Analytically Integratable Zero-restlength Springs for Capturing Dynamic Modes unrepresented by Quasistatic Neural Networks

Yongxu Jin
yxjin@stanford.edu
Stanford University
Stanford, California, USA
Epic Games
Cary, North Carolina, USA

Yushan Han
yushanh1@math.ucla.edu
University of California, Los Angeles
Los Angeles, California, USA
Epic Games
Cary, North Carolina, USA

Zhenglin Geng
zhenglin.geng@epicgames.com
Epic Games
Cary, North Carolina, USA

Joseph Teran
jteran@math.ucdavis.edu
University of California, Davis
Davis, California, USA
University of California, Los Angeles
Los Angeles, California, USA
Epic Games
Cary, North Carolina, USA

Ronald Fedkiw
rfedkiw@stanford.edu
Stanford University
Stanford, California, USA
Epic Games
Cary, North Carolina, USA

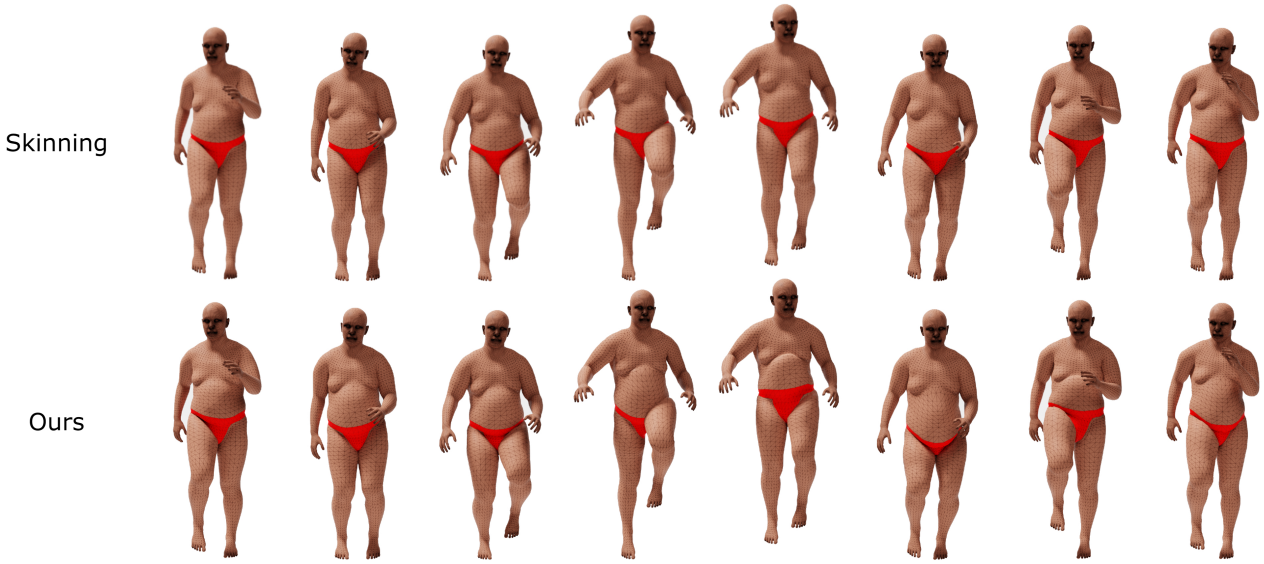


Figure 1: Our method enhances standard skinning with a configuration-only quasistatic neural network (QNN) that approximates quasistatic hyperelasticity as well as analytically integratable zero-restlength springs trained that approximates inertial effects. The QNN fixes well-known skinning artifacts (e.g. in the shoulder regions) and the zero-restlength springs add ballistic motion (e.g. in the belly region). We refer readers to our supplementary video which is far more compelling than still images.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGGRAPH '22 Conference Proceedings, August 7–11, 2022, Vancouver, BC, Canada

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-9337-9/22/08…$15.00
https://doi.org/10.1145/3528233.3530705

ABSTRACT

We present a novel paradigm for modeling certain types of dynamic simulation in real-time with the aid of neural networks. In order to significantly reduce the requirements on data (especially time-dependent data), as well as decrease generalization error, our approach utilizes a data-driven neural network *only* to capture quasistatic information (instead of dynamic or time-dependent information). Subsequently, we augment our quasistatic neural network (QNN) inference with a (real-time) dynamic simulation layer. Our key insight is that the dynamic modes lost when using

a QNN approximation can be captured with a quite simple (and decoupled) zero-restlength spring model, which can be integrated analytically (as opposed to numerically) and thus has no time-step stability restrictions. Additionally, we demonstrate that the spring constitutive parameters can be robustly learned from a surprisingly small amount of dynamic simulation data. Although we illustrate the efficacy of our approach by considering soft-tissue dynamics on animated human bodies, the paradigm is extensible to many different simulation frameworks.

CCS CONCEPTS

• **Computing methodologies** → **Animation**; **Neural networks**; **Physical simulation**.

KEYWORDS

Zero-restlength spring, soft-tissue dynamics, human body animation

ACM Reference Format:
Yongxu Jin, Yushan Han, Zhenglin Geng, Joseph Teran, and Ronald Fedkiw. 2022. Analytically Integratable Zero-restlength Springs for Capturing Dynamic Modes unrepresented by Quasistatic Neural Networks. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings (SIGGRAPH '22 Conference Proceedings), August 7–11, 2022, Vancouver, BC, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3528233.3530705

1 INTRODUCTION

Recently, there has been a lot of interest in using neural networks to approximate dynamic simulation (see e.g. [Holden et al. 2019; Pfaff et al. 2020; Sanchez-Gonzalez et al. 2020; Santesteban et al. 2020]), especially because neural network inference has the potential to run in real-time (on high-end GPUs). Unfortunately, one requires an exorbitant amount of training data in order to represent all the possible temporal transitions between states that these networks aim to model. These networks do not typically generalize well when not enough training data is used. Even if one had access to the exorbitant amount of training data required, an unwieldy amount of network parameters would be required to prevent underfitting.

Some aspects of a dynamic simulation depend mostly on the configuration, whereas others more strongly depend on the time history of configuration to configuration transitions; thus, we propose the following paradigm. Firstly, we construct a neural network that depends only on the configurations (and as such cannot capture dynamic modes). Secondly, we subtract this configuration-only model from the full dynamics in order to obtain a dynamics layer. Thirdly, we propose a dynamic simulation model that can approximate the dynamics layer. Theoretically, a well-approximated dynamics layer has the potential to augment the configuration-only neural network in a way that exactly matches the original simulations. Moreover, if the configuration-only neural network can capture enough of the non-linearities, then the dynamics layer has the potential to be quite simple (and thus real-time).

In this paper, we propose using a quasistatic physics simulation neural network (see e.g. [Bertiche et al. 2020; Geng et al. 2020; Jin et al. 2020; Luo et al. 2018]) as the configuration-only neural network. Since quasistatic neural networks (QNNs) do not have time dependency, they require far less training data and as such can use a much simpler network structure with far fewer parameters than a network that attempts to model temporal transitions. Using less training data on a network designed to capture temporal transitions leads to overfitting and poor generalization to unseen data. Using a simpler network structure with less parameters on a network designed to capture transitions leads to underfitting of the training data (and poor generalization).

Although we expect that an entire cottage industry could be developed around the modelling and real-time simulation of dynamics layers, we propose only a very simple demonstrational model here (but note that it works surprisingly well). Importantly, the zero-restlength spring approximation to the dynamics layer can be integrated analytically and thus has zero truncation error and no time step stability restrictions, making it quite fast and accurate. Furthermore (as shown in Section 7), one can (automatically) robustly learn spring constitutive parameters from a very small amount of dynamic simulation data.

2 RELATED WORK

Stated-based methods. We first discuss prior works that generate elastic deformation directly from spatial state without considering temporal or configurational history. Many works aim to upsample a low-resolution simulation to higher resolution: [Feng et al. 2010] trains a regressor to upsample, [Kavan et al. 2011] learns an upsampling operator, and [Chen et al. 2018] rasterizes the vertex positions into an image before upsampling it and interpolating new vertex positions. [Wang et al. 2010; Xu et al. 2014; Zurdo et al. 2012] use example-based methods to synthesize fine-scale wrinkles from a database. [Patel et al. 2020] predicts a low-frequency mesh with a fully connected network and uses a mixture model to add wrinkles. [Chentanez et al. 2020] upsamples with graph convolutional neural networks. [Wu et al. 2021b] recovers high-frequency geometric details with perturbations of texture. [Lahner et al. 2018] uses a generative adversarial network (GAN) to upsample a cloth normal map for improved rendering. [Bailey et al. 2020, 2018] use neural networks to drive fine scale details from a coarse character rig. Many works aim to learn equilibrium configurations from boundary conditions: [Luo et al. 2018] uses a neural network to add non-linearity to a linear elasticity model. [Mendizabal et al. 2020] learns the non-linear mapping from contact forces to displacements. Such approaches are particularly common in virtual surgery applications, e.g. [De et al. 2011; Liu et al. 2020; Pfeiffer et al. 2019; Roewer-Despres et al. 2018; Salehi and Giannacopoulos 2021]. [Jin et al. 2020] trains a CNN to infer a displacement map which adds wrinkles to skinned cloth, and [Wu et al. 2020] improves the accuracy of this approach by embedding the cloth into a volumetric tetrahedral mesh. [Bertiche et al. 2020] adds physics to the loss function, a common approach in physics-inspired neural networks (PINNs), see e.g. [Raissi et al. 2019]. To avoid the soft constraints of PINNs that only coerce physically-inspired behaviour, [Geng et al. 2020; Srinivasan et al. 2021] add quasistatic simulation as the final layer of a neural network in order to constrain output to physically attainable manifolds.

Transition-based methods. Here we discuss prior works that use a temporal history of states, typically for resolving dynamic/inertia

related behaviors. In one of the earliest works (before the deep learning era) [Grzeszczuk et al. 1998] uses a neural network to learn temporal transitions and leverage back propagation to optimize control parameters. [De Aguiar et al. 2010] incorporates an approximation to the quasistatic equilibrium that serves as a control for a dynamics layer. [Guan et al. 2012] predicts a cloth mesh from body poses and previous frames, solving a linear system to fix penetrations. [Hahn et al. 2014] uses dynamic subspace simulation on an adaptive selected basis generated from the current body pose. [Holden et al. 2019] computes a linear subspace of configurations with principal component analysis (PCA) and learns subspace simulations from previous frames with a fully connected network. [Fulton et al. 2019; Tan et al. 2018, 2020] obtain nonlinear subspaces with autoencoder networks. Similar methods are commonly used to animate fluids using regression forests [Ladickỳ et al. 2015] or recurrent neural networks (RNNs) [Wiewel et al. 2019]. [Pfaff et al. 2020] and [Sanchez-Gonzalez et al. 2020] use graph networks to learn simulations with both fixed and changing topology. [Chentanez et al. 2020] proposes a transition-based model with position and linear/angular velocity of the body as network input (in addition to a state-based model). [Meister et al. 2020] uses a fully connected network to predict node-wise acceleration for total Lagrangian explicit dynamics. [Deng et al. 2020] proposes a convolutional long short-term memory (LSTM) layer to capture elastic force propagation. [Zhang et al. 2021] uses an image based approach to enhance detail in low resolution dynamic simulations.

Secondary dynamics for characters. Numerical methods that resolve the dynamic effects of inertia-driven deformation have a long history in computer graphics skin and flesh animation. We refer interested readers to only a few papers and a plethora of references therein (e.g. [Capell et al. 2002; Sheen et al. 2021; Wang et al. 2020; Xu and Barbič 2016; Zhang et al. 2020]). We note that any of these techniques could be used to generate training data for learning-based methods. Secondary dynamics for characters have also been added using data-driven methods: [Pons-Moll et al. 2015] provides a motion capture dataset with dynamic surface meshes, and proposes a linear auto-regressive model to capture dynamic displacements compressed by PCA. [Loper et al. 2015] extends this method to the SMPL human model. See also [Casas and Otaduy 2018; Santesteban et al. 2020; Seo et al. 2021]. [Kim et al. 2017] proposes a two layer approach which skins a volumetric body model as an inner layer and simulates a tetrahedral mesh as an outer layer. The constitutive parameters of the outer layer are learned from 4D scan data. [Zheng et al. 2021] trains a network to approximate per-vertex displacements from temporal one-ring state using backward Euler simulation data of primitive shapes. [Deng et al. 2020] also uses a one-ring based approach and trains with forward Euler.

Proportional-derivative control. Our analytic zero-restlength spring targeting method resembles proportional-derivative (PD) control algorithms used in both computer graphics and robotics. We refer interested readers to several papers leveraging PD control and control parameter optimization for various usages [Allen et al. 2011; De Luca et al. 2005; Hodgins et al. 1995; Wang et al. 2012; Weinstein et al. 2007].

3 QUASISTATIC NEURAL NETWORK

We use the (freely available) MetaHuman [Epic Games 2021] which has 122 joints and 13575 vertices as our human model. Given joint angles $\boldsymbol{\theta}$, we use a skinning function $\mathbf{x}^{skin}(\boldsymbol{\theta})$ to get the skinned position for each surface vertex. Any reasonable skinning approach (e.g. linear blend skinning [Lewis et al. 2000; Magnenat-Thalmann et al. 1988] and dual quaternion skinning [Kavan et al. 2007]) may be used.

Starting from $\boldsymbol{\theta}$ and $\mathbf{x}^{skin}(\boldsymbol{\theta})$, we aim to train a neural network that predicts a more realistic surface mesh $\mathbf{x}^{net}(\boldsymbol{\theta})$. Generally speaking, we could add our analytically intergratable zero-restlength springs directly on top of the skinning result (and there are many interesting skinning-related methods being proposed recently, e.g. [Wu et al. 2021a]), although our proposed dynamics layer (likely) works best when the shape of the surface skin mesh is approximated as accurately as possible. We obtain ground truth for $\mathbf{x}^{net}(\boldsymbol{\theta})$ via two different approaches: quasistatic simulation (as discussed in Section 3.1) and 4D scanning (which will be discussed in a future paper). Both approaches worked rather well in our experiences.

3.1 Quasistatic Simulation

First, we use Tetgen [Si 2015] (alternatively, [Hu et al. 2018],[Shewchuk 1998] could be used) to create a volumetric tetrahedron mesh whose boundary corresponds to the Metahuman surface mesh in a reference A-pose. Next, we interpolate skinning weights from the Metahuman surface vertices to the tetrahedron mesh boundary vertices, and subsequently solve a Poisson equation on the tetrahedron mesh to propagate the skinning weights to interior vertices [Cong et al. 2015]. Then, we use a geometric approximation to a skeleton in order to specify which interior vertices of the tetrahedron mesh should follow their skinned positions with either Dirichlet boundary conditions or zero-restlength spring penalty forces.

Our training dataset includes about 5000 poses genetared randomly, from motion capture data, and manually specified animations. Given any target pose, specified by a set of joint angles $\boldsymbol{\theta}$, we solve for the equilibrium configuration of the volumetric tetrahedron mesh using the method from [Teran et al. 2005] in order to avoid issues with indefiniteness and the method from [Marquez et al. 2022] to enforce contact boundary conditions on the surface of the tetrahedron mesh. Although simulation can be time-consuming, quasistatic simulation is much faster than dynamic simulation. Furthermore, the amount of simulation required is significantly smaller than that which would be needed to obtain similar efficacy for a network aiming to capture temporal information, since such a network would require far more parameters to prevent underfitting.

3.2 QNN

Instead of inferring the positions of the surface vertices directly, we augment the skinning result $\mathbf{x}^{skin}(\boldsymbol{\theta})$ with per-vertex displacements $\mathbf{d}(\boldsymbol{\theta})$ so that the non-linearities from joint rotations $\boldsymbol{\theta}$ are mostly captured by the skinning. This reduces the demands on the neural network allowing for a smaller model and thus requiring less training data. Given ground truth displacements $\mathbf{d}(\boldsymbol{\theta})$, we train our quasistatic neural network (QNN) to minimize the loss between $\mathbf{d}(\boldsymbol{\theta})$ and the network inferred result $\mathbf{d}^{net}(\boldsymbol{\theta})$. We follow an approach similar to [Jin et al. 2020] rasterizing the per-vertex displacements

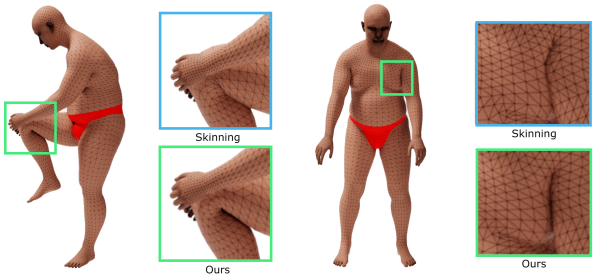


Figure 2: Our QNN resolves well-known skinning collision artifacts. We demonstrate this in extreme poses involving the back of the knee and the armpit.

into a displacement map image so that a convolutional neural network (CNN) can be used. Of course, one could alternatively use PCA with a fully connected network; however, GPUs are more amenable to the image-based frameworks used by CNNs (see e.g. [Wang 2021], which discusses the benefit of using data structures that resemble images on GPUs). Our QNN can fix skinning artifacts like interpenetration and volume loss (see Figure 2), thus providing a simpler dynamics layer for analytic zero-restlength springs to capture (see Section 7 for discussions). Since the QNN is not the main contribution of this paper, we refer readers to the original paper [Jin et al. 2020] for technical details (network architectures, optimizers, hyperparameters, etc.). The QNN used in this paper can also be easily replaced with other state-based models.

4 KINEMATICS

The skeletal animation will be queried at a user-specified time scale (likely proportional to the frame rate). While these samples are inherently discrete, our approach utilizes the analytic solution of temporal ODEs; therefore, we extend these discrete samples to the continuous time domain. Specifically, given a sequence of skeletal joint angles $\{\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \ldots\}$, we construct a target function of surface vertex positions $\hat{\mathbf{x}}(t)$ defined for all $t \geq 0$. Options include e.g. Heaviside (discontinous), piecewise linear (C^0), or cubic (C^1) interpolation. We utilize cubic interpolation given its relative simplicity and favorable continuity. Between sample n at time t^n and sample $n+1$ at time $t^n + \Delta t$, we define

$$\hat{\mathbf{x}}(t^n + s\Delta t) = \hat{\mathbf{q}}^n(s\Delta t)^3 + \hat{\mathbf{a}}^n(s\Delta t)^2 + \hat{\mathbf{b}}^n s\Delta t + \hat{\mathbf{c}}^n \tag{1}$$

$$= \mathbf{q}^n s^3 + \mathbf{a}^n s^2 + \mathbf{b}^n s + \mathbf{c}^n, \tag{2}$$

where $s \in [0,1]$ and Equation 2 absorbs the powers of Δt into the non-hatted variables for simplicity of exposition. Enforcing C^1 continuity at times t^n and t^{n+1} requires the following position and derivative constraints

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{q}^n \\ \mathbf{a}^n \\ \mathbf{b}^n \\ \mathbf{c}^n \end{bmatrix} = \begin{bmatrix} \mathbf{x}^{net}(\boldsymbol{\theta}^n) \\ \frac{1}{2}(\mathbf{x}^{net}(\boldsymbol{\theta}^{n+1}) - \mathbf{x}^{net}(\boldsymbol{\theta}^{n-1})) \\ \mathbf{x}^{net}(\boldsymbol{\theta}^{n+1}) \\ \frac{1}{2}(\mathbf{x}^{net}(\boldsymbol{\theta}^{n+2}) - \mathbf{x}^{net}(\boldsymbol{\theta}^{n})) \end{bmatrix}, \tag{3}$$

which can readily be solved to determine $\mathbf{q}^n, \mathbf{a}^n, \mathbf{b}^n, \mathbf{c}^n$. Here, $\mathbf{x}^{net}(\boldsymbol{\theta}^n) = \mathbf{x}^{skin}(\boldsymbol{\theta}^n) + \mathbf{d}^{net}(\boldsymbol{\theta}^n)$ are QNN-inferred surface vertex positions at time t^n. Note, in the first interval, $\frac{1}{2}(\mathbf{x}^{net}(\boldsymbol{\theta}^{n+1}) - \mathbf{x}^{net}(\boldsymbol{\theta}^{n-1}))$ is replaced by the one-sided difference $\mathbf{x}^{net}(\boldsymbol{\theta}^{n+1}) - \mathbf{x}^{net}(\boldsymbol{\theta}^n)$.

5 DYNAMICS

We connect a particle (with mass m) to each kinematic vertex $\hat{\mathbf{x}}(t^n + s\Delta t)$ using a zero-restlength spring (although other analytically integratable dynamic models could be used). The position of each simulated particle obeys Hooke's law,

$$\ddot{\mathbf{x}}(t) = k_s(\hat{\mathbf{x}}(t) - \mathbf{x}(t)) + k_d(\dot{\hat{\mathbf{x}}}(t) - \dot{\mathbf{x}}(t)), \tag{4}$$

where k_s and k_d are the spring stiffness and damping (both divided by the mass m) respectively. This equation can be analytically integrated (separately for each particle) to determine a closed form solution, which varies per interval because $\mathbf{q}^n, \mathbf{a}^n, \mathbf{b}^n, \mathbf{c}^n$ vary. Consider one interval $[t^n, t^{n+1}]$ with initial conditions

$$\mathbf{x}^n = \mathbf{x}(t^n) \tag{5}$$

$$\dot{\mathbf{x}}^n = \dot{\mathbf{x}}(t^n) \tag{6}$$

determined from the previous interval; then, the closed form solution in this interval can be written as

$$\mathbf{x}(t^n + s\Delta t) = e^{-\frac{k_d}{2}\Delta t s}\mathbf{g}(t^n + s\Delta t) + \mathbf{p}(t^n + s\Delta t) \tag{7}$$

where $s \in [0,1]$. Here $\mathbf{p}(t^n + s\Delta t)$ is the particular solution associated with the inhomogeneous terms arising from the targets $\hat{\mathbf{x}}(t^n + s\Delta t)$

$$\mathbf{p}(t^n + s\Delta t) = \hat{\mathbf{x}}(t^n + s\Delta t) - \frac{6\mathbf{q}^n s + 2\mathbf{a}^n}{k_s \Delta t^2} + \frac{6k_d\mathbf{q}^n}{k_s^2\Delta t^3} \tag{8}$$

The spring is overdamped when $k_d^2 - 4k_s > 0$, underdamped when $k_d^2 - 4k_s < 0$, and critically damped when $k_d^2 - 4k_s = 0$. Defining a (unitless) ϵ for both the overdamped case, $\epsilon = \frac{\Delta t s}{2}\sqrt{k_d^2 - 4k_s}$, and the underdamped case, $\epsilon = \frac{\Delta t s}{2}\sqrt{4k_s - k_d^2}$, allows us to write

$$\mathbf{g_o}(t^n + s\Delta t) = \boldsymbol{\gamma}_1^n \frac{e^{\epsilon} + e^{-\epsilon}}{2} + \boldsymbol{\gamma}_2^n \Delta t s \frac{e^{\epsilon} - e^{-\epsilon}}{2\epsilon} \tag{9}$$

$$\mathbf{g_u}(t^n + s\Delta t) = \boldsymbol{\gamma}_1^n \cos\epsilon + \boldsymbol{\gamma}_2^n \Delta t s \frac{\sin\epsilon}{\epsilon} \tag{10}$$

$$\mathbf{g_c}(t^n + s\Delta t) = \boldsymbol{\gamma}_1^n + \boldsymbol{\gamma}_2^n \Delta t s \tag{11}$$

where $\mathbf{g_o}$ is the overdamped case, $\mathbf{g_u}$ is the underdamped case, and $\mathbf{g_c}$ is the critically damped case. As $\epsilon \to 0$, we obtain $\frac{e^{\epsilon}+e^{-\epsilon}}{2} \to 1$, $\frac{e^{\epsilon}-e^{-\epsilon}}{2\epsilon} \to 1$, $\cos\epsilon \to 1$, $\frac{\sin\epsilon}{\epsilon} \to 1$; thus, $\mathbf{g_o} \to \mathbf{g_c}$ and $\mathbf{g_u} \to \mathbf{g_c}$. In all cases,

$$\boldsymbol{\gamma}_1^n = \mathbf{x}^n - \mathbf{p}(t^n) \tag{12}$$

$$\boldsymbol{\gamma}_2^n = \dot{\mathbf{x}}^n + \frac{k_d}{2}\boldsymbol{\gamma}_1^n - \dot{\mathbf{p}}(t^n). \tag{13}$$

6 LEARNING THE CONSTITUTIVE PARAMETERS

Given one or more temporal sequences $\{\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \ldots, \boldsymbol{\theta}^N\}$ and corresponding dynamic simulation or motion capture results $\{\mathbf{x}_D^1, \mathbf{x}_D^2, \ldots, \mathbf{x}_D^N\}$, we automatically learn constitutive parameters k_s and k_d for each spring. For each such temporal sequence, we create a loss

function of the form

$$\mathcal{L} = \sum_{n=1}^{N} \|\mathbf{x}(t^n) - \mathbf{x}_D^n\|_2^2 \tag{14}$$

where $\mathbf{x}(t^n)$ is determined as described in Section 5. When there is more than one temporal sequence, the loss function can simply be added together. Notably, the loss can be minimized separately for each particle in a highly parallel and efficient manner. We use gradient descent, where initial guesses are obtained from a few iterations of a genetic algorithm [Holland 1992].

The gradient of $\mathcal{L}$ with respect to the parameters k_d and k_s requires the gradient of $\mathbf{x}(t^n)$ with respect to k_d and k_s, i.e. $\frac{\partial \mathbf{x}}{\partial k_d}$ and $\frac{\partial \mathbf{x}}{\partial k_s}$. From Equation 7, one can readily see that the chain rule takes the form

$$\frac{\partial \mathbf{x}}{\partial k_s} = e^{-\frac{k_d}{2}\Delta ts}\frac{\partial \mathbf{g}}{\partial k_s} + \frac{\partial \mathbf{p}}{\partial k_s} \tag{15}$$

$$\frac{\partial \mathbf{x}}{\partial k_d} = e^{-\frac{k_d}{2}\Delta ts}\frac{\partial \mathbf{g}}{\partial k_d} - \frac{\Delta ts}{2}e^{-\frac{k_d}{2}\Delta ts}\mathbf{g} + \frac{\partial \mathbf{p}}{\partial k_d} \tag{16}$$

where $\frac{\partial \mathbf{g}}{\partial k_s}$, $\frac{\partial \mathbf{g}}{\partial k_d}$, and $\mathbf{g}$ all vary based on ϵ, i.e. based on whether k_s and k_d admit overdamping, underdamping, or critically damping. As we have seen (see Equation 9, 10, 11 and the discussion thereafter), $\mathbf{g}$ is continuous in the 2-dimensional k_s-k_d phase space; however, one needs to carefully implement $\frac{\sin\epsilon}{\epsilon}$ and $\frac{e^{\epsilon}-e^{-\epsilon}}{2\epsilon}$ to replace potentially spurious floating point divisions by the asymptotic result when ϵ is small. One can similarly show that $\frac{\partial \mathbf{g}}{\partial k_s}$ and $\frac{\partial \mathbf{g}}{\partial k_d}$ are continuous, and thus $\frac{\partial \mathbf{x}}{\partial k_s}$ and $\frac{\partial \mathbf{x}}{\partial k_d}$ are continuous.

To see that $\frac{\partial \mathbf{g}}{\partial k_s}$ and $\frac{\partial \mathbf{g}}{\partial k_d}$ are continuous, we expand them via the chain rule

$$\frac{\partial \mathbf{g}}{\partial k_s} = \frac{\partial \mathbf{g}}{\partial \boldsymbol{\gamma}_1^n}\frac{\partial \boldsymbol{\gamma}_1^n}{\partial k_s} + \frac{\partial \mathbf{g}}{\partial \boldsymbol{\gamma}_2^n}\frac{\partial \boldsymbol{\gamma}_2^n}{\partial k_s} + \left(\frac{1}{\epsilon}\frac{\partial \mathbf{g}}{\partial \epsilon}\right)\left(\epsilon\frac{\partial \epsilon}{\partial k_s}\right) \tag{17}$$

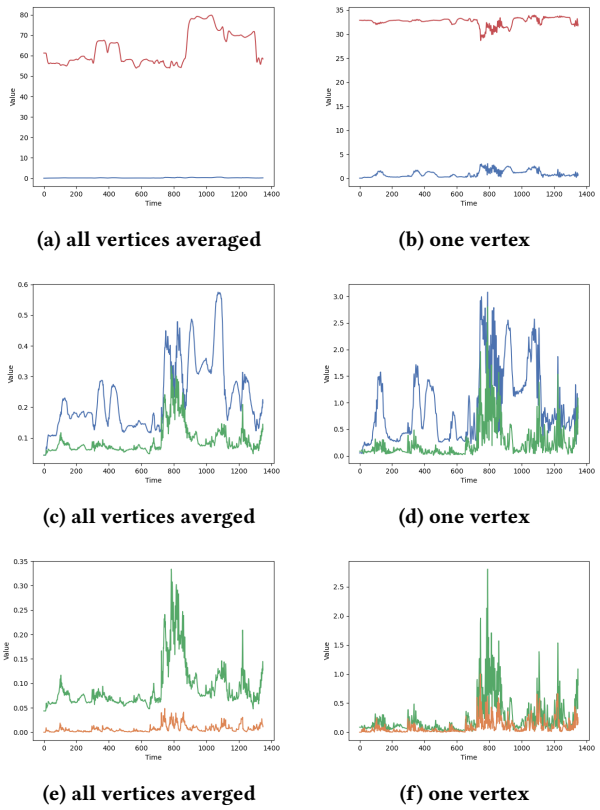
$$\frac{\partial \mathbf{g}}{\partial k_d} = \frac{\partial \mathbf{g}}{\partial \boldsymbol{\gamma}_1^n}\frac{\partial \boldsymbol{\gamma}_1^n}{\partial k_d} + \frac{\partial \mathbf{g}}{\partial \boldsymbol{\gamma}_2^n}\frac{\partial \boldsymbol{\gamma}_2^n}{\partial k_d} + \left(\frac{1}{\epsilon}\frac{\partial \mathbf{g}}{\partial \epsilon}\right)\left(\epsilon\frac{\partial \epsilon}{\partial k_d}\right) \tag{18}$$

and note that $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\gamma}_1^n}$ and $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\gamma}_2^n}$ are continuous for the same reasons that $\mathbf{g}$ is. As can be seen in Equations 12 and 13, $\frac{\partial \boldsymbol{\gamma}_1^n}{\partial k_s}$, $\frac{\partial \boldsymbol{\gamma}_1^n}{\partial k_d}$, $\frac{\partial \boldsymbol{\gamma}_2^n}{\partial k_s}$, and $\frac{\partial \boldsymbol{\gamma}_2^n}{\partial k_d}$ recursively depend on the prior interval via $\mathbf{x}^n$ and $\dot{\mathbf{x}}^n$ (and eventually the initial conditions) but add no new discontinuities of their own. We inserted $\frac{1}{\epsilon}$ and ϵ into the last term in both Equations 17 and 18 so that $\epsilon\frac{\partial \epsilon}{\partial k_s} = \mp\frac{1}{2}\Delta t^2 s^2$ and $\epsilon\frac{\partial \epsilon}{\partial k_d} = \pm\frac{1}{4}\Delta t^2 s^2 k_d$ are robust to compute (the $\mp$ and $\pm$ signs represent overdamping/underdamping respectively). Then, we write

$$\frac{1}{\epsilon}\frac{\partial \mathbf{g_o}}{\partial \epsilon} = \boldsymbol{\gamma}_1^n\frac{e^{\epsilon}-e^{-\epsilon}}{2\epsilon} + \boldsymbol{\gamma}_2^n\Delta ts\frac{(\epsilon-1)e^{\epsilon}+(\epsilon+1)e^{-\epsilon}}{2\epsilon^3} \tag{19}$$

$$\frac{1}{\epsilon}\frac{\partial \mathbf{g_u}}{\partial \epsilon} = -\left(\boldsymbol{\gamma}_1^n\frac{\sin\epsilon}{\epsilon} + \boldsymbol{\gamma}_2^n\Delta ts\frac{\sin\epsilon-\epsilon\cos\epsilon}{\epsilon^3}\right) \tag{20}$$

to identify two more functions that must be carefully implemented (as $\epsilon \to 0$, $\frac{(\epsilon-1)e^{\epsilon}+(\epsilon+1)e^{-\epsilon}}{2\epsilon^3} \to \frac{1}{3}$ and $\frac{\sin\epsilon-\epsilon\cos\epsilon}{\epsilon^3} \to \frac{1}{3}$). The sign difference between Equation 19 and 20 matches that in $\epsilon\frac{\partial \epsilon}{\partial k_s}$ and $\epsilon\frac{\partial \epsilon}{\partial k_d}$ showing that both $\frac{\partial \mathbf{g}}{\partial \epsilon}\frac{\partial \epsilon}{\partial k_s}$ and $\frac{\partial \mathbf{g}}{\partial \epsilon}\frac{\partial \epsilon}{\partial k_d}$ are continuous.



(a) all vertices averaged (b) one vertex

(c) all vertices averged (d) one vertex

(e) all vertices averged (f) one vertex

Figure 3: Red curve: ℓ_2 norm of vertex positions in the pelvis coordinate system. Blue curve: ℓ_2 norm of displacements from skinning to dynamics. Green curve: ℓ_2 norm of displacements from QNN to dynamics. Orange curve: ℓ_2 norm of displacements from QNN to zero-restlength springs.

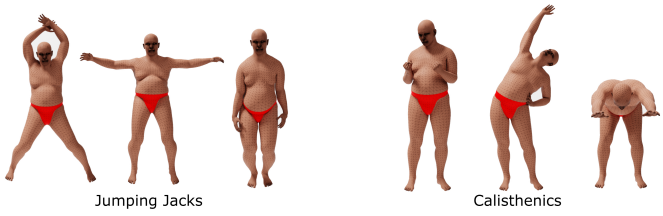


Figure 4: Dynamic simulation sequences used to learn zero-restlength spring constitutive parameters.

Finally, it is worth noting that a 2-dimensional gradient cannot be computed on the codimension-1 curve associated with critically damping; however, taking the dot product of the continuous (between overdamping and underdamping) gradient with the tangent to the codimension-1 curve (and adjusting for either k_s or k_d parameterization) matches the derivative along the curve as expected.

7 RESULTS AND DISCUSSION

Figure 3 quantitatively illustrates how our approach alleviates the demand on the neural network for a particular dynamic simulation example ("calisthenics"). Figure 3a shows the ℓ_2 norm of the vertex positions (red curve) measured relative to a coordinate system whose origin is placed on the pelvis joint. Figure 3b shows the same result for a single vertex on the belly. The ℓ_2 norm of the displacements from the skinned result (blue curve) is vastly smaller (as shown in Figures 3a and 3b), indicating that most of this function is readily captured via skinning (a number of authors have utilized this approach [Jin et al. 2020; Loper et al. 2015; Pons-Moll et al. 2015; Santesteban et al. 2020; Seo et al. 2021]). In Figures 3c and 3d, we change the scale so that the blue curve can be more readily examined. In addition, we also plot the ℓ_2 norm of the displacements from our QNN result (green curve) as the dynamics layer we want to approximate. This dynamics layer has a relatively small magnitude and low variance (comparably), which is readily approximated/learned based on a few dynamic simulations of training data. Figures 3e and 3f show the dynamics layer approximated by our zero-restlength springs (orange curve). Our approach captures the approximated shape, but with smaller magnitude, due to regularization. However, even with regularization, our method still outputs quite compelling dynamics (as can be seen in the supplementary video).

As mentioned in Section 6, we learn our spring constitutive parameters using a (surprisingly) small amount (less than 100 frames) of ground truth simulation data. We obtain the dynamic simulation results $\{\mathbf{x}_D^1, \mathbf{x}_D^2, \ldots, \mathbf{x}_D^N\}$ via backward Euler simulation. Figure 4 shows examples of two dynamic simulation sequences ("jumping jacks" and "calisthenics") we use to learn zero-restlength spring constitutive parameters. Note that any reasonable animation sequence with dynamics can be used, even motion capture data (see e.g. [Pons-Moll et al. 2015]). Although we use a dataset with 5000 data samples in order to train a robust QNN (see Section 3), only a few dynamic simulation examples are required in order to learn zero-restlength spring constitutive parameters that generalize well to unseen animations. This also means that we only need to engineer the network architectures and hyperparameters for the configuration-only QNN, which is much easier than engineering a network that captures configuration transitions (see Section 1 for the discussion about underfitting and overfitting of transition-based methods). Previous methods that add secondary motions to characters [Casas and Otaduy 2018; Pons-Moll et al. 2015; Santesteban et al. 2020; Seo et al. 2021] usually require a large dataset with thousands of data samples to not overfit their network. In comparison, the minimal need of data from our method is a great ease for the data generation process. Our method is also unconditionally stable thanks to its analytic nature, and its optimized constitutive parameters are physically interpretable.

7.1 Examples

Our analytic zero-restlength spring model generalizes very well to unseen animations and does not face severe underfitting or overfitting, which is common in machine learning methods if the network architecture is not carefully designed and trained on a plethora of data. Figure 5 qualitatively shows two example frames comparing

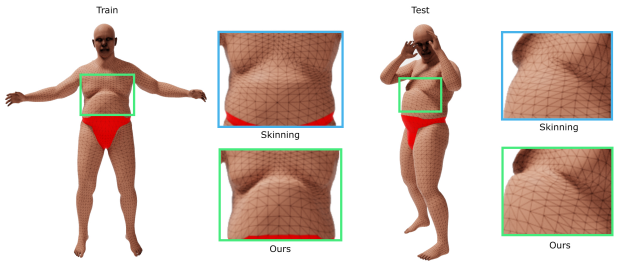
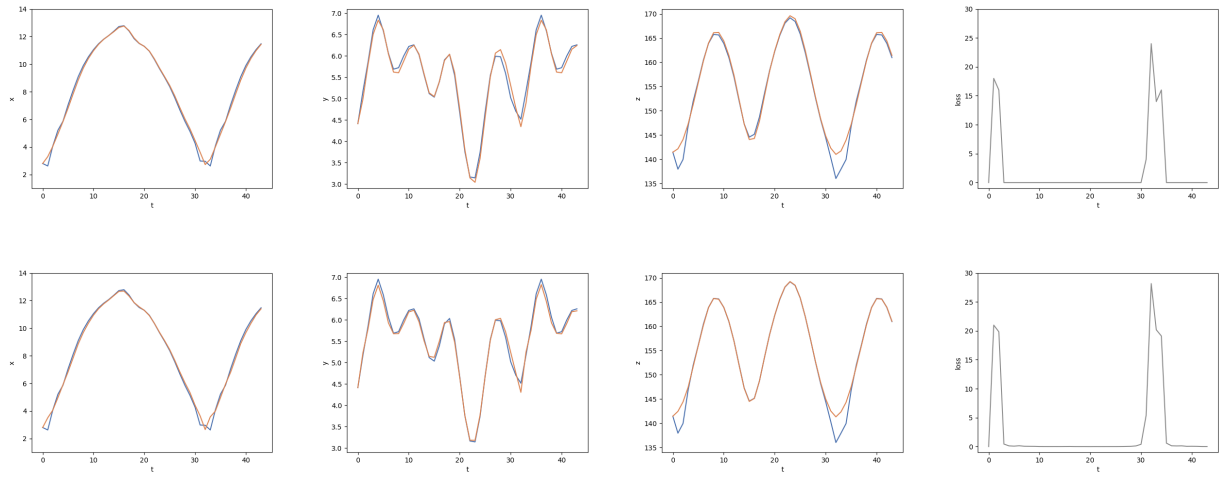


Figure 5: Comparison of our trained zero-restlength spring ballistic motion with the corresponding skinned result. Left: a motion sequence included in training. Right: a motion sequence not included in training. The ability to train on "jumping jacks" and generalize to "shadow boxing" would be impossible for a typical neural network approach.

Figure 6: Secondary dynamics are added to a low-poly ankylosaurus. Notice how the zero-restlength springs (second row) manage to add dynamic motions on top of quasistatic result (first row), especially around the ears, tail, and back region.

a skinning-only result with our analytic zero-restlength springs added on top of our QNN. The frame on the left ("jumping jacks") is taken from an animation sequence used in training while the frame on the right ("shadow boxing") is taken from an animation sequence not used in training. In both examples, our method successfully recovers ballistic motion (e.g. in the belly). Our method runs in real-time (30-90 fps, or even faster pending optimizations) and emulates the effects of accurate, but costly, dynamic backward Euler simulation remarkably well (the dynamic backward Euler simulation we use to generate training examples take about 16 minutes per frame with self-collision enabled). Our approach can be easily applied to different mesh topologies. Figure 6 shows the secondary dynamics added to a low-poly ankylosaurus. We refer readers to our supplementary video for a compelling demonstration, particularly of the secondary inertial motion.

Full dynamic simulation is costly and prone to instabilities. Often this results in a few simulated frames with visible errors. To avoid such artifacts, we modify our training procedure to avoid overfitting to poorly converged frames (that would lead to poor generalization).



(a) X axis (b) Y axis (c) Z axis (d) Loss

Figure 7: Robust training in the presence of simulation errors. Subfigures in columns (a)-(c) are per-axis trajectories of an example vertex in the jumping jack sequence. The backward Euler trajectory is shown in blue and our analytic zero-restlength spring trajectory is shown in orange. The high-frequencies in Frames 31-34 are caused by poorly converged dynamics in the presence of collisions. Subfigures in column (d) show the ℓ_2 loss between the zero-restlength springs and backward Euler. The first row is the initial training result and the second row is the re-trained result with the 10% highest-loss frames ignored. The second row more closely follows the backward Euler trajectory for the frames that don't have simulation errors.

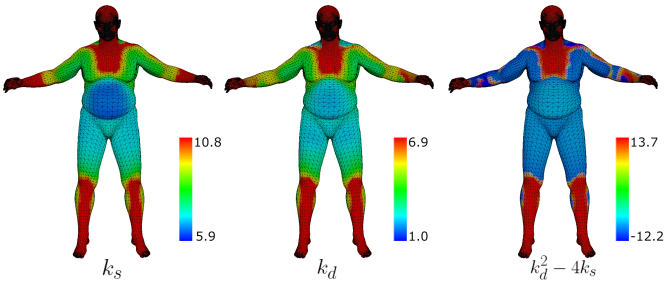


Figure 8: Heatmap visualization (logarithm scale) of stiffness k_s, damping k_d, and $k_d^2 - 4k_s$ which determines overdamping/underdamping, respectively. In heavily constrained regions the springs are stiffer and more overdamped, while in fleshy regions the springs are softer and more underdamped. Note that more constrained regions occur based on proximity to the bones used in the dynamic simulation training data (e.g. chest, forearms, shins, etc.).

See Figure 7. We note that similar approaches are common in the computer vision community (see e.g. random sample consensus [Fischler and Bolles 1981]).

Figure 8 shows a heatmap visualization of learned k_s, k_d and the overdamping/ underdamping indicator $k_d^2 - 4k_s$, respectively. Note how symmetric our optimization result is, even if we optimize each particle separately. In regions where rigid motion dominates (e.g. hands, feet, head, etc.), the optimization results in overdamped springs with large stiffness. The code can be accelerated by replacing the constitutive parameters of all such springs with a single set of consitutive parameters. In regions where soft-tissue dynamics dominates (e.g. belly, thigh, etc.), the optimization results in underdamped springs with small stiffness. Since our optimization is per particle decoupled, it is easy to troubleshoot (if necessary).

Some artifacts of our methods appear when the QNN is not well trained, resulting in physically incorrect quasistatic meshes during inference (interpenetrations, not preserving volume, etc.). This can be constantly improved by better QNN architecture and more extensive experiments on hyperparameter tuning, and is not the main focus of this paper. Collision artifacts might also appear in the dynamic step (although not noticeable in our experiments), since our zero-restlength springs method does not handle collisions for efficiency.

As a final note, one could obviously add our zero-restlength springs on top of the skinned result directly; however, we obtained better results using our QNN to fix skinning artifacts due to volume loss and collision.

8 CONCLUSION AND FUTURE WORK

We present an analytically integratable physics model that can recover dynamic modes in real-time. The main takeaway is that the problem can be separated into a configuration-only quasistatic layer and a transition-dependent dynamics layer, where the dynamics layer can be well approximated by a simple physics model. The constitutive parameters of the physics model can be robustly learned from only a few backward Euler simulation examples. In

particular, determining k_s and k_d requires a gradient that can erroneously overflow/underflow near the critical damping manifold in k_s-k_d phase space. We quite robustly addressed this by isolating non-dimensionalized functions that were trivially carefully implemented to obtain the correct asymptotic result in *all* cases. For more discussions on both numerical and analytical issues with gradients, we refer the interested readers to [Johnson and Fedkiw 2022; Metz et al. 2021].

ACKNOWLEDGMENTS

Research supported in part by ONR N00014-13-1-0346, ONR N00014-17-1-2174, DOE 1655264, NSF-NRT 1829071 and Epic Games. We would like to thank Epic Games (especially including Michael Lentine, Kim Libreri and Tim Sweeney) for supporting our research combining machine learning with physics simulations, aiming towards democratization of such technologies for the metaverse.

REFERENCES

Brian F Allen, Michael Neff, and Petros Faloutsos. 2011. Analytic proportional-derivative control for precise and compliant motion. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 6039–6044.

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

Stephen W Bailey, Dave Otte, Paul Dilorenzo, and James F O'Brien. 2018. Fast and deep deformation approximations. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Hugo Bertiche, Meysam Madadi, and Sergio Escalera. 2020. PBNS: Physically Based Neural Simulator for Unsupervised Garment Pose Space Deformation. *arXiv preprint arXiv:2012.11310* (2020).

Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. 2002. Interactive skeleton-driven dynamic deformations. *ACM transactions on graphics (TOG)* 21, 3 (2002), 586–593.

Dan Casas and Miguel A Otaduy. 2018. Learning nonlinear soft-tissue dynamics for interactive avatars. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 1 (2018), 1–15.

Lan Chen, Juntao Ye, Liguo Jiang, Chengcheng Ma, Zhanglin Cheng, and Xiaopeng Zhang. 2018. Synthesizing cloth wrinkles by CNN-based geometry image superresolution. *Computer Animation and Virtual Worlds* 29, 3-4 (2018), e1810.

Nuttapong Chentanez, Miles Macklin, Matthias Müller, Stefan Jeschke, and Tae-Yong Kim. 2020. Cloth and skin deformation with a triangle mesh based convolutional neural network. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 123–134.

Matthew Cong, Michael Bao, Jane L E, Kiran S Bhat, and Ronald Fedkiw. 2015. Fully automatic generation of anatomical face simulation models. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 175–183.

Suvranu De, Dhannanjay Deo, Ganesh Sankaranarayanan, and Venkata S Arikatla. 2011. A physics-driven neural networks-based simulation system (phynness) for multimodal interactive virtual environments involving nonlinear deformable objects. *Presence* 20, 4 (2011), 289–308.

Edilson De Aguiar, Leonid Sigal, Adrien Treuille, and Jessica K Hodgins. 2010. Stable spaces for real-time clothing. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 1–9.

Alessandro De Luca, Bruno Siciliano, and Loredana Zollo. 2005. PD control with on-line gravity compensation for robots with elastic joints: Theory and experiments. *automatica* 41, 10 (2005), 1809–1819.

Congyue Deng, Tai-Jiang Mu, and Shi-Min Hu. 2020. Alternating convlstm: Learning force propagation with alternate state updates. *arXiv preprint arXiv:2006.07818* (2020).

Epic Games. 2021. *MetaHuman Creator*. https://www.unrealengine.com/en-US/digital-humans

Wei-Wen Feng, Yizhou Yu, and Byung-Uck Kim. 2010. A deformation transformer for real-time cloth animation. *ACM transactions on graphics (TOG)* 29, 4 (2010), 1–9.

Martin A Fischler and Robert C Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (1981), 381–395.

Lawson Fulton, Vismay Modi, David Duvenaud, David IW Levin, and Alec Jacobson. 2019. Latent-space Dynamics for Reduced Deformable Simulation. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 379–391.

Zhenglin Geng, Daniel Johnson, and Ronald Fedkiw. 2020. Coercing machine learning to output physically accurate results. *J. Comput. Phys.* 406 (2020), 109099.

Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. 1998. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 9–20.

Peng Guan, Loretta Reiss, David A Hirshberg, Alexander Weiss, and Michael J Black. 2012. Drape: Dressing any person. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–10.

Fabian Hahn, Bernhard Thomaszewski, Stelian Coros, Robert W Sumner, Forrester Cole, Mark Meyer, Tony DeRose, and Markus Gross. 2014. Subspace clothing simulation using adaptive bases. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–9.

Jessica K Hodgins, Wayne L Wooten, David C Brogan, and James F O'Brien. 1995. Animating human athletics. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 71–78.

Daniel Holden, Bang Chi Duong, Sayantan Datta, and Derek Nowrouzezahrai. 2019. Subspace neural physics: Fast data-driven interactive simulation. In *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 1–12.

John H Holland. 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* (2018).

Ning Jin, Yilin Zhu, Zhenglin Geng, and Ronald Fedkiw. 2020. A Pixel-Based Framework for Data-Driven Clothing. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 135–144.

Daniel Johnson and Ronald Fedkiw. 2022. Smoothing Discontinuous Root-Finding for Subsequent Differentiability in Learning, Inverse Problems, and Control. *preprint* (2022).

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2007. Skinning with dual quaternions. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*. 39–46.

Ladislav Kavan, Dan Gerszewski, Adam W Bargteil, and Peter-Pike Sloan. 2011. Physics-inspired upsampling for cloth simulation in games. In *ACM SIGGRAPH 2011 papers*. 1–10.

Meekyoung Kim, Gerard Pons-Moll, Sergi Pujades, Seungbae Bang, Jinwook Kim, Michael J Black, and Sung-Hee Lee. 2017. Data-driven physics for human soft tissue animation. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

L'ubor Ladickỳ, SoHyeon Jeong, Barbara Solenthaler, Marc Pollefeys, and Markus Gross. 2015. Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–9.

Zorah Lahner, Daniel Cremers, and Tony Tung. 2018. Deepwrinkles: Accurate and realistic clothing modeling. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 667–684.

John P Lewis, Matt Cordner, and Nickson Fong. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 165–172.

Haolin Liu, Ye Han, Daniel Emerson, Houriyeh Majditehran, Qi Wang, Yoed Rabin, and Levent Burak Kara. 2020. Real-time Prediction of Soft Tissue Deformations Using Data-driven Nonlinear Presurgical Simulations. *arXiv preprint arXiv:2010.13823* (2020).

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2015. SMPL: A skinned multi-person linear model. *ACM transactions on graphics (TOG)* 34, 6 (2015), 1–16.

Ran Luo, Tianjia Shao, Huamin Wang, Weiwei Xu, Xiang Chen, Kun Zhou, and Yin Yang. 2018. NNWarp: Neural network-based nonlinear deformation. *IEEE transactions on visualization and computer graphics* 26, 4 (2018), 1745–1759.

Nadia Magnenat-Thalmann, Richard Laperrire, and Daniel Thalmann. 1988. Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface'88*. Citeseer.

Alan Marquez, Yizhou Chen, Yushan Han, Steven Gagniere, Michael Tupek, and Joseph Teran. 2022. A Momentum Conserving Hybrid Particle/Grid Iteration for Volumetric Elastic Contact. *preprint* (2022).

Felix Meister, Tiziano Passerini, Viorel Mihalef, Ahmet Tuysuzoglu, Andreas Maier, and Tommaso Mansi. 2020. Deep learning acceleration of total lagrangian explicit dynamics for soft tissue mechanics. *Computer Methods in Applied Mechanics and Engineering* 358 (2020), 112628.

Andrea Mendizabal, Pablo Márquez-Neila, and Stéphane Cotin. 2020. Simulation of hyperelastic materials in real-time using deep learning. *Medical image analysis* 59 (2020), 101569.

Luke Metz, C Daniel Freeman, Samuel S Schoenholz, and Tal Kachman. 2021. Gradients are Not All You Need. *arXiv preprint arXiv:2111.05803* (2021).

Chaitanya Patel, Zhouyingcheng Liao, and Gerard Pons-Moll. 2020. Tailornet: Predicting clothing in 3d as a function of human pose, shape and garment style. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7365–7375.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. 2020. Learning Mesh-Based Simulation with Graph Networks. In *International Conference on Learning Representations*.

Micha Pfeiffer, Carina Riediger, Jürgen Weitz, and Stefanie Speidel. 2019. Learning soft tissue behavior of organs for surgical navigation with convolutional neural networks. *International journal of computer assisted radiology and surgery* 14, 7 (2019), 1147–1155.

Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J Black. 2015. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–14.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* 378 (2019), 686–707.

Francois Roewer-Despres, Najeeb Khan, and Ian Stavness. 2018. Towards finite element simulation using deep learning. In *15th international symposium on computer methods in biomechanics and biomedical engineering*.

Yasmin Salehi and Dennis Giannacopoulos. 2021. PhysGNN: A Physics-Driven Graph Neural Network Based Model for Predicting Soft Tissue Deformation in Image-Guided Neurosurgery. *arXiv preprint arXiv:2109.04352* (2021).

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. 2020. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*. PMLR, 8459–8468.

Igor Santesteban, Elena Garces, Miguel A Otaduy, and Dan Casas. 2020. SoftSMPL: Data-driven Modeling of Nonlinear Soft-tissue Dynamics for Parametric Humans. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 65–75.

Hyewon Seo, Kaifeng Zou, and Frederic Cordier. 2021. DSNet: Dynamic Skin Deformation Prediction by Recurrent Neural Network. In *Computer Graphics International Conference*. Springer, 365–377.

Seung Heon Sheen, Egor Larionov, and Dinesh K Pai. 2021. Volume Preserving Simulation of Soft Tissue with Skin. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 4, 3 (2021), 1–23.

Jonathan Richard Shewchuk. 1998. Tetrahedral Mesh Generation by Delaunay Refinement. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*. 86–95.

Hang Si. 2015. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)* 41, 2 (2015), 1–36.

Sangeetha Grama Srinivasan, Qisi Wang, Junior Rojas, Gergely Klár, Ladislav Kavan, and Eftychios Sifakis. 2021. Learning active quasistatic physics-based models from data. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14.

Qingyang Tan, Lin Gao, Yu-Kun Lai, Jie Yang, and Shihong Xia. 2018. Mesh-based autoencoders for localized deformation component analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

Qingyang Tan, Zherong Pan, Lin Gao, and Dinesh Manocha. 2020. Realtime simulation of thin-shell deformable materials using CNN-based mesh embedding. *IEEE Robotics and Automation Letters* 5, 2 (2020), 2325–2332.

Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. 2005. Robust Quasistatic Finite Elements and Flesh Simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 181–190.

Bohan Wang, Mianlun Zheng, and Jernej Barbič. 2020. Adjustable Constrained Soft-Tissue Dynamics. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 69–79.

Huamin Wang. 2021. GPU-based simulation of cloth wrinkles at submillimeter levels. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14.

Huamin Wang, Florian Hecht, Ravi Ramamoorthi, and James F O'Brien. 2010. Example-based wrinkle synthesis for clothing animation. In *ACM SIGGRAPH 2010 papers*. 1–8.

Jack M Wang, Samuel R Hamner, Scott L Delp, and Vladlen Koltun. 2012. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–11.

Rachel Weinstein, Eran Guendelman, and Ronald Fedkiw. 2007. Impulse-based control of joints and muscles. *IEEE transactions on visualization and computer graphics* 14, 1 (2007), 37–46.

Steffen Wiewel, Moritz Becher, and Nils Thuerey. 2019. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 71–82.

Jane Wu, Zhenglin Geng, Hui Zhou, and Ronald Fedkiw. 2020. Skinning a parameterization of three-dimensional space for neural network cloth. *arXiv preprint arXiv:2006.04874* (2020).

Jane Wu, Yongxu Jin, Zhenglin Geng, Hui Zhou, and Ronald Fedkiw. 2021b. Recovering geometric information with learned texture perturbations. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 4, 3 (2021), 1–18.

Nannan Wu, Qianwen Chao, Yanzhen Chen, Weiwei Xu, Chen Liu, Dinesh Manocha, Wenxin Sun, Yi Han, Xinran Yao, and Xiaogang Jin. 2021a. AgentDress: Realtime Clothing Synthesis for Virtual Agents using Plausible Deformations. *IEEE Transactions on Visualization and Computer Graphics* 27, 11 (2021), 4107–4118.

Hongyi Xu and Jernej Barbič. 2016. Pose-space subspace dynamics. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–14.

Weiwei Xu, Nobuyuki Umetani, Qianwen Chao, Jie Mao, Xiaogang Jin, and Xin Tong. 2014. Sensitivity-optimized rigging for example-based real-time clothing synthesis. *ACM Trans. Graph.* 33, 4 (2014), 107–1.

Jiayi Eris Zhang, Seungbae Bang, David I.W. Levin, and Alec Jacobson. 2020. Complementary Dynamics. *ACM Transactions on Graphics* (2020).

Meng Zhang, Tuanfeng Y Wang, Duygu Ceylan, and Niloy J Mitra. 2021. Dynamic neural garments. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–15.

Mianlun Zheng, Yi Zhou, Duygu Ceylan, and Jernej Barbic. 2021. A Deep Emulator for Secondary Motion of 3D Characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5932–5940.

Javier S Zurdo, Juan P Brito, and Miguel A Otaduy. 2012. Animating wrinkles by example on non-skinned cloth. *IEEE Transactions on Visualization and Computer Graphics* 19, 1 (2012), 149–158.