NEUROK: Generative 4D Neural Object Kinematics

Chen Geng[1,*] Guangzhao He[3,*] Yue Gao[1,*]
Yunzhi Zhang[1] Shangzhe Wu[2] Jiajun Wu[1]

[1]Stanford University [2]University of Cambridge [3]Cornell University

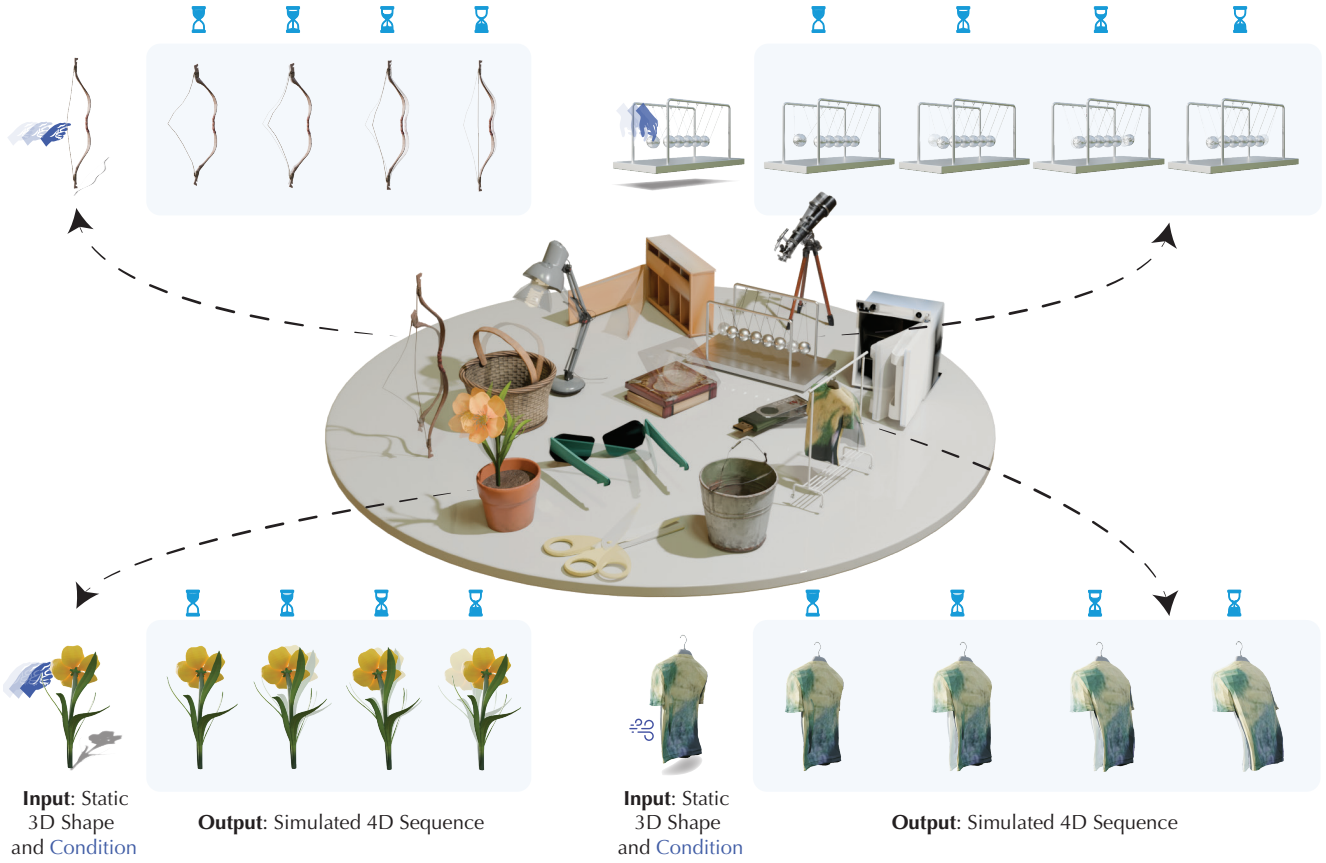


Figure 1. We present a versatile and scalable framework for generating simulative 4D dynamics of static 3D objects under physical conditions (*e.g.*, forces, actions, velocities). Trained on a large-scale 4D shape dataset without any explicit physical annotations, our method does not rely on any inductive bias of the object's dynamic structure and therefore can be applied to various types of dynamic objects, ranging from elastic bodies, cloth, and continuum bodies, to multi-body objects. Project page: https://chen-geng.com/neurok

Abstract

Data-driven approaches have revolutionized 3D vision, enabling transformers to effectively reconstruct and generate static 3D objects. However, generating simulative 4D dynamics—realistic temporal deformations of static objects under various physical conditions—remains challenging and often ad hoc, despite its importance in building comprehensive 3D world models. Most existing methods assume a predefined physical model and use system identification to estimate parameters, restricting these methods to specific categories and small-scale datasets.

*We propose that these restrictions can be overcome by learning a data-driven kinematic state parameterization for object-centric physical systems. Specifically, we learn both a latent space representing all possible states of the object and a decoder that maps any sampled latent to a plausibly deformed shape of the object. We refer to this parameterization as **Neur**al **O**bject **K**inematics (*NEUROK*), and learn a transformer-based encoder-decoder model on a curated large-scale 4D dataset. This formulation and the learned model significantly simplify the generation of simulative dy-*

namics since we only need to consider the dynamics within a low-dimensional latent space from the Lagrangian mechanics' perspective in classical physics. We demonstrate the effectiveness and generality of this neural simulation framework across diverse dynamic object types, showing clear advantages over prior works.

1. Introduction

> These quantities need not be the Cartesian co-ordinates of the particles, and the conditions of the problem may render some other choice of coordinates more convenient.
>
> — *L. Landau & E. Lifshitz, in Mechanics, 1960*

Given a 3D geometric snapshot of a dynamic object, humans can intuitively imagine how the object would react under different physical conditions, even without precise knowledge of the governing physical equations. However, in the community of generative AI, generating such 4D reactive behaviors with no reliance on any category-specific physical priors is far from trivial, despite the importance of this capability in constructing 3D world models for embodied AI or robotics [63, 92].

A long-standing view holds that generating such 4D simulative dynamics demands a comprehensive physical understanding of the object. This is epitomized by most existing works [121, 130] that generate 4D dynamics by adopting predefined category-specific physical models and estimating their parameters with system identification. While this paradigm is effective for target object categories (*e.g.*, articulated objects, continuum bodies, and cloth), it struggles to generalize beyond these predefined categories, and, more importantly, offers limited scalability to large-scale 4D datasets comprising diverse dynamic structures.

Is it possible to build a general-purpose simulator that generates such 4D motions without any category-specific inductive bias? We argue that this is achievable by reconsidering a critical yet long-overlooked piece: the **kinematic state parameterization** of dynamic objects. As illustrated in Fig. 2, a kinematic state parameterization defines the configuration space of state vectors that fully specify an object's geometry. Most existing approaches [81, 121, 130] adopt a kinematic state parameterization naturally inherited from the object's shape representation, *e.g.*, a dense particle set derived from mesh discretizations. While effective, this choice leads to an over-parameterized system, and thus necessitates category-specific physical constraints to prevent the system from being under-determined.

We revisit this important factor by introducing an automatically-discovered kinematic state parameterization scheme — **Neur**al **O**bject **K**inematics (NEUROK) — a latent space from which any vector sampled can be decoded into a plausible deformation of the modeled object. With this learned parameterization, the physical system can be greatly simplified: we only need to model the transition between low-dimensional latent vectors, similar to how a pendulum system can be simplified through a symbolic parameterization (Fig. 2(a)). This data-driven parameterization leads to a universal framework that simulates system dynamics from the perspective of Lagrangian mechanics [58], where category-agnostic energy functions are defined over the latent states and dynamics are directly derived using Euler-Lagrange equations.

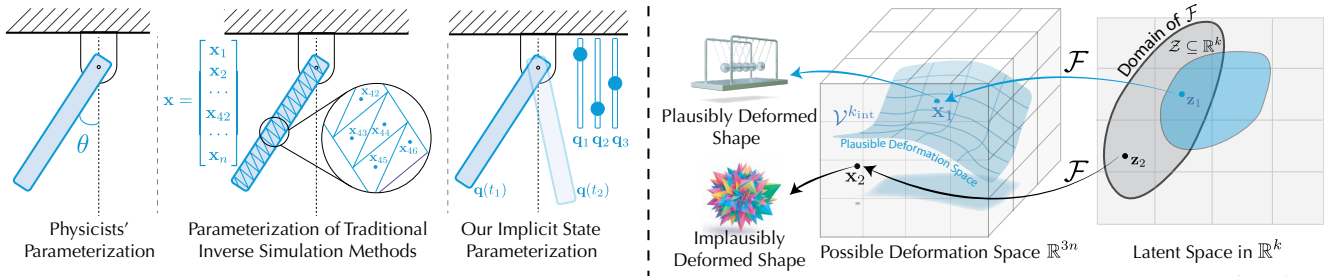
This framework forms a versatile and scalable pipeline for generative simulation of dynamic objects. Its core learning component, NEUROK, adopts a transformer-based [111] encoder-decoder architecture that learns to encode a static 3D object into a latent distribution over its possible kinematic states and to decode any sampled latent vector into a corresponding deformation field. The model can be trained solely on 4D geometric trajectories of 3D objects, eliminating any need for physical or action annotations. Moreover, this framework relies on a minimal inductive bias — that the object's deformation space is low-dimensional — making it broadly applicable to diverse dynamic objects.

We validate our framework by curating a large-scale 4D object dataset, training a feed-forward NEUROK model, and generating 4D dynamics across a wide range of objects. We evaluate its performance by comparing against existing methods, demonstrating its superior generalizability and effectiveness. To the best of our knowledge, this is the first data-driven framework capable of simulating object-centric physical systems without any reliance on heuristic priors or physical annotations.

2. Related Work

Physically-Inspired 4D Generation. Existing approaches to generating 4D simulative dynamics typically follow a two-step paradigm: finding a physical model of the targeted domain, and determining its parameters with system identification. This includes directly modeling physical properties of rigid objects [85, 123, 128]; modeling elastic objects with MPM [13, 17, 26, 52, 55, 67, 68, 77, 79–81, 93, 121, 130, 131], projective dynamics [10, 31, 100], or geometry-agnostic elastic simulation methods [18, 34, 94]; using spring-mass to model deformable objects [53, 135]; predicting articulations to model articulated objects [23, 54, 56, 59, 62, 82, 83, 91, 95, 118, 119, 122]; and building physical models for cloth [41, 69, 71, 74, 78, 89, 134]. While they perform well within specific domains, none can generate 4D motions without assuming a predefined dynamic structure. Our framework removes such structural biases, enabling general 4D simulative dynamics generation.

Reduced-Order Simulation. Model reduction is a com-



(a) Different Kinematic Parameterizations of the Same Physical System

(b) Illustration of an Abstract Kinematic State Parameterization $(\mathcal{Z}, \mathcal{F})$

Figure 2. **Kinematic state parameterization. (a)** Several kinematic state parameterizations can be used to describe a physical system. The symbolic parameterizations used in classical mechanics are concise yet not accessible in inverse problems. Traditional inverse simulation approaches use geometry-derived parameterizations, yet require dense physical constraints to solve the over-parameterized system. We instead learn low-dimensional parameterizations that are both compact and learnable from data. **(b)** As formally defined in Def. 1, a kinematic state parameterization studied in this paper is a pair $(\mathcal{Z}, \mathcal{F})$ which contains a latent manifold $\mathcal{Z}$ and a decoder $\mathcal{F}$ that maps a sampled latent to a vertex configuration. This definition explicitly includes those kinematic state parameterizations that are not compact.

mon technique in forward computer graphics, yet the focus is *efficiency* rather than versatility. The goal of such approaches [5, 16, 21, 22, 37, 50, 51, 61, 75, 94, 103, 108, 110, 112, 114, 137] is typically to accelerate an existing physical simulation system where all physical constraints are known, in contrast to our category-agnostic setting. These approaches typically train instance-specific neural networks to represent the reduced-order kinematic space for a specific object, rather than learning a generalizable, amortized-inference model on a large dataset as in our framework.

Machine Learning for Dynamic Systems. Beyond 3D vision, machine learning has also been used to model non-visual dynamic systems, typically through either physics-agnostic or physics-aware approaches. Physics-agnostic methods [3, 15, 64, 70, 72, 99, 106, 107, 109] learn dynamics end-to-end — often via GNNs — using synthetic datasets of action-state pairs. Although effective in controlled settings, they struggle to generalize to real-world objects due to the scarcity of action-labeled data. In contrast, our method relies solely on 4D geometric supervision, offering greater scalability for graphics and 3D vision applications. Physics-aware methods assume known physical models and use neural networks to solve PDEs [20, 73, 76, 102], learn constitutive laws [88, 93], and learn discretization schemes [2]. While demonstrating potential in producing accurate solutions, these approaches are unsuitable for our setting which makes no assumptions about dynamic structures. Closer to our formulation are methods that use neural networks [6, 25, 35, 87] to model systems within the Lagrangian mechanics framework, but their focus is learning the system's Lagrangian from synthetic data rather than learning data-driven kinematic state parameterizations.

Neural Deformation Priors. Several graphics systems have also explored learning data-driven priors over object deformations, but most are category-specific (*e.g.*, for humans [86, 98], faces [1, 8, 9], and animals [138]) and target other tasks — most dominantly, for animating characters [11, 40, 43, 45, 49, 90, 96, 97, 116, 124–127, 136] and controlling embodied agents [66, 84, 104]. We instead formalize this idea through the concept of kinematic state parameterization and demonstrate its huge potential as a general interface in physically-inspired 4D generation.

3. Overview

3.1. Formulation and Concepts

This paper studies generating simulative dynamics of 3D object-centric[1] physical systems. Our pipeline takes a static snapshot of a 3D dynamic object and a set of physical conditions (*e.g.,* actions, forces, initial velocities) as inputs, and generates a sequence of temporally evolving 3D shapes. As a single 3D snapshot of an object cannot fully determine its physical parameters, our goal is to *generate* one plausible 4D sequence that satisfies one valid physical configuration and conforms to human physical intuition [4]. We assume no kinematic or physical priors on the dynamic structure of the modeled object. The object can be articulated, rigid, a continuum body, or even a heterogeneous combination of several dynamic types, like the examples shown in Fig. 1.

The geometry of the modeled object is represented as a mesh $\mathcal{M}_0 = (V_0, F)$ with n vertices. We denote by $\mathbf{x}^0 \in \mathbb{R}^{3n}$ the concatenated vertex positions in V_0. Our pipeline outputs a sequence of deformed meshes with timestamps ranging from 1 to T, denoted as $\{\mathcal{M}_1, \cdots, \mathcal{M}_T\}$, where $\mathcal{M}_t = (V_t, F)$, and the concatenated vertex positions are represented by $\mathbf{x}^t \in \mathbb{R}^{3n}$.

While the vertices of the mesh $\mathcal{M}_0$ can theoretically take arbitrary positions in $\mathbb{R}^{3n}$, only a small subset of these configurations correspond to plausibly re-posed shapes. In fact, a randomly sampled deformation vector from $\mathbb{R}^{3n}$ will almost certainly yield a deformed mesh far outside the distribution of valid object poses. Empirically, the set of plausible vertex position vectors of a dynamic object forms a low-dimensional configuration manifold $\mathcal{V}^{k_{\text{int}}}$ embedded in

[1] We colloquially define an object-centric physical system as one in which most motion arises from a single dominant deformable object.

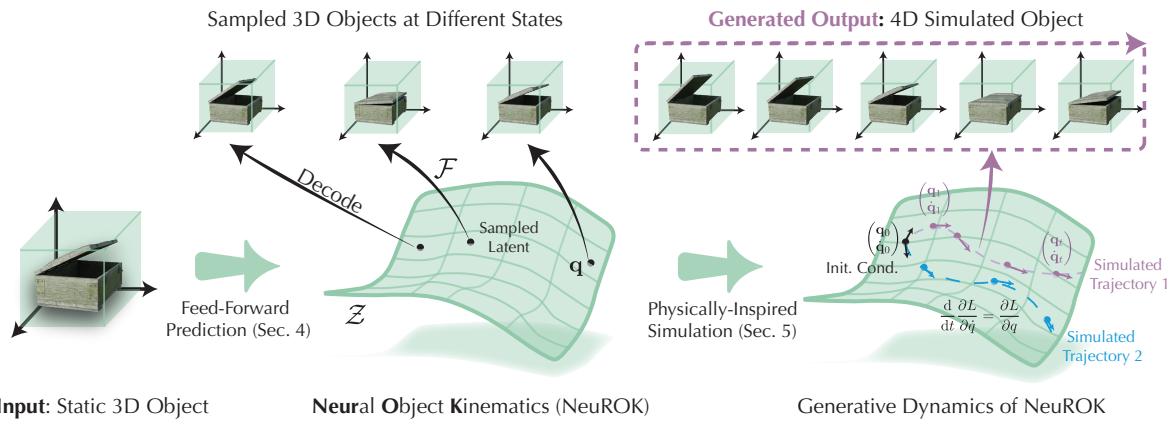


Figure 3. **Overview of our framework.** Given a static 3D shape, NeuROK uses a transformer-based encoder to predict an instance-specific latent space to represent different kinematic states of this object. Each sampled latent on the learned manifold can be decoded to a corresponding state of the input object. Under different physical conditions (*e.g.*, forces, actions, velocities), our method generates dynamic trajectories of latents by solving a physically-inspired ODE.

$\mathbb{R}^{3n}$, where k_{int} denotes the intrinsic degrees of freedom of the deformation space and $k_{\text{int}} \ll 3n$.

When studying these object-centric physical systems containing a deformable mesh with n vertices, we need to define a parameterization scheme for its kinematic states, which in turn determines the solution space for a physical simulator. We formulate this with the following definition:

Definition 1: Kinematic State Parameterization

A k-dimensional *kinematic state parameterization* for a dynamic object is a pair $(\mathcal{Z}, \mathcal{F})$, where $\mathcal{Z} \subseteq \mathbb{R}^k$ is the state space of the parameterization, and $\mathcal{F} : \mathcal{Z} \to \mathbb{R}^{3n}$ maps any state vector $\mathbf{z} \in \mathcal{Z}$ to a vertex configuration of $\mathcal{M}_0$ with n vertices.

Determining a kinematic state parameterization is the first step when studying a physical system, and it dictates how the system should be solved. As in Fig. 2(a), concise symbolic parameterizations are commonly used to simplify the solution space, but such representations are generally inaccessible in 4D generation where only the raw 3D geometry $\mathcal{M}_0$ is given. Consequently, most approaches adopt geometry-derived parameterizations, such as the high-dimensional particles (material points) used in MPM [52]. Such parameterizations are commonly redundant and under-constrained since some configurations will yield implausibly deformed shapes, as in Fig. 2(b).

To solve dynamics in high-dimensional solution space defined by the redundant parameterization, prior works introduce category-specific physical equations and constraints to prevent the system from being under-determined. These formulations are effective in targeted domains, yet they struggle to model objects beyond the designated category.

3.2. Proposed Solution

We address the above-discussed problem by introducing a kinematic state parameterization learned from data:

Definition 2: Neural Object Kinematics

If a k-dimensional kinematic state parameterization $(\mathcal{Z}, \mathcal{F})$ uses a neural network to represent $\mathcal{F}$, and the range of $\mathcal{F}$ is $\mathcal{V}^{k_{\text{int}}}$, we refer to this pair as a **Neural Object Kinematics** (NeuROK) of a dynamic object.

We train an encoder-decoder model to infer NeuROK of a given object $\mathcal{M}_0$. The model comprises an encoder that encodes $\mathcal{M}_0$ to an instance-specific latent space $\mathcal{Z}$ of the object's kinematic states and a decoder $\mathcal{F}$ that decodes any sampled latent to a plausibly deformed shape. This model is learned with a generative objective, as detailed in Sec. 4.

A successfully learned NeuROK greatly simplifies the solution space of the physical system, since we only need to model the dynamics between latent vector $\mathbf{z}$ in a low-dimensional space. It also eliminates the need for inter-particle physical equations employed in mainstream simulation approaches to keep the deformed shape intact and plausible, as any sampled latent can be mapped into a validly deformed mesh. This allows us to study the system as a whole by considering the energy landscape over different kinematic states of an entire system.

Formalizing this intuition, we simulate this system from the Lagrangian mechanics' perspective in classical physics. The learned NeuROK can be seen as the *generalized-coordinates* of the object-centric physical system, and such systems can be solved in a generic manner by defining the Lagrangian function of the system and solving Euler-Lagrange equations [58]. We detail this process in Sec. 5.

An overview of our framework can be found in Fig. 3.

4. Generative Learning of NeuROK

This section discusses the methodology of learning an encoder-decoder model to predict a NeuROK $(\mathcal{Z}(\mathcal{M}_0), \mathcal{F}(\cdot; \mathcal{M}_0))$ from an input mesh $\mathcal{M}_0$ of a 3D snapshot of a dynamic object. We model the latent

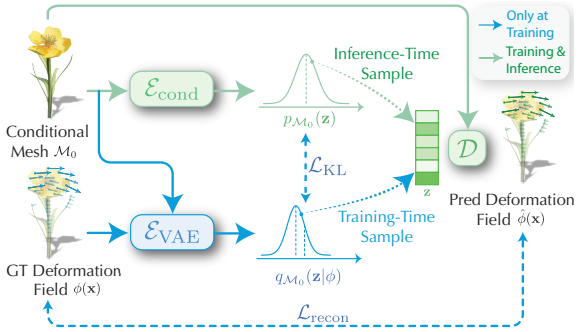


Figure 4. **Generative learning of NEUROK.** During training, we randomly sample an instance mesh and one of its possible deformation fields from the training set, and supervise all three models with KL and reconstruction targets. During inference, we only use $\mathcal{E}_{\text{cond}}$ to obtain the prior distribution $p_{\mathcal{M}_0}(\mathbf{z})$ for the instance $\mathcal{M}_0$ and sample from this distribution a latent, which is further decoded to a predicted deformation field with decoder $\mathcal{D}$.

state space $\mathcal{Z}(\mathcal{M}_0)$ associated with $\mathcal{M}_0$ by studying a surrogate task: learning a generative distribution $p_{\mathcal{M}_0}(\phi)$ over all plausible deformation fields[2] $\phi(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}^3$ of $\mathcal{M}_0$. Concretely, we train a conditional variational auto-encoder [57] to learn three models to approximate the instance-specific prior distribution $p_{\mathcal{M}_0}(\phi)$:

1. A **kinematic prior encoder** $\mathcal{E}_{\text{cond}}(\mathcal{M}_0)$ that takes in the conditioning input mesh and outputs the parameters for a prior distribution $p_{\mathcal{M}_0}(\mathbf{z})$ over the latent space $\mathbb{R}^k$.
2. A **variational deformation encoder** $\mathcal{E}_{\text{VAE}}(\phi, \mathcal{M}_0)$ that takes in a deformation field ϕ and a conditional mesh $\mathcal{M}_0$ and produces the parameters of a posterior distribution $q_{\mathcal{M}_0}(\mathbf{z} \mid \phi)$.
3. A **deformation decoder** $\mathcal{D}(\mathbf{z}, \mathcal{M}_0)$ that takes in a sampled latent $\mathbf{z}$ from the conditional prior distribution $p_{\mathcal{M}_0}(\mathbf{z})$ and decodes it into a deformed mesh $\mathcal{M}_{\mathbf{z}}$.

After learning these three models, we extract the high-density region of the latent probability distribution $p_{\mathcal{M}_0}(\mathbf{z})$ as the NEUROK kinematic state space $\mathcal{Z}(\mathcal{M}_0)$, and use the probabilistic decoder $\mathcal{D}(\mathbf{z}, \mathcal{M}_0)$ as the NEUROK mapping $\mathcal{F}(\cdot; \mathcal{M}_0)$.

An overview of these models can be found in Fig. 4. We design these three models with scalable transformer-based architectures and train them on a large-scale 4D dataset to let them learn generalizable kinematic priors.

4.1. Model Architecture

We now discuss the model architectures of $\mathcal{E}_{\text{cond}}, \mathcal{E}_{\text{VAE}}$, and $\mathcal{D}$. As a general principle, we use transformers [111] as backbones to ensure that they scale well to large-scale datasets.

Kinematic Prior Encoder. $\mathcal{E}_{\text{cond}}$ takes a conditional mesh $\mathcal{M}_0$ as input and outputs the kinematic prior distribution for $\mathcal{M}_0$. To encode $\mathcal{M}_0$, we evenly sample n_{sample} points from the surface of the input mesh to form a point cloud V_{sample}. We then use the position embedding layer following 3DShape2Vecset [129] to obtain point-wise features $\mathbf{F}_{\text{cond}} \in \mathbb{R}^{n_{\text{sample}} \times F_{\text{pos}}}$, where F_{pos} is the feature dimension of position embeddings. To allow the encoder to take varying numbers of point samples from a single mesh during encoding, we adopt a perceiver-based architecture [48, 132] and store a series of learnable tokens $\{\mathbf{e}_i\}_{i=1}^K$, where K is the number of tokens. With the learnable tokens, we apply multiple blocks of cross-attention and self-attention layers to obtain K encoded features $\{\mathbf{f}_i\}_{i=1}^K$. We flatten the features to form $\mu_{\text{cond}} \in \mathbb{R}^{K \times F_{\text{token}}}$, where F_{token} is the dimension of each token. We use the normal distribution $\mathcal{N}(\mu_{\text{cond}}, \mathbf{I})$ as the instance-specific prior distribution $p_{\mathcal{M}_0}(\mathbf{z})$.

Variational Deformation Encoder. $\mathcal{E}_{\text{VAE}}$ outputs posterior distribution $q_{\mathcal{M}_0}(\mathbf{z} \mid \phi)$ by taking two inputs: a deformation field ϕ and an instance-specific mesh $\mathcal{M}_0$. To parameterize these inputs, at training time, we sample a deformed mesh of $\mathcal{M}_0$. This deformed mesh is represented as $\mathcal{M}_{\mathbf{z}} = (V_{\mathbf{z}}, F)$ with a shared topology F as $\mathcal{M}_0 = (V_0, F)$. Similar to $\mathcal{E}_{\text{cond}}$, we sample a point cloud V_{sample} on the surface of $\mathcal{M}_0$. We then compute the vertex deformation[3] $\delta_{\mathbf{z}} = V_{\mathbf{z}} - V_0$ from $\mathcal{M}_0$ to $\mathcal{M}_{\mathbf{z}}$ and use barycentric interpolation to compute the deformation vector $\delta_{\text{sample}} \in \mathbb{R}^{d_{\text{deform}} \times n_{\text{sample}}}$ of the sampled points, where d_{deform} is the dimensionality of the deformation representation. We then concatenate δ_{sample} with the position vectors of V_{sample} and encode it using the position embedding layer [129] to get the point-wise feature $\mathbf{F}_{\text{VAE}}$ as inputs to the transformer. We similarly use a perceiver-based [48] architecture and store $2 \times K$ learnable tokens. These tokens are mapped to $2 \times K$ features $\{\mathbf{f}_i^{\text{VAE}}\}_{i=1}^{2 \times K}$. We separate those features into two sets and flatten them to represent the mean $\mu_{\text{VAE}} \in \mathbb{R}^{K \times F_{\text{token}}}$ and variance $\sigma_{\text{VAE}} \in \mathbb{R}^{K \times F_{\text{token}}}$ of the posterior. The output posterior distribution $q_{\mathcal{M}_0}(\mathbf{z} \mid \phi)$ is modeled as a Gaussian distribution $\mathcal{N}(\mu_{\text{VAE}}, \sigma_{\text{VAE}})$.

Deformation Decoder. $\mathcal{D}(\mathbf{z}, \mathcal{M}_0)$ is a decoder that decodes sampled latent $\mathbf{z}$ to a deformed mesh from $\mathcal{M}_0$. To implement this, we sample n_{sample} points from the surface of the input mesh to form a query point cloud V_{query}. As the latent space has a dimensionality of $K \times F_{\text{token}}$, we reshape $\mathbf{z}$ into K latent tokens $\{\mathbf{e}_i\}_{i=1}^K$, each with F_{token} dimensions. We then pass the query point cloud and the latent tokens to several blocks of self-attention and cross-attention layers, and predict n_{sample} features $\{\mathbf{f}_i\}_{i=1}^{n_{\text{sample}}}$. We further pass the features into an MLP to get the final deformation vectors $\delta_{\text{pred}} = \{\delta_i\}_{i=1}^{n_{\text{sample}}}$. We deform V_{query} using the predicted deformation vectors, and drive the mesh vertices V_0 by aver-

[2] To parameterize deformation fields for use in neural networks, we sample points on the mesh and treat their deformations as the parameterization of ϕ.

[3] Practically, we parameterize the deformation of each point with dual quaternions. See Appendix C.1 for more discussion.

aging the deformations over K_{drive} nearest sampled points.

4.2. Dataset and Training

All three models are trained simultaneously on a large-scale 4D dataset of deforming meshes of dynamic objects. We construct this dataset by curating instances from existing works [27, 119] and physical simulation. The details of the dataset can be found in Appendix D.

At each training iteration, we randomly sample an instance from all training instances of the dataset. For this instance, we randomly select two frames in its deformation sequence and obtain two meshes with shared topology. We use the first mesh as $\mathcal{M}_0$ and sample the deformation from the first mesh to the second mesh to form the sampled deformation vector δ_{sample}. These are passed into three models to get the reconstructed deformation δ_{pred}. The models are supervised with the standard conditional VAE target:

$$\mathcal{L} = ||\delta_{\text{sample}} - \delta_{\text{pred}}||_2^2 + \lambda D_{KL}(q_{\mathcal{M}_0}(\mathbf{z} \mid \phi)||p_{\mathcal{M}_0}(\mathbf{z})), \quad (1)$$

where λ is a hyper-parameter and we set $\lambda = 0.01$.

4.3. Dimension Reduction

The raw latent space of the learned VAE can be high-dimensional. To obtain a reduced-order latent space, we further perform a dimension reduction process to compress $\mathcal{Z} \subseteq \mathbb{R}^k$ to a lower-dimensional latent space $\mathcal{Q} \subseteq \mathbb{R}^{k_q}$, where $k_q \ll k$.

We perform the dimension reduction through the Active Subspace Method [24] that reduces the dimensionality of a high-dimensional space $\mathcal{Z}$ by considering a surrogate function $\mathcal{G}(\mathbf{z}) = g(A\mathbf{z} + \epsilon(\mathbf{z}))$, where $G : \mathbb{R}^k \to \mathbb{R}$, $A \in \mathbb{R}^{k_q \times k}$, and $g : \mathbb{R}^{k_q} \to \mathbb{R}$. In this way, the span of the rows of A identifies the directions that matter for $\mathcal{G}$ [7]. We define G in a way that identifies the influence of $\mathbf{z} \in \mathcal{Z}$ on the predicted deformation. Therefore, we formalize G as the 2-norm of δ_{pred} predicted from a set of sampled points on $\mathcal{M}_0$.

5. Generative 4D Simulation

With the predicted NEUROK, our initial task of generating a dynamic sequence of meshes is converted into generating a series of $\{\mathbf{z}_i\}_{i=1}^T$, with $\mathbf{z} \in \mathcal{Z}(\mathcal{M}_0)$. Note that our mapping $\mathcal{F}$ in the learned NEUROK will map any sampled latent $\mathbf{z}$ to a plausibly deformed shape that corresponds to a valid configuration of the studied object-centric physical system. This observation motivates us to use methods from Lagrangian mechanics [58] to generate such dynamics.

5.1. Preliminaries: Lagrangian Mechanics

Lagrangian mechanics studies a physical system by defining a set of parameters $\{\mathbf{q}_1, \cdots, \mathbf{q}_k\}$ that completely define the state of the system in a *configuration space*. Such parameters are called *generalized coordinates* of the system, and their time derivatives $\dot{\mathbf{q}}$ are called *generalized velocities*.

From this perspective, $\mathcal{Z}(\mathcal{M}_0)$ effectively forms a configuration space of the studied object-centric physical system, and any $\mathbf{z} \in \mathcal{Z}(\mathcal{M}_0)$ is a vector of generalized coordinates of the system. Therefore, we can generate the dynamics of $\mathbf{z}$ by using principles in Lagrangian mechanics.

Lagrangian mechanics solves the dynamics of generalized coordinates by defining a smooth function L over the latent space and solving the Euler-Lagrange equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\mathbf{z}}} = \frac{\partial L}{\partial \mathbf{z}}. \quad (2)$$

For most physical systems we study in this paper, we define Lagrangian function $L(\mathbf{z}, \dot{\mathbf{z}}) = T(\mathbf{z}, \dot{\mathbf{z}}) - V(\mathbf{z})$ using the kinetic energy T and potential energy V of the system.

5.2. Euler-Lagrange Equations for NEUROK

With the defined Lagrangian functions and the learned NEUROK, we solve the dynamics of $\mathbf{z} \in \mathcal{Z}(\mathcal{M}_0)$ with:

$$mG(\mathbf{z})\ddot{\mathbf{z}} + C(\mathbf{z}, \dot{\mathbf{z}}) + \nabla_{\mathbf{z}} V = 0, \quad (3)$$

where $G(\mathbf{z}) = J_{\mathbf{z}}^T J_{\mathbf{z}}$, $J_{\mathbf{z}}$ is the Jacobian of $\mathcal{F}$, $C_i = m\sum_{j,k}\Gamma_{ijk}(\mathbf{z})\dot{\mathbf{z}}_j\dot{\mathbf{z}}_k$, Γ_{ijk} is the Christoffel symbol. Its derivation can be found in Appendix B.3. We solve it with numerical solvers and get the trajectory of $\{\mathbf{z}_i\}_{i=1}^T$.

5.3. Boundary Conditions

Our system takes in conditions such as actions to generate 4D dynamics. They are incorporated by optimizing $(\mathbf{z}_0, \dot{\mathbf{z}}_0)$ to minimize $||\mathbf{x}_0 - \mathcal{F}(\mathbf{z}_0)||_2^2 + ||\dot{\mathbf{x}}_0 - J_{\mathbf{z}}\dot{\mathbf{z}}_0||_2^2$, where $\mathbf{x}_0, \dot{\mathbf{x}}_0$ are input positions and velocities of selected particles of the system. After solving $(\mathbf{z}_0, \dot{\mathbf{z}}_0)$, they serve as the initial condition for solving Eq. (3). See Appendix B.6 for details.

6. Experiments

6.1. Neural Object Kinematics Learning

We evaluate the effectiveness of NEUROK for learning kinematic state parameterization. Given an object and a target pose of the same object, we estimate an initial kinematic state and identify the optimal latent state vector that deforms the input shape to match the target. For quantitative assessment, we use the test dataset of PartNet-Mobility [119], and evaluate reconstruction accuracy using Chamfer distances [33] and a volumetric consistency metric (IoU).

Baselines. We evaluate two types of baselines for learning object kinematics. NeuralDeformationGraphs (NDG) [11], CANOR [43], and KeyPointDeformer (KPD) [49] model kinematics using implicit representations. FreeArt3D [19] and SINGAPO [83] explicitly learn articulation structures, limiting them to specific object categories.

Results. As shown in Tab. 1 and Fig. 5, our framework consistently outperforms existing methods in inverse kinematics, demonstrating its flexibility and effectiveness.

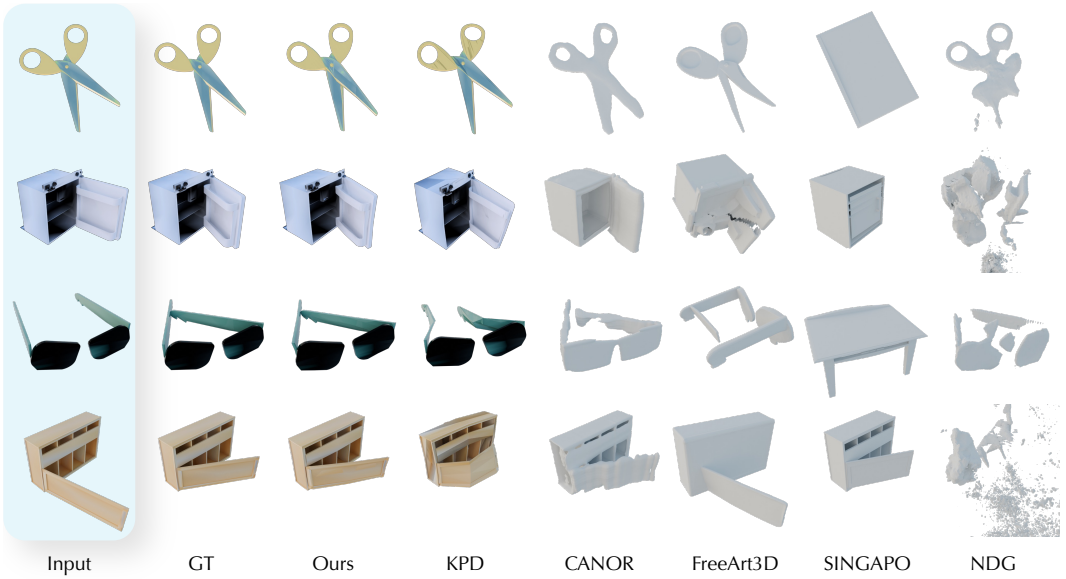


Figure 5. **Qualitative comparison on learning object kinematics.** We evaluate different methods on learning compact and smooth kinematic spaces. Given an input object and the shape of a target pose, we perform inverse kinematics and find the best-matching kinematic state. We compare how well the reconstructed shape decoded from the obtained state vectors matches the target.

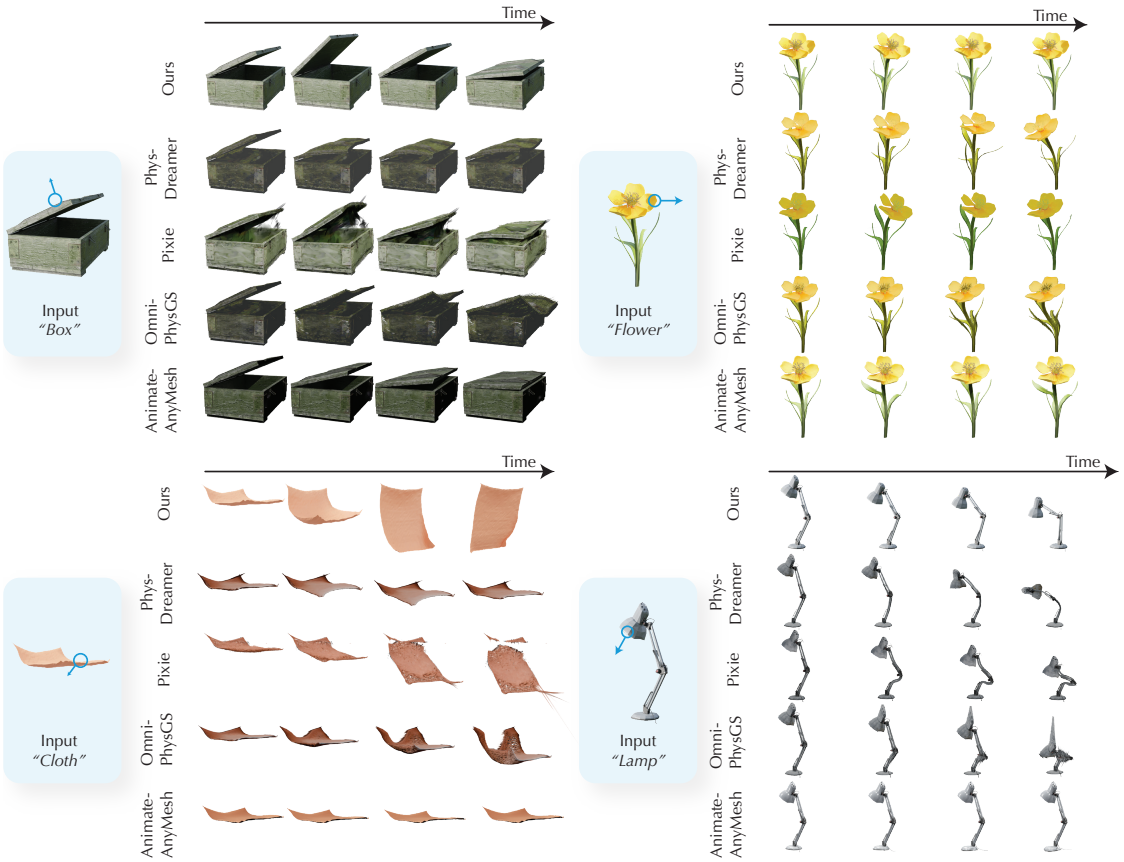


Figure 6. **Qualitative comparison on physically-inspired 4D generation.** We compare against baselines on the task of generating physically-plausible 4D motion given a single shape and conditioning actions.

6.2. Generative 4D Simulation

We show that our pipeline generates 4D simulative dynamics for diverse objects, evaluated across eight objects.

Baselines. We compare against representative methods for generating 4D dynamics from 3D shapes. PhysDreamer [130] distills physical parameters from video

Table 1. **Quantitative comparison on inverse-kinematics optimization.**

	Chamfer (L1) ↓	Chamfer (L2) ↓	IoU ↑
NeuralDeformationGraphs [11]	0.670	0.724	0.289
SINGAPO [83]	0.313	0.200	0.091
FreeArt3D [19]	0.169	0.139	0.354
CANOR [43]	0.082	0.067	0.568
KeyPointDeformer [49]	0.067	0.067	0.570
NeuROK (ours)	**0.028**	**0.028**	**0.764**
NeuROK w/o Model Reduction	0.045	0.059	0.711
NeuROK w/o Data Augmentation	0.036	0.041	0.724
NeuROK w/o Dual-Quaternion	0.033	0.037	0.728

Table 2. **Quantitative comparison on physically-inspired generation.** We report user study preferences along with metrics from VBench [46] and WorldScore [32]. AQ: Aesthetic Quality, DD: Dynamic Degrees, IQ: Imaging Quality, CLIP: CLIP score [101], MM: Motion Magnitude.

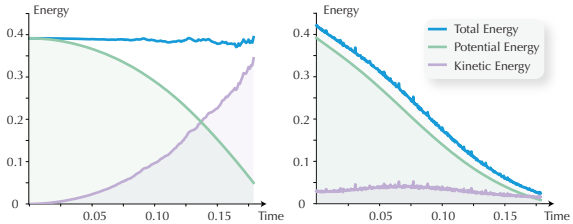
	User Study		VBench [46]			WorldScore [32]	
	Alignment ↑	Realism ↑	AQ ↑	DD ↑	IQ ↑	CLIP ↑	MM ↑
PhysDreamer [130]	5.95%	5.36%	0.362	0.500	48.432	0.716	0.783
OmniPhysGS [81]	1.67%	0.48%	0.380	0.625	48.937	0.690	0.544
Pixie [60]	5.12%	4.17%	0.392	0.625	46.177	0.659	0.857
AnimateAnyMesh [117]	5.83%	6.67%	0.450	0.625	48.370	0.730	0.889
NeuROK (ours)	**81.43%**	**83.33%**	**0.483**	**0.750**	**51.100**	**0.761**	**2.343**

Figure 7. **Simulating real objects.** Our model can be used to simulate real-captured objects. See our project page for more results.

models, Pixie [60] predicts simulation parameters using amortized-inference networks, and OmniPhysGS [81] represents each asset with material-aware Constitutive Gaussians for general physics-based dynamics. AnimateAnyMesh [117] is an end-to-end 4D generator trained on large-scale 4D data.

Metrics. Predicting 4D dynamics from 3D shapes is inherently ambiguous, so we evaluate plausibility and visual quality of the generated motions. A user study with 105 users assesses action alignment and realism. We also report metrics from VBench [46] and WorldScore [32].

Results. Quantitative and qualitative comparisons in Tab. 2 and Fig. 6 show that existing baselines perform well only within their specialized domains. Physically based methods (PhysDreamer [130], OmniPhysGS [81], Pixie [60]) can handle certain material categories but generalize poorly, while end-to-end methods (AnimateAnyMesh [117]) lack fine-grained conditioning and struggle on rarely encountered object types. Across all settings, our method consistently generates the most physically plausible and visually realistic 4D dynamics, demonstrating strong generalization to diverse object categories.



(a) Energy Evolution of Generated Traj. with Euler-Lagrangian Dynamics

(b) Energy Evolution of Interpolated Traj. w/o Euler-Lagrangian Dynamics

Figure 8. **Analysis of energy conservation.** Our approach maintains physical consistency in the generated trajectories through Euler–Lagrangian modeling. Under this formulation, the total energy of the simulated motion remains approximately constant.

Figure 9. **Generalization on unseen categories.** Our model can generalize to novel object categories that are completely not present in the training data.

Simulating Real Objects. Our pipeline can also simulate and manipulate real scenes. We scan a real scene and apply our approach to simulate the dynamics of the objects within it. As shown in Fig. 7, our method successfully simulates the closing motion of the laptop on the desk.

6.3. Analysis and Ablation Studies

Analysis of Physical Consistency. We analyze the physical consistency of the proposed framework in Fig. 8. As demonstrated, our method preserves the basic conservation law of energy by leveraging a physically-inspired framework from Lagrangian mechanics.

Generalization on Unseen Categories. Our method learns common dynamic structures from the training dataset and successfully generalizes them to entirely new object categories. As shown in Fig. 9, a NeuROK variant trained only on PartNet-Mobility [119] categories can still generate plausible dynamics for unseen object types.

Ablation Studies. We evaluate the impact of key design choices in Tab. 1. The results show that model reduction, training data augmentation, and our deformation parameterization each contribute significantly to the overall performance of the proposed framework.

7. Conclusion

We have introduced a novel framework, NeuROK, for generating 4D simulative dynamics from static 3D shapes, bridging physical principles and learned latent spaces through a physically inspired formulation. Our work opens up promising future research directions and introduces a new research paradigm in 4D visual generation.

Appendix

In this appendix, we provide detailed theory, formulations, implementation notes, and experimental settings that could not be included in the main paper due to space constraints. We expand on the core theoretical results and formulations introduced in the main text, describe additional implementation details and the dataset construction process, and present a comprehensive account of our experimental setup and results. We also report supplementary experiments, offer further analysis, and discuss limitations and directions for future work. To keep this part self-contained, we occasionally reuse text and figures from the main paper. We refer readers to our project page at https://chen-geng.com/neurok.

Contents

A. Notations

The following table summarizes the main symbols used throughout this paper.

Table 3. Summary of the main symbols and notations used in the paper.

Symbol	Meaning
A	Projection matrix to reduced subspace.
$C_i(\mathbf{z}, \dot{\mathbf{z}})$	Coriolis-like term.
D_{KL}	KL divergence.
δ_z	Vertex deformation from $\mathcal{M}_0$ to $\mathcal{M}_{\mathbf{z}}$.
δ_{pred}	Predicted sampled-point deformations.
δ_{sample}	Deformation vectors of the sampled points on the surface.
$\mathcal{D}(\mathbf{z}, M_0)$	Decoder that decodes a latent $\mathbf{z}$ to a deformation field ϕ.
d_{deform}	Dimension of deformation representation. For this paper, it is 8 due to the dual-quaternion representation.
$\mathcal{E}_{\mathrm{VAE}}(\phi, \mathcal{M}_0)$	Posterior encoder producing the posterior distribution $q_{\mathcal{M}_0}(\mathbf{z}\|\phi)$.
$\mathcal{E}_{\text{cond}}(\mathcal{M}_0)$	Encoder producing prior kinematic distribution for $\mathcal{M}_0$.
$\{\mathbf{e}_i\}_{i=1}^K$	Learnable latent tokens.
F	Face set shared across meshes.
F_{pos}	Positional feature dimension of the positional embedding.
F_{token}	Dimensionality of each token.
$\mathbf{F}_{\mathrm{VAE}}$	Point-wise features in the encoding process of $\mathcal{E}_{\mathrm{VAE}}$.
$\mathbf{F}_{\text{cond}}$	Point-wise features in the encoding process of $\mathcal{E}_{\text{cond}}$.
$\mathbf{F}(\mathbf{x}, t)$	External physical-space force field applied to the object.
$\mathcal{F} : \mathcal{Z} \to \mathbb{R}^{3n}$	Decoder mapping state vector $\mathbf{z}$ to vertex configuration of n vertices.
$\{\mathbf{f}_i\}_{i=1}^K$	Transformer token features.
$\mathcal{G}(z)$	Function for sensitivity of the Active Subspace Method.
$G(z) = J_z^\top J_z$	Latent-manifold metric.
Γ_{ijk}	Christoffel symbols.
$g(\cdot)$	Low-dimensional surrogate in the Active Subspace Method.
J_z	Decoder Jacobian of $\mathcal{F}$.
K	Number of latent transformer tokens.
K_{drive}	Number of nearest driver points for vertex deformation.
k, k_q	Raw latent dimension and reduced dimension.
k_{int}	Intrinsic DOFs of the deformation space.
$L(z)$	Lagrangian function in the Lagrangian Mechanics.
λ	KL weighting hyperparameter.
$\mathcal{L}$	VAE training loss.
$\mathcal{M}_t = (V_t, F)$	Deformed mesh at time step t.
$\mathcal{M}_z = (V_z, F)$	Mesh decoded from latent state z.
$\mathcal{M}_0 = (V_0, F)$	Canonical reference mesh of the dynamic object with vertex set V_0 and face set F.
$\mu_{\mathrm{VAE}},\ \sigma_{\mathrm{VAE}}$	Parameters of the Gaussian distribution predicted by $\mathcal{E}_{\mathrm{VAE}}$.
μ_{cond}	Mean of the prior distribution $p_{\mathcal{M}_0}(\mathbf{z})$.
m	Mass parameter.
$\nabla_z V$	Gradient of potential.
n	Number of vertices for a certain mesh. Note that different meshes can have different number of vertices.

Continued on next page

Symbol	Meaning
n_{sample}	Number of sampled points of a point cloud from a surface.
$\mathbf{P}(\mathbf{z})$	Jacobian $\partial\pi/\partial\mathbf{z}$ of the reduction map.
π	Reduction map from the redundant latent $\mathbf{z}$ to the minimal generalized coordinates $\mathbf{q}$.
Ψ	Embedding of the minimal coordinates $\mathbf{q}$ into the physical vertex space $\mathbb{R}^{3n}$.
$\phi(\mathbf{x})$	A deformation field that maps any point $\mathbf{x} \in \mathbb{R}^3$ to the deformation of this point.
$p_{\mathcal{M}_0}(\phi)$	Distribution over plausible deformations ϕ for $\mathcal{M}_0$.
$p_{\mathcal{M}_0}(\mathbf{z})$	Instance-specific prior distribution of $\mathcal{M}_0$ over latent states.
$Q \subseteq \mathbb{R}^{k_q}$	Reduced latent space via Active Subspace Method.
$\mathbf{Q}(\mathbf{z}, \dot{\mathbf{z}}, t)$	Generalized force conjugate to $\mathbf{z}$, given by the pullback $J^T(\mathbf{z})\mathbf{F}$.
$\mathbf{q}_i,\ \dot{\mathbf{q}}_i$	Minimal generalized coordinates and velocities in Lagrangian mechanics (the redundant learned latent is $\mathbf{z}$).
$q_{\mathcal{M}_0}(\mathbf{z} \mid \phi)$	Posterior over latent states.
T	Length of mesh sequence.
$T(z), V(z)$	Kinetic and potential energy.
V_0, V_t, V_z	Vertex sets of $\mathcal{M}_0$, $\mathcal{M}_t$, $\mathcal{M}_z$.
V_{query}	Query point cloud.
V_{sample}	Sampled surface point cloud to encode a 3D shape.
$\mathcal{V}^{k_{\text{int}}}$	The manifold of all plausible deformations for $\mathcal{M}_0$.
$\mathbf{x}^0, \mathbf{x}^t$	Concatenated vertex coordinates of M_0 and M_t.
$\mathbf{x}_0, \dot{\mathbf{x}}_0$	Initial particle positions/velocities.
$(\mathbf{z}_0, \dot{\mathbf{z}}_0)$	Initial conditions.
$\mathcal{Z} \subseteq \mathbb{R}^k$	The domain of a kinematic state parameterization with k dimensions
$\mathcal{Z}(\mathcal{M}_0)$ or $\mathcal{Z}_{\mathcal{M}_0}$	Object-specific latent state space associated with $\mathcal{M}_0$.
$\mathbf{z}$	A sampled state vector from $\mathcal{Z}$.
$\{\mathbf{z}_t\}_{i=1}^T$	Latent trajectory.

B. Theory, Methodology and Examples

In this section, we provide a more comprehensive discussion on the concepts, theory, and methodologies proposed in the main paper. We reused some of the figures and texts to keep the content self-contained.

B.1. Formulation and Concepts

Problem formulation. This paper studies generating simulative dynamics of 3D object-centric physical systems. In this paper, we colloquially define an object-centric physical system as one in which most observed motion arises from a single dominant deformable object. Our pipeline takes a static snapshot of a 3D dynamic object and a set of physical conditions (*e.g.,* actions, forces, initial velocities) as inputs, and generates a sequence of temporally evolving 3D shapes.

Scope of the problem. As a single 3D snapshot of an object cannot fully determine its physical parameters, our goal is to *generate* one plausible 4D sequence that satisfies one valid physical configuration and conforms to human physical intuition [4]. We assume no kinematic or physical priors on the dynamic structure of the modeled object. The object can be articulated, rigid, a continuum body, or even a heterogeneous combination of several dynamic types.

We assume that the modeled object has a low-dimensional kinematic space and a concrete shape. Therefore, we do not model phenomena involving fluid in this paper. We also assume that different kinematic states of the input object can be deformed from the shape of any other kinematic states with a shared mesh topology. Therefore, this paper does not handle objects which change their topologies during transformations (*e.g.*, flower blooming).

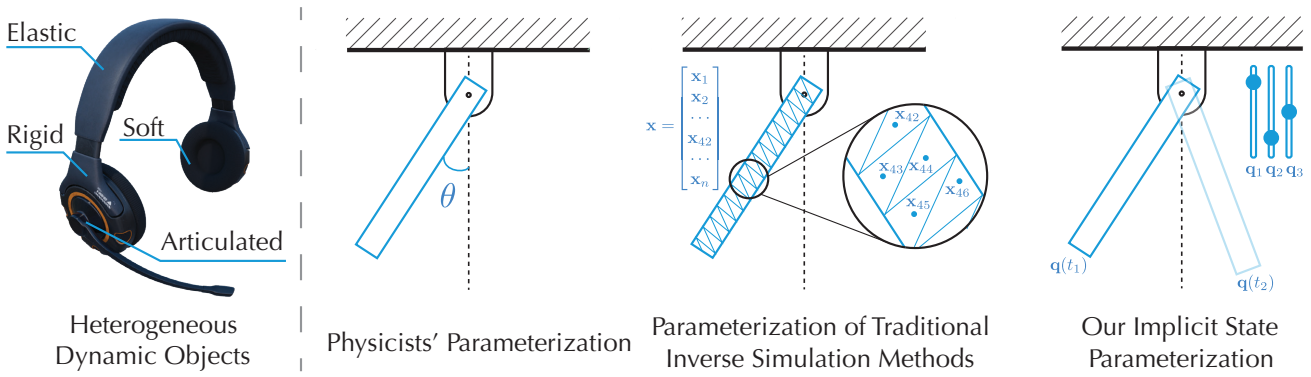


Figure 10. **Examples. (Left)** A dynamic object in our daily life can have diverse and heterogenous dynamic structures. Traditional approaches model each type of dynamic structure individually with domain-specific physical equations, which is not scalable to in-the-wild 4D datasets. **(Right)** Several kinematic state parameterizations can be used to describe a physical system. The symbolic parameterization used in classical mechanics are concise yet not accessible in inverse problems. Traditional inverse simulation approaches use geometry-derived parameterizations, yet require dense physical constraints to solve the over-parameterized system. We instead learn a low-dimensional parameterizations that are both compact and learnable from data.

Discussion. In this setting, ambiguities in the generated dynamics are expected because the input contains only static 3D objects. Our goal is visual 4D generation with motion that looks physically reasonable, which is a common aim in the 4D generation community [77, 117, 130]. We are not trying to predict exact physical trajectories, a task that is not feasible when starting from static inputs alone. Even so, this generative setup is useful for downstream tasks. In robotics, for example, researchers often need to create objects with different plausible physical properties to build simulation environments that help policies generalize [63, 119]. Similarly, in AR/VR and visual content creation, perfect physical accuracy is usually not the main concern; what matters most is that the generated motion looks plausible.

Geometry representation. The input geometry of the modeled object is represented as a mesh $\mathcal{M}_0 = (V_0, F)$ with n vertices. We denote by $\mathbf{x}^0 \in \mathbb{R}^{3n}$ the concatenated vertex positions in V_0. Our pipeline outputs a sequence of deformed meshes with timestamps ranging from 1 to T, denoted as $\{\mathcal{M}_1, \cdots, \mathcal{M}_T\}$, where $\mathcal{M}_t = (V_t, F)$, and the concatenated vertex positions are represented by $\mathbf{x}^t \in \mathbb{R}^{3n}$.

While the vertices of the mesh $\mathcal{M}_0$ can theoretically take arbitrary positions in $\mathbb{R}^{3n}$, only a small subset of these configurations correspond to plausibly reposed shapes. In fact, a randomly sampled deformation vector from $\mathbb{R}^{3n}$ will almost certainly yield a deformed mesh far outside the distribution of valid object poses. Empirically, the set of plausible vertex position vectors of a dynamic object forms a low-dimensional configuration manifold $\mathcal{V}^{k_{\text{int}}}$ embedded in $\mathbb{R}^{3n}$, where k_{int} denotes the intrinsic degrees of freedom of the deformation space and $k_{\text{int}} \ll 3n$.

Kinematic state parameterization. When studying these object-centric physical systems containing a deformable mesh with n vertices, we need to define a parameterization scheme for its kinematic states, which in turn determines the solution space for a physical simulator. We generalize this with the following definition:

Definition 3: Kinematic State Parameterization

A k-dimensional *kinematic state parameterization* for a dynamic object is a pair $(\mathcal{Z}, \mathcal{F})$, where $\mathcal{Z} \subseteq \mathbb{R}^k$ is the state space of the parameterization, and $\mathcal{F} : \mathcal{Z} \to \mathbb{R}^{3n}$ is a mapping that maps any state vector $\mathbf{z} \in \mathcal{Z}$ to a vertex configuration of $\mathcal{M}_0$.

As illustrated in Fig. 11, we do not assume the image of $\mathcal{F}$ is $\mathcal{V}^{k_{\text{int}}}$, therefore some kinematic state $\mathbf{z} \in \mathcal{Z}$ can be mapped to pure noisy shapes.

Some parameterizations are compact, and some are redundant. Consider the following example:

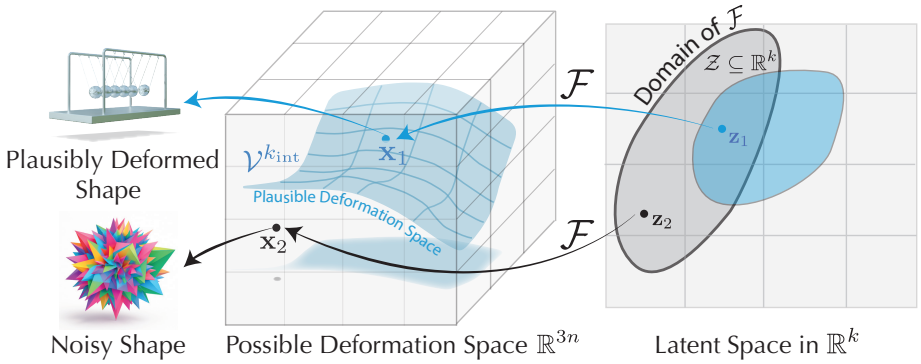


Figure 11. **Illustration of Kinematic State Parameterization.** As formally defined in Def. 3, a kinematic state parameterization studied in this paper is a pair $(\mathcal{Z}, \mathcal{F})$ which contains a latent manifold $\mathcal{Z}$ and a decoder $\mathcal{F}$ that maps a sampled latent to a deformation vector. This definition explicitly include those kinematic state parameterizations that are not compact. For a non-compact kinematic state parameterization, it is possible that some sampled latent $\mathbf{z}$ is mapped to implausibly deformed shapes.

Example 1: A pendulum with shape

In Fig. 10, a swinging pendulum's dynamic geometry can be either parameterized with its swing angle θ:

$$\mathcal{Z} = \mathbb{R}^1, \mathcal{F}(\theta) = (\mathbf{r}\sin\theta, -\mathbf{r}\cos\theta),$$

or with a trivial parameterization:

$$\mathcal{Z} = \mathbb{R}^{3n}, \mathcal{F}(\mathbf{x}) = \mathbf{x}.$$

For the trivial parameterization in Ex. 1, a randomly sampled $\mathbf{z} \in \mathcal{Z} = \mathbb{R}^{3n}$ will almost certainly produce a noisy shape. Therefore, for methods adapting similarly dense parameterizations, constraints or physical equations need to be introduced to keep the shape plausibly deformed. For example, if using MPM [52] to model this pendulum, strong elastic forces need to be introduced to keep particles from breaking apart. We formulating this intuition with the following claim.

Choice of kinematic state parameterization dictates the problem complexity. When simulating physical systems, physicists typically find concise parameterizations (*generalized coordinates* [58]) to simplify the solution space, similar to θ in Ex. 1. However, such symbolic parameterizations are generally inaccessible in an inverse-graphics setting, where only the raw 3D geometry $\mathcal{M}_0$ is given. Consequently, most approaches [81, 130] adopt natural, geometry-driven parameterization schemes, such as the high dimensional particles (material points) used in MPM [52].

To solve dynamics in the high-dimensional solution space defined by the redundant parameterization, researchers have introduced physical equations and constraints to prevent the system from being under-determined. These formulations have proven effective in forward graphics settings, where full scene descriptions are available as input. However, they become problematic in inverse-graphics scenarios when no physical priors on the modeled object are assumed, since the imposed constraints are not universal. Consequently, prior approaches [77, 130] remain restricted to specific target domains instead of generalizing across arbitrary dynamic objects.

B.2. Neural Object Kinematics

Motivation. We solve the above-discussed problem inspired by a simple observation: humans can imagine the dynamic structure of a dynamic object just by watching its shape and leveraging prior knowledge. If a neural network is trained on a large-scale of 4D data on how 3D shapes deform, it can theoretically learn similar priors. To this end, we introduce the following concept:

Method	$\mathcal{Z}$ (state space)	$\mathcal{F}: \mathcal{Z} \to \mathbb{R}^{3n}$	Application domain	Constraints
MPM [52]	$\mathbb{R}^{3N_p}$ (material particle positions)	Particles reconstructed into a mesh	Elastoplastic materials, granular media	Continuum mechanics PDE
Full-space FEM	$\mathbb{R}^{3n}$ (nodal displacements)	$\mathbf{x}^{\text{rest}} + \mathbf{u}$	Elastoplastic materials, cloth	Elastic energy
PBD / XPBD	$\mathbb{R}^{3n}$ (vertex positions)	$\mathbf{x}$	Real-time cloth, hair, soft bodies	Distance, bending, volume constraints
Projective Dynamics	$\mathbb{R}^{3n}$ (vertex positions)	$\mathbf{x}$	Cloth, soft bodies, shells	Strain energies, inertia term, iterative projections
Spring-based models	$\mathbb{R}^{3N_m}$ (mass positions)	Mass positions mapped to mesh geometry	Cloth, hair, simple soft bodies	Hookean springs, damping, fixed-point and collision constraints
Rigid bodies	$\mathbb{R}^{6N_b}$ (translations + orientations)	Rigid transforms applied to rest vertices	Rigid-body simulation	Joint constraints, non-penetration, friction, contact
Articulated + skinning	$\mathbb{R}^k$ (joint DOFs)	Skinning of rest mesh (LBS/DQS)	Characters, robots, articulated objects	Joint limits, torques, optional dynamics
Reduced FEM	$\mathbb{R}^k$	$\mathbf{x}^{\text{rest}} + \mathbf{Bq}$	Fast reduced deformable simulation	Reduced space from the defined physical constraints for a single system, projected elastic energy
NEUROK	$\mathbb{R}^k$	$\mathcal{D}(\mathbf{z}, \mathcal{M}_0)$	Objects with low-dimensional kinematic space	Learned prior embedded in $(\mathcal{Z}, \mathcal{F})$ with Lagrangian mechanics

Table 4. **Kinematic state parameterizations of typical simulation methods.** Most forward simulation techniques use a task-specific parameterization designed for a particular physical domain, but such parameterizations are usually unavailable in an inverse setting where no category-specific priors can be assumed. Existing inverse-graphics approaches [130] typically adopt one of these simulation formulations directly, which restricts them to that domain. Our approach instead learns physical constraints from large-scale 4D object data and reduces reliance on manual, category-specific modeling. Reduced FEM methods are seemingly similar to NEUROK in that they learn a low-dimensional state space, but they focus on accelerating a known physical system (e.g., elastic objects with continuum mechanics). Consequently, they learn a reduced model for a specific physical system rather than an amortized-inference feed-forward model for generalizable object priors, and their simulations still depend on the physical constraints of the underlying full system.

Definition 4: Neural Object Kinematics

If a k-dimensional kinematic state parameterization $(\mathcal{Z}, \mathcal{F})$ uses a neural network to represent $\mathcal{F}$, and the range of $\mathcal{F}$ is $\mathcal{V}^{k_{\text{int}}}$, we refer to it as a **Neur**al **O**bject **K**inematics (NEUROK) of a dynamic object.

A successfully learned NEUROK simplifies the physical system. After NEUROK is successfully learned, any latent vector $\mathbf{z}$ sampled from $\mathcal{Z}$ will be mapped to a validly deformed input mesh. This greatly simplifies the simulation process, as we no longer need to add category-specific constraints to maintain geometric plausibility.

Using this concept, our proposed generative 4D simulation pipeline comprises two components: NEUROK learning and generating latent dynamics. The first component uses a transformer-based encoder-decoder model to learn an NEUROK for dynamic objects from a large-scale 4D dataset, and the second component provides a physically-based approach to generate a dynamic sequence $\{\mathbf{z}_1, \cdots, \mathbf{z}_T\}$ to be decoded back into dynamics meshes.

As a concrete example on why NEUROK can simplify the simulation, let's again consider the example in Ex. 1. If we directly model the particles forming the 3D shape of the rigid pendulum, we need to model the microscopic interactions between particles by discretizing the geometric volume and apply Newton's law. Using NEUROK as our kinematic state parameterization allows us to view the problem from a global, systematic perspective: we only need to consider the global

energy of the system, as any configuration with NEUROK will yield a plausible shape. We no longer needs constraints just to keep the particles from breaking apart since these constraints have been learned from data.

NEUROK as generalized coordinates in Lagrangian mechanics. One key insight in this paper is that the learned NEUROK can enable physically-inspired 4D generation from Lagrangian Mechanics' perspective from the literature of classical physics. Lagrangian mechanics is an alternative formulation in classical mechanics. Compared to the widely-used Newtonian mechanics, it uses a configuration space as well as a smoothly defined function over the configuration space (named *Lagrangian*) to model a mechanical system. It eliminates the need of modeling individual forces in a complex physical system as opposed to the classical Newtonian mechanics. Lagrangian mechanics uses energy as its central element, and solves dynamics with an ODE formed with Euler-Lagrangian equations [58].

The framework of Lagrangian mechanics fits the general theme of our approach perfectly. The latent state space $\mathcal{Z}$ learned NEUROK as defined by Def. 4 constitutes a configuration space for the studied object-centric physical system, and the framework of Lagrangian mechanics allows us to only analyze the system's energy without defining category-specific forces between dense particles. Therefore, we study leveraging the principles in Lagrangian mechanics to generate the dynamics $\{\mathbf{z}_i\}_i^T$ on $\mathbf{z} \in \mathcal{Z}$. We detail this process as follows.

B.3. Lagrange Mechanics on NEUROK

In this section, we will first show that the Euler-Lagrangian equation still holds on the learned (potentially over-parameterized) manifold $\mathcal{Z}_{\mathcal{M}_0}$.

In this section, we discuss the process of using Lagrangian mechanics to solve dynamics of $\mathbf{z} \in \mathcal{Z}_{\mathcal{M}_0}$. As discussed above, $\mathbf{z}$ can be considered as the generalized coordinates of the physical system. Lagrangian mechanics solve the dynamics of such systems with the Euler-Lagrange equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\mathbf{z}}} = \frac{\partial L}{\partial \mathbf{z}}. \tag{4}$$

We need to define Lagrangian L for the studied physical systems to make it generalizable and category agnostic for most dynamic objects. Following conventions in classical mechanics, the Lagrangian $L(\mathbf{z})$ is defined as:

$$L(\mathbf{z}) = T(\mathbf{z}) - V(\mathbf{z}), \tag{5}$$

where $T(\mathbf{z})$ is the kinetic energy of the system under the configuration $\mathbf{z}$, and $V(\mathbf{z})$ is the potential of the system under this setting.

To define the energy of a physical system of a deformed mesh, we sample k_{sample} points $\mathbf{x} \in \mathbb{R}^{3\times k_{\text{sample}}}$ from the volume of the deformed object $\mathcal{M}_0$. We denote the kinetic and potential energy of the system parameterized with the sampled points with $T_{\mathbf{x}}(\cdot)$ and $V_{\mathbf{x}}(\cdot)$, and:

$$L(\mathbf{z}) = T_{\mathbf{x}}(\mathcal{D}(\mathbf{z}, \mathcal{M}_0)(\mathbf{x})) - V_{\mathbf{x}}(\mathcal{D}(\mathbf{z}, \mathcal{M}_0)(\mathbf{x})). \tag{6}$$

For the following, we denote the deformation field $\mathcal{D}(\mathbf{z}, \mathcal{M}_0)$ as $\mathcal{F}$ to simplify the notations.

We then define $T_{\mathbf{x}}$ and $V_{\mathbf{x}}$ such that they are general and category-agnostic for most dynamic objects of our interest. Note that the goal of this paper is to *generate* one possible 4D motion sequence for a given static 3D object. Therefore, to simplify the following derivation, without loss of generality, we choose one plausible configuration of the system of assuming an isotropic distribution of mass in the volume of the object, with m as mass parameter for each sampled particle. In this way, the $T_{\mathbf{x}}$ is defined as follows:

$$T_{\mathbf{x}}(\mathbf{x}) = \frac{1}{2} m ||\dot{\mathbf{x}}||_2^2 \tag{7}$$

$$= \frac{1}{2} m \dot{\mathbf{x}}^T \dot{\mathbf{x}}, \tag{8}$$

where we use $\dot{\mathbf{x}}$ to denote $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$ following convention in physics. In this way, $T(\mathbf{z})$ can be defined as:

$$T(\mathbf{z}) = T_{\mathbf{x}}(\mathcal{F}(\mathbf{z})) \tag{9}$$
$$= \frac{1}{2} m (\frac{\mathrm{d}}{\mathrm{d}t} \mathcal{F}(\mathbf{z}))^T \cdot (\frac{\mathrm{d}}{\mathrm{d}t} \mathcal{F}(\mathbf{z})) \tag{10}$$
$$= \frac{1}{2} m \dot{\mathbf{z}}^T J^T(\mathbf{z}) J(\mathbf{z}) \dot{\mathbf{z}} \tag{11}$$
$$= \frac{1}{2} m \dot{\mathbf{z}}^T G(\mathbf{z}) \dot{\mathbf{z}}, \tag{12}$$

where $J(\mathbf{z}) = \frac{\partial \mathcal{F}}{\partial \mathbf{z}}$ is the Jacobian of $\mathcal{F}$, $G(\mathbf{z}) = J^T(\mathbf{z}) \cdot J(\mathbf{z})$.

Remark. In this paper, we focus on generating one plausible 4D motion that conforms to intuitive physical reasoning. Therefore, we adapt one of possible physical parameters of the system to simplify the derivation. One can certainly define a general mass matrix for certain applications with the same derivation framework, yet this is beyond the scope of this 4D generation paper.

The potential $V_{\mathbf{x}}$ is designed with the same principle as $T_{\mathbf{x}}$: it should be general and category-agnostic, working for most dynamic objects within our interest, and corresponds to one possible physical configurations of the system. Therefore, we design the following potential for all physical systems we study in this paper:

$$V_{\mathbf{x}}(\mathbf{x}) = m\mathbf{g} \cdot \mathbf{x} + \frac{k_e}{2} \sum_{j \in \mathcal{N}(i)} (||\mathbf{x}_i - \mathbf{x}_j||_2 - ||\mathbf{x}_i^0 - \mathbf{x}_j^0||_2)^2, \tag{13}$$

where $\mathbf{g}$ is the gravitational acceleration vector, k_e is a constant, $\mathbf{x}^0$ denotes the initial locations of particles in $\mathcal{M}_0$, and $\mathcal{N}(i)$ denotes the neighbor points of point i. This energy is designed by analyzing the global behavior of the system, and empirically works for all physical systems studied in this paper for different categories (rigid body, articulated, cloth, elastic, etc.)

The generality of this energy comes from the learned physical prior in the latent state space $\mathcal{Z}$. As constraints on deformation have been fully learned in $\mathcal{Z}$, the energy function only plays an role in driving latent vectors to form a dynamic trajectory in the plausible deformation space. As an example, for most rigid objects, the second term of this energy function will be near to zero, as the predicted kinematic states of the object will not produce inter-particle deformations due to the prior learned in data. For articulated objects, we do not need to introduce the joint constraints as they are also learned in $\mathcal{Z}$ by training on a large-scale 4D dataset. With this potential:

$$V(\mathbf{z}) = V_{\mathbf{x}}(\mathcal{F}(\mathbf{z})). \tag{14}$$

And the final Lagrangian $L(\mathbf{z})$ is defined as:

$$L(\mathbf{z}) = T(\mathbf{z}) - V(\mathbf{z}) \tag{15}$$
$$= \frac{1}{2} m \dot{\mathbf{z}}^T G(\mathbf{z}) \dot{\mathbf{z}} - V(\mathbf{z}). \tag{16}$$

We then derive the ODE on $\mathbf{z}$ to solve the dynamic trajectory with Eq. (4). Analyzing the left hand side of the equation, we have:

$$\frac{\mathrm{d}}{\mathrm{d}t} (\frac{\partial L}{\partial \dot{\mathbf{z}}}) = mG(\mathbf{z})\ddot{\mathbf{z}} + m \sum_k \frac{\partial G}{\partial \mathbf{z}_k} \dot{\mathbf{z}} \cdot \dot{\mathbf{z}} \tag{17}$$
$$= m(G(\mathbf{z}) \cdot \ddot{\mathbf{z}} + \dot{G}(\mathbf{z}) \cdot \dot{\mathbf{z}}), \tag{18}$$

where:

$$\dot{G}(\mathbf{z}) = \sum_k \frac{\partial G}{\partial \mathbf{z}_k} \dot{\mathbf{z}}_k. \tag{19}$$

Analyzing the right hand side of Eq. (4), we have:

$$\frac{\partial L}{\partial \mathbf{z}} = \frac{1}{2} m \dot{\mathbf{z}}^T \frac{\partial G}{\partial \mathbf{z}} \dot{\mathbf{z}} - \nabla_{\mathbf{z}} V(\mathbf{z}) \tag{20}$$

$$= \frac{1}{2} m \dot{\mathbf{z}}^T \frac{\partial G}{\partial \mathbf{z}} \dot{\mathbf{z}} - J^T(\mathbf{z}) \nabla_x V(\mathbf{x}). \tag{21}$$

Combining Eq. (18) and Eq. (21), we have:

$$m(G(\mathbf{z})\ddot{\mathbf{z}} + \dot{G}(\mathbf{z}) \cdot \dot{\mathbf{z}}) = \frac{1}{2} m \dot{\mathbf{z}}^T \frac{\partial G}{\partial \mathbf{z}} \dot{\mathbf{z}} - J^T(\mathbf{z}) \nabla_{\mathbf{x}} V(\mathbf{x}). \tag{22}$$

Organizing this equation, we have:

$$mG(\mathbf{z})\ddot{\mathbf{z}} + m(\dot{G}(\mathbf{z})\dot{\mathbf{z}} - \frac{1}{2}\dot{\mathbf{z}}^T \frac{\partial G}{\partial \mathbf{z}} \dot{\mathbf{z}}) + J^T(\mathbf{z}) \nabla_{\mathbf{x}} V(\mathbf{x}) = 0. \tag{23}$$

To simplify this equation, we analyze the i-th term of $(\dot{G}(\mathbf{z})\dot{\mathbf{z}} - \frac{1}{2}\dot{\mathbf{z}}^T \frac{\partial G}{\partial \mathbf{z}} \dot{\mathbf{z}})$:

$$(\dot{G}(\mathbf{z})\dot{\mathbf{z}} - \frac{1}{2}\dot{\mathbf{z}}^T \frac{\partial G}{\partial \mathbf{z}} \dot{\mathbf{z}})_i = \sum_{jk} \frac{\partial G_{ij}}{\partial \mathbf{z}_k} \dot{\mathbf{z}}_k \dot{\mathbf{z}}_j - \frac{1}{2} \sum_{jk} \frac{\partial G_{jk}}{\partial \mathbf{z}_i} \dot{\mathbf{z}}_j \dot{\mathbf{z}}_k \tag{24}$$

$$= \sum_{jk} \frac{\partial G_{ij}}{\partial \mathbf{z}_k} \dot{\mathbf{z}}_k \dot{\mathbf{z}}_j - \frac{1}{2} \sum_{jk} \frac{\partial G_{jk}}{\partial \mathbf{z}_i} \dot{\mathbf{z}}_j \dot{\mathbf{z}}_k + \frac{1}{2} \sum_{jk} \frac{\partial G_{ik}}{\partial \mathbf{z}_j} \dot{\mathbf{z}}_j \dot{\mathbf{z}}_k - \frac{1}{2} \sum_{jk} \frac{\partial G_{ik}}{\partial \mathbf{z}_j} \dot{\mathbf{z}}_j \dot{\mathbf{z}}_k \tag{25}$$

$$= \frac{1}{2} \sum_{jk} \frac{\partial G_{ij}}{\partial \mathbf{z}_k} \dot{\mathbf{z}}_k \dot{\mathbf{z}}_j - \frac{1}{2} \sum_{jk} \frac{\partial G_{jk}}{\partial \mathbf{z}_i} \dot{\mathbf{z}}_j \dot{\mathbf{z}}_k + \frac{1}{2} \sum_{jk} \frac{\partial G_{ik}}{\partial \mathbf{z}_j} \dot{\mathbf{z}}_j \dot{\mathbf{z}}_k \tag{26}$$

$$= \frac{1}{2} \left[\frac{\partial G_{ij}}{\partial \mathbf{z}_k} + \frac{\partial G_{ik}}{\partial \mathbf{z}_j} - \frac{\partial G_{jk}}{\partial \mathbf{z}_i} \right] \dot{\mathbf{z}}_j \dot{\mathbf{z}}_k \tag{27}$$

$$= \sum_{jk} \Gamma_{ijk}(\mathbf{z}) \dot{\mathbf{z}}_j \dot{\mathbf{z}}_k, \tag{28}$$

where:

$$\Gamma_{ijk}(\mathbf{z}) = \frac{1}{2} \left[\frac{\partial G_{ij}}{\partial \mathbf{z}_k} + \frac{\partial G_{ik}}{\partial \mathbf{z}_j} - \frac{\partial G_{jk}}{\partial \mathbf{z}_i} \right]. \tag{29}$$

Plugging Eq. (28) into Eq. (23), we obtain the equation of motion governing the latent dynamics of NEUROK, which we summarize as the following result.

Theorem 1: Euler-Lagrange Equation on the Latent State Space $\mathcal{Z}_{\mathcal{M}_0}$

The dynamics of the generalized coordinates $\mathbf{z} \in \mathcal{Z}_{\mathcal{M}_0}$ under the Lagrangian $L(\mathbf{z}) = T(\mathbf{z}) - V(\mathbf{z})$ are governed by the second-order ODE

$$mG(\mathbf{z})\ddot{\mathbf{z}} + C(\mathbf{z}, \dot{\mathbf{z}}) + \nabla_{\mathbf{z}} V(\mathbf{z}) = 0, \tag{30}$$

where $G(\mathbf{z}) = J^T(\mathbf{z})J(\mathbf{z})$ is the latent metric induced by the decoder Jacobian $J(\mathbf{z}) = \partial \mathcal{F}/\partial \mathbf{z}$, and $C_i(\mathbf{z}, \dot{\mathbf{z}}) = m \sum_{jk} \Gamma_{ijk}(\mathbf{z}) \dot{\mathbf{z}}_j \dot{\mathbf{z}}_k$ with Γ_{ijk} the Christoffel symbols of G defined above.

B.4. Covariance of Euler-Lagrangian Equation under Redundant Coordinates

A subtlety underlies the derivation above. By Def. 4, the learned latent space $\mathcal{Z}_{\mathcal{M}_0} \subseteq \mathbb{R}^k$ is allowed to be *over-parameterized*: its dimensionality k may exceed the intrinsic degrees of freedom k_{int} of the object's deformation manifold $\mathcal{V}^{k_{\text{int}}}$ (Fig. 11). It is therefore natural to ask whether it is legitimate to solve the dynamics directly in the redundant latent space $\mathcal{Z}_{\mathcal{M}_0}$, as we do in Eq. (4). We answer this in the affirmative by showing that the Euler-Lagrange equation is *covariant* under such a redundant reparameterization: solving it in the over-parameterized latent coordinates recovers the same physical dynamics as solving it in a minimal set of generalized coordinates, provided a mild full-rank condition on the reduction map holds.

Setup. Let $\mathbf{q} = (\mathbf{q}_1, \ldots, \mathbf{q}_{k_{\text{int}}}) \in \mathbb{R}^{k_{\text{int}}}$ be a *minimal* set of generalized coordinates that parameterizes $\mathcal{V}^{k_{\text{int}}}$ exactly, and let $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_k) \in \mathbb{R}^k$ with $k > k_{\text{int}}$ be the learned, *redundant* latent coordinates. Since both parameterize the same manifold, there exists a smooth surjective reduction map

$$\pi : \mathbb{R}^k \to \mathbb{R}^{k_{\text{int}}}, \qquad \mathbf{q} = \pi(\mathbf{z}), \tag{31}$$

through which the decoder factors as $\mathcal{F} = \Psi \circ \pi$, with $\Psi : \mathbb{R}^{k_{\text{int}}} \to \mathbb{R}^{3n}$ embedding the minimal coordinates into the physical vertex space. We write the Jacobian of the reduction map as $\mathbf{P}(\mathbf{z}) = \partial\pi/\partial\mathbf{z} \in \mathbb{R}^{k_{\text{int}} \times k}$. A Lagrangian $L(\mathbf{q}, \dot{\mathbf{q}})$ expressed in the minimal coordinates pulls back to the latent coordinates as $\tilde{L}(\mathbf{z}, \dot{\mathbf{z}}) = L\big(\pi(\mathbf{z}),\, \mathbf{P}(\mathbf{z})\,\dot{\mathbf{z}}\big)$.

A one-dimensional example. Before stating the general result, we build intuition with a minimal example.

Example 2: Redundant coordinates of a free particle

Consider a free particle with Lagrangian $L = \frac{1}{2}m\dot{x}^2$, where $x \in \mathbb{R}$ is the single true coordinate ($k_{\text{int}} = 1$). Its Euler-Lagrange equation gives $m\ddot{x} = 0$, i.e., $x(t) = x_0 + \dot{x}_0\, t$. Now over-parameterize x with two redundant coordinates (u, v) via $x = u + v$ ($k = 2$). The pulled-back Lagrangian becomes $\tilde{L} = \frac{1}{2}m(\dot{u} + \dot{v})^2$, and the Euler-Lagrange equations for u and v both reduce to

$$m(\ddot{u} + \ddot{v}) = m\ddot{x} = 0.$$

Both redundant equations collapse to the single true equation $m\ddot{x} = 0$. The system is under-determined in (u, v) — only the physical combination $x = u + v$ is constrained — yet every solution decodes to the correct physical motion.

Covariance. The behavior observed in Ex. 2 holds in general.

Theorem 2: Covariance under redundant coordinates

Let $\tilde{L}(\mathbf{z}, \dot{\mathbf{z}}) = L\big(\pi(\mathbf{z}),\, \mathbf{P}(\mathbf{z})\dot{\mathbf{z}}\big)$ be the pullback of a Lagrangian $L(\mathbf{q}, \dot{\mathbf{q}})$ under a smooth reduction map $\mathbf{q} = \pi(\mathbf{z})$ with Jacobian $\mathbf{P}(\mathbf{z}) = \partial\pi/\partial\mathbf{z}$. Then the Euler-Lagrange residuals

$$\tilde{R} = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial\tilde{L}}{\partial\dot{\mathbf{z}}} - \frac{\partial\tilde{L}}{\partial\mathbf{z}} \in \mathbb{R}^k, \qquad R = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial\dot{\mathbf{q}}} - \frac{\partial L}{\partial\mathbf{q}} \in \mathbb{R}^{k_{\text{int}}},$$

are related by $\tilde{R} = \mathbf{P}(\mathbf{z})^T R$. In particular, if $\mathbf{z}(t)$ satisfies the Euler-Lagrange equation in the latent coordinates ($\tilde{R} = \mathbf{0}$) and $\mathbf{P}(\mathbf{z})$ has full row rank k_{int} along the trajectory, then $\mathbf{q}(t) = \pi(\mathbf{z}(t))$ satisfies the Euler-Lagrange equation in the minimal coordinates ($R = \mathbf{0}$).

Proof. Writing the reduction in components, $\mathbf{q}_i = \mathbf{q}_i(\mathbf{z})$ and $\dot{\mathbf{q}}_i = \sum_l \frac{\partial\mathbf{q}_i}{\partial\mathbf{z}_l}\dot{\mathbf{z}}_l$, which yields the two standard identities

$$\frac{\partial\dot{\mathbf{q}}_i}{\partial\dot{\mathbf{z}}_j} = \frac{\partial\mathbf{q}_i}{\partial\mathbf{z}_j}, \qquad \frac{\partial\dot{\mathbf{q}}_i}{\partial\mathbf{z}_j} = \sum_l \frac{\partial^2\mathbf{q}_i}{\partial\mathbf{z}_j\partial\mathbf{z}_l}\dot{\mathbf{z}}_l. \tag{32}$$

Since $\mathbf{q}_i$ does not depend on $\dot{\mathbf{z}}_j$, the first identity gives

$$\frac{\partial\tilde{L}}{\partial\dot{\mathbf{z}}_j} = \sum_i \frac{\partial L}{\partial\dot{\mathbf{q}}_i}\frac{\partial\dot{\mathbf{q}}_i}{\partial\dot{\mathbf{z}}_j} = \sum_i \frac{\partial L}{\partial\dot{\mathbf{q}}_i}\frac{\partial\mathbf{q}_i}{\partial\mathbf{z}_j}. \tag{33}$$

Differentiating in time,

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial\tilde{L}}{\partial\dot{\mathbf{z}}_j} = \sum_i \frac{\partial\mathbf{q}_i}{\partial\mathbf{z}_j}\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial\dot{\mathbf{q}}_i} + \sum_i \frac{\partial L}{\partial\dot{\mathbf{q}}_i}\sum_l \frac{\partial^2\mathbf{q}_i}{\partial\mathbf{z}_j\partial\mathbf{z}_l}\dot{\mathbf{z}}_l. \tag{34}$$

Using both identities in Eq. (32),

$$\frac{\partial\tilde{L}}{\partial\mathbf{z}_j} = \sum_i \frac{\partial L}{\partial\mathbf{q}_i}\frac{\partial\mathbf{q}_i}{\partial\mathbf{z}_j} + \sum_i \frac{\partial L}{\partial\dot{\mathbf{q}}_i}\sum_l \frac{\partial^2\mathbf{q}_i}{\partial\mathbf{z}_j\partial\mathbf{z}_l}\dot{\mathbf{z}}_l. \tag{35}$$

Subtracting Eq. (35) from Eq. (34), the second-order terms cancel by the symmetry of mixed partials, leaving

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \tilde{L}}{\partial \dot{\mathbf{z}}_j} - \frac{\partial \tilde{L}}{\partial \mathbf{z}_j} = \sum_i \frac{\partial \mathbf{q}_i}{\partial \mathbf{z}_j}\left(\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\mathbf{q}}_i} - \frac{\partial L}{\partial \mathbf{q}_i}\right), \tag{36}$$

which is exactly $\tilde{R}_j = \sum_i \mathbf{P}_{ij}(\mathbf{z})\, R_i = \big(\mathbf{P}(\mathbf{z})^T R\big)_j$. Finally, $\tilde{R} = \mathbf{0}$ implies $R \in \ker\big(\mathbf{P}(\mathbf{z})^T\big)$; when $\mathbf{P}(\mathbf{z})$ has full row rank k_{int}, $\mathbf{P}(\mathbf{z})^T$ has full column rank and $\ker\big(\mathbf{P}(\mathbf{z})^T\big) = \{\mathbf{0}\}$, hence $R = \mathbf{0}$. □

Physical validity of latent trajectories. Thm. 2 is stated in terms of the abstract reduction map π, which we never observe directly. The following corollary recasts its hypothesis as a concrete condition on the learned *decoder* $\mathcal{F}$, and concludes that a trajectory solved in the latent space is physically valid.

Corollary 1: Physical validity of decoded trajectories

Let $\mathcal{F}$ be a NEUROK (Def. 4), so that $\operatorname{range}\mathcal{F} = \mathcal{V}^{k_{\text{int}}}$, and let $\mathbf{z}(t)$ be a latent trajectory obtained by solving the Euler-Lagrange equation in $\mathcal{Z}_{\mathcal{M}_0}$. If $\mathcal{F}$ is a *local submersion* onto $\mathcal{V}^{k_{\text{int}}}$ along $\mathbf{z}(t)$ — that is, $\mathcal{F}$ is locally surjective onto the deformation manifold, or equivalently its Jacobian $J(\mathbf{z}) = \partial\mathcal{F}/\partial\mathbf{z}$ has full rank k_{int} at every point of $\mathbf{z}(t)$ — then the decoded trajectory $\mathbf{x}(t) = \mathcal{F}(\mathbf{z}(t))$ is a physically valid motion: it satisfies the Euler-Lagrange equations of the underlying physical system on $\mathcal{V}^{k_{\text{int}}}$, independent of the chosen latent parameterization.

Proof. Since $\mathcal{F} = \Psi \circ \pi$ with Ψ an embedding, $J(\mathbf{z}) = (\partial\Psi/\partial\mathbf{q})\,\mathbf{P}(\mathbf{z})$, where the left factor $\partial\Psi/\partial\mathbf{q}$ has full column rank k_{int}. Left-multiplication by a full-column-rank matrix preserves rank, so $\operatorname{rank} J(\mathbf{z}) = \operatorname{rank}\mathbf{P}(\mathbf{z})$. Hence $\mathcal{F}$ being a local submersion onto $\mathcal{V}^{k_{\text{int}}}$ ($\operatorname{rank} J(\mathbf{z}) = k_{\text{int}}$) is equivalent to $\mathbf{P}(\mathbf{z})$ having full row rank, i.e., to the reduction map π being a submersion. Thm. 2 then gives $R = \mathbf{0}$, i.e., $\mathbf{q}(t) = \pi(\mathbf{z}(t))$ satisfies the Euler-Lagrange equations in the minimal coordinates, and the decoded motion $\mathbf{x}(t) = \Psi(\mathbf{q}(t))$ is its image under the embedding Ψ. □

Remark (relation to $G(\mathbf{z})$ and dimension reduction). Intuitively, Cor. 1 asks that the decoder not collapse intrinsic motion directions along the trajectory — its Jacobian must keep spanning the full k_{int}-dimensional tangent space of $\mathcal{V}^{k_{\text{int}}}$. Note that when $k > k_{\text{int}}$ the latent metric $G(\mathbf{z}) = J^T(\mathbf{z})J(\mathbf{z})$ is necessarily rank-deficient even under this condition, so the equation of motion (Eq. (30)) does not pin down $\ddot{\mathbf{z}}$ uniquely — mirroring the under-determination of $(\ddot{u}, \ddot{v})$ in Ex. 2. Thm. 2 guarantees this residual freedom lies entirely in the physically irrelevant kernel directions, so every latent solution decodes to the same physical trajectory. In practice, we further apply the Active Subspace reduction [24] to obtain a compact k_q-dimensional latent whose Jacobian is better conditioned, which improves the numerical stability of the solver and helps maintain the rank condition of Cor. 1.

Remark (no orthogonality constraint on the learned latent). A useful practical consequence of Thm. 2 is that the learned latent coordinates need *not* be mutually orthogonal: the decoder Jacobian $J(\mathbf{z})$ need not have orthonormal columns, and equivalently the metric $G(\mathbf{z}) = J^T(\mathbf{z})J(\mathbf{z})$ need not be diagonal. As with generalized coordinates in classical mechanics, the latent axes may be skewed and coupled; the off-diagonal entries of $G(\mathbf{z})$ encode this non-orthogonality, and the Coriolis-like term $C(\mathbf{z}, \dot{\mathbf{z}})$ — built from the Christoffel symbols of G — accounts for the induced inter-coordinate coupling, leaving the decoded physical trajectory unchanged. The learning objective of NEUROK (Appendix C) therefore does not need to impose any orthogonality, disentanglement, or isometry penalty on $\mathcal{Z}_{\mathcal{M}_0}$: the network is free to discover whatever (possibly entangled, non-orthogonal, and over-parameterized) parameterization best fits the data. Orthogonality affects only the numerical conditioning of the solver, not the correctness of the dynamics, and is handled separately by the dimension reduction above.

B.5. Generalized Forces

The development so far assumes a conservative system whose dynamics are determined entirely by the Lagrangian $L = T - V$. To support effects that cannot be written as the gradient of a potential — external pushes, drags, actuation, or dissipative contact — we extend the formulation with *generalized forces*, following the standard treatment in analytical mechanics [58, 105]. The Euler-Lagrange equation then takes the forced form

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\mathbf{z}}} - \frac{\partial L}{\partial \mathbf{z}} = \mathbf{Q}(\mathbf{z}, \dot{\mathbf{z}}, t), \tag{37}$$

where $\mathbf{Q}(\mathbf{z}, \dot{\mathbf{z}}, t) \in \mathbb{R}^k$ is the generalized force conjugate to the latent coordinates $\mathbf{z}$.

A generalized force is obtained from a physical-space force by the principle of virtual work. Let $\mathbf{F}(\mathbf{x}, t)$ be a force field acting on the sampled material points $\mathbf{x} = \mathcal{F}(\mathbf{z})$. Under an infinitesimal latent displacement $\delta\mathbf{z}$, the points move by $\delta\mathbf{x} = J(\mathbf{z})\,\delta\mathbf{z}$, and the virtual work done is

$$\delta W = \mathbf{F} \cdot \delta\mathbf{x} = \mathbf{F} \cdot \big(J(\mathbf{z})\,\delta\mathbf{z}\big) = \big(J^T(\mathbf{z})\,\mathbf{F}\big) \cdot \delta\mathbf{z}, \tag{38}$$

so the generalized force is the pullback of the physical force through the decoder Jacobian,

$$\mathbf{Q}(\mathbf{z}, \dot{\mathbf{z}}, t) = J^T(\mathbf{z})\,\mathbf{F}\big(\mathcal{F}(\mathbf{z}), t\big). \tag{39}$$

Substituting $L(\mathbf{z}, \dot{\mathbf{z}}) = \frac{1}{2} m\,\dot{\mathbf{z}}^T G(\mathbf{z})\,\dot{\mathbf{z}} - V(\mathbf{z})$ into Eq. (37) and repeating the derivation of Thm. 1 yields the forced equation of motion

$$mG(\mathbf{z})\,\ddot{\mathbf{z}} + C(\mathbf{z}, \dot{\mathbf{z}}) + \nabla_{\mathbf{z}} V(\mathbf{z}) = \mathbf{Q}(\mathbf{z}, \dot{\mathbf{z}}, t), \tag{40}$$

which we integrate for $\ddot{\mathbf{z}} = \big[mG(\mathbf{z})\big]^{-1}\big(\mathbf{Q} - C - \nabla_{\mathbf{z}} V\big)$. In practice we evaluate the pullback in Eq. (39) with a single vector-Jacobian product, so the generalized force costs one extra backward pass through the decoder and never requires forming $J(\mathbf{z})$ explicitly. Finally, because $\mathbf{Q}$ acts through the same Jacobian that defines the metric, it transforms covariantly under reparameterization (Thm. 2): the forced dynamics, like the unforced ones, are independent of the chosen latent parameterization.

B.6. Boundary Conditions and Action Conditioning

The proposed neural-symbolic framework has formed an elegant bridge to communicate the physical world and learned latent space with the Euler-Lagrangian equation. As the latent space is connected to a physical 3D representation with $\mathcal{F}$, it enables native support for diverse types of physical conditioning. We expand on the following types of conditioning.

External Non-conservative Force. Using the extended formulation of Appendix B.5, our system natively supports conditioning on external, possibly non-conservative, forces. The user specifies a force field $\mathbf{F}(\mathbf{x}, t)$ over a chosen region of the object and time window, and the system applies the corresponding generalized force $\mathbf{Q}(\mathbf{z}, \dot{\mathbf{z}}, t) = J^T(\mathbf{z})\,\mathbf{F}(\mathcal{F}(\mathbf{z}), t)$ in Eq. (40). We support constant forces, forces gated to a spatial region — selected by a bounding box in either the world frame or the rest (canonical) frame of the object — and a time interval, as well as forces that ramp smoothly over their active window. Unlike the conservative *uniform and constant force* below, which can equivalently be absorbed into the potential, this mechanism handles localized, time-varying, and genuinely non-conservative forcing that admits no potential representation.

Uniform and constant force Our system supports uniform force $\mathbf{F}$ by adding an additional force-inherited potential energy term $V_F(\cdot)$ to $V_{\mathbf{x}}(\mathbf{x})$:

$$V_F(\cdot) = -\mathbf{F} \cdot \mathbf{x}, \tag{41}$$

and solving the updated Lagrangian mechanical system with commonly defined $V(\cdot)$ and $V_F(\cdot)$.

Initial velocity The most common conditioning of the system is selecting a region of the input mesh $\mathcal{M}_0$ as well as sampled points $\mathbf{x}_{\text{sample}}$ in this selected region, and providing the initial velocities $\dot{\mathbf{x}}_{\text{sample}}$ for these sampled points. This provides the initial conditions $(\mathbf{z}_0, \dot{\mathbf{z}}_0)$ by solving the following projective optimization problem:

$$\min_{\mathbf{z}_0, \dot{\mathbf{z}}_0} ||\mathbf{x}_0 - \mathcal{F}(\mathbf{z}_0)||_2^2 + ||\dot{\mathbf{x}}_0 - J_{\mathbf{z}} \dot{\mathbf{z}}_0||_2^2. \tag{42}$$

Quasistatic action Our system also supports dragging-style quasistatic action conditioning. This is by providing the positions of sampled points $\mathbf{x}^t_{\text{sample}}$ from a specific region at different timestamps t, and solving the trajectory by optimizing:

$$\min_{\mathbf{z}_t} \mathbb{E}_t \left[||\mathbf{x}_t - \mathcal{F}(\mathbf{z}_t)||_2^2 \right]. \tag{43}$$

B.7. Collision Handling

Collisions are incorporated into the same energy-based formulation: each contact is modeled as a soft, one-sided barrier potential added to $V(\mathbf{z})$, so that the repulsive response $-\nabla_{\mathbf{z}} V$ arises automatically through Eq. (40). This keeps collision handling fully within the Lagrangian framework and requires no explicit constraint projection. The energy dissipated at impact is then modeled by a per-step velocity restitution, described at the end of this section.

Self-collision. Although the modeled system is object-centric, a deforming object may collide with itself. We detect this in a topology-free, geometry-agnostic manner using a pairwise barrier on the sampled points, analogous to the non-adjacent-primitive filter of Incremental Potential Contact [65]. Once, from the rest pose, we precompute the set of point pairs

$$\mathcal{P} = \left\{ (i,j) : \|\mathbf{x}_i^0 - \mathbf{x}_j^0\| > d_{\text{filt}} \right\}, \tag{44}$$

whose rest-pose distance exceeds a threshold d_{filt}; pairs that are already close at rest are expected to remain close and are excluded so that they do not saturate the barrier from the start. The self-collision potential then penalizes any retained pair that approaches within an activation distance $\hat{d}$:

$$V_{\text{self}}(\mathbf{x}) = \frac{k_{\text{self}}}{2} \sum_{(i,j)\in\mathcal{P}} \max\big(0,\ \hat{d} - \|\mathbf{x}_i - \mathbf{x}_j\|\big)^2, \tag{45}$$

where k_{self} is the barrier stiffness.

Inter-object collision. Collisions with the surrounding environment (*e.g.*, a ground plane or other static obstacles) follow the same soft-barrier principle. For a half-space obstacle bounded along axis a at height h, we add a one-sided quadratic penalty over the sampled points,

$$V_{\text{obs}}(\mathbf{x}) = k_{\text{obs}} \sum_i \max\big(0,\ h + \epsilon - \mathbf{x}_{i,a}\big)^2, \tag{46}$$

with stiffness k_{obs} and a small margin ϵ. More general static colliders are handled analogously by penalizing penetration of their signed-distance field.

Contact restitution. The barrier potentials above are conservative and hence energy-preserving. To model the energy lost at impact, we additionally apply a velocity restitution whenever a contact is active: at each integration step, if any contact potential is non-zero ($V_{\text{self}} + V_{\text{obs}} > 0$), we rescale the generalized velocity $\dot{\mathbf{z}} \leftarrow \rho\, \dot{\mathbf{z}}$ with a restitution coefficient $\rho \in (0, 1]$, where $\rho = 1$ is perfectly elastic and $\rho < 1$ dissipates energy at each contact. This produces stable, visually plausible bounces without the full machinery of impulse-based contact resolution. Optionally, to keep the trajectory within the region where the decoder is well-behaved, we apply the same reflective treatment in latent space: when a coordinate leaves a prescribed range, $|\mathbf{z}_i| > \mathbf{z}_{\max}$, we reflect it back into the range and flip the corresponding velocity.

B.8. Numerical Integration

We integrate the (forced) equation of motion (Eq. (40)) with a semi-implicit, or *symplectic*, Euler scheme, which is our default solver. Let Δt be the step size and

$$\ddot{\mathbf{z}}_n = \big[mG(\mathbf{z}_n)\big]^{-1}\big(\mathbf{Q}_n - C(\mathbf{z}_n, \dot{\mathbf{z}}_n) - \nabla_{\mathbf{z}} V(\mathbf{z}_n)\big) \tag{47}$$

the acceleration evaluated at step n. Each step updates the velocity first, then advances the position using the *updated* velocity:

$$\dot{\mathbf{z}}_{n+1} = \dot{\mathbf{z}}_n + \Delta t\, \ddot{\mathbf{z}}_n, \tag{48}$$

$$\mathbf{z}_{n+1} = \mathbf{z}_n + \Delta t\, \dot{\mathbf{z}}_{n+1}. \tag{49}$$

Although only first-order accurate, this scheme is symplectic: it preserves phase-space volume and keeps the energy error bounded over long horizons, whereas the explicit (forward) Euler update $\mathbf{z}_{n+1} = \mathbf{z}_n + \Delta t\, \dot{\mathbf{z}}_n$ drifts unboundedly. This property underlies the approximately constant total energy observed in our energy-conservation analysis. After each step we apply the post-step updates of Appendix B.7 — contact restitution and the optional latent-space reflection — together with an optional small global velocity damping for additional stability. We use $\Delta t = 5 \times 10^{-4}$ by default.

C. Generative Learning of NEUROK

In this section, we cover more details of the implementation of NEUROK that cannot be covered in the main paper due to the space limit.

C.1. Dual-Quaternion Based Deformation Representation

Our method predicts deformation fields ϕ to drive mesh $\mathcal{M}_0$ to other kinematic states. We find that choosing a right representation for $\phi(\mathbf{x})$ is essential to the success of the model.

Specifically, assuming that in the training data, we have:

$$\phi(\mathbf{x}) = T\mathbf{x} \tag{50}$$
$$= R\mathbf{x} + t, \tag{51}$$

where $T \in SE(3)$, we parameterize this deformation field with the dual quaternion representation. A dual quaternion is an extension for the common quaternion representation by introducing a virtual ϵ. Formally, the conversion between dual quaternion and $SE(3)$ matrix representation (R, t) can be computed with the following process:

Given an element of $SE(3)$,

$$T = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix}, \qquad R \in SO(3), \quad \mathbf{t} = (t_x, t_y, t_z)^\top \in \mathbb{R}^3, \tag{52}$$

we convert the rotation matrix R to a unit quaternion

$$q_r = (q_w, q_x, q_y, q_z), \qquad \|q_r\| = 1. \tag{53}$$

The translation is embedded as a pure quaternion

$$q_t = (0, t_x, t_y, t_z). \tag{54}$$

The dual part of the dual quaternion is obtained via the Hamilton product

$$q_d = \frac{1}{2}\, q_t \otimes q_r. \tag{55}$$

Writing $q_r = (q_w, \mathbf{v})$ with $\mathbf{v} = (q_x, q_y, q_z)$, the Hamilton product

$$(0, \mathbf{t}) \otimes (q_w, \mathbf{v}) = \big(-\mathbf{t}\cdot\mathbf{v},\ q_w\,\mathbf{t} + \mathbf{t} \times \mathbf{v}\big), \tag{56}$$

gives the explicit dual component

$$q_d = \big(-\tfrac{1}{2}\,\mathbf{t}\cdot\mathbf{v},\ \tfrac{1}{2}\big(q_w\mathbf{t} + \mathbf{t} \times \mathbf{v}\big)\big)\,. \tag{57}$$

Thus the SE(3) transform T is represented by the dual quaternion

$$\widehat{q} = q_r + \epsilon\, q_d \tag{58}$$
$$= (q_w, q_x, q_y, q_z)\ +\ \epsilon\,(q_{dw}, q_{dx}, q_{dy}, q_{dz}), \tag{59}$$

where $(q_{dw}, q_{dx}, q_{dy}, q_{dz}) = q_d$. We found that this representation gives a smoother latent space $\mathcal{Z}$, as demonstrated in our ablation studies in the main paper.

C.2. Details of Model Architectures

In this section, we provide additional model architecture details for training NEUROK, including the concrete instantiation of the conditional VAE and hyperparameters.

All three components in Fig. 4 of the main paper — the kinematic prior encoder $\mathcal{E}_{\text{cond}}$, the variational deformation encoder $\mathcal{E}_{\text{VAE}}$, and the decoder $\mathcal{D}$ — share a common transformer-based perceiver [48, 111] architecture.

Table 5. Architecture hyperparameters for the NEUROK conditional VAE implementation.

Component	**Setting**
Latent token count K	128
Latent token channels (bottleneck channels)	1024
Point embed hidden size (Fourier basis width)	8
Point embed output dimension	512
Number of motion encoder transformer blocks	4
Number of deformation decoder transformer blocks	4
Attention heads per transformer block	16
MLP expansion ratio in transformer blocks	4.0
Conditional codebook (for $\mathcal{E}_{\text{cond}}$)	learnable, size 128×1024
Prior codebook (for $\mathcal{E}_{\text{VAE}}$ mean)	learnable, size 128×1024
Log-var codebook (for $\mathcal{E}_{\text{VAE}}$ logvar)	learnable, size 128×1024
Log-var output head	linear layer on bottleneck tokens
Output head of decoder $\mathcal{D}$	linear layer from 512 to deformation dim

For both $\mathcal{E}_{\text{cond}}$ and $\mathcal{E}_{\text{VAE}}$, we first sample N points on the mesh surface and embed them using a sinusoidal 3D point encoder [129], which applies a fixed Fourier basis followed by an MLP to obtain point-wise features. These features are then consumed by a stack of cross-attention and self-attention "perceiver" blocks with a learned set of latent tokens, as described in Sec. 4.1 of the main paper. When implementing attention layers, we use RMSNorm to stablize the training.

The latent tokens produced by $\mathcal{E}_{\text{cond}}$ parameterize the instance-specific Gaussian prior $p_{M_0}(\mathbf{z})$, while those from $\mathcal{E}_{\text{VAE}}$ parameterize the posterior $q_{M_0}(z \mid \phi)$ via separate sets of tokens for the mean and log-variance. The decoder $\mathcal{D}$ reshapes $\mathbf{z}$ back into a sequence of tokens and uses them as context for a second perceiver-style transformer that predicts point-wise deformation vectors on the sampled query points, which are then transferred to mesh vertices by averaging over K_{drive} nearest neighbors. The concrete hyperparameters of this architecture are summarized in Tab. 5.

Latent Distribution and Log-Variance Clamping The VAE models the latent distribution for each instance as a factorized Gaussian over the $K \times 1024$ latent tokens. $\mathcal{E}_{\text{cond}}$ outputs the mean of the prior $p_{\mathcal{M}_0}(\mathbf{z})$, while $\mathcal{E}_{\text{VAE}}$ outputs both mean and log-variance for the posterior $q_{\mathcal{M}_0}(\mathbf{z} \mid \phi)$ via separate transformer branches and a dedicated linear "logvar head."

To prevent numerical instabilities from excessively large or small variances, we apply a smooth upper clamp to the log-variance using a softplus-based function. This ensures that the exponentiated variance remains within a bounded range during training, while preserving differentiability and avoiding hard clipping.

C.3. Model Training

We train the conditional VAE jointly over $\mathcal{E}_{\text{cond}}$, $\mathcal{E}_{\text{VAE}}$, and $\mathcal{D}$ using the standard conditional VAE objective. Training is performed with AdamW optimizer, a cosine-annealing-with-restarts scheduler, adaptive gradient clipping, and bfloat16 mixed precision. We use a fixed per-GPU batch size and train for a large number of optimization steps to fully exploit the scale of our curated 4D dataset. Full optimization hyperparameters are summarized in Tab. 6. Our model is trained on 4 NVIDIA H100 GPUs for 7 days to arrive final convergence.

Data Augmentation The conditional VAE is trained on a large 4D mesh dataset of deforming objects, as described in Sec. 4.2 of the main paper. For each training instance we randomly select two frames sharing the same topology, treat the first as the conditioning mesh M_0, and extract the deformation field from the first to the second frame.

During training, we perform randomized $SE(3)$ augmentation of the object. The main sampling and augmentation hyper-parameters are given in Tab. 7.

Table 6. Optimization and training hyperparameters for learning NEUROK.

Component	Setting
Optimizer	AdamW
Initial learning rate	8×10^{-5}
Weight decay	8×10^{-6}
Learning rate schedule	CosineAnnealingWarmRestarts
Cosine schedule T_0	1,000,000 steps
Cosine schedule T_{mult}	2
Cosine schedule $\eta_{\min}$	8×10^{-7}
Batch size	512
Mixed precision mode	bfloat16

Table 7. Dataset and augmentation hyperparameters for training the conditional VAE.

Component	Setting
Number of surface points sampled per shape	512
Bounding-box scale range	$[0.5, 1.5]$
In-plane rotation range around z-axis	$[0°, 360°]$
Translation range along x	$[-0.1, 0.1]$
Translation range along y	$[-0.1, 0.1]$

C.4. Mesh Driving

Given the predicted deformation field $\phi(\mathbf{x})$, we use following algorithm inspired by KNN and RANSAC [30] to obtain the vertex deformation $\delta_{\mathbf{z}}$ from $\mathcal{M}_0$ to $\mathcal{M}_{\mathbf{z}}$.

Concretely, we first sample N_{drive} points from the surface of $\mathcal{M}_0$ and obtain their deformations δ_{pred} with $\mathcal{D}(\mathbf{z}, \mathcal{M}_0)$ with the sampled latent $\mathbf{z}$.

We then compute K_1 nearest neighbors for each vertex based on Euclidean distance and obtain their corresponding deformations δ_{neighbor}. We then use the RANSAC algorithm to fit the deformation vector for each vertices. For vertices with a high inlier ratio, their deformation is returned as in this step. For other N' vertices, we continue to do the following steps.

We perform another round of KNN to find $\lambda_{\text{drive}} K_1$ nearest points for the remaining vertices. We then compute the *geodesic* distance between the $(N', \lambda_{\text{drive}} K_1)$ pairs of vertices. We then multiply the euclidean distance with the computed geodesic distance, and annotate the deformation of points that are not connected to the computed vertice with noise.

Finally, we run the third round of KNN on the updated distance to filter K_1 points for each of the remaining vertices. We run the second round of RANSAC with the updated neighbor point sets from the third round of KNN to obtain the deformation for every vertices of $\mathcal{M}_{\mathbf{z}}$. The detailed algorithm description can be found at Alg. 1.

D. Dataset Construction

In this section, we detail the pipeline we use to construct the training dataset for the NeuROK network. Our training data consists of three parts:

1. Dynamic objects from PartNet-Mobility [119],
2. General dynamic objects from Objaverse [27],
3. Synthetic physical simulation data for cloth and continuum bodies.

D.1. Processing of PartNet-Mobility

PartNet-Mobility [119] is a dataset of 2,372 synthetic articulated objects spanning everyday categories such as laptops, glasses, and chairs. Each object is annotated with a URDF articulation file, link-level meshes, and basic textures. For training

Input: Reference mesh $\mathcal{M}_0$, latent code $\mathbf{z}$, predicted deformation field $\phi(\mathbf{x})$, number of driving points N_{drive}, base neighbor count K_1, expansion factor λ_{drive}
Output: Vertex deformations $\delta_{\mathbf{z}}$ from $\mathcal{M}_0$ to $\mathcal{M}_{\mathbf{z}}$

Step 1: Sample driving points and deformations
Sample N_{drive} points $\{\mathbf{x}_i\}_{i=1}^{N_{\text{drive}}}$ on the surface of $\mathcal{M}_0$.
Compute their predicted deformations using the decoder. Store the set of driving pairs $\mathcal{S} = \{(\mathbf{x}_i, \delta_{\text{pred},i})\}_{i=1}^{N_{\text{drive}}}$.

Step 2: First KNN and RANSAC pass
foreach *vertex* v *of* $\mathcal{M}_{\mathbf{z}}$ **do**
  Find the K_1 nearest driving points in $\mathcal{S}$ w.r.t. Euclidean distance in $\mathbb{R}^3$.
  Let $\mathcal{N}_1(v)$ be the corresponding set of deformation vectors.
  Run RANSAC on $\mathcal{N}_1(v)$ to estimate a deformation $\hat{\delta}(v)$ and an inlier set $\mathcal{I}_1(v)$.
  if *inlier ratio* $\frac{|\mathcal{I}_1(v)|}{K_1}$ *is above a threshold* **then**
    Accept $\hat{\delta}(v)$ as the deformation for v.
  else
    Mark v as *unresolved*.
  end
end
Let $\mathcal{V}'$ be the set of unresolved vertices (of size N').

Step 3: Geodesic-aware neighbor refinement
foreach *vertex* $v \in \mathcal{V}'$ **do**
  `// (a) Expanded KNN in Euclidean space`
  Find the $\lambda_{\text{drive}} K_1$ nearest driving points in $\mathcal{S}$ w.r.t. Euclidean distance.
  Denote this set by $\mathcal{N}_{\text{exp}}(v)$.
  `// (b) Compute geodesic distances and update metric`
  foreach *driving point* $(\mathbf{x}_i, \delta_{pred,i}) \in \mathcal{N}_{exp}(v)$ **do**
    Compute geodesic distance $d_{\text{geo}}(v, \mathbf{x}_i)$ on the mesh.
    Compute Euclidean distance $d_{\text{euc}}(v, \mathbf{x}_i)$.
    Define an updated distance
$$d_{\text{upd}}(v, \mathbf{x}_i) = d_{\text{euc}}(v, \mathbf{x}_i) \cdot (1 + d_{\text{geo}}(v, \mathbf{x}_i)).$$
    If v and $\mathbf{x}_i$ are not connected on the mesh (e.g. d_{geo} is infinite or above a large threshold), mark $(\mathbf{x}_i, \delta_{\text{pred},i})$ as noise for vertex v.
  end
  `// (c) Third KNN pass with updated distances`
  From $\mathcal{N}_{\text{exp}}(v)$, select the K_1 points with smallest $d_{\text{upd}}(v, \mathbf{x}_i)$ that are not marked as noise.
  Let this filtered neighbor set be $\mathcal{N}_2(v)$.
end

Step 4: Second RANSAC pass
foreach *vertex* $v \in \mathcal{V}'$ **do**
  Run RANSAC on the deformation vectors in $\mathcal{N}_2(v)$ to estimate a final deformation $\hat{\delta}(v)$.
  Assign $\hat{\delta}(v)$ as the deformation for vertex v.
end

return *the deformation field* $\delta_{\mathbf{z}}$ *defined by* $\hat{\delta}(v)$ *for all vertices* v *of* $\mathcal{M}_{\mathbf{z}}$.

Algorithm 1: The detailed algorithm for mesh driving via KNN and RANSAC.

our method on PartNet-Mobility, we removed corrupted or incomplete assets and split the remaining objects into 2,093

training instances and 254 test instances. To generate training and evaluation data, we use the Urchin [36] URDF parsing library to load articulation definitions and perform forward kinematics. During training, we sample joint configurations on the fly within their respective joint limits to ensure broad coverage of each object's kinematic space. Although PartNet-Mobility includes textures, they are generally simple and unrealistic. We augment each object with additional textures generated using the open-source 3D texturing model Hunyuan3D 2.1 [47]. While these enhanced textures are not used during training, they significantly improve the diversity and visual fidelity of our simulated renderings.

D.2. Processing of Objaverse

Objaverse-XL [27] is not only a static 3D object dataset — it actually contains rich dynamic objects, as evidenced by previous works [29]. Inspired by these works, we enrich our training dataset with objects from Objaverse. The filtering pipeline is as follows.

Filtering out objects without animations. We only use models from Sketchfab of Objaverse-XL to ensure the quality. We use the meta-data from Sketchfab to judge whether there is animation inside the model. For those models without animation metadata, we download its *glb* file and use a GLTF parser to check whether there are animations. After this step, we filter in only objects with at least one animation.

VLM filtering on static model quality. We then filter out models that do not contain motions of our interest. In this work, we only consider motions of dynamic *objects*. However, the majority of Objaverse-XL is humanoid characters. To solve this, we use a Gemini to filter all objects with their static 3D model rendering with the following prompt:

Prompt for the first step of VLM filtering:

You are an expert CG artist. Your task is to analyze the provided 3D model, identify its primary subject(s), and classify it using a specific category and label. You must follow the decision process and output format below precisely.
Decision Process
Step 1: Is the model a scene or a single subject?
First, determine the overall type. It is a scene if either of the following is true:
- It depicts two or more distinct primary subjects (e.g., a person standing next to a car; a lamp on a table).
- It features a detailed background (anything other than a single, solid, uniform color).
If neither of these conditions is met, the model depicts a single subject.
(Note on Accessories: Items that are clearly part of a main subject (like a knight's sword and shield, a character's clothing, or a guitar held by a person) do not count as separate subjects. The knight is one subject, not three.)
Step 2: Choose the correct category.
Based on your decision in Step 1, select one category from the appropriate list below.
Scene Category
- scene — Use this category if you determined the model was a scene in Step 1.
Subject Categories (Use only if the model is a single subject)
- character — A single human, humanoid, or anthropomorphic figure (e.g., astronaut, elf, talking animal, humanoid robot).
- animal — A single non-humanoid creature, real or fantasy (e.g., dog, dragon, insect, fish).
- dynamic-object — A single man-made object designed with moving, articulating, or deforming parts (e.g., car, book, foldable chair, refrigerator, gun).
- static-object — A single man-made object that is rigid and not designed to move or deform (e.g., mug, table, sword, statue, brick).
- dynamic-nature — A single natural object that is living, flexible, or has moving parts (e.g., a tree with leaves, a blooming flower, a jellyfish).
- static-nature — A single natural object that is non-living and rigid (e.g., rock, crystal, mountain).
- other — Any subject that does not fit the categories above (e.g., abstract art, fractal pattern, text).
- unknown — Use this only if the subject cannot be identified.
Step 3: Define the output.
Your response must be a single line in the format major_category:specific_subject, with no extra text.
major_category: The category you chose in Step 2.

Category	Count	Percentage
animal	2800	6.69
character	15054	35.99
dynamic-nature	522	1.25
dynamic-object	5892	14.09
failed	805	1.92
other	1404	3.36
scene	8866	21.20
static-nature	530	1.27
static-object	5553	13.28
unknown	403	0.96
TOTAL	**41829**	**100.00**

Table 8. Category distribution summary of dynamic objects in Objaverse after the first step of filtering.

specific_subject: A concise (1-3 words) lowercase description of the model's content.
- Examples:
A model of a single tiger: animal:tiger
A model of a knight holding a sword: character:knight with sword
A model of a single windmill: dynamic-object:windmill
A model of a simple teacup: static-object:teacup
A model of a cup sitting on a saucer (two distinct objects): scene:cup on saucer
A model of a car parked inside a detailed garage (detailed background): scene:car in garage

After this filtering, the resulting dataset has categories as in Tab. 8. We selected *dynamic-objects* to move to the next step of filtering.

Filtering out animations with only global transformations. Each object in Objaverse can have various animations, yet some of them are ill-posed. In this step, we filter out animations with only rigid, global motions, such as the global rotation or translation of the object. This is done by analyzing the GLTF file and check whether all animations are changing transformations of the root node of the GLTF tree.

Filtering out purely static animations. Some animations are purely static, and we also need to filter out these animations. We randomly sample several frames in the GLTF file, and compare whether the changes in the vertex positions are within a threshold. If all changes are within the threshold, we consider the animation to be static.

Filtering out non-ideal motion types After the previous two filtering steps, the remaining animations are mostly usable. Yet, some of them are not demonstrating the functionality of the object. For example, some of them are just explosion view of the object to demonstrate the internal structure. To find out these animations, we again use Gemini to judge whether the animation of an object is showing its functional deformations. We use the following prompt:

Prompt for the second step of VLM filtering: You are an expert CG artist. Your task is to analyze the type of motion of the provided 3D model, and classify it using a specific category and label. You must follow the decision process and output format below precisely.
You will be provided with 30 images the same 3D model. The first 10 images are videos rendered from the first viewpoint, the second 10 images are videos rendered from the second viewpoint, and the last 10 images are videos rendered from the third viewpoint.

Category	Count	Percentage
explosion	820	9.05
other_motion	68	0.75
static	1215	13.41
explosion_as_demonstration	1	0.01
embodied_behavior	505	5.57
other	1039	11.46
functional	5415	59.75
TOTAL	**9063**	**100.00**

Table 9. Category distribution summary for animations of objects after filtering.

Decision Process
Step 1: What rough type is the provided motion? Decide the major category.
First, determine the overall type. It is a motion of a 3D model with either of the following types:
- Static motion. The model is not moving or deforming at all. In this case, you should classify it as ”static”.
- Explosion as demonstration. The model is exploding or blowing up to show its different parts or to provide a part segmentation. In this case, you should classify it as ”explosion”.
- Embodied behavior. The model is moving or deforming by itself, without any external force. In this case, you should classify it as ”embodied_behavior”.
- Functional motion. The model is moving or deforming to show its functionality. In this case, you should classify it as ”functional”.
- Other motion. The model is moving or deforming in a way that is not one of the above. In this case, you should classify it as ”other”.
Step 2: Determine the specific motion.
Then, you should determine the specific type of motion.
Step 3: Define the output.
Your response must be a single line in the format major_category:specific_motion, with no extra text.
major_category: The category you chose in Step 2.
specific_motion: A concise (1-3 words) lowercase description of the model’s motion.
- Examples:
A tiger walking: embodied_behavior:tiger walking
A bicycle with its pedals moving: functional:bicycle with pedals moving
A complex mechanical system is exploding to show its exploded-view drawing: explosion:complex mechanical system exploding
A human is not moving: static:human not moving

We append animations identified as *functional* by the VLM to our dataset. The resulting statistics after this step can be found at Tab. 9.

D.3. Processing of Physical Simulation Data

To construct the third component of our dataset, we generate synthetic simulation data using Blender to capture non-rigid dynamics. This data focuses on two categories: cloth and continuum bodies.

Cloth Simulation. We sourced 30 high-fidelity, textured shirt models from the BlenderKit asset library. We employ Blender’s cloth simulation engine to generate realistic fabric dynamics. To maintain the global position of the object, we select vertices in the shoulder region as pin points. We then introduce external forces by applying two random directional wind fields sequentially. The simulation timeline consists of an initial wind impulse, a relaxation phase where the cloth settles under gravity, followed by a second wind impulse and a final gravitational settling phase.

Soft Body Simulation. We collected 30 high-quality plant and flower models, also via BlenderKit. To simulate soft-body dynamics efficiently, we adopt a procedural animation technique rather than a full physics simulation. We apply a "Simple Deform" modifier to the stem of the vegetation to induce bending. By keyframing the deformation parameters, we approximate the natural oscillatory motion of plants swaying in the wind without the computational overhead of finite element analysis.

E. Details on Experiment Setting

E.1. Neural Object Kinematics Learning

We conduct the object kinematics learning experiment to evaluate the effectiveness of our learned neural object kinematics. Following the evaluation of CANOR [43], we first predict kinematic state parameterizations $\mathcal{Z}$ from a given object instance $\mathcal{M}_0$, and then optimize such representations so that the decoded deformed objects $\mathcal{M}_z$ best align the shape of an unseen pose of the same instance $\mathcal{M}_t$. The accuracy of such alignment across diverse poses reflects the expressiveness of the kinematic state parameterization, while the optimization process provides insight into the smoothness and regularization properties of the learned kinematic space.

For this experiment, due the need of ground-truth target mesh, we use PartNet-Mobility by sampling 50 objects from our test split and selecting two poses per object as the input and target, respectively. We limit the evaluation to 50 objects because some of the baselines require per-object optimization, which is computationally expensive. For methods that rely on multiple posed frames as input, we additionally sample extra unseen poses, distinct from the target pose, to serve as their inputs.

We compare our proposed method with a wide range of baselines that learn object kinematics, namely CANOR [43], KeyPointDeformer [49], NeuralDeformationGraphs [11], SINGAPO [83], FreeArt3D [19]. CANOR [43] is a feed-forward model that represents object kinematics as a series of blobs. For evaluation, we optimize the pose-dependent blob attributes, including position and rotation, and use occupancy prediction loss as objective. KeyPointDeformer [49] is a deformation-based shape editing method that predicts keypoints for representing object kinematics and applies cage-based skinning driven by these keypoints. In our setting, we use the keypoints predicted by the network and optimize their positions. NeuralDeformationGraphs [11] is a per-scene optimization method that models 3D object motion using a learned deformation graph with local shape functions defined at each node. For evaluation, we sample five poses for each test object and train NeuralDeformationGraphs on these poses. After training, we freeze the local shape functions and optimize only the node positions and rotations to fit the remaining unseen test pose. SINGAPO [83] performs feed-forward articulation prediction from a single image. Since it was originally developed on a limited set of object categories, we adjusted its LLM prompts to ensure that it produces valid predictions for all test inputs. For each test object, we render a front-view image and use SINGAPO to infer its URDF structure. We then align the predicted object to the input shape and subsequently optimize its articulation angles using Limited-memory BFGS (L-BFGS) optimization [12] to align with the target. FreeArt3D [19] is a per-object optimization method that distills part shape and articulations from a 3D generation model Trellis [120]. Since it requires multiple posed frames as input, we randomly sample five distinct poses for each object and render a front-view image for each one. Similar to SINGAPO, we first align the predicted object to the input shape and then optimize its articulation parameters to match the target pose.

After optimizing the kinematic space for all methods, we decode each representation back into a posed object mesh and evaluate the reconstruction quality by measuring how well the optimized mesh aligns with the ground-truth target. We compute both L1 and L2 Chamfer Distance, and additionally run a manifolding algorithm [44] on both the predicted and ground-truth meshes before computing IoU, since some objects are not watertight and this can affect IoU calculation.

E.2. Physically-Inspired 4D Generation

We conduct the physically-inspired 4D generation experiment on 8 different objects, *Box, Flower, Newton, Shirt, Cloth, Bow, Laptop,* and *Lamp*. These objects cover diverse dynamic types, including rigid body, articulated bodies, elastic objects, and cloth. Our approach can generate plausible 4D motion for every dynamic type, yet other approaches can only successfully model their designated categories. The tested objects are collect from in-the-wild Internet 3D model database and do not exist in any of the training datasets.

The animated results of these 4D generations can be found on our project page at `https://chen-geng.com/neurok`. We compare with diverse baselines on the task of generating 4D motion only from 3D objects. We use their official implementations and released checkpoints to perform generation on our test cases. For AnimateAnyMesh [117], as the input is text and it does not support action conditioning, we describe the action with text to condition the generative model.

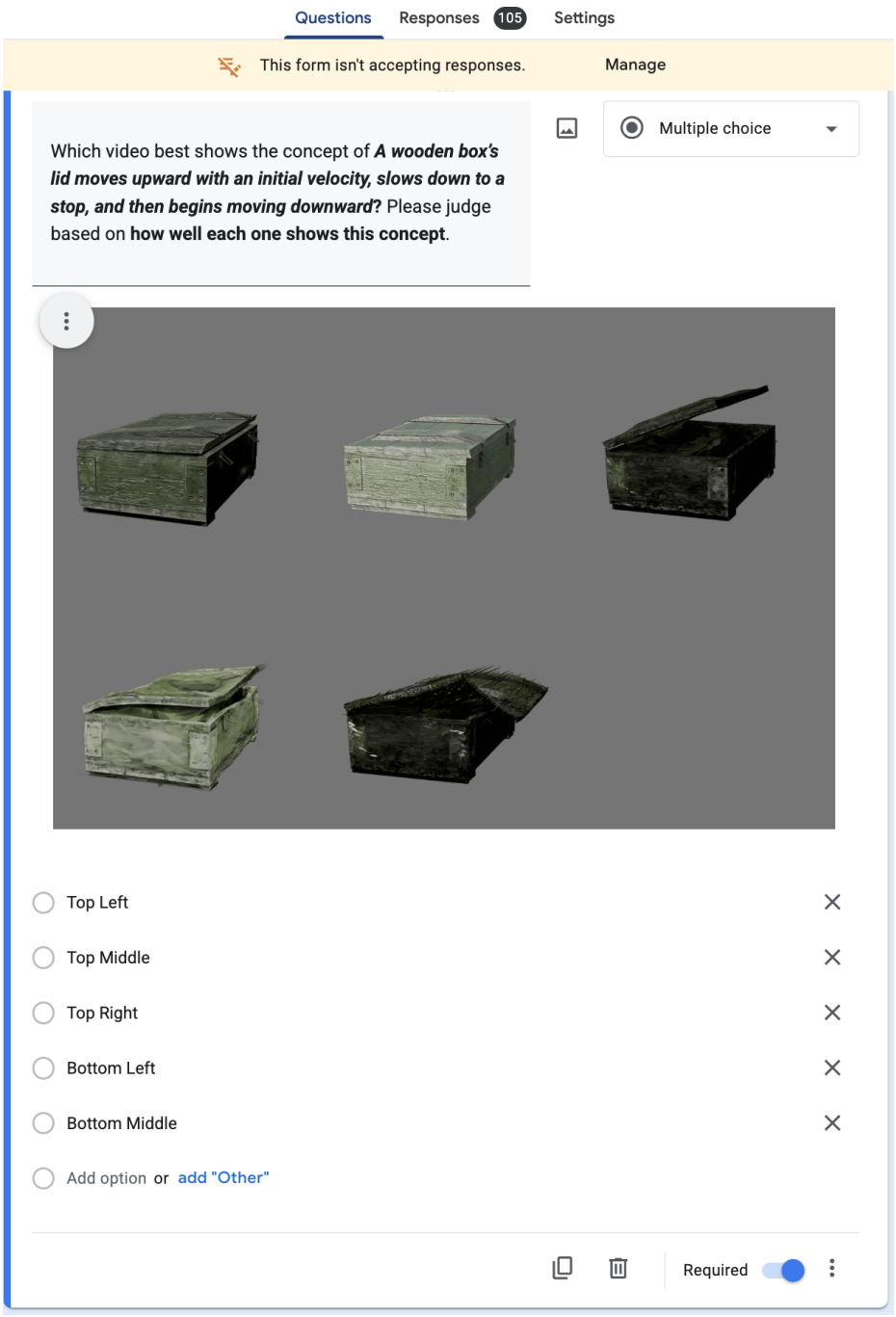


Figure 12. **Screenshot of the user study page.** We use Google forms and Prolfiic to conduct user study, and this page shows the backend management page for the first type of questions.

It is inherently ambiguous to generate 4D dynamics given only 3D shapes as input, and the goal of our paper is to perform 4D generation instead of physical simulation. Therefore we compare whether the generated 4D motion is visually plausible and physically consistent.

This property is evaluated with two categories of quantitative metrics: user study and objective metrics. For user study, we recruit 105 participants in the Prolific platform, and show them five videos of the rendering of the 4D generation results

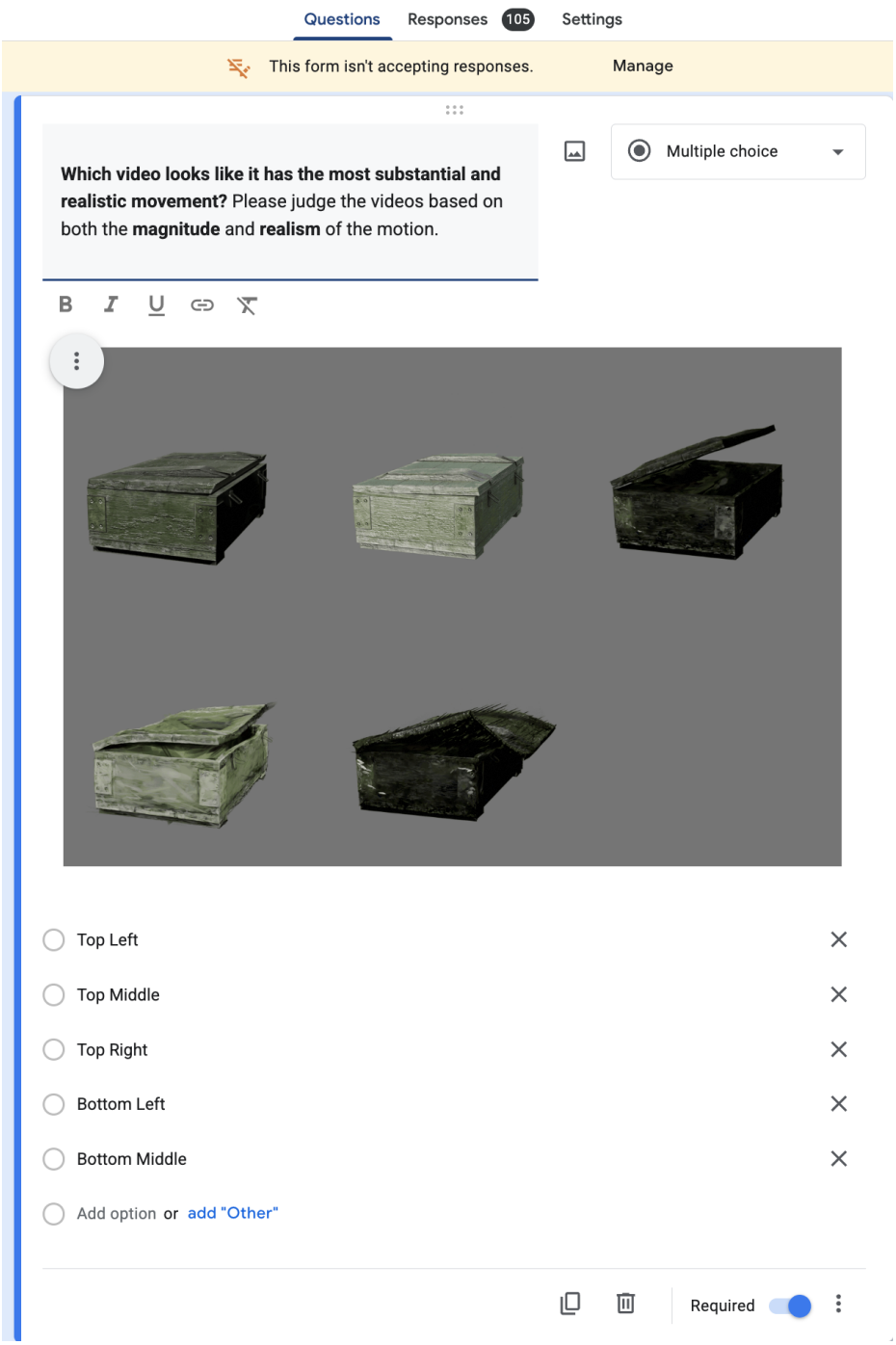


Figure 13. **Screenshot of the user study page.** We use Google forms and Prolfiic to conduct user study, and this page shows the backend management page for the second types of questions.

from different approaches. They are asked to answer two questions:

- "Which video best shows the concept of XXX? Please judge based on how well each one shows this concept." The answer to this question will be used to evaluate "Alignment".
- "Which video looks like it has the most substantial and realistic movement? Please judge the videos based on both the magnitude and realism of the motion." The answer to this question is used to evaluate "Realism".

Table 10. User study results for Prompt Alignment and Motion Realism. Each row shows vote counts for five tested methods.

Metric	Case	AnimateAnyMesh	Ours	PhysDreamer	Pixie	OmniPhysGS	Total
Alignment	Box	7	93	1	0	4	105
	Flower	21	59	13	10	2	105
	Newton	0	105	0	0	0	105
	Shirt	1	80	24	0	0	105
	Cloth	13	49	3	33	7	105
	Bow	6	93	5	0	1	105
	Laptop	0	105	0	0	0	105
	Lamp	1	100	4	0	0	105
	Average	6.125	85.5	6.25	5.375	1.75	105
Realism	Box	13	90	0	2	0	105
	Flower	24	55	10	14	2	105
	Newton	0	105	0	0	0	105
	Shirt	3	77	25	0	0	105
	Cloth	13	68	3	19	2	105
	Bow	3	98	4	0	0	105
	Laptop	0	102	3	0	0	105
	Lamp	0	105	0	0	0	105
	Average	7	87.5	5.625	4.375	0.5	105

We show the vote of participants on different methods and different test cases in Tab. 10. Our method consistently surpasses all the baselines. We show the screenshots of the user study pages in Fig. 12 and Fig. 13

We complement our user study with quantitative evaluations using VBench [46] and WorldScore [32]. Since these benchmarks are primarily designed for video generation models, we adapt their official evaluation toolkits to assess the final rendered videos of 4D results directly, bypassing the standard benchmark pipelines that require text prompts or other input conditions. For VBench [46], we use version 1.0 (as v2 [133] does not support custom inputs) and report three key metrics: *Dynamic Degree*, which measures the magnitude of deliberate motion and differentiates rich dynamics from near-static scenes; *Aesthetic Quality*, which reflects overall artistic appeal, including framing and visual composition; and *Imaging Quality*, which evaluates technical fidelity such as sharpness and artifact levels. We omit background-consistency metrics because our experiments focus on object-centric dynamics. For WorldScore [32], we report *CLIP-Score* to evaluate semantic content alignment and *Flow Magnitude* to quantify motion strength. For CLIP-Score [101], we use a simple text template, `A video of CATEGORY MOTION`, such as, `A video of a flower swinging in the wind`. We exclude perceptual metrics (*CLIP-IQA+* and *CLIP-Aesthetic*) to avoid redundancy, as visual fidelity is already assessed by VBench [46]. We also omit metrics designed to proxy 3D consistency or background stability (*e.g.*, reprojection error, Flow-AEPE, or Gram-style consistency), as they are inapplicable in our setting where all evaluated videos are rendered directly from dynamic 3D objects.

E.3. Analysis of Energy Conservation

In Fig. 8 the main paper, we analyzed whether the resulting framework is conserving energy and ablated the usage of Lagrangian mechanics modeling of latent dynamics. We detail the settings in this experiment in this section.

We conduct this experiment in the simulation of *Box* example. From a stationary pose, the top lid of the box is falling down due to gravity, and its kinetic energy increase during this process. The total energy of this process is roughly constant, showcasing the physical consistency of our approach.

We compare the generated latent dynamics with a simple trajectory generated by the linear interpolation of two latent state vectors. The resulting trajectory is not conserving a constant total energy, and therefore this generated trajectory is not physically consistent.

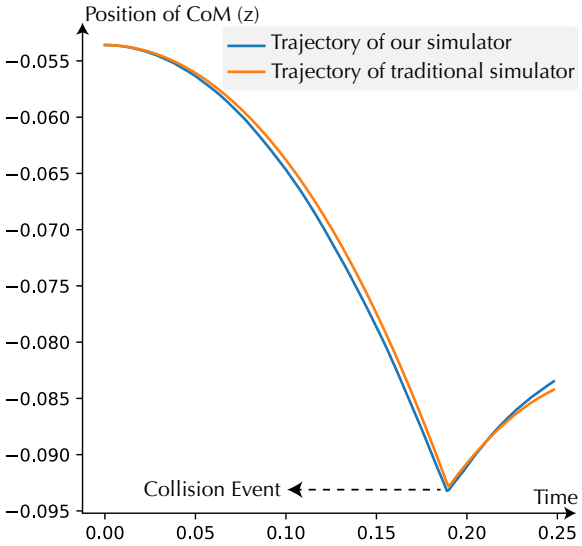


Figure 14. **Comparison with ground-truth physical simulation.** On the *Box* example, we compare the center-of-mass trajectory of our generated 4D motion against a ground-truth sequence produced by a traditional simulator with manual physical annotations. The two trajectories align closely, and our result accurately captures the timing and position of the collision event.

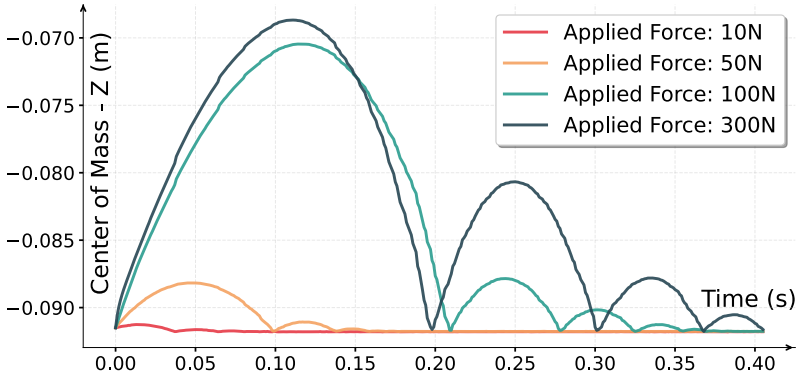


Figure 15. **Generation under forces of varying magnitude.** On the *Box* example, we condition the generation on external forces of different magnitudes and plot the resulting center-of-mass trajectories. Larger forces yield more pronounced and repeated bouncing motions, while the smallest force (10N) produces almost no motion, consistent with physical intuition.

E.4. Comparison with Ground-Truth Physical Simulation

To further assess the physical fidelity of our generation, we compare our generated 4D dynamics on the *Box* example against a ground-truth trajectory computed by a traditional physical simulator with manually specified physical parameters. As shown in Fig. 14, our method accurately reproduces the dynamics, including the timing and position of the collision event. Quantitatively, the average Chamfer distance between the generated and ground-truth mesh sequences is 0.021 (L_1) and 2.7×10^{-4} (L_2^2) within a normalized $[-1, 1]^3$ space, corresponding to a relative deviation of $\approx 1\%$. The close alignment of the center-of-mass trajectories indicates that the motion produced by our latent-space Euler-Lagrange formulation (Thm. 1 and Cor. 1) is not only visually plausible but also physically grounded.

E.5. Generation under Forces of Varying Magnitude

Because our framework conditions on rigorously defined physical quantities rather than abstract directional cues, it naturally responds to the magnitude of the applied force. On the *Box* example, we apply external forces of varying magnitude and plot the resulting center-of-mass trajectories in Fig. 15. Larger forces produce more pronounced motions with higher and repeated bounces, whereas the smallest force (10N) yields almost no displacement. This behavior is consistent with physical intuition and shows that our method supports fine-grained, physically meaningful conditioning.

Table 11. **Ablation study on different dimensions of model reduction.**

	Chamfer (L1) ↓	Chamfer (L2) ↓	IoU ↑
NEUROK w/o Model Reduction	0.045	0.059	0.711
NEUROK w/ $k_q = 5$	0.032	0.034	0.711
NEUROK w/ $k_q = 25$ (default)	0.028	0.028	0.764
NEUROK w/ $k_q = 100$	0.023	0.025	0.806
NEUROK w/ $k_q = 250$	0.021	0.023	0.814
NEUROK w/ $k_q = 500$	0.018	0.018	0.830
NEUROK w/ $k_q = 1000$	0.014	0.012	0.833

E.6. Analysis of Model Reduction

The latent space learned by the VAE is high-dimensional. To obtain a more compact reduced-order representation, we perform dimensionality reduction to compress the original latent space $\mathcal{Z} \subseteq \mathbb{R}^k$ into a lower-dimensional subspace $\mathcal{Q} \subseteq \mathbb{R}^{k_q}$, where $k_q \ll k$, using the Active Subspace Method [24]. We additionally conduct an ablation study to evaluate the effect of such model-reduction and the effect of different reduced dimensionality k_q, as reported in Table 11. The results show that model reduction significantly improves optimization robustness and accuracy, and that performance gains as k_q increases. That being said, a large k_q will increase the computational overhead, therefore we choose $k_q = 25$ as our default configuration.

F. Discussions

F.1. Comparison to Other Paradigm

NEUROK and reduced-order models As discussed in the main paper, existing reduced-order simulation methods [5, 16, 21, 22, 37, 50, 51, 61, 75, 94, 103, 108, 110, 112, 114, 137] are designed to accelerate simulations for known physical systems. They primarily address the forward problem: given system configurations and governing equations, predict the resulting dynamics. These approaches are highly effective in that regime. However, in computer vision we rarely have access to system configurations in the first place—the setting is inherently an inverse problem.

This difference motivates the core distinction between our method and reduced-order models. Our approach adopts a computer-vision and inverse-graphics perspective: we prioritize generalizability, minimal conditioning, and visually plausible 4D generation rather than strict physical fidelity. Technically, reduced-order approaches rely on an explicit physical system specification, and their latent dynamics are solved under known physical constraints. In contrast, our formulation learns dynamics through Lagrangian mechanics defined over raw 4D observations without assuming prior knowledge of the underlying system.

From a modeling standpoint, reduced-order techniques typically overfit to a single, fixed physical system, whereas our model performs amortized inference across previously unseen shapes. Ultimately, these differences arise from fundamentally different goals: reduced-order models solve forward simulation problems, while our method targets generative AI and inverse problems where the physical system itself is unknown.

NEUROK and learning-based constitutive laws A recent line of work [88, 93] has made impressive progress by introducing learning-based constitutive laws to enhance continuum-body simulators. These methods represent an important advance in physical modeling, providing improved accuracy and flexibility compared to traditional hand-crafted constitutive formulations. Nevertheless, because they remain rooted in continuum-body mechanics, they are not designed to handle objects with discontinuities or the articulated and multi-body rigid objects studied in this paper. Their objectives also differ from ours: whereas these approaches aim for increasingly accurate physical simulation, we focus on visually plausible 4D generation without requiring explicit physical supervision or ground-truth dynamic trajectories. This difference in goals and assumptions makes direct comparison infeasible, as constitutive-law learning typically operates on a fixed object with known training trajectories, while our approach performs amortized inference on objects that are unseen during training.

NEUROK and end-to-end dynamic models Another way to pursue category-agnostic simulative dynamics learning is through fully end-to-end models, typically based on GNNs [3, 15, 64, 70, 72, 99, 106, 107, 109]. These methods have achieved impressive results in controlled settings, but they generally require large numbers of action–state pairs for training—data that is rarely available in real-world scenarios. As a result, such approaches tend to struggle when applied to

in-the-wild objects, whereas our method remains robust across diverse real-world instances, as demonstrated on our project page. This mismatch in assumptions also makes direct comparison infeasible: end-to-end dynamic models rely on explicit physical annotations and action trajectories, while our method does not assume any such supervision and instead learns directly from raw 4D observations.

F.2. Limitations

The current system is built on the assumption that the dynamic structure of the modeled object is low-dimensional and the geometry of the object is concrete, therefore it cannot be used to model phenomena involving fluids [38]. We also assume that there is no geometric growth, elimination, and addition for different kinematic states. Therefore, it cannot be used to model natural phenomena such as the growth of plants [39].

These limitations can be addressed by using more flexible geometry representations other than Lagrangian deformations, and we leave this to future works.

F.3. Future works

Our NEUROK opens a new research paradigm for physically-inspired 4D generation, and there are various exciting future directions to explore within this clean and general framework.

More flexible geometry representations. The first future work as discussed above is more flexible geometric representations. The current version of NEUROK uses a reference mesh and Lagrangian deformations to model different kinematic state of objects. A fully implicit representation [14, 120] can enable more flexible kinematic state parameterization, empowering this framework to model more diverse dynamic objects.

Equivariant network architecture design. Our current neural network design uses plain transformers [111] to model the deformation fields. It would be more data-efficient if we could leverage the $SE(3)$ equivariance of this output modality. This can be achieved by using equivariant architectures [28] in network design.

Scaling up using 2D video models and 3D foundation models. This is arguably the most exciting direction for extending our paradigm. As video generative models [113] and 3D/4D reconstruction models [115] continue to advance, we expect substantially richer 4D data to become available in the near future. Because our pipeline does not rely on physical supervision or action annotations—and instead trains purely from raw 4D observations—it is naturally positioned to scale with these developments. Its data requirements align directly with the trend toward large, diverse 4D datasets, making it a strong candidate for training general-purpose neural object simulators and forming key building blocks for universal 3D world models [42].

Acknowledgments. This work is in part supported by NSF RI #2211258 and #2338203, ONR YIP N00014-24-1-2117, ONR MURI N00014-22-1-2740, the Stanford Institute for Human-Centered AI (HAI), and the Magic Grant from the Brown Institute for Media Innovation. We acknowledge the compute support from the NSF ACCESS program #CIS250696, Stanford Data Science and Marlowe Computing Platform, and the AMD University Program for AI & HPC Cluster. We thank Robyn Lockwood (Stanford Language Center) for editorial and writing suggestions that improved the clarity of the manuscript. We thank Chong Zeng and Ruocheng Wang for early feedback on the manuscript and members of Stanford Vision and Learning Lab and Stanford Graphics Lab for fruitful discussion.

References

[1] Stephen W Bailey, Dalton Omens, Paul Dilorenzo, and James F O'Brien. Fast and deep facial deformations. *ACM Transactions on Graphics (TOG)*, 39(4):94–1, 2020. 3

[2] Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019. 3

[3] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016. 3, 34

[4] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the national academy of sciences*, 110(45):18327–18332, 2013. 3, 11

[5] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015. 3, 34

[6] Ravinder Bhattoo, Sayan Ranu, and NM Krishnan. Learning articulated rigid body dynamics with lagrangian graph neural network. *Advances in Neural Information Processing Systems*, 35:29789–29800, 2022. 3

[7] David Bindel. Numerical methods for data science, 2018. 6

[8] Volker Blanz and Thomas Vetter. Face recognition based on fitting a 3d morphable model. *IEEE Transactions on pattern analysis and machine intelligence*, 25(9):1063–1074, 2003. 3

[9] James Booth, Anastasios Roussos, Stefanos Zafeiriou, Allan Ponniah, and David Dunaway. A 3d morphable model learnt from 10,000 faces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5543–5552, 2016. 3

[10] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. Projective dynamics: fusing constraint projections for fast simulation. *ACM Transactions on Graphics (TOG)*, 33(4):1–11, 2014. 2

[11] Aljaz Bozic, Pablo Palafox, Michael Zollhofer, Justus Thies, Angela Dai, and Matthias Nießner. Neural deformation graphs for globally-consistent non-rigid reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1450–1459, 2021. 3, 6, 8, 29

[12] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, 1995. 29

[13] Junhao Cai, Yuji Yang, Weihao Yuan, Yisheng He, Zilong Dong, Liefeng Bo, Hui Cheng, and Qifeng Chen. Gic: Gaussian-informed continuum for physical property identification and simulation. *Advances in Neural Information Processing Systems*, 37: 75035–75063, 2024. 2

[14] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16123–16133, 2022. 35

[15] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016. 3, 34

[16] Yue Chang, Peter Yichen Chen, Zhecheng Wang, Maurizio M Chiaramonte, Kevin Carlberg, and Eitan Grinspun. Licrom: Linear-subspace continuous reduced order modeling with neural fields. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–12, 2023. 3, 34

[17] Boyuan Chen, Hanxiao Jiang, Shaowei Liu, Saurabh Gupta, Yunzhu Li, Hao Zhao, and Shenlong Wang. Physgen3d: Crafting a miniature interactive world from a single image. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6178–6189, 2025. 2

[18] Chuhao Chen, Zhiyang Dou, Chen Wang, Yiming Huang, Anjun Chen, Qiao Feng, Jiatao Gu, and Lingjie Liu. Vid2sim: Generalizable, video-based reconstruction of appearance, geometry and physics for mesh-free simulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26545–26555, 2025. 2

[19] Chuhao Chen, Isabella Liu, Xinyue Wei, Hao Su, and Minghua Liu. Freeart3d: Training-free articulated object generation using 3d diffusion. In *SIGGRAPH Asia 2025 Conference Papers*, 2025. 6, 8, 29

[20] Honglin Chen, Rundi Wu, Eitan Grinspun, Changxi Zheng, and Peter Yichen Chen. Implicit neural spatial representations for time-dependent pdes. In *International Conference on Machine Learning*, pages 5162–5177. PMLR, 2023. 3

[21] Peter Yichen Chen, Jinxu Xiang, Dong Heon Cho, Yue Chang, GA Pershing, Henrique Teles Maia, Maurizio M Chiaramonte, Kevin Carlberg, and Eitan Grinspun. Crom: Continuous reduced-order modeling of pdes using implicit neural representations. *arXiv preprint arXiv:2206.02607*, 2022. 3, 34

[22] Peter Yichen Chen, Maurizio M Chiaramonte, Eitan Grinspun, and Kevin Carlberg. Model reduction for the material point method via an implicit neural representation of the deformation map. *Journal of Computational Physics*, 478:111908, 2023. 3, 34

[23] Zoey Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Karthikeya Vemuri, Alan Wu, Dieter Fox, and Abhishek Gupta. URDFormer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024. 2

[24] Paul G Constantine, Eric Dow, and Qiqi Wang. Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524, 2014. 6, 19, 34

[25] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020. 3

[26] Rishit Dagli, Donglai Xiang, Vismay Modi, Charles Loop, Clement Fuji Tsang, Anka He Chen, Anita Hu, Gavriel State, David IW Levin, and Maria Shugrina. Vomp: Predicting volumetric mechanical property fields. *arXiv preprint arXiv:2510.22975*, 2025. 2

[27] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36:35799–35813, 2023. 6, 24, 26

[28] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12200–12209, 2021. 35

[29] Yufan Deng, Yuhao Zhang, Chen Geng, Shangzhe Wu, and Jiajun Wu. Anymate: A dataset and baselines for learning 3d object rigging. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–10, 2025. 26

[30] Konstantinos G Derpanis. Overview of the ransac algorithm. *Image Rochester NY*, 4(1):2–3, 2010. 24

[31] Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Diffpd: Differentiable projective dynamics. *ACM Transactions on Graphics (ToG)*, 41(2):1–21, 2021. 2

[32] Haoyi Duan, Hong-Xing Yu, Sirui Chen, Li Fei-Fei, and Jiajun Wu. Worldscore: A unified evaluation benchmark for world generation. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2025. 8, 32

[33] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 6

[34] Yutao Feng, Yintong Shang, Xuan Li, Tianjia Shao, Chenfanfu Jiang, and Yin Yang. Pie-nerf: Physics-based interactive elastodynamics with nerf. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4450–4461, 2024. 2

[35] Marc Finzi, Ke Alexander Wang, and Andrew G Wilson. Simplifying hamiltonian and lagrangian neural networks via explicit constraints. *Advances in neural information processing systems*, 33:13880–13889, 2020. 3

[36] fishbotics. urchin. https://github.com/fishbotics/urchin. 26

[37] Lawson Fulton, Vismay Modi, David Duvenaud, David IW Levin, and Alec Jacobson. Latent-space dynamics for reduced deformable simulation. In *Computer graphics forum*, pages 379–391. Wiley Online Library, 2019. 3, 34

[38] Yue Gao, Hong-Xing Yu, Bo Zhu, and Jiajun Wu. Fluidnexus: 3d fluid reconstruction and prediction from a single video. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26091–26101, 2025. 35

[39] Chen Geng, Yunzhi Zhang, Shangzhe Wu, and Jiajun Wu. Birth and death of a rose. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26102–26113, 2025. 35

[40] Simon Giebenhain, Tobias Kirschstein, Markos Georgopoulos, Martin Rünz, Lourdes Agapito, and Matthias Nießner. Learning neural parametric head models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21003–21012, 2023. 3

[41] Michelle Guo, Matt Jen-Yuan Chiang, Igor Santesteban, Nikolaos Sarafianos, Hsiao-yu Chen, Oshri Halimi, Aljaž Božič, Shunsuke Saito, Jiajun Wu, C Karen Liu, et al. Pgc: Physics-based gaussian cloth from a single pose. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21215–21225, 2025. 2

[42] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3), 2018. 35

[43] Guangzhao He, Chen Geng, Shangzhe Wu, and Jiajun Wu. Category-agnostic neural object rigging. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22078–22088, 2025. 3, 6, 8, 29

[44] Jingwei Huang, Hao Su, and Leonidas J. Guibas. Robust watertight manifold surface generation method for shapenet models. *CoRR*, abs/1802.01698, 2018. 29

[45] Qixing Huang, Xiangru Huang, Bo Sun, Zaiwei Zhang, Junfeng Jiang, and Chandrajit Bajaj. Arapreg: An as-rigid-as possible regularization loss for learning deformable shape generators. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5815–5825, 2021. 3

[46] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818, 2024. 8, 32

[47] Team Hunyuan3D, Shuhui Yang, Mingxin Yang, Yifei Feng, Xin Huang, Sheng Zhang, Zebin He, Di Luo, Haolin Liu, Yunfei Zhao, Qingxiang Lin, Zeqiang Lai, Xianghui Yang, Huiwen Shi, Zibo Zhao, Bowen Zhang, Hongyu Yan, Lifu Wang, Sicong Liu, Jihong Zhang, Meng Chen, Liang Dong, Yiwen Jia, Yulin Cai, Jiaao Yu, Yixuan Tang, Dongyuan Guo, Junlin Yu, Hao Zhang, Zheng Ye, Peng He, Runzhou Wu, Shida Wei, Chao Zhang, Yonghao Tan, Yifu Sun, Lin Niu, Shirui Huang, Bojian Zheng, Shu Liu, Shilin Chen, Xiang Yuan, Xiaofeng Yang, Kai Liu, Jianchen Zhu, Peng Chen, Tian Liu, Di Wang, Yuhong Liu, Linus, Jie Jiang, Jingwei Huang, and Chunchao Guo. Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready PBR material. *CoRR*, abs/2506.15442, 2025. 26

[48] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021. 5, 22

[49] Tomas Jakab, Richard Tucker, Ameesh Makadia, Jiajun Wu, Noah Snavely, and Angjoo Kanazawa. Keypointdeformer: Unsupervised 3d keypoint discovery for shape control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12783–12792, 2021. 3, 6, 8, 29

[50] Doug L James and Dinesh K Pai. Dyrt: Dynamic response textures for real time deformation simulation with graphics hardware. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 582–585, 2002. 3, 34

[51] Doug L James, Jernej Barbič, and Dinesh K Pai. Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics (TOG)*, 25(3):987–995, 2006. 3, 34

[52] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 courses*, pages 1–52, 2016. 2, 4, 13, 14

[53] Hanxiao Jiang, Hao-Yu Hsu, Kaifeng Zhang, Hsin-Ni Yu, Shenlong Wang, and Yunzhu Li. Phystwin: Physics-informed reconstruction and simulation of deformable objects from videos. *ICCV*, 2025. 2

[54] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5616–5626, 2022. 2

[55] Takuhiro Kaneko. Improving physics-augmented continuum neural radiance field-based geometry-agnostic system identification with lagrangian particle optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5480, 2024. 2

[56] Justin Kerr, Chung Min Kim, Mingxuan Wu, Brent Yi, Qianqian Wang, Ken Goldberg, and Angjoo Kanazawa. Robot see robot do: Imitating articulated object manipulation with monocular 4d reconstruction. *arXiv preprint arXiv:2409.18121*, 2024. 2

[57] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 5

[58] Lev Davidovich Landau and Evgeniĭ Mikhaĭlovich Lifshitz. *Mechanics*. CUP Archive, 1960. 2, 4, 6, 13, 15, 19

[59] Long Le, Jason Xie, William Liang, Hung-Ju Wang, Yue Yang, Yecheng Jason Ma, Kyle Vedder, Arjun Krishna, Dinesh Jayaraman, and Eric Eaton. Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. *arXiv preprint arXiv:2410.13882*, 2024. 2

[60] Long Le, Ryan Lucas, Chen Wang, Chuhao Chen, Dinesh Jayaraman, Eric Eaton, and Lingjie Liu. Pixie: Fast and generalizable supervised learning of 3d physics from pixels. *arXiv preprint arXiv:2508.17437*, 2025. 8

[61] Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020. 3, 34

[62] Jiahui Lei, Congyue Deng, William B Shen, Leonidas J Guibas, and Kostas Daniilidis. Nap: Neural 3d articulated object prior. *Advances in Neural Information Processing Systems*, 36:31878–31894, 2023. 2

[63] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR, 2023. 2, 12

[64] Chenchang Li, Zihao Ai, Tong Wu, Xiaosa Li, Wenbo Ding, and Huazhe Xu. Deformnet: Latent space modeling and dynamics prediction for deformable object manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14770–14776. IEEE, 2024. 3, 34

[65] Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M Kaufman. Incremental potential contact: intersection- and inversion-free, large-deformation dynamics. *ACM Transactions on Graphics (TOG)*, 39(4):49, 2020. 21

[66] Sizhe Lester Li, Annan Zhang, Boyuan Chen, Hanna Matusik, Chao Liu, Daniela Rus, and Vincent Sitzmann. Controlling diverse robots by inferring jacobian fields with deep networks. *Nature*, pages 1–7, 2025. 3

[67] Xuan Li, Yadi Cao, Minchen Li, Yin Yang, Craig Schroeder, and Chenfanfu Jiang. Plasticitynet: Learning to simulate metal, sand, and snow for optimization time integration. *Advances in Neural Information Processing Systems*, 35:27783–27796, 2022. 2

[68] 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. 2

[69] Xuan Li, Chang Yu, Wenxin Du, Ying Jiang, Tianyi Xie, Yunuo Chen, Yin Yang, and Chenfanfu Jiang. Dress-1-to-3: Single image to simulation-ready 3d outfit with diffusion prior and differentiable physics. *ACM Transactions on Graphics (TOG)*, 44(4):1–16, 2025. 2

[70] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018. 3, 34

[71] Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. Diffcloth: Differentiable cloth simulation with dry frictional contact. *ACM Transactions on Graphics (TOG)*, 42(1):1–20, 2022. 2

[72] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. In *Conference on Robot Learning*, pages 112–123. PMLR, 2022. 3, 34

[73] Yichen Li, Peter Yichen Chen, Tao Du, and Wojciech Matusik. Learning preconditioners for conjugate gradient pde solvers. In *International Conference on Machine Learning*, pages 19425–19439. PMLR, 2023. 3

[74] Yifei Li, Hsiao-yu Chen, Egor Larionov, Nikolaos Sarafianos, Wojciech Matusik, and Tuur Stuyck. Diffavatar: Simulation-ready garment optimization with differentiable simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4368–4378, 2024. 2

[75] Yue Li, Gene Wei-Chin Lin, Egor Larionov, Aljaz Bozic, Doug Roble, Ladislav Kavan, Stelian Coros, Bernhard Thomaszewski, Tuur Stuyck, and Hsiao-Yu Chen. Self-supervised learning of latent space dynamics. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 8(4):1–18, 2025. 3, 34

[76] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020. 3

[77] Zizhang Li, Hong-Xing Yu, Wei Liu, Yin Yang, Charles Herrmann, Gordon Wetzstein, and Jiajun Wu. Wonderplay: Dynamic 3d scene generation from a single image and actions. *arXiv preprint arXiv:2505.18151*, 2025. 2, 12, 13

[78] Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. *Advances in neural information processing systems*, 32, 2019. 2

[79] Jiajing Lin, Zhenzhong Wang, Dejun Xu, Shu Jiang, YunPeng Gong, and Min Jiang. Phys4dgen: Physics-compliant 4d generation with multi-material composition perception. *arXiv preprint arXiv:2411.16800*, 2024. 2

[80] Jiajing Lin, Shu Jiang, Qingyuan Zeng, Zhenzhong Wang, and Min Jiang. Visionlaw: Inferring interpretable intrinsic dynamics from visual observations via bilevel optimization. *arXiv preprint arXiv:2508.13792*, 2025.

[81] Yuchen Lin, Chenguo Lin, Jianjin Xu, and Yadong Mu. Omniphysgs: 3d constitutive gaussians for general physics-based dynamics generation. *arXiv preprint arXiv:2501.18982*, 2025. 2, 8, 13

[82] Jiayi Liu, Ali Mahdavi-Amiri, and Manolis Savva. Paris: Part-level reconstruction and motion analysis for articulated objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 352–363, 2023. 2

[83] Jiayi Liu, Denys Iliash, Angel X Chang, Manolis Savva, and Ali Mahdavi-Amiri. Singapo: Single image controlled generation of articulated parts in objects. *arXiv preprint arXiv:2410.16499*, 2024. 2, 6, 8, 29

[84] Ruoshi Liu, Alper Canberk, Shuran Song, and Carl Vondrick. Differentiable robot rendering. *arXiv preprint arXiv:2410.13851*, 2024. 3

[85] Shaowei Liu, Zhongzheng Ren, Saurabh Gupta, and Shenlong Wang. Physgen: Rigid-body physics-grounded image-to-video generation. In *European Conference on Computer Vision*, pages 360–378. Springer, 2024. 2

[86] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: a skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, 34(6):1–16, 2015. 3

[87] Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490*, 2019. 3

[88] Pingchuan Ma, Peter Yichen Chen, Bolei Deng, Joshua B Tenenbaum, Tao Du, Chuang Gan, and Wojciech Matusik. Learning neural constitutive laws from motion observations for generalizable pde dynamics. In *International Conference on Machine Learning*, pages 23279–23300. PMLR, 2023. 3, 34

[89] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. Xpbd: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, pages 49–54, 2016. 2

[90] Arman Maesumi, Paul Guerrero, Noam Aigerman, Vladimir Kim, Matthew Fisher, Siddhartha Chaudhuri, and Daniel Ritchie. Explorable mesh deformation subspaces from unstructured 3d generative models. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11, 2023. 3

[91] Zhao Mandi, Yijia Weng, Dominik Bauer, and Shuran Song. Real2code: Reconstruct articulated objects via code generation. *arXiv preprint arXiv:2406.08474*, 2024. 2

[92] Zhao Mandi, Yifan Hou, Dieter Fox, Yashraj Narang, Ajay Mandlekar, and Shuran Song. Dexmachina: Functional retargeting for bimanual dexterous manipulation. *arXiv preprint arXiv:2505.24853*, 2025. 2

[93] Himangi Mittal, Peiye Zhuang, Hsin-Ying Lee, and Shubham Tulsiani. Uniphy: Learning a unified constitutive model for inverse physics simulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16208–16218, 2025. 2, 3, 34

[94] Vismay Modi, Nicholas Sharp, Or Perel, Shinjiro Sueda, and David IW Levin. Simplicits: Mesh-free, geometry-agnostic elastic simulation. *ACM Transactions on Graphics (TOG)*, 43(4):1–11, 2024. 2, 3, 34

[95] Atsuhiro Noguchi, Umar Iqbal, Jonathan Tremblay, Tatsuya Harada, and Orazio Gallo. Watch it move: Unsupervised discovery of 3d joints for re-posing of articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3677–3687, 2022. 2

[96] Pablo Palafox, Aljaž Božič, Justus Thies, Matthias Nießner, and Angela Dai. Npms: Neural parametric models for 3d deformable shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12695–12705, 2021. 3

[97] Pablo Palafox, Nikolaos Sarafianos, Tony Tung, and Angela Dai. Spams: Structured implicit parametric models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12851–12860, 2022. 3

[98] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10975–10985, 2019. 3

[99] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International conference on learning representations*, 2020. 3, 34

[100] Yi-Ling Qiao, Alexander Gao, and Ming Lin. Neuphysics: Editable neural geometry and physics from monocular videos. *Advances in Neural Information Processing Systems*, 35:12841–12854, 2022. 2

[101] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 8, 32

[102] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. 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, 2019. 3

[103] Cristian Romero, Dan Casas, Jesús Pérez, and Miguel Otaduy. Learning contact corrections for handle-based subspace dynamics. *ACM Transactions on Graphics (ToG)*, 40(4):1–12, 2021. 3, 34

[104] Quanyuan Ruan, Jiabao Lei, Wenhao Yuan, Yanglin Zhang, Dekun Lu, Guiliang Liu, and Kui Jia. Prof. robot: Differentiable robot rendering without static and self-collisions. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22562–22572, 2025. 3

[105] JL Safko, Herbert Goldstein, and C Poole. Classical mechanics, 2002. 19

[106] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020. 3, 34

[107] Connor Schenck and Dieter Fox. Spnets: Differentiable fluid dynamics for deep neural networks. In *Conference on Robot Learning*, pages 317–335. PMLR, 2018. 3, 34

[108] Nicholas Sharp, Cristian Romero, Alec Jacobson, Etienne Vouga, Paul Kry, David IW Levin, and Justin Solomon. Data-free learning of reduced-order kinematics. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–9, 2023. 3, 34

[109] Bokui Shen, Zhenyu Jiang, Christopher Choy, Silvio Savarese, Leonidas J Guibas, Anima Anandkumar, and Yuke Zhu. Action-conditional implicit visual dynamics for deformable object manipulation. *The International Journal of Robotics Research*, 43(4): 437–455, 2024. 3, 34

[110] Siyuan Shen, Yang Yin, Tianjia Shao, He Wang, Chenfanfu Jiang, Lei Lan, and Kun Zhou. High-order differentiable autoencoder for nonlinear model reduction. *arXiv preprint arXiv:2102.11026*, 2021. 3, 34

[111] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 5, 22, 35

[112] Hrishikesh Viswanath, Yue Chang, Aleksey Panas, Julius Berner, Peter Yichen Chen, and Aniket Bera. Reduced-order neural operators: Learning lagrangian dynamics on highly sparse graphs. *arXiv preprint arXiv:2407.03925*, 2024. 3, 34

[113] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 35

[114] Jiahong Wang, Yinwei Du, Stelian Coros, and Bernhard Thomaszewski. Neural modes: Self-supervised learning of nonlinear modal subspaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23158–23167, 2024. 3, 34

[115] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025. 35

[116] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3dn: 3d deformation network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1038–1046, 2019. 3

[117] Zijie Wu, Chaohui Yu, Fan Wang, and Xiang Bai. Animateanymesh: A feed-forward 4d foundation model for text-driven universal mesh animation. *arXiv preprint arXiv:2506.09982*, 2025. 8, 12, 29

[118] Hongchi Xia, Entong Su, Marius Memmel, Arhan Jain, Raymond Yu, Numfor Mbiziwo-Tiapo, Ali Farhadi, Abhishek Gupta, Shenlong Wang, and Wei-Chiu Ma. Drawer: Digital reconstruction and articulation with environment realism. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21771–21782, 2025. 2

[119] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020. 2, 6, 8, 12, 24

[120] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21469–21480, 2025. 29, 35

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

[122] Xinli Xu, Wenhang Ge, Dicong Qiu, ZhiFei Chen, Dongyu Yan, Zhuoyun Liu, Haoyu Zhao, Hanfeng Zhao, Shunsi Zhang, Junwei Liang, et al. Gaussianproperty: Integrating physical properties to 3d gaussians with lmms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7231–7240, 2025. 2

[123] Zhenjia Xu, Jiajun Wu, Andy Zeng, Joshua B Tenenbaum, and Shuran Song. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. *arXiv preprint arXiv:1906.03853*, 2019. 2

[124] Haitao Yang, Xiangru Huang, Bo Sun, Chandrajit Bajaj, and Qixing Huang. Gencorres: Consistent shape matching via coupled implicit-explicit shape generative models. *arXiv preprint arXiv:2304.10523*, 2023. 3

[125] Haitao Yang, Bo Sun, Liyan Chen, Amy Pavel, and Qixing Huang. Geolatent: A geometric approach to latent space design for deformable shape generators. *ACM Transactions on Graphics (TOG)*, 42(6):1–20, 2023.

[126] Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 75–83, 2020.

[127] Seungwoo Yoo, Juil Koo, Kyeongmin Yeo, and Minhyuk Sung. Neural pose representation learning for generating and transferring non-rigid object poses. *arXiv preprint arXiv:2406.09728*, 2024. 3

[128] Albert J Zhai, Yuan Shen, Emily Y Chen, Gloria X Wang, Xinlei Wang, Sheng Wang, Kaiyu Guan, and Shenlong Wang. Physical property understanding from language-embedded feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28296–28305, 2024. 2

[129] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions On Graphics (TOG)*, 42(4):1–16, 2023. 5, 23

[130] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T Freeman. Physdreamer: Physics-based interaction with 3d objects via video generation. In *European Conference on Computer Vision*, pages 388–406. Springer, 2024. 2, 7, 8, 12, 13, 14

[131] Haoyu Zhao, Hao Wang, Xingyue Zhao, Hao Fei, Hongqiu Wang, Chengjiang Long, and Hua Zou. Physsplat: Efficient physics simulation for 3d scenes via mllm-guided gaussian splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5242–5252, 2025. 2

[132] Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. *Advances in neural information processing systems*, 36:73969–73982, 2023. 5

[133] Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Lulu Gu, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, et al. Vbench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *arXiv preprint arXiv:2503.21755*, 2025. 32

[134] Yang Zheng, Qingqing Zhao, Guandao Yang, Wang Yifan, Donglai Xiang, Florian Dubost, Dmitry Lagun, Thabo Beeler, Federico Tombari, Leonidas Guibas, et al. Physavatar: Learning the physics of dressed 3d avatars from visual observations. In *European Conference on Computer Vision*, pages 262–284. Springer, 2024. 2

[135] Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and simulation of elastic objects with spring-mass 3d gaussians. In *European Conference on Computer Vision*, pages 407–423. Springer, 2024. 2

[136] Keyang Zhou, Bharat Lal Bhatnagar, and Gerard Pons-Moll. Unsupervised shape and pose disentanglement for 3d meshes. In *European conference on computer vision*, pages 341–357. Springer, 2020. 3

[137] Zeshun Zong, Xuan Li, Minchen Li, Maurizio M Chiaramonte, Wojciech Matusik, Eitan Grinspun, Kevin Carlberg, Chenfanfu Jiang, and Peter Yichen Chen. Neural stress fields for reduced-order elastoplasticity and fracture. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11, 2023. 3, 34

[138] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3