
| shoe (Fig. 6) | $48 \times 30 \times 20$ | 28K | 36 | 0.001 | 5 | 100 | NH | 2.34×10^{-4} | 1.08 |

Table 1: **Experiments statistics.** We report detailed settings of our experiments. **#DoFs**: the average number of DOFs involved in the optimization. Δt : the size of timestep. **#R1**: the average loop count of RNN-1 for each step. **#R2**: the average number of RNN-2 loops for each timestep. **# latent**: the dimension of latent layer in the subspace encoder. **EM**: the elastic materials including Neo-Hookean (NH), StVK, and co-rotational (CR) models. **Fitting error**: the loss of NeuralMTL in Eq. (9). **t/frame**: the seconds needed for each frame.

Quantitative validation of NeuralMTL

ElastoGen replicates the behavior of complex hyperelastic materials with varying parameters. We quantitatively compare ElastoGen’s results with those from the finite element method (FEM) using a cantilever beam bending test. ElastoGen predicts the trajectories for three classic materials co-rotational (Brogan 1986), Neo-Hookean (Wu et al. 2001), and StVK (Barbič and James 2005). Each material is tested at three Poisson’s ratios, with a fixed Young’s modulus. (Poisson’s ratio alters the material response more nonlinearly than Young’s modulus). The results of ElastoGen, as shown in Fig. 4 (b), align well with the results obtained from the classic method of FEM. Both overlap nearly perfectly. Such superior accuracy is due to our NeuralMTL prediction. As shown in Fig. 4 (a), the diffusion-generated strain from NeuralMTL closely matches the ground truth (GT) with the correlation coefficient r being larger than 0.98 (calculated as $r = \frac{\sum_{i=1}^n (g_i - \bar{g})(f_i - \bar{f})}{\sqrt{\sum_{i=1}^n (g_i - \bar{g})^2 \sum_{i=1}^n (f_i - \bar{f})^2}}$ for each sample point f_i and g_i on neural strain and the ground truth curve, and \bar{f} and \bar{g} are their averages). We also plot the total neural energy variation over time for those materials ($\nu = 0.32$) in Fig. 4 (c).

Versatility

ElastoGen is a general-purpose generative AI model. As long as a 3D object can be rasterized, ElastoGen deals with both explicit, *e.g.*, as shown in Fig. 5, and implicit shape representations. For instance, when ElastoGen readily takes an implicit neural radiance field (NeRF) (Mildenhall et al. 2021) based model. One can conveniently employ the Poisson-disk sampling as described in Feng et al. (2023) to obtain the rasterized model. Given user-specified external forces or position constraints, ElastoGen generates its further dynamics directly via a neural network without resorting to an underlying physics simulator as used in PIE-NeRF (Feng et al. 2023). Similarly, a 3DGS (3D Gaussian splatting)-based model (Kerbl et al. 2023b) can also feed to ElastoGen for 4D generation. We demonstrate this using NeRF dataset images in Fig. 3.

ElastoGen enables artists and animators to quickly produce high-quality 4D animations, even for complex mod-

els. Examples of high-resolution, triangle mesh objects are shown in Fig. 5, where ElastoGen generates visually accurate dynamics while preserving fine structural details. Additionally, material parameters can be inversely learned from video to ensure consistency with observed dynamics.

ElastoGen can also serve as a downstream model for 3D generation frameworks to accomplish true 4D generation tasks. After rasterizing the 3D object generated by the upstream model, it can be used as the input to ElastoGen, which then produces physically plausible animations driven by forces or constraints. As shown in Fig. 6, by integrating ElastoGen with ARM (Feng et al. 2024a), we achieve high-quality and physically accurate 4D elastodynamics generation.

More comparisons

Comparison with ground truth. In addition to Fig. 4, we further compare ElastoGen with the FEM simulation under large-scale nonlinear twisting. The comparison is based on the Neo-Hookean material. For highly nonlinear instances, the physical accuracy of ElastoGen relies on the RNN loops — more loops at both RNN-1 and RNN-2 effectively converge ElastoGen to the ground truth. Nevertheless, for general-purpose generation, fewer iterations also yield good results. Detailed experimental results and error plots are provided in the supplementary materials (Feng et al. 2025).

Comparison with SOTA competitors. We compare ElastoGen with existing 4D generative models, including Gen-2 (Inc. 2024) and PhysDreamer (Zhang et al. 2024b) in Fig. 7, showing that ElastoGen excels in both physical accuracy and geometric consistency. Gen-2 generates moderate motion with limited nonlinearity, such as rotation and bending, and fails to maintain geometric consistency over time, causing changes in both color and shape. This is a common issue for observation-based 4D models, where complex visual correlations in training data are difficult to decouple in a monolithic deep model. PhysDreamer produces plausible elastodynamics only at small time steps ($\Delta t < 6.0 \times 10^{-5}$) due to the underlying explicit integration, which is known to be unstable under large time steps. In contrary, ElastoGen is able to generalize on large time steps. In Tab. 2, we present a quantitative comparison of error using the Intersection over

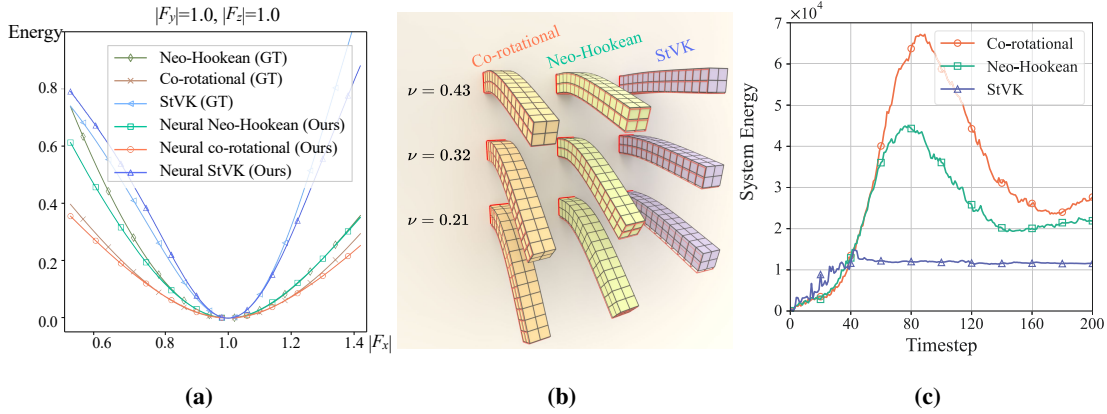


Figure 4: **Quantitative validation of NeuralMTL.** (a) Comparison between the energy computed from NeuralMTL strain and the ground truth energy. (b) Visualization of the final static state of the cantilever beam generated by our method, with different materials and material parameters. (c) Plots of the elastic energy during the prediction.



Figure 5: **Experiments on high-resolution mesh objects.** ElastoGen generates realistic nonlinear elastic dynamics under external forces. Both high-frequency local details and low-frequency global deformations are effectively captured with a carefully designed nested RNN architecture.



Figure 6: **4D generation with ARM (Feng et al. 2024a).** We combine ElastoGen with a 3D generation model ARM (Feng et al. 2024a) to perform 4D elastodynamics generation. The first row shows a monster shaking its head, while the second row shows a shoe being squeezed.

Union (IoU) metric between ElastoGen, Gen-2 (Inc. 2024), and PhysDreamer (Zhang et al. 2024b). The reference data is generated using (Feng et al. 2023). Our method demonstrates superior accuracy in comparison to the others.

Conclusion

ElastoGen is a knowledge-driven deep model that embeds physical principles and numerical procedures into the network design. As a result, it is remarkably lightweight and compact. Each module is designed for a specific computational task aimed at minimizing the total variational energy. This modular design enables decoupled training, removing the need for large-scale training datasets. The accuracy of

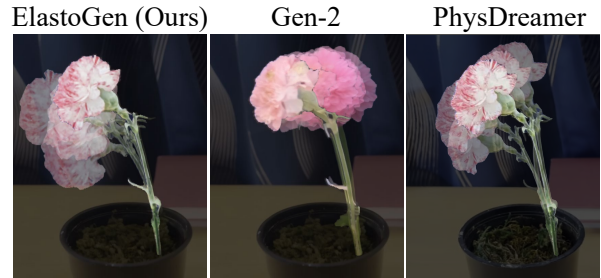


Figure 7: **Comparison (trajectory) between ElastoGen, Gen-2 (Inc. 2024) and PhysDreamer (Zhang et al. 2024b).** We visualize the trajectory of a swinging carnation using ElastoGen, Gen-2, and PhysDreamer. Note that PhysDreamer can only produce plausible elastodynamics with tiny time steps.

| ElastoGen (Ours) | Gen-2 | PhysDreamer |
|------------------|-------|-------------|
| 94% | 64% | 75% |

Table 2: **Comparison of quantitative error between ElastoGen, Gen-2 and PhysDreamer.** We compute the Intersection over Union (IoU) using reference data generated by (Feng et al. 2023). Higher IoU values indicate greater accuracy.

ElastoGen can be controlled by NeuralMTL, which predicts the strain based on observed numerical computations.

ElastoGen also has limitations. It currently lacks collision support and is less efficient for thin geometries due to excessive convolution operations on empty voxels. It may also fail to converge with highly stiff materials, such as near-rigid objects. Future improvements will include adding dynamics for more physical phenomena, integrating collision support, and automating the setting of physical parameters to enable real-world dynamics with minimal input.

Acknowledgements

We thank anonymous reviewers and AC for their constructive comments. This work is supported by NSF China (62322209, 62421003, 62332015), the gift from Adobe Research, the XPLOER PRIZE and the 100 Talents Program of Zhejiang University. Yin Yang is partially supported by NSF 2301040.

References

- Ajay, A.; Wu, J.; Fazeli, N.; Bauza, M.; Kaelbling, L. P.; Tenenbaum, J. B.; and Rodriguez, A. 2018. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3066–3073. IEEE.
- Bahmani, S.; Liu, X.; Wang, Y.; Skorokhodov, I.; Rong, V.; Liu, Z.; Liu, X.; Park, J. J.; Tulyakov, S.; Wetzstein, G.; et al. 2024a. TC4D: Trajectory-Conditioned Text-to-4D Generation. *arXiv preprint arXiv:2403.17920*.
- Bahmani, S.; Skorokhodov, I.; Rong, V.; Wetzstein, G.; Guibas, L.; Wonka, P.; Tulyakov, S.; Park, J. J.; Tagliasacchi, A.; and Lindell, D. B. 2024b. 4D-fy: Text-to-4D Generation Using Hybrid Score Distillation Sampling. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Barbič, J.; and James, D. L. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM transactions on graphics (TOG)*, 24(3): 982–990.
- Battaglia, P.; Pascanu, R.; Lai, M.; Jimenez Rezende, D.; et al. 2016. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29.
- Blattmann, A.; Rombach, R.; Ling, H.; Dockhorn, T.; Kim, S. W.; Fidler, S.; and Kreis, K. 2023. Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models. *arXiv:2304.08818*.
- Boggs, P. T.; and Tolle, J. W. 1995. Sequential quadratic programming. *Acta numerica*, 4: 1–51.
- Bouaziz, S.; Martin, S.; Liu, T.; Kavan, L.; and Pauly, M. 2014. Projective dynamics: fusing constraint projections for fast simulation. *ACM Trans. Graph.*, 33(4).
- Brogan, F. 1986. An Element Independent Corotational Procedure for the Treatment of Large Rotations. *Journal of Pressure Vessel Technology*, 108: 165.
- Chan, E. R.; Monteiro, M.; Kellnhofer, P.; Wu, J.; and Wetzstein, G. 2021. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5799–5809.
- Chang, M.; Ullman, T. D.; Torralba, A.; and Tenenbaum, J. B. 2017. A Compositional Object-Based Approach to Learning Physical Dynamics. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.
- Child, R. 2021. Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- Chu, M.; Liu, L.; Zheng, Q.; Franz, E.; Seidel, H.-P.; Theobalt, C.; and Zayer, R. 2022. Physics informed neural fields for smoke reconstruction with sparse data. *ACM Trans. Graph.*, 41(4).
- de Avila Belbute-Peres, F.; Smith, K.; Allen, K.; Tenenbaum, J.; and Kolter, J. Z. 2018. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31.
- Degrave, J.; Hermans, M.; Dambre, J.; and Wyffels, F. 2019. A differentiable physics engine for deep learning in robotics. *Frontiers in neurorobotics*, 13: 6.
- Feng, X.; Yu, C.; Bi, Z.; Shang, Y.; Gao, F.; Wu, H.; Zhou, K.; Jiang, C.; and Yang, Y. 2024a. ARM: Appearance Reconstruction Model for Relightable 3D Generation. *arXiv:2411.10825*.
- Feng, Y.; Feng, X.; Shang, Y.; Jiang, Y.; Yu, C.; Zong, Z.; Shao, T.; Wu, H.; Zhou, K.; Jiang, C.; et al. 2024b. Gaussian Splashing: Dynamic Fluid Synthesis with Gaussian Splatting. *arXiv preprint arXiv:2401.15318*.
- Feng, Y.; Shang, Y.; Feng, X.; Lan, L.; Zhe, S.; Shao, T.; Wu, H.; Zhou, K.; Jiang, C.; and Yang, Y. 2025. ElastoGen: 4D Generative Elastodynamics. *arXiv:2405.15056*.
- Feng, Y.; Shang, Y.; Li, X.; Shao, T.; Jiang, C.; and Yang, Y. 2023. PIE-NeRF: Physics-based Interactive Elastodynamics with NeRF. *arXiv preprint arXiv:2311.13099*.
- Geng, Z.; Johnson, D.; and Fedkiw, R. 2020. Coercing machine learning to output physically accurate results. *J. Comput. Phys.*, 406: 109099.
- Gibou, F.; Hyde, D.; and Fedkiw, R. 2019. Sharp interface approaches and deep learning techniques for multiphase flows. *Journal of Computational Physics*, 380: 442–463.
- Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.
- Harvey, W.; Naderiparizi, S.; Masrani, V.; Weilbach, C.; and Wood, F. 2022. Flexible diffusion modeling of long videos. *Advances in Neural Information Processing Systems*, 35: 27953–27965.
- Ho, J.; Chan, W.; Saharia, C.; Whang, J.; Gao, R.; Gritsenko, A.; Kingma, D. P.; Poole, B.; Norouzi, M.; Fleet, D. J.; et al. 2022a. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*.
- Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising Diffusion Probabilistic Models. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Ho, J.; Salimans, T.; Gritsenko, A.; Chan, W.; Norouzi, M.; and Fleet, D. J. 2022b. Video diffusion models. *Advances in Neural Information Processing Systems*, 35: 8633–8646.
- Hu, Y.; Anderson, L.; Li, T.-M.; Sun, Q.; Carr, N.; Ragan-Kelley, J.; and Durand, F. 2019. DiffTaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*.
- Imambi, S.; Prakash, K. B.; and Kanagachidambaresan, G. 2021. PyTorch. *Programming with TensorFlow: Solution for Edge Computing Applications*, 87–104.
- Inc., R. A. 2024. Text/Image to Video Gen-2. <https://app.runwayml.com/video-tools>. Accessed: 2024-08-04.
- Jain, A.; Mildenhall, B.; Barron, J. T.; Abbeel, P.; and Poole, B. 2022. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 867–876.
- Jiang, Y.; Yu, C.; Xie, T.; Li, X.; Feng, Y.; Wang, H.; Li, M.; Lau, H.; Gao, F.; Yang, Y.; et al. 2024. VR-GS: A Physical Dynamics-Aware Interactive Gaussian Splatting System in Virtual Reality. *arXiv preprint arXiv:2401.16663*.
- Karras, J.; Holynski, A.; Wang, T.-C.; and Kemelmacher-Shlizerman, I. 2023. Dreampose: Fashion image-to-video synthesis via stable diffusion. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 22623–22633. IEEE.

- Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023a. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4).
- Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023b. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.*, 42(4): 139–1.
- Kingma, D. P.; and Welling, M. 2014. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*.
- Kipf, T.; Fetaya, E.; Wang, K.-C.; Welling, M.; and Zemel, R. 2018. Neural relational inference for interacting systems. In *International conference on machine learning*, 2688–2697. PMLR.
- Li, X.; Qiao, Y.; Chen, P. Y.; Jatavallabhula, K. M.; Lin, M. C.; Jiang, C.; and Gan, C. 2023. PAC-NeRF: Physics Augmented Continuum Neural Radiance Fields for Geometry-Agnostic System Identification. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Li, Y.; He, H.; Wu, J.; Katabi, D.; and Torralba, A. 2019a. Learning compositional koopman operators for model-based control. *arXiv preprint arXiv:1910.08264*.
- Li, Y.; Wu, J.; Tedrake, R.; Tenenbaum, J. B.; and Torralba, A. 2019b. Learning Particle Dynamics for Manipulating Rigid Bodies, Deformable Objects, and Fluids. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.
- Li, Y.; Wu, J.; Zhu, J.-Y.; Tenenbaum, J. B.; Torralba, A.; and Tedrake, R. 2019c. Propagation networks for model-based control under partial observation. In *2019 International Conference on Robotics and Automation (ICRA)*, 1205–1211. IEEE.
- Lin, C.-H.; Gao, J.; Tang, L.; Takikawa, T.; Zeng, X.; Huang, X.; Kreis, K.; Fidler, S.; Liu, M.-Y.; and Lin, T.-Y. 2023. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 300–309.
- Ling, H.; Kim, S. W.; Torralba, A.; Fidler, S.; and Kreis, K. 2023. Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. *arXiv preprint arXiv:2312.13763*.
- Liu, M.; Xu, C.; Jin, H.; Chen, L.; Varma, T. M.; Xu, Z.; and Su, H. 2024. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances in Neural Information Processing Systems*, 36.
- Liu, R.; Wu, R.; Van Hoorick, B.; Tokmakov, P.; Zakharov, S.; and Vondrick, C. 2023. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9298–9309.
- Liu, T.; Bargteil, A. W.; O’Brien, J. F.; and Kavan, L. 2013. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG)*, 32(6): 1–7.
- Metzer, G.; Richardson, E.; Patashnik, O.; Giryas, R.; and Cohen-Or, D. 2023. Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12663–12673.
- Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.
- Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1): 99–106.
- Müller, M.; Heidelberger, B.; Hennix, M.; and Ratcliff, J. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2): 109–118.
- Müller, M.; Heidelberger, B.; Teschner, M.; and Gross, M. 2005. Meshless deformations based on shape matching. *ACM transactions on graphics (TOG)*, 24(3): 471–478.
- Ni, H.; Shi, C.; Li, K.; Huang, S. X.; and Min, M. R. 2023. Conditional Image-to-Video Generation with Latent Flow Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18444–18455.
- Pakravan, S.; Mistani, P. A.; Aragon-Calvo, M. A.; and Gibou, F. 2021. Solving inverse-PDE problems with physics-aware neural networks. *Journal of Computational Physics*, 440: 110414.
- Poole, B.; Jain, A.; Barron, J. T.; and Mildenhall, B. 2022. DreamFusion: Text-to-3D using 2D Diffusion. In *The Eleventh International Conference on Learning Representations*.
- Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378: 686–707.
- Razavi, A.; Van den Oord, A.; and Vinyals, O. 2019. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32.
- Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.
- Sanchez-Gonzalez, A.; Heess, N.; Springenberg, J. T.; Merel, J.; Riedmiller, M.; Hadsell, R.; and Battaglia, P. 2018. Graph networks as learnable physics engines for inference and control. In *International conference on machine learning*, 4470–4479. PMLR.
- Shen, L.; Li, X.; Sun, H.; Peng, J.; Xian, K.; Cao, Z.; and Lin, G. 2023. Make-It-4D: Synthesizing a Consistent Long-Term Dynamic Scene Video from a Single Image. In *Proceedings of the 31st ACM International Conference on Multimedia*, 8167–8175.
- Singer, U.; Sheynin, S.; Polyak, A.; Ashual, O.; Makarov, I.; Kokkinos, F.; Goyal, N.; Vedaldi, A.; Parikh, D.; Johnson, J.; et al. 2023. Text-to-4d dynamic scene generation. *arXiv preprint arXiv:2301.11280*.
- Sohl-Dickstein, J.; Weiss, E. A.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In Bach, F. R.; and Blei, D. M., eds., *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, 2256–2265. JMLR.org.
- Um, K.; Brand, R.; Fei, Y. R.; Holl, P.; and Thuerey, N. 2020. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in Neural Information Processing Systems*, 33: 6111–6122.
- Wang, Z.; Lu, C.; Wang, Y.; Bao, F.; Li, C.; Su, H.; and Zhu, J. 2024. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36.
- Wu, X.; Downes, M. S.; Goktekin, T.; and Tendick, F. 2001. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Computer Graphics Forum*, volume 20, 349–358. Wiley Online Library.
- Xie, T.; Zong, Z.; Qiu, Y.; Li, X.; Feng, Y.; Yang, Y.; and Jiang, C. 2023. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *arXiv preprint arXiv:2311.12198*.

- Xu, D.; Liang, H.; Bhatt, N. P.; Hu, H.; Liang, H.; Plataniotis, K. N.; and Wang, Z. 2024. Comp4D: LLM-Guided Compositional 4D Scene Generation. *arXiv preprint arXiv:2403.16993*.
- Yang, S.; He, X.; and Zhu, B. 2020. Learning physical constraints with neural projections. *Advances in Neural Information Processing Systems*, 33: 5178–5189.
- Yin, Y.; Xu, D.; Wang, Z.; Zhao, Y.; and Wei, Y. 2023. 4dgen: Grounded 4d content generation with spatial-temporal consistency. *arXiv preprint arXiv:2312.17225*.
- Zhang, B.; Luo, C.; Yu, D.; Li, X.; Lin, H.; Ye, Y.; and Zhang, B. 2024a. MetaDiff: Meta-Learning with Conditional Diffusion for Few-Shot Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 16687–16695.
- Zhang, T.; Yu, H.-X.; Wu, R.; Feng, B. Y.; Zheng, C.; Snavely, N.; Wu, J.; and Freeman, W. T. 2024b. PhysDreamer: Physics-Based Interaction with 3D Objects via Video Generation. *arxiv*.
- Zhu, M.; Pan, P.; Chen, W.; and Yang, Y. 2019. Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5802–5810.
- Zienkiewicz, O. C.; and Morice, P. 1971. *The finite element method in engineering science*, volume 1977. McGraw-hill London.
- Zienkiewicz, O. C.; Taylor, R. L.; and Zhu, J. Z. 2005. *The finite element method: its basis and fundamentals*. Elsevier.