COMBINING NEURAL NETWORKS AND PHYSICS-BASED SIMULATION FOR CLOTH AND FLESH DYNAMICS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Yongxu Jin
July 2024

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Ron Fedkiw, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Kayvon Fatahalian

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Karen Liu

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format.

Abstract

Physics-based simulation techniques have long been established for various offline applications, yet real-time dynamic simulation remains a formidable challenge. Despite advancements in modern game engines like Unreal Engine 5, achieving high-resolution, real-time simulation capabilities remains limited. Recently, researchers have shown interest in using neural networks to approximate dynamic simulation, thanks to their fast inference on GPUs. However, although effective at approximating kinematics or quasistatic simulation, purely data-driven approaches often struggle with dynamic simulations (involving velocity and momentum information) due to potential overfitting and poor generalization with time series data. This limitation makes them unsuitable for real-world applications. Other efforts have tried using neural networks to upsample real-time, low-resolution simulations, but achieving good low-resolution results with conventional methods is very challenging.

This thesis aims to pioneer a paradigm for real-time, high-fidelity physics simulation, with a focus on practical implementation within current game engines. Motivated by recent advancements in neural networks for capturing quasistatic simulations (referred to as quasistatic neural networks, or QNNs), we propose to rethink the need for dynamic components given QNN-based enhancements and redesign real-time physics models to primarily capture the ballistic motion of full dynamics, which can then be enhanced by QNNs to obtain the full shape. These meticulously designed physics models ensure stability, robustness, and performance that surpass real-time requirements. Concurrently, the lightweight QNNs can capture quasistatic shapes, facilitating ease of training and robust generalization. This thesis comprises two primary papers: one addressing human flesh simulation and the other focusing on the simulation of loose-fitting clothing.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my PhD advisor, Ron Fedkiw, for his unwavering support and invaluable guidance throughout my PhD journey. Your insights and vision have been instrumental in shaping my research path, transforming me from a novice into a confident researcher. Thank you for always supporting my career, connecting me with various professionals, and providing excellent career advice. I am especially grateful for your flexibility and understanding during difficult times, offering not only mentorship but also exceptional mental support.

I am also deeply grateful to my committee members. My sincere thanks to Mykel Kochenderfer for serving as my committee chair and to Karen Liu, Kayvon Fatahalian, and Jiajun Wu for their valuable insights and support in completing this dissertation. A special thank you to Karen Liu for welcoming me as a first-year rotation student and introducing me to the exciting field of robotics and reinforcement learning. I also extend my gratitude to Silvio Savarese and Fei-Fei Li for allowing me to rotate in the Stanford Vision and Learning Lab during my first year.

I would like to extend my heartfelt thanks to all my research collaborators. Thank you, Joseph Teran and Yushan Han, for collaborating with me remotely throughout most of my PhD and for your numerous contributions and insights into physics simulation. I am grateful to Roberto Martín-Martín, Fei Xia, William Shen, and Yifeng Jiang for their guidance and support during my first-year rotations in the Stanford Vision and Learning Lab and The Movement Lab. Special thanks to Bo Zhu and Xubo Yang, my research advisors during my undergraduate studies, for their incredible support and mentorship, which helped launch my research career.

Beyond Stanford, I would like to thank my industry collaborators. My heartfelt thanks to Michael Lentine for offering me an internship opportunity at Epic Games throughout my PhD and for providing valuable industry insights. Collaborating with Epic Games has

profoundly influenced my understanding of impactful and practical research. Thank you, Kenji Tashiro at Sony, and Hui Zhou at JD, for your interest in machine learning for clothing and for collaborating with Ron's lab. I also appreciate Tuur Stuyck and Petr Kadlecek for accepting me as a summer intern twice at Meta Reality Labs, allowing me to explore various aspects of physics simulation in VR. Thank you, Jenny Jin and Olivier Soares, for the opportunity to intern at Apple and delve into the frontier of spatial computing with Vision Pro. It was an incredibly enriching experience, and I am grateful for the return offer, which allows me to continue exploring this exciting field.

A very special thank you goes to all my labmates for their support, camaraderie, and the enjoyable moments we shared. Thank you, Zhenglin Geng, Winnie Lin, Dan Johnson, Jane Wu, Kevin Li, Yilin Zhu, Dalton Omens, Trevor Maxfield, Sarah Jobalia, Demi Guo, Haodi He, Zhengfei Kuang, and Yitong Deng, for being such an amazing and supportive group. Special thanks to Jane Wu for her mentorship during my first year, not only helping me start my research career in Stanford but also helping me adjust to life in the States, and to Zhenglin Geng for his invaluable help throughout my PhD, especially with the machine learning cloth project. Special thanks to Dalton Omens, Dan Johnson, and Trevor Maxfield for being incredible collaborators. I am deeply grateful to all of you for making my PhD journey enjoyable and rewarding.

Lastly, I would like to thank all my friends and family. To my parents, thank you for always being my unwavering support system. Your constant encouragement and belief in me have been my greatest strength. I hope to make you proud.

Contents

Abstract **iv**

Acknowledgments **v**

1 Introduction **1**

1.1 Thesis Overview 3

2 Flesh Simulation **5**

2.1 Introduction 5

2.2 Related Work 7

2.3 Quasistatic Neural Network 8

2.3.1 Quasistatic Simulation 9

2.3.2 QNN 10

2.4 Kinematics 11

2.5 Dynamics 11

2.6 Learning the constitutive parameters 13

2.7 Results and Discussion 14

2.7.1 Examples 17

2.8 Conclusion and Future Work 20

3 Cloth Simulation **22**

3.1 Introduction 22

3.2 Related Work 25

3.3 Rope Chain Simulation 27

3.4 Velocity Update 30

3.5 Position Update 34

3.6 Collisions 36
3.7 Neural Skinning 39
3.8 Neural Shape Inference 42
3.9 Results and Discussion 43
3.10 Conclusion and Future Work 47

4 Conclusion and Future Work **52**
4.1 Conclusion 52
4.2 Future Work 53
4.2.1 Collisions 53
4.2.2 Differentiable Simulation 53
4.2.3 Network Architecture 53
4.2.4 Generalization 54

Bibliography **55**

List of Figures

2.1 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. 6

2.2 Our QNN resolves well-known skinning collision artifacts. We demonstrate this in extreme poses involving the back of the knee and the armpit. 10

2.3 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. 15

2.4 Dynamic simulation sequences used to learn zero-restlength spring constitutive parameters. 16

2.5 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. 17

2.6 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. 18

2.7 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.). 18

2.8 Robust training in the presence of simulation errors. Subfigures in rows (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 row (d) show the ℓ_2 loss between the zero-restlength springs and backward Euler. The first column is the initial training result and the second column is the re-trained result with the 10% highest-loss frames ignored. The second column more closely follows the backward Euler trajectory for the frames that don't have simulation errors. 19

3.1 We introduce a rope chain simulation approach to efficiently model cloth dynamics using a small number of degrees of freedom. Analytic signed distance functions are used to efficiently manage collisions with the body mesh. Neural networks are utilized to skin a mesh from the simulated degrees of freedom and to capture the detailed mesh shapes. Our results (the second and fifth images) not only produce dynamics similar to full numerical simulations (the third and sixth images) but also do not suffer from the locking and/or overstretching typical of real-time physics-based simulations. 23

3.2 The loose-fitting garments that we use for numerical experiments. The cape mesh consists of 12690 vertices, and we define 10 rope chains with 13-15 virtual bones in each chain (reducing the total DOFs by a factor of approximately 90). The skirt mesh consists of 18546 vertices, and we define 26 rope chains with 9 virtual bones in each chain (reducing the total DOFs by a factor of approximately 80). Importantly, the large reduction in the number of DOFs is highly beneficial for both RAM and cache performance, not just CPU performance. Note that the unattached virtual bones near the top of both the cape and the skirt are not simulated, but they will be used for skinning (see Section 3.7). 25

3.3 The far left subfigure shows a a coarsely discretized mesh with position constraints on the five red nodes. The next three subfigures show the results of steady state simulations using a decreasing spring stiffness (from left to right). The final subfigure shows a rope chain simulation (the rope chains are shaded green) of the same degrees of freedom. The mass-spring simulations lock with stiffer springs and overstrech with weaker springs. The spring stiffness in the middle subfigure was chosen to approximately match the downward stretching extent of the rope chain simulation, which is what one would expect without overstretching; however, locking occurs since the springs are still too stiff. 28

3.4 In this figure, we simulate two virtual bones connected by a single rope with the top virtual bone fixed and the bottom virtual bone free to rock back and forth as a pendulum. The results compare well to the analytic solution for pendulum motion for both shorter times (left) and longer times (right), illustrating the efficacy of our numerical approach. 29

3.5 Motion of a single swinging rope chain, showcasing varying numbers of Gauss-Seidel iterations. Left to right: 1, 5, 10 iterations, and iterating until the relative error is smaller than a tolerance of 10^{-6}. With more iterations, the rope chain is less damped (as expected). The same number of iterations (or the same tolerance) was used for both the tension and impulse computations. Note that it might be more efficient (depending on the application) to use a different number of iterations on the tension and impulse computations. . . 32

3.6 For the sake of computational efficiency, we represent the volumetric body (left) with a number of analytically defined SDFs (right). Although other representations (three dimensional grid discretizations, neural representations, etc.) are compatible with our approach, they incur higher computational costs. 36
3.7 Time evolution of a sphere colliding with six particles (6 frames are depicted). The red particles use the $\nabla\phi$ direction for both push out and velocity projection (as is typical), the purple particles use $\hat{r}$ for push out and $\nabla\phi$ for velocity projection, and the green particles use $\hat{r}$ for both push out and velocity projection. Note how replacing $\nabla\phi$ with $\hat{r}$ prevents the particles from quickly working their way around the sphere. In fact, the green particles maintain a persistent contact with the sphere until it stops moving shortly before the last frame (even though friction is not used in this example). The behavior of the green particles is highly preferable to that of the red (and purple) particles when considering collisions between clothing and the human body. 38
3.8 Top row: first five principle components of the neural skinning PCA model for the cape. Bottom row: first five principle components of the neural shape PCA model for the cape. All of the images depict the augmentation of the rest shape cloth mesh by the PCA displacements, even though the displacements are added to the skinned mesh (not the rest state mesh) during neural shape inference. The PCA model used for skinning tends to capture low-frequency deformations, while the PCA model used for shape inference is better suited for capturing higher-frequency deformations. 42
3.9 The first row shows the results of the neural skinning network, which can be compared to the ground truth training data in the second row. In order to demonstrate that the network does have the ability to match the ground truth data, the third row shows the overfit results obtained via overtraining; of course, an overfit network will not generalize well to unseen data. 44
3.10 The first row shows the results of the neural skinning network (identical to the first row in Figure 3.9). The second row shows the results of the neural shape network applied on top of the neural skinning result. The third row shows the ground truth training data. The fourth row shows how overtraining the neural shape network leads to results that well match (albeit overfit, similar to the last row in Figure 3.9) the ground truth training data. 45

3.11 The first row shows the results of the neural skinning network, and the second row shows the results of the neural shape network applied on top of the neural skinning result. These holdout frames from the training set give some indication of how the network will perform on unseen data. Note that the network needs to generalize to out-of-distribution data (from rope chain simulations, as is discussed in the last paragraph in Section 3.7), not just to holdout frames from the training set. 46
3.12 Comparing the result of a neural skinning network trained without and with the PINN-style collision loss. 47
3.13 We trained a separate RNN for each rope chain (10 RNNs in total). The RNN results (first row) are quite noisy compared to the ground truth training data (second row). The thrid row shows the overfit results obtained via overtraining. The fourth row shows the result of applying our neural skinning and neural shape inference to the RNN-inferred virtual bone positions (from the first row). Comparing these results to the ground truth training data (fifth row) illustrates that our networks generalize well to this noisy RNN input. 48
3.14 The main issue with the RNN is that it generalizes very poorly to the holdout data. Presumably, this issue could be fixed by collecting more and more training data, but that excessively increases the cost incurred in both data collection (via numerical simulation) and training (via optimization). Even though our skinning and shape networks can smooth the noise in this RNN output (as they did in Figure 3.13), the dynamics will still be completely wrong. 49

3.15 Simulation of a cape on an animation of a running person. First row: rope chain simulation. The rope chains are shaded green, and the weak lateral springs are visualized as purple edges. Second row: neural skinning, based on the first row. Third row: neural shape inference, based on the second row. Fourth row: Houdini cloth simulation with approximately 12K vertices (as a reference). The Houdini simulation and our rope chain approach both produce good dynamics, although the Houdini simulation does exhibit visually displeasing erroneous over-stretching artifacts; in addition, the Houdini simulation is an order of magnitude slower than our currently unoptimized code. 50

3.16 Simulation of a skirt on an animation of a dancing person. First row: rope chain simulation. The rope chains are shaded green, and the weak lateral springs are visualized as purple edges. Second row: neural skinning, based on the first row. Third row: neural shape inference, based on the second row. Fourth row: Houdini cloth simulation with approximately 18K vertices (as a reference). The skirt is rendered with a blue color in order to differentiate it from the yellow skirt figures, which use barycentrically embedded virtual bone positions instead of rope chain simulations as the network input. . . . 51

Chapter 1

Introduction

Physics-based simulation has made remarkable progress in recent decades, becoming a cornerstone technology in the graphics industry. Researchers have developed numerous algorithms to simulate various materials, such as fluids [27, 30, 76], muscles [119, 120, 110], cloth [5, 10, 11], and hair [6, 102, 81]. These simulations have found widespread applications, particularly in offline scenarios like visual effects in Hollywood movies [113, 32]. However, in real-time applications such as video games or AR/VR, existing physics simulation techniques face challenges in producing high-resolution, high-fidelity results. This is primarily due to the necessity of using low-resolution meshes and large time steps to maintain real-time performance. Even in advanced game engines like Unreal Engine 5 [26], achieving high-fidelity simulation in real-time remains a significant technical challenge, despite impressive advancements in real-time rendering driven by NVIDIA's real-time ray-tracing technology [79].

To address these challenges, there has been growing interest in both academia and industry in using neural networks to approximate high-fidelity physics simulation [43, 85, 97, 98, 82, 20, 100]. This interest is driven by the real-time inference capabilities of neural networks on modern GPUs. Research in this area can be broadly classified into two categories: approximating quasi-static simulation and approximating dynamic simulation.

The first category involves using neural networks to approximate quasi-static simulation. In this scenario, the neural network aims to approximate the result of a steady-state simulation at each frame without considering temporal dynamics. This means the network generates mesh deformation directly from the current spatial state without needing historical information as input. This approach is particularly useful for animating tightly-fitting

clothing, such as t-shirts and pants [47, 100, 84], as well as for modeling muscle deformations like bulging or skin wrinkling [4, 3].

The second category focuses on using neural networks to approximate dynamic simulation, which is crucial for generating animations with ballistic or secondary motion, such as the swinging of loose-fitting clothing like capes or skirts [20, 82, 53], or the jiggling of flesh and soft tissues on the human body [98, 88]. To approximate dynamic simulation, the network needs to understand how velocity and momentum work, as it must approximate the entire trajectory of a second-order ordinary differential equation (ODE) or partial differential equation (PDE). Thus, the input to the network must include a temporal history of spatial states.

Using neural networks to approximate quasi-static simulation has recently seen significant success and has already been implemented in some real-time industry applications [62, 25]. In this thesis, I will refer to all neural networks used to estimate quasi-static simulations as quasistatic neural networks (QNNs). QNNs are successful because they do not need to capture temporal dynamics, making them easy to train and capable of generalizing well. In fact, many works use simple fully connected networks as the QNN architecture.

On the other hand, using neural networks to infer dynamic simulation remains a challenging problem. Although there have been many recent papers on this topic, they are often not robust or practical enough for industry applications. Recent works usually employ two strategies to approximate dynamic simulation. The first strategy involves directly using a time series model, particularly recurrent neural networks (RNNs), to model dynamic simulation. RNNs can memorize information from previous frames and learn to produce secondary ballistic motion. A popular option is the gated recurrent unit (GRU) [16], used in several recent papers [82, 98, 142, 23]. More recently, researchers are exploring Transformer architectures [123] for this purpose [105]. However, training RNNs is challenging due to their need for temporal information, often leading to overfitting or underfitting and resulting in poor generalization. Consequently, RNNs struggle to extrapolate to out-of-distribution data and cannot produce reasonable, physically accurate results over long sequences.

The second strategy involves performing a low-resolution simulation initially and then using a neural network to upsample it to a high-resolution simulation [53]. This approach is appealing because the neural network can be a QNN, as secondary dynamics are captured in the low degrees of freedom. This makes the network easier to train and generalize. However, achieving a good low-resolution simulation result with conventional methods is challenging.

Low-resolution simulations often suffer from artifacts such as locking or overstretching, which depend on the stiffness of the chosen material. Consequently, the dynamics of low-resolution simulations are often inaccurate.

1.1 Thesis Overview

The goal of my thesis is to achieve high-fidelity dynamic simulation in real-time, with an algorithm practical enough for potential industry applications. Over the years, we have gained extensive knowledge about performing cost-effective, low-resolution simulations. Additionally, we have learned that neural networks are much better at capturing quasistatic simulations than dynamic simulations.

Thus, given the success of neural networks for quasistatics, we have rethought and redesigned real-time physics models for simulating dynamics. On one hand, we still utilize existing neural network architectures solely to capture quasistatic information. The QNN architecture is designed to be very lightweight, making it easy to train and highly efficient during inference. On the other hand, we use physics simulation to capture the secondary dynamics that are missing from the quasistatic network results. Since QNNs have recently proven to be highly successful and far better than previous non-neural network methods at capturing quasistatics, we have re-evaluated the requirements for dynamic components. Specifically, unlike traditional methods, our physics models primarily handle the ballistic motion of full dynamics, followed by QNN-based enhancement to recover the full shape. This allows us to design our physics models to be simple, robust, stable, and free from the artifacts typically seen in mass-spring simulations, which is crucial for real-time applications. Both our quasistatic and dynamic components can be easily implemented in applications for video games and AR/VR.

Our focus is on dynamic simulation related to digital humans, given its significant potential in many real-time graphics applications. Therefore, this thesis primarily contains two main papers: one on human flesh simulation [48], and another on loose-fitting clothing simulation [49]. Both of these projects build upon recent advancements in both the physics simulation and machine learning communities, which have provided the tools and frameworks necessary for our hybrid approach. Specifically, we leverage QNNs to capture the quasistatic shape of the mesh. Given the success of QNNs, we directly use well-tested network architectures from previous works [47]. For dynamic simulation, instead of relying

on simple low-resolution mass-spring simulations, we develop two novel physics models: one for flesh and one for cloth.

For human flesh (see Chapter 2), we find that a set of zero-rest-length springs is surprisingly sufficient to model the jiggling motion of the flesh. For loose-fitting garments (see Chapter 3), we model the cloth as a set of low degrees of freedom, one-dimensional rope chains, and define an inequality constraint on each rope, inspired by a previous work on low-resolution cloth simulation [46]. In particular, we designed the rope chain physics model to simulate loose-fitting clothing that displays large-scale ballistic motion (such as skirts or capes), rather than loose-fitting clothing that merely deviates from the body and exhibits minimal ballistic motion (such as an oversized shirt). Both physics models can be time-integrated in real-time, and we carefully derive the mathematics to ensure that both methods are not only fast and stable but also practical enough to be implemented in current game engines.

Chapter 2

Flesh Simulation

2.1 Introduction

Recently, there has been a lot of interest in using neural networks to approximate dynamic simulation (see e.g. [43, 85, 97, 98]), 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.

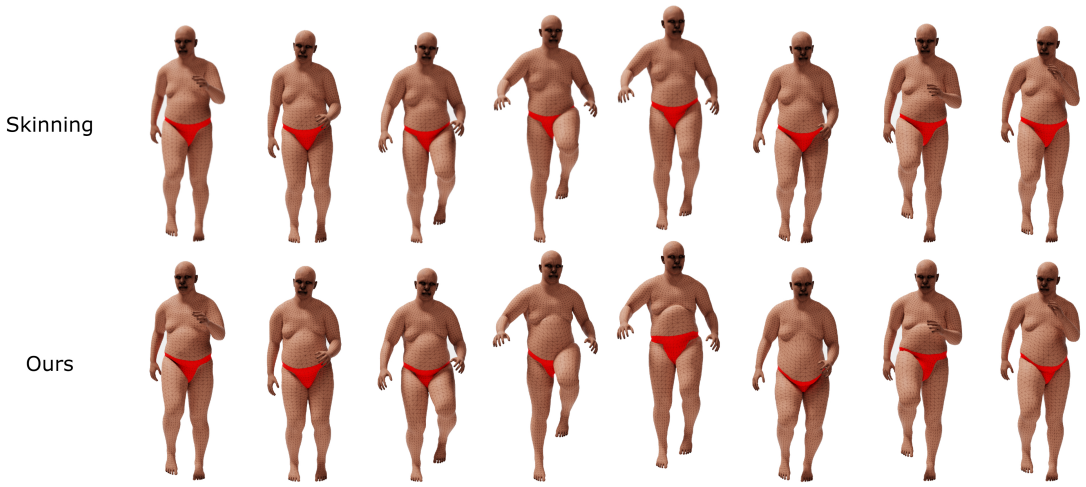


Figure 2.1: Our method enhances standard skinning with a configuration-only quasistatic neural network (QNN) that approximates quasistatic hyperelasticity as well as analytically integratable zero-restlength springs trained that approximates inertial effects. The QNN fixes well-known skinning artifacts (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.

[47, 70, 33, 7]) 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 2.7), one can (automatically) robustly learn spring constitutive parameters from a very small amount of dynamic simulation data.

2.2 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: [28] trains a regressor to upsample, [53] learns an upsampling operator, and [14] rasterizes the vertex positions into an image before upsampling it and interpolating new vertex positions. [126, 144, 139] use example-based methods to synthesize fine-scale wrinkles from a database. [84] predicts a low-frequency mesh with a fully connected network and uses a mixture model to add wrinkles. [15] upsamples with graph convolutional neural networks. [133] recovers high-frequency geometric details with perturbations of texture. [59] uses a generative adversarial network (GAN) to upsample a cloth normal map for improved rendering. [4, 3] use neural networks to drive fine scale details from a coarse character rig. Many works aim to learn equilibrium configurations from boundary conditions: [70] uses a neural network to add non-linearity to a linear elasticity model. [74] learns the non-linear mapping from contact forces to displacements. Such approaches are particularly common in virtual surgery applications, e.g. [68, 19, 96, 93, 87]. [47] trains a CNN to infer a displacement map which adds wrinkles to skinned cloth, and [132] improves the accuracy of this approach by embedding the cloth into a volumetric tetrahedral mesh. [7] adds physics to the loss function, a common approach in physics-inspired neural networks (PINNs), see e.g. [90]. To avoid the soft constraints of PINNs that only coerce physically-inspired behaviour, [33, 111] 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) [36] uses a neural network to learn temporal transitions and leverage back propagation to optimize control parameters. [20] incorporates an approximation to the quasistatic equilibrium that serves as a control for a dynamics layer. [37] predicts a cloth mesh from body poses and previous frames, solving a linear system to fix penetrations. [40] uses dynamic subspace simulation on an adaptive selected basis generated from the current body pose. [43] computes a linear subspace of configurations with principal component analysis (PCA) and learns subspace simulations from previous frames with a fully connected network. [31, 115, 114] obtain nonlinear subspaces with autoencoder

networks. Similar methods are commonly used to animate fluids using regression forests [58] or recurrent neural networks (RNNs) [131]. [85] and [97] use graph networks to learn simulations with both fixed and changing topology. [15] proposes a transition-based model with position and linear/angular velocity of the body as network input (in addition to a state-based model). [73] uses a fully connected network to predict node-wise acceleration for total Lagrangian explicit dynamics. [22] proposes a convolutional long short-term memory (LSTM) layer to capture elastic force propagation. [141] 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. [124, 140, 106, 138, 12]). 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: [88] provides a motion capture dataset with dynamic surface meshes, and proposes a linear auto-regressive model to capture dynamic displacements compressed by PCA. [69] extends this method to the SMPL human model. See also [13, 98, 104]. [55] 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. [143] trains a network to approximate per-vertex displacements from temporal one-ring state using backward Euler simulation data of primitive shapes. [22] 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 [1, 130, 127, 42, 21].

2.3 Quasistatic Neural Network

We use the (freely available) MetaHuman [24] 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 [71, 63] and dual quaternion skinning [51]) 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. [135]), 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.

2.3.1 Quasistatic Simulation

First, we use Tetgen [109] (alternatively, [45],[108] 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 [18]. 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 [121] in order to avoid issues with indefiniteness and the method from [91] 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.

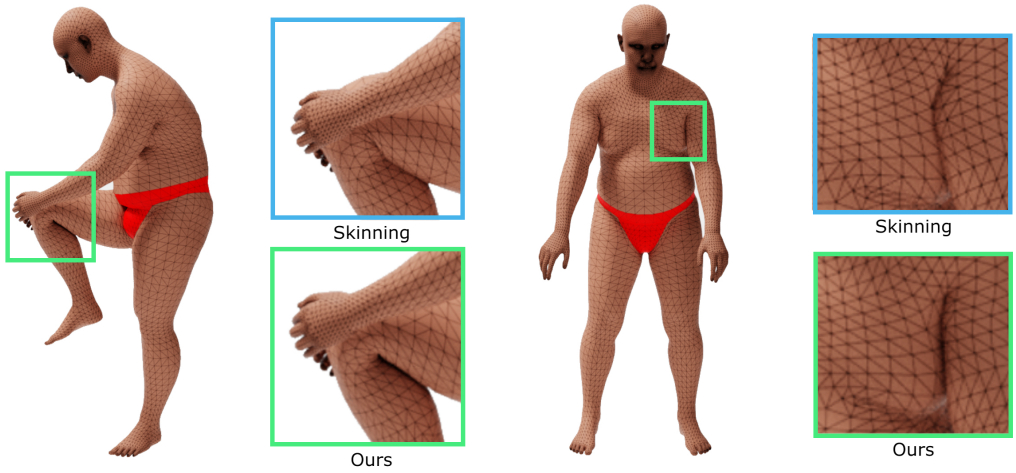


Figure 2.2: Our QNN resolves well-known skinning collision artifacts. We demonstrate this in extreme poses involving the back of the knee and the armpit.

2.3.2 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 [47] rasterizing the per-vertex displacements 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. [125], 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 2.2), thus providing a simpler dynamics layer for analytic zero-restlength springs to capture (see Section 2.7 for discussions). Since the QNN is not the main contribution of this paper, we refer readers to the original paper [47] for technical details (network architectures, optimizers, hyperparameters, etc.). The QNN used in this paper can also be easily replaced with other state-based models.

2.4 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

$$\hat{\mathbf{x}}(t^n + s\Delta t) = \hat{\mathbf{q}}^n (s\Delta t)^3 + \hat{\mathbf{a}}^n (s\Delta t)^2 + \hat{\mathbf{b}}^n s\Delta t + \hat{\mathbf{c}}^n \tag{2.1}$$

$$= \mathbf{q}^n s^3 + \mathbf{a}^n s^2 + \mathbf{b}^n s + \mathbf{c}^n, \tag{2.2}$$

where $s \in [0, 1]$ and Equation 2.2 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

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{q}^n \\ \mathbf{a}^n \\ \mathbf{b}^n \\ \mathbf{c}^n \end{bmatrix} = \begin{bmatrix} \mathbf{x}^{net}(\boldsymbol{\theta}^n) \\ \frac{1}{2}(\mathbf{x}^{net}(\boldsymbol{\theta}^{n+1}) - \mathbf{x}^{net}(\boldsymbol{\theta}^{n-1})) \\ \mathbf{x}^{net}(\boldsymbol{\theta}^{n+1}) \\ \frac{1}{2}(\mathbf{x}^{net}(\boldsymbol{\theta}^{n+2}) - \mathbf{x}^{net}(\boldsymbol{\theta}^{n})) \end{bmatrix}, \tag{2.3}$$

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 $\mathbf{x}^{net}(\boldsymbol{\theta}^{n+1}) - \mathbf{x}^{net}(\boldsymbol{\theta}^{n})$.

2.5 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,

$$\ddot{\mathbf{x}}(t) = k_s(\hat{\mathbf{x}}(t) - \mathbf{x}(t)) + k_d(\dot{\hat{\mathbf{x}}}(t) - \dot{\mathbf{x}}(t)), \tag{2.4}$$

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

$$\mathbf{x}^n = \mathbf{x}(t^n) \tag{2.5}$$

$$\dot{\mathbf{x}}^n = \dot{\mathbf{x}}(t^n) \tag{2.6}$$

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

$$\mathbf{x}(t^n + s\Delta t) = e^{-\frac{k_d}{2}\Delta t s}\mathbf{g}(t^n + s\Delta t) + \mathbf{p}(t^n + s\Delta t) \tag{2.7}$$

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 $\hat{\mathbf{x}}(t^n + s\Delta t)$

$$\mathbf{p}(t^n + s\Delta t) = \hat{\mathbf{x}}(t^n + s\Delta t) - \frac{6\mathbf{q}^n s + 2\mathbf{a}^n}{k_s \Delta t^2} + \frac{6k_d\mathbf{q}^n}{k_s^2\Delta t^3} \tag{2.8}$$

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

$$\mathbf{g_o}(t^n + s\Delta t) = \boldsymbol{\gamma}_\mathbf{1}^n \frac{e^\epsilon + e^{-\epsilon}}{2} + \boldsymbol{\gamma}_\mathbf{2}^n \Delta t s \frac{e^\epsilon - e^{-\epsilon}}{2\epsilon} \tag{2.9}$$

$$\mathbf{g_u}(t^n + s\Delta t) = \boldsymbol{\gamma}_\mathbf{1}^n \cos\epsilon + \boldsymbol{\gamma}_\mathbf{2}^n \Delta t s \frac{\sin\epsilon}{\epsilon} \tag{2.10}$$

$$\mathbf{g_c}(t^n + s\Delta t) = \boldsymbol{\gamma}_\mathbf{1}^n + \boldsymbol{\gamma}_\mathbf{2}^n \Delta t s \tag{2.11}$$

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,

$$\boldsymbol{\gamma}_\mathbf{1}^n = \mathbf{x}^n - \mathbf{p}(t^n) \tag{2.12}$$

$$\boldsymbol{\gamma}_\mathbf{2}^n = \dot{\mathbf{x}}^n + \frac{k_d}{2}\boldsymbol{\gamma}_\mathbf{1}^n - \dot{\mathbf{p}}(t^n). \tag{2.13}$$

2.6 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

$$\mathcal{L} = \sum_{n=1}^{N} \|\mathbf{x}(t^n) - \mathbf{x}_D^n\|_2^2 \tag{2.14}$$

where $\mathbf{x}(t^n)$ is determined as described in Section 2.5. 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 [44].

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 2.7, one can readily see that the chain rule takes the form

$$\frac{\partial \mathbf{x}}{\partial k_s} = e^{-\frac{k_d}{2}\Delta t s}\frac{\partial \mathbf{g}}{\partial k_s} + \frac{\partial \mathbf{p}}{\partial k_s} \tag{2.15}$$

$$\frac{\partial \mathbf{x}}{\partial k_d} = e^{-\frac{k_d}{2}\Delta t s}\frac{\partial \mathbf{g}}{\partial k_d} - \frac{\Delta t s}{2}e^{-\frac{k_d}{2}\Delta t s}\mathbf{g} + \frac{\partial \mathbf{p}}{\partial k_d} \tag{2.16}$$

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 2.9, 2.10, 2.11 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

$$\frac{\partial \mathbf{g}}{\partial k_s} = \frac{\partial \mathbf{g}}{\partial \boldsymbol{\gamma}_\mathbf{1}^n}\frac{\partial \boldsymbol{\gamma}_\mathbf{1}^n}{\partial k_s} + \frac{\partial \mathbf{g}}{\partial \boldsymbol{\gamma}_\mathbf{2}^n}\frac{\partial \boldsymbol{\gamma}_\mathbf{2}^n}{\partial k_s} + \left(\frac{1}{\epsilon}\frac{\partial \mathbf{g}}{\partial \epsilon}\right)\left(\epsilon\frac{\partial \epsilon}{\partial k_s}\right) \tag{2.17}$$

$$\frac{\partial \mathbf{g}}{\partial k_d} = \frac{\partial \mathbf{g}}{\partial \boldsymbol{\gamma}_\mathbf{1}^n}\frac{\partial \boldsymbol{\gamma}_\mathbf{1}^n}{\partial k_d} + \frac{\partial \mathbf{g}}{\partial \boldsymbol{\gamma}_\mathbf{2}^n}\frac{\partial \boldsymbol{\gamma}_\mathbf{2}^n}{\partial k_d} + \left(\frac{1}{\epsilon}\frac{\partial \mathbf{g}}{\partial \epsilon}\right)\left(\epsilon\frac{\partial \epsilon}{\partial k_d}\right) \tag{2.18}$$

and note that $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\gamma}_\mathbf{1}^n}$ and $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\gamma}_\mathbf{2}^n}$ are continuous for the same reasons that $\mathbf{g}$ is. As can be seen

in Equations 2.12 and 2.13, $\frac{\partial \boldsymbol{\gamma}_1^n}{\partial k_s}$, $\frac{\partial \boldsymbol{\gamma}_1^n}{\partial k_d}$, $\frac{\partial \boldsymbol{\gamma}_2^n}{\partial k_s}$, 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 2.17 and 2.18 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

$$\frac{1}{\epsilon}\frac{\partial \mathbf{g_o}}{\partial \epsilon} = \boldsymbol{\gamma}_1^n \frac{e^\epsilon - e^{-\epsilon}}{2\epsilon} + \boldsymbol{\gamma}_2^n \Delta t s \frac{(\epsilon-1)e^\epsilon + (\epsilon+1)e^{-\epsilon}}{2\epsilon^3} \tag{2.19}$$

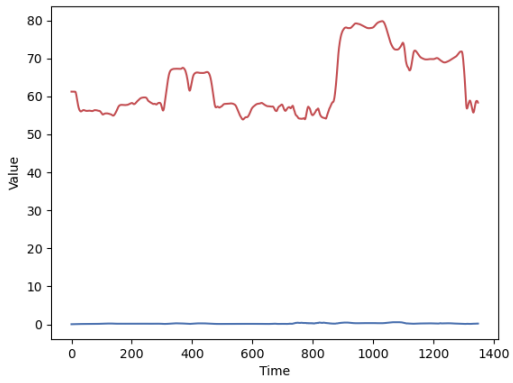
$$\frac{1}{\epsilon}\frac{\partial \mathbf{g_u}}{\partial \epsilon} = -\left(\boldsymbol{\gamma}_1^n \frac{\sin\epsilon}{\epsilon} + \boldsymbol{\gamma}_2^n \Delta t s \frac{\sin\epsilon - \epsilon\cos\epsilon}{\epsilon^3}\right) \tag{2.20}$$

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 2.19 and 2.20 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.

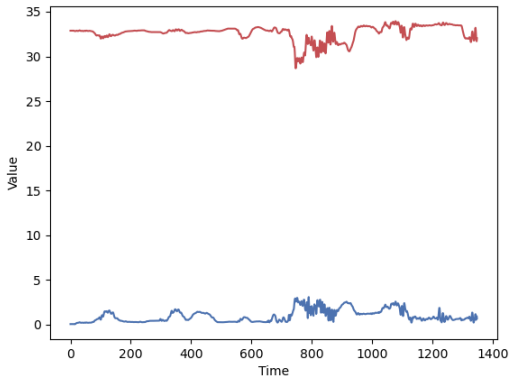
Finally, it is worth noting that a 2-dimensional gradient cannot be computed on the codimension-1 curve associated with critically damping; however, taking the dot product of the continuous (between overdamping and underdamping) gradient with the tangent to the codimension-1 curve (and adjusting for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

2.7 Results and Discussion

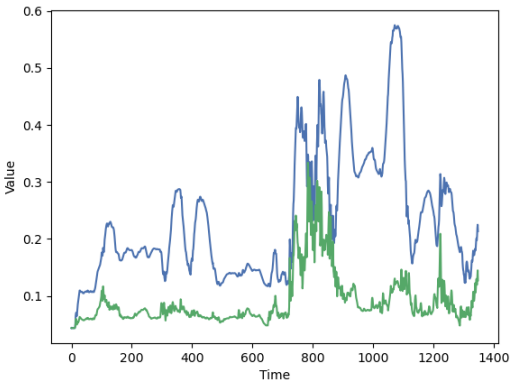
Figure 2.3 quantitatively illustrates how our approach alleviates the demand on the neural network for a particular dynamic simulation example ("calisthenics"). Figure 2.3a shows the ℓ_2 norm of the vertex positions (red curve) measured relative to a coordinate system whose origin is placed on the pelvis joint. Figure 2.3b 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 2.3a and 2.3b), indicating that most of this function is readily captured via skinning (a number of authors have utilized this approach [88, 98, 104, 47, 69]). In Figures 2.3c and 2.3d, 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 2.3e and 2.3f show the dynamics layer approximated by our zero-restlength springs (orange



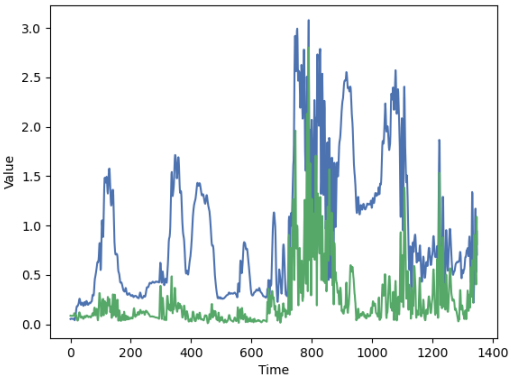
(a) all vertices averaged



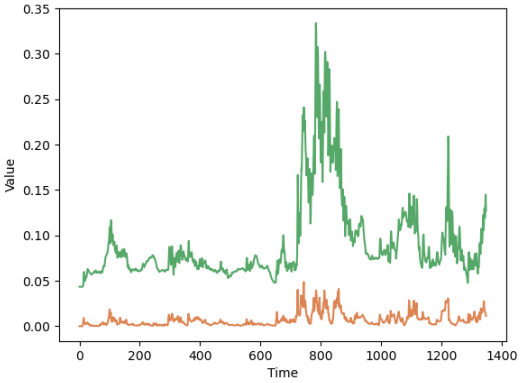
(b) one vertex



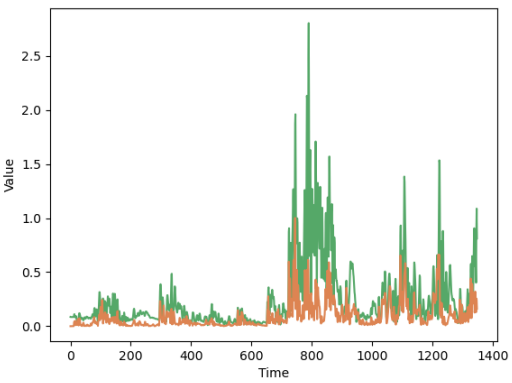
(c) all vertices averaged



(d) one vertex



(e) all vertices averaged



(f) one vertex

Figure 2.3: 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.

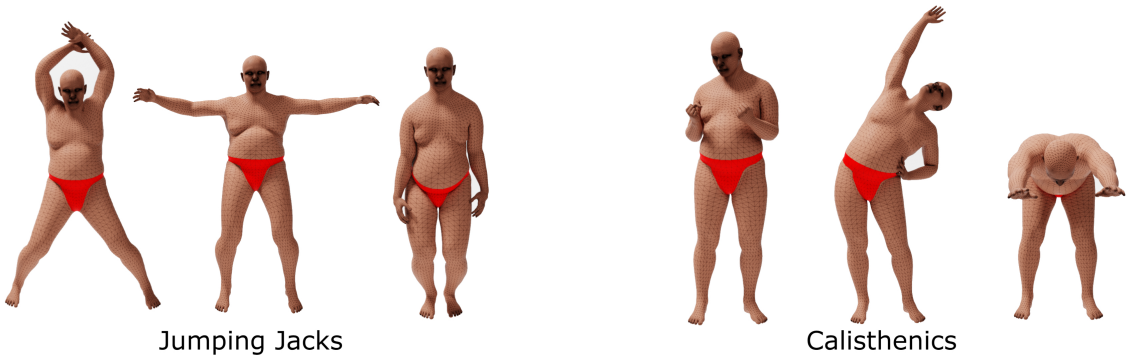


Figure 2.4: Dynamic simulation sequences used to learn zero-restlength spring constitutive parameters.

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 2.6, 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 2.4 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. [88]). Although we use a dataset with 5000 data samples in order to train a robust QNN (see Section 2.3), 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 2.1 for the discussion about underfitting and overfitting of transition-based methods). Previous methods that add secondary motions to characters [88, 13, 98, 104] 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.

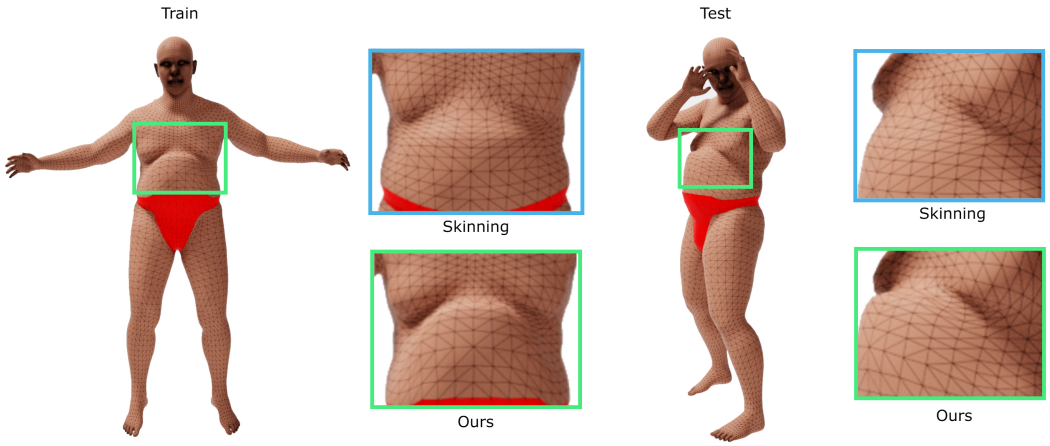


Figure 2.5: Comparison of our trained zero-restlength spring ballistic motion with the corresponding skinned result. Left: a motion sequence included in training. Right: a motion sequence not included in training. The ability to train on "jumping jacks" and generalize to "shadow boxing" would be impossible for a typical neural network approach.

2.7.1 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 2.5 qualitatively shows two example frames comparing 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 2.6 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.

Figure 2.7 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

Figure 2.6: 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.

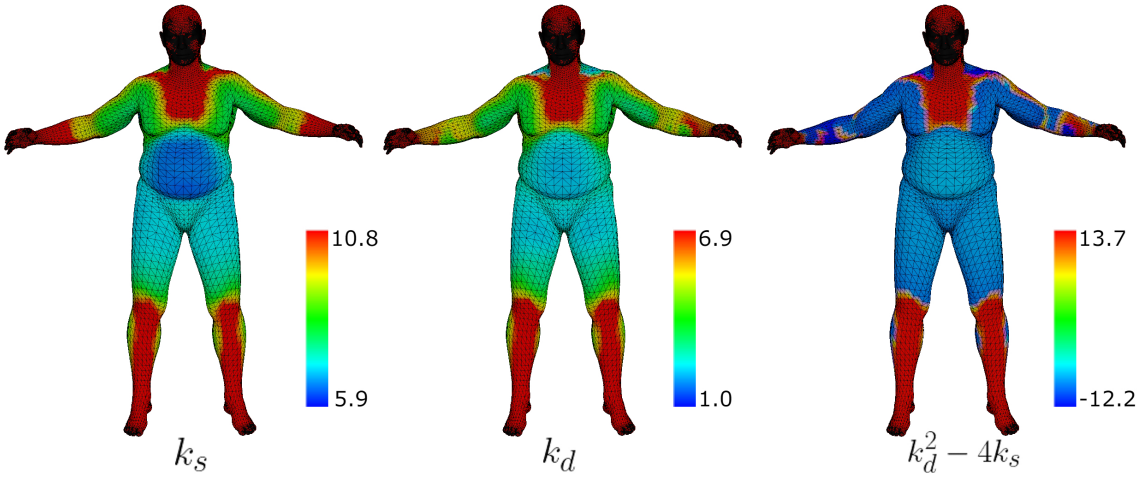


Figure 2.7: Heatmap visualization (logarithm scale) of stiffness k_s, damping k_d, and $k_d^2 - 4k_s$ which determines overdamping/underdamping, respectively. In heavily constrained regions the springs are stiffer and more overdamped, while in fleshy regions the springs are softer and more underdamped. 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.).

(a) X axis

(b) Y axis

(c) Z axis

(d) Loss

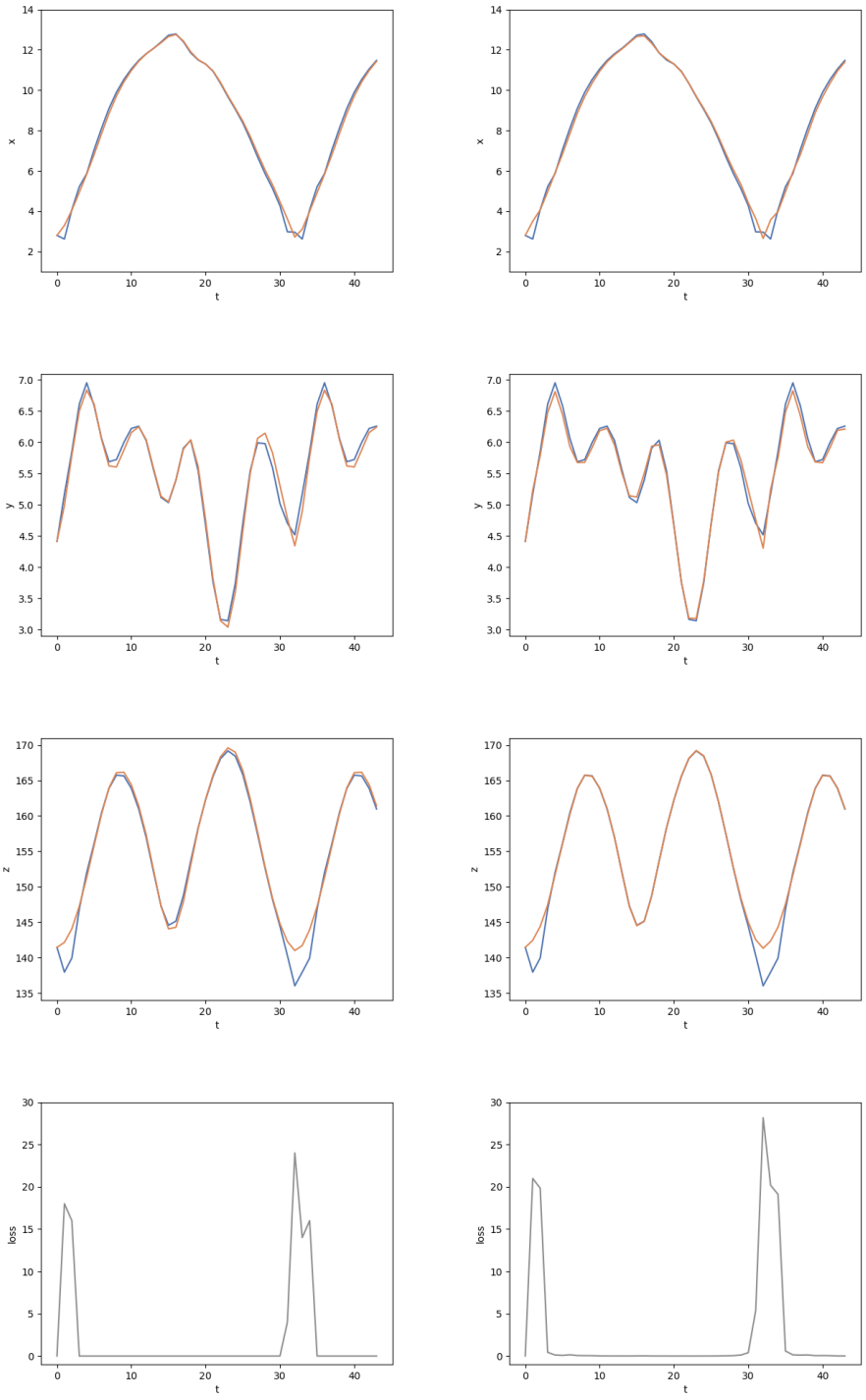


Figure 2.8: Robust training in the presence of simulation errors. Subfigures in rows (a)-(c) are per-axis trajectories of an example vertex in the jumping jack sequence. The backward Euler trajectory is shown in blue and our analytic zero-restlength spring trajectory is shown in orange. The high-frequencies in Frames 31-34 are caused by poorly converged dynamics in the presence of collisions. Subfigures in row (d) show the ℓ_2 loss between the zero-restlength springs and backward Euler. The first column is the initial training result and the second column is the re-trained result with the 10% highest-loss frames ignored. The second column more closely follows the backward Euler trajectory for the frames that don't have simulation errors.

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).

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). See Figure 2.8. We note that similar approaches are common in the computer vision community (see e.g. random sample consensus [29]).

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.

2.8 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 [50, 75].

Chapter 3

Cloth Simulation

3.1 Introduction

Animation of digital clothing has been a captivating research area for decades due to its importance in creating realistic digital humans. While traditional physics-based simulation methods [5, 10, 17] can generate high-fidelity results, the inability to achieve real-time performance at high resolutions restricts their utility in contemporary real-time applications such as video games, VR/AR, and virtual try-on systems. In light of recent advancements in GPU hardware, there has been a surge of interest in leveraging neural networks to approximate physics-based simulations, see e.g. [90, 99, 84, 97, 62, 86]

Tight or close fitting garments such as shirts, pants, etc. typically exhibit only subtle dynamic behaviors; thus, capturing quasistatic shape information is more important than modelling the ballistic motion, see e.g. [59, 47, 84, 7]. Quasistatic shape information can be captured by various skinning techniques, by neural networks that lack temporal information (we will refer to these as quasistatic neural networks or QNNs), or by a combination of skinning and quasistatic neural network approaches. While some low-frequency vibration of tight-fitting clothing can be captured by underlying flesh motion (see e.g. [48]), modelling the pronounced ballistic motion associated with loose-fitting clothing such as skirts, dresses, capes, etc. is more challenging. Skinning and QNN based approaches have not been able to successfully model pronounced ballistic motion; thus, researchers have explored alternative network architectures that incorporate temporal history, most notably recurrent neural networks (RNNs), see e.g. [100, 141, 82]. Unfortunately, the amount of training data required to robustly model temporal transitions between states (especially when considering

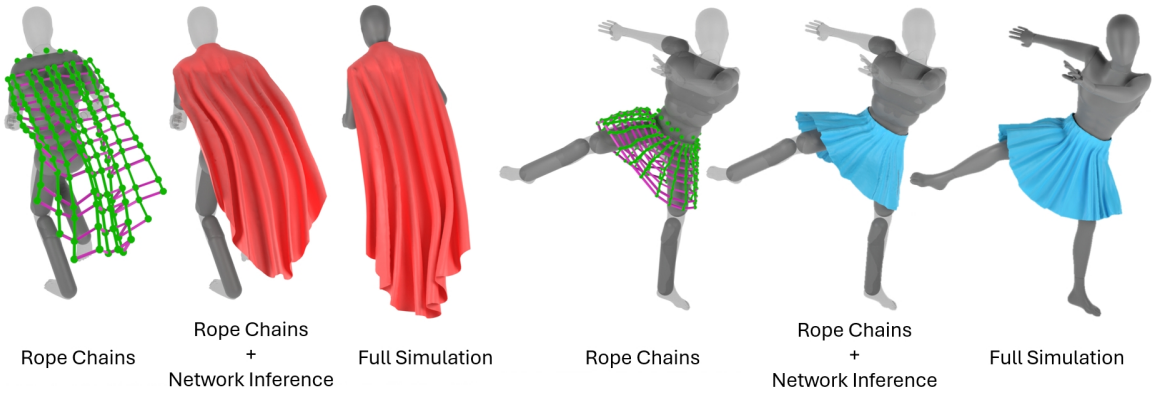


Figure 3.1: We introduce a rope chain simulation approach to efficiently model cloth dynamics using a small number of degrees of freedom. Analytic signed distance functions are used to efficiently manage collisions with the body mesh. Neural networks are utilized to skin a mesh from the simulated degrees of freedom and to capture the detailed mesh shapes. Our results (the second and fifth images) not only produce dynamics similar to full numerical simulations (the third and sixth images) but also do not suffer from the locking and/or overstretching typical of real-time physics-based simulations.

the need for generalization) is significantly greater than that required to model the states themselves. Moreover, increased network capacity is required in order to properly capture the numerous state transitions present in this increased volume of training data. These issues typically cause recurrent neural network approaches to overfit and thus generalize poorly.

Since a physics-based simulation readily models ballistic motions but does not efficiently scale to a large number of degrees of freedom and neural networks readily deal with a large number of degrees of freedom but struggle with temporal state transitions, we pursue a more optimal hybrid approach that uses a small number of degrees of freedom physics simulation to capture dynamics and a neural network to capture a high resolution shape. In particular, we physically simulate such a low number of degrees of freedom that they are best viewed as virtual bones as opposed to being viewed as a subset of a higher resolution mesh (similar to [82, 142]); thus, we rely on skinning (driven by a neural network) in order to construct a coarse approximation to the desired mesh from the simulated degrees of freedom. Given this coarsely approximated dynamic mesh, a quasistatic neural network is then used to obtain a more desirable higher resolution mesh.

Since we rely on the neural network's ability to capture the shape of the cloth, the physics

simulation does not require all the usual (and computationally expensive) techniques for simulating stretching, bending, compression, etc.; thus, we devise a novel approach that connects our virtual bones (or rigid frames) into vertical rope chains. Each rope is designed to be inextensible yet shrinkable, avoiding undesirable locking or rubbery buckling artifacts (see [46] for detailed insights). Optional spring forces can be included in order to softly constrain the distances between different rope chains and/or to regulate the shrinking of each rope (if desired). Compared to low resolution cloth simulation, the rope chains can be robustly simulated with large time steps thus enabling real-time performance. Instead of simulating the rotational (in additional to translational) degrees of freedom typically required in order to skin a coarse mesh from virtual bones (or rigid frames), we utilize a neural network so that the garment can be skinned directly from the simulated translational degrees of freedom.

Collisions between the body and the rope chain degrees of freedom are facilitated via signed distance functions (SDFs), see e.g. [11]. In order to avoid computationally expensive grid-based SDFs, the SDF can be defined either by a set of closed-form primitives (desirable for real-time applications like video games) or by a neural network (see e.g. [83, 92]). Similar collision treatment can also be applied to the degrees of freedom of the full cloth mesh. Both the skinning neural network and the QNN are trained with an additional PINN-style [90] collision loss (using the SDF) in order to obtain network parameters that favor collision-free cloth mesh degrees of freedom; importantly, adding collisions in this fashion does not require modifications to the network architecture nor does it add any computational expense to inference.

To summarize our contributions:

- We propose a hybrid framework for animating loose-fitting clothing that blends together the efficacy of physics simulation for capturing ballistic motion and the efficiency of neural networks for skinning and shape inference.

- In particular, we propose a novel simulation method for low resolution (and loose-fitting) clothing via ballistic rope chains, which are used to reconsturct a full cloth mesh with the aid of neural networks for both skinning and shape inference.

- We propose a novel collision handling method with analytic SDFs, using history-based information in order to improve both robustness and efficiency for the sake of real-time applications.

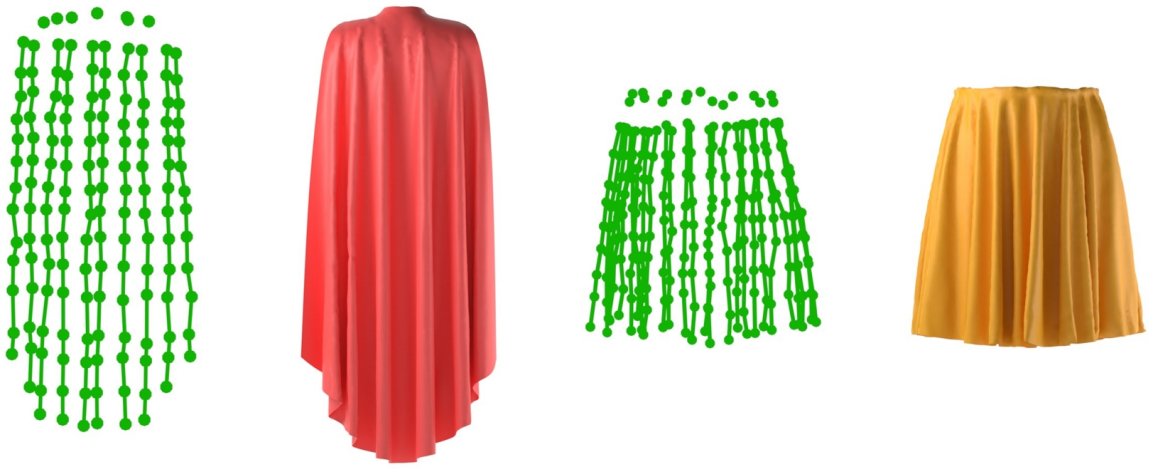


Figure 3.2: The loose-fitting garments that we use for numerical experiments. The cape mesh consists of 12690 vertices, and we define 10 rope chains with 13-15 virtual bones in each chain (reducing the total DOFs by a factor of approximately 90). The skirt mesh consists of 18546 vertices, and we define 26 rope chains with 9 virtual bones in each chain (reducing the total DOFs by a factor of approximately 80). Importantly, the large reduction in the number of DOFs is highly beneficial for both RAM and cache performance, not just CPU performance. Note that the unattached virtual bones near the top of both the cape and the skirt are not simulated, but they will be used for skinning (see Section 3.7).

3.2 Related Work

Physics Simulation: The physical simulation of cloth has a long hisory in computer graphics, dating back to [122]; however, it gained significant popularity due to the implicit time integration approach proposed in [5]. In order to overcome the overly-damped (underwater) appearance of implicit time integration, [10, 11] introduced a semi-implicit time integration approach (central differencing in computational mechanics) that is explicit on the elastic vibrational modes and implicit on the damping modes. Although offline methods could simulate very high resolution cloth even 15 years ago (see [103]), position based dynamics (starting with [77]) has been the method of choice for most real-time applications. Other interesting contributions include discussions on the buckling instability [17], overcoming locking [46], etc.

Neural Physics: Many researchers have aimed to mimic physics simulations via neural networks and various other data-driven techniques. Early works include: [20, 40] used a PCA subspace, [37] predicted a cloth mesh from pose history and body shape, [54] used

motion graphs, and [88, 69] used linear auto-regression. As deep learning gained popularity, a number of authors embraced neural networks. [70] used a neural network to add non-linear displacements on top of linear elasticity. [43] used a neural network in a PCA subspace to replace time integration. [31, 115, 128] all aimed to integrate state transitions in a latent space (see also [141]). [97, 15, 86, 67] all used graph neural networks, which can embrace the typical physics based simulation notion of a stencil (see also [143]). [100, 35] used a PINN-style objective function. Although PINNs (first proposed in [90]) are self-supervised and as such do not require ground truth data, there is nothing special about their architecture implying that they need just as much training data as any other method in order to properly generalize to unseen data. [105] used Transformers. [66] used three networks to capture static, coarse, and wrinkle dynamics separately. Most similar to our approach, [82, 142, 23] used virtual bones, skinning, and/or super-resolution techniques; however, (unlike us) they used RNNs to animate the virtual bones.

Rigging and Skinning: There is a long history of rigging and skinning in computer graphics, although not necessarily geared towards cloth animation. We refer interested readers to [71] for linear blend skinning, [63] for pose space deformation, [57] for weighted pose space deformation, [51, 52] for dual quaternion skinning, and [94] for a survey. Other interesting works include the parametric body models in [2] (SCAPE) and [69] (SMPL) and the joint extraction in [60] (SSDR). It is difficult to skin clothing (especially when it is loose-fitting) using standard (non-neural) skinning techniques; however, see e.g. [126, 139].

Neural Shape: There have been a number of efforts to infer shape (and appearance, see [59]) using neural networks. [4] added per-vertex displacements on top of the skinned body mesh, and [3] similarly added per-vertex displacements on top of a skinned face mesh. [48] also added per-vertex displacements on top of skinned body mesh, but augmented these with dynamic motion from analytic springs. For cloth, there have been various attempts to create a high resolution mesh from the physics simulation of a coarser mesh, see e.g. [53, 80, 15]. Inferring cloth from body pose and shape alone (without a coarse simulation mesh) is significantly more difficult, see e.g. [99, 39, 84, 47, 62, 7, 116, 61, 65].

Collisions: The successful approach to cloth-cloth self collisions using continuous collision detection (CCD) in [10] (which leveraged [89]) led to a plethora of work on improving the efficiency of $O(n \log n)$ collision detection (see e.g. [34, 112, 117, 118, 64]); however, for

cloth-body collisions, signed distance functions (SDFs) remain prevalent due to their more efficient $O(1)$ cost (see e.g. [11]). The main drawback of SDFs is that the three dimensional discretization of the volumetric field is expensive to load and store in memory; thus, analytic SDFs are more popular for real-time applications (see e.g. [9, 137, 78]). Neural network approximations to SDFs have the potential to be efficient enough to be used in real-time applications. Inference is typically fast enough, but the amount of network parameters required (or new network parameters required, when switching from one SDF to another) needs to be small enough to be efficiently loaded and/or stored in memory. We refer the interested reader to [95, 83, 92, 107, 72, 134] for various details (and discussions on differentiability).

Physics-Informed Neural Networks (PINNs): For neural network inferenced cloth, the computational burden from processing collisions can be alleviated to some degree by utilizing interpenetration -free training data. Unfortunately, regularization (which is generally necessary and desirable) prevents inferenced cloth from being interpenetration-free even when the training data is interpenetration-free. Thus, PINN-style [90] collision losses can be quite useful during training (see [101, 7, 8, 39, 100]), as they allow one to increase the penalty on cloth-body interpenetrations without forcing overfitting to the interpenetration-free training data. The PINN losses can be made as stiff as desired while otherwise maintaining desirable regularization on the deviation of the cloth from the training data positions.

3.3 Rope Chain Simulation

Given a garment mesh, we define a set of virtual bones distributed across the garment (either manually or by a procedural algorithm such as SSDR [60]); then, the virtual bones are interconnected vertically to form a set of simulatable rope chains (see Figure 3.2). Not only does this significantly reduce the number of degrees of freedom that need to be simulated, but (we argue that) the rope chains provide a much better approximation to the desired ballistic degrees of freedom than a rubbery mass-spring mesh does: Rope chains can be made to swing and rotate freely, whereas a mass-spring system follows linearized rotations resisted by spring stretching. Rope chains can be made to buckle freely, whereas a mass-spring system over-resists buckling (locking when under-discretized, see e.g. [46]).

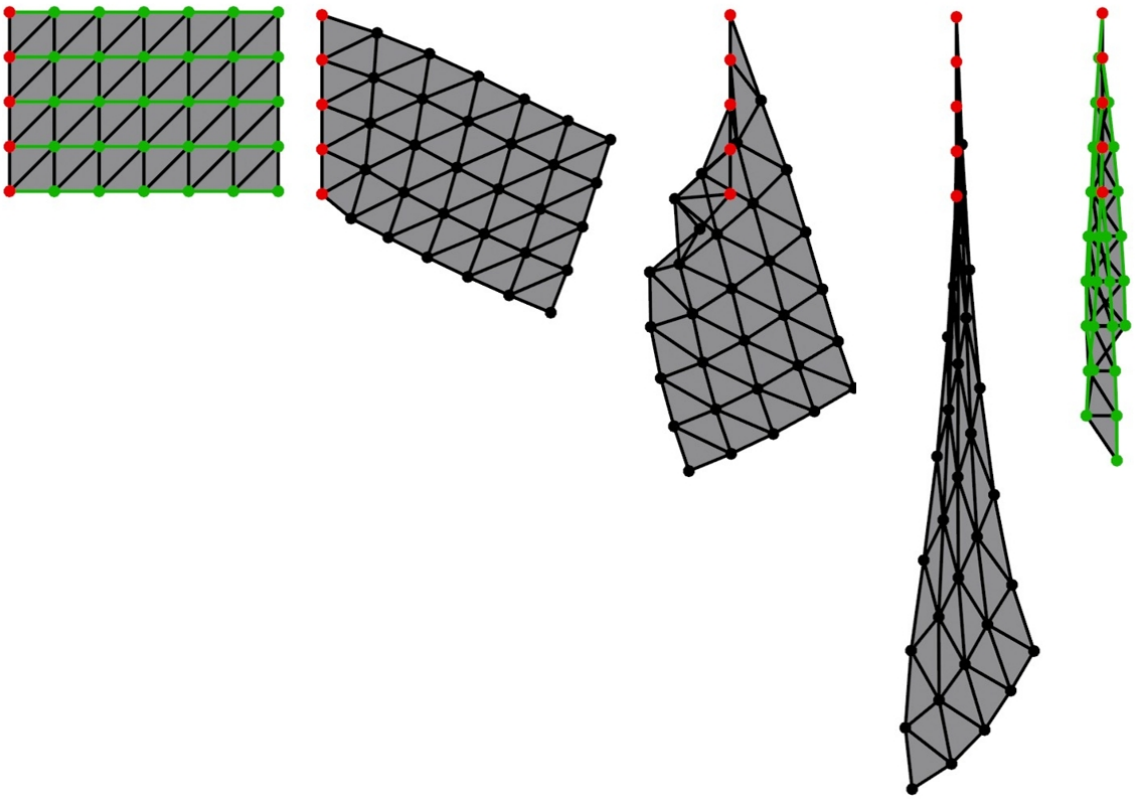


Figure 3.3: The far left subfigure shows a a coarsely discretized mesh with position constraints on the five red nodes. The next three subfigures show the results of steady state simulations using a decreasing spring stiffness (from left to right). The final subfigure shows a rope chain simulation (the rope chains are shaded green) of the same degrees of freedom. The mass-spring simulations lock with stiffer springs and overstrech with weaker springs. The spring stiffness in the middle subfigure was chosen to approximately match the downward stretching extent of the rope chain simulation, which is what one would expect without overstretching; however, locking occurs since the springs are still too stiff.

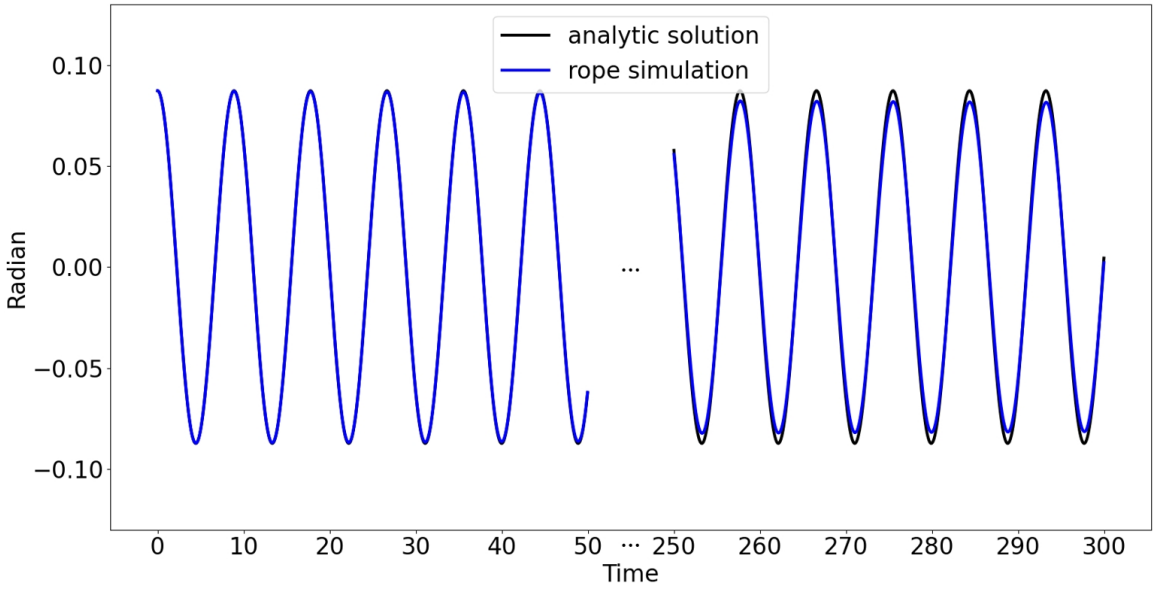


Figure 3.4: In this figure, we simulate two virtual bones connected by a single rope with the top virtual bone fixed and the bottom virtual bone free to rock back and forth as a pendulum. The results compare well to the analytic solution for pendulum motion for both shorter times (left) and longer times (right), illustrating the efficacy of our numerical approach.

Enforcing inextensibility is trivial (from root to tip) for a rope chain, whereas various ad-hoc techniques are required in order to prevent over-stretching for a mass-spring simulation. Etc. See Figure 3.3.

A semi-implicit Newmark style time integration scheme is used separately for each rope chain (see e.g. [11]):

1. $\vec{v}^{n+\frac{1}{2}} = \vec{v}^{n} + \frac{\Delta t}{2}\frac{\vec{F}(\vec{x}^{n},\vec{v}^{n})}{M}$
2. $\vec{x}^{n+1} = \vec{x}^{n} + \Delta t \vec{v}^{n+\frac{1}{2}}$
3. Resolve collisions, perturbing $\vec{x}^{n+1}$ and $\vec{v}^{n+\frac{1}{2}}$
4. $\vec{v}^{n+1} = \vec{v}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\frac{\vec{F}(\vec{x}^{n+1},\vec{v}^{n+\frac{1}{2}})}{M}$

This is often referred to as central differencing in the computational mechanics literature. Central differencing preserves rich high-frequency dynamic motion significantly better than fully-implicit methods (such as backward Euler) do. The velocity updates (steps 1 and 4) are discussed in detail in Section 3.4, the position update (step 2) is discussed in detail in Section 3.5, and collisions (step 3) are discussed in detail in Section 3.6. See Figure 3.4.

3.4 Velocity Update

Given a rope chain, let $\vec{x}_0, ..., \vec{x}_m$ denote the virtual bones from root to tip where $\vec{x}_0$ is kinematically constrained to follow some part of the body (or some other object). The magnitude of each $\vec{l_i} = \vec{x}_i - \vec{x}_{i-1}$ should never exceed the maximal length $l_{max,i}$ of the corresponding rope; however, there is no penalty for slack ($|\vec{l_i}| = l_i < l_{max,i}$) in the rope. Except for $\vec{x}_0$, various external forces $\vec{F}_{ext,i}$ are applied to the virtual bones. Examples include gravity $\vec{F}_g = M_i\vec{g}$, wind drag $\vec{F}_{wind} = -c_{wind}(\vec{v}_i - \vec{v}_{wind})$, etc. In order to prevent a rope chain from deviating too far from its neighboring rope chains, springs can be attached laterally connecting each virtual bone to its neighbors on neighboring rope chains. We treat these as soft constraints, meant to influence but not overly dictate the simulation; thus, they are added to $\vec{F}_{ext,i}$ and given the same relative importance as gravity, wind drag, and other similar forces. One may also desire forces that aim to preserve rest angles between consecutive pairs of virtual bones, in order to coerce the cloth towards its rest shape. These forces would also be included in $\vec{F}_{ext,i}$.

When a rope reaches its maximal length $l_{max,i}$, the two virtual bones it attaches to will rotate around each other. The magnitude of the centripetal force required to maintain this rotation is

$$\begin{aligned} F_{c,i} &= M_{i-1}\frac{|(\vec{v}_{i-1}-\vec{v}_c)-\left((\vec{v}_{i-1}-\vec{v}_c)\cdot\hat{l}_i\right)\hat{l}_i|^2}{|\vec{x}_{i-1}-\vec{x}_c|} \\ &= M_i\frac{|(\vec{v}_i-\vec{v}_c)-\left((\vec{v}_i-\vec{v}_c)\cdot\hat{l}_i\right)\hat{l}_i|^2}{|\vec{x}_i-\vec{x}_c|} \end{aligned} \tag{3.1}$$

where $\vec{x}_c = \frac{M_{i-1}\vec{x}_{i-1}+M_i\vec{x}_i}{M_{i-1}+M_i}$ is the center of mass, $\vec{v}_c = \frac{M_{i-1}\vec{v}_{i-1}+M_i\vec{v}_i}{M_{i-1}+M_i}$ is the velocity at the center of mass, and $\hat{l}_i = \frac{\vec{l}_i}{l_i}$ is unit length (i.e. a direction). For the first rope, which connects the kinematic $\vec{x}_0$ with $\vec{x}_1$, $\vec{x}_c = \vec{x}_0$ and $\vec{v}_c = \vec{v}_0$ due to $M_0 = \infty$ (note that second line of Equation 3.1 needs to be used to calculate $F_{c,1}$, in order to avoid dealing with L'Hopital's rule in the first line).

When a rope reaches its maximal length $l_{max,i}$, an additional tension force with magnitude $T_i \geq 0$ is added to the non-kinematic virtual bones it attaches to (the kinematic root ignores these forces). The net force in the virtual bones can be defined via

$$\vec{F}_{net,i} = \vec{F}_{ext,i} - T_i\hat{l}_i + T_{i+1}\hat{l}_{i+1} \tag{3.2a}$$

$$\vec{F}_{net,m} = \vec{F}_{ext,m} - T_m\hat{l}_m \tag{3.2b}$$

where $i \in \{1, ..., m-1\}$ and $T_i = 0$ whenever $l_i < l_{max,i}$. In order to preserve the rotational motion for each taut rope,

$$\vec{F}_{net,1}\cdot\hat{l}_1 \leq -F_{c,1} + M_1\ddot{\vec{x}}_0\cdot\hat{l}_1 \tag{3.3a}$$

$$\vec{F}_{net,i}\cdot\hat{l}_i - \vec{F}_{net,i-1}\cdot\hat{l}_i \leq -2F_{c,i} \tag{3.3b}$$

where $i \in \{2, ..., m\}$ and the $\ddot{\vec{x}}_0$ term accounts for the motion of the kinematic root. Substituting Equations 3.2a and 3.2b into Equations 3.3a and 3.3b gives

$$-T_1 + \hat{l}_2\cdot\hat{l}_1T_2 \leq -\vec{F}_{ext,1}\cdot\hat{l}_1 - F_{c,1} + M_1\ddot{\vec{x}}_0\cdot\hat{l}_1 \tag{3.4a}$$

$$\hat{l}_{i-1}\cdot\hat{l}_iT_{i-1} - 2T_i + \hat{l}_{i+1}\cdot\hat{l}_iT_{i+1} \leq (\vec{F}_{ext,i-1} - \vec{F}_{ext,i})\cdot\hat{l}_i - 2F_{c,i} \tag{3.4b}$$

$$\hat{l}_{m-1}\cdot\hat{l}_mT_{m-1} - 2T_m \leq (\vec{F}_{ext,m-1} - \vec{F}_{ext,m})\cdot\hat{l}_m - 2F_{c,m} \tag{3.4c}$$

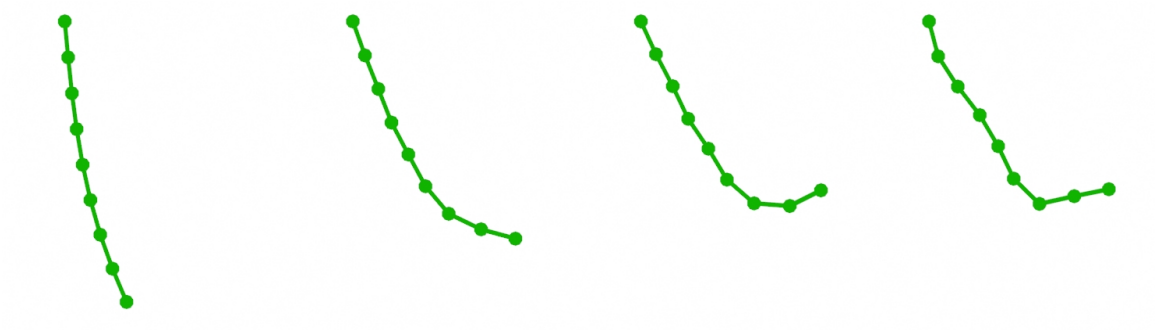


Figure 3.5: Motion of a single swinging rope chain, showcasing varying numbers of Gauss-Seidel iterations. Left to right: 1, 5, 10 iterations, and iterating until the relative error is smaller than a tolerance of 10^{-6}. With more iterations, the rope chain is less damped (as expected). The same number of iterations (or the same tolerance) was used for both the tension and impulse computations. Note that it might be more efficient (depending on the application) to use a different number of iterations on the tension and impulse computations.

where $i \in \{2, ..., m-1\}$. This tri-diagonal linear system of inequalities (for the unknown tension) decouples into separate blocks whenever a slack rope has $l_i < l_{max,i}$ or two consecutive ropes are orthogonal (with a dot product of zero). For computational efficiency, we iterate these equations in tip-to-root order (from bottom to top in Equation 3.4) using Gauss-Seidel; for most real-time applications, typically only one tip-to-root sweep is required. In particular, we rewrite Equations 3.4a, 3.4b, and 3.4c in reverse order as

$$T_m \geq \frac{(\vec{F}_{ext,m} - \vec{F}_{ext,m-1}) \cdot \hat{l}_m + 2F_{c,m} + \hat{l}_{m-1} \cdot \hat{l}_m T_{m-1}}{2} \tag{3.5a}$$

$$T_i \geq \frac{(\vec{F}_{ext,i} - \vec{F}_{ext,i-1}) \cdot \hat{l}_i + 2F_{c,i} + \hat{l}_{i-1} \cdot \hat{l}_i T_{i-1} + \hat{l}_{i+1} \cdot \hat{l}_i T_{i+1}}{2} \tag{3.5b}$$

$$T_1 \geq \vec{F}_{ext,1} \cdot \hat{l}_1 + F_{c,1} - M_1 \ddot{\vec{x}}_0 \cdot \hat{l}_1 + \hat{l}_2 \cdot \hat{l}_1 T_2 \tag{3.5c}$$

and enforce them sequentially (from top to bottom in Equation 3.5) by choosing each T_i equal to the larger between zero and right hand side.

After solving for T_i via Equation 3.5, Equation 3.2 can be used to find the net force $\vec{F}_{net,i}$ on each non-kinematic virtual bone. Given $\vec{F}_{net,i}$ for each non-kinematic virtual bone, the velocity can be updated in any order (for both step 1 and step 4 of the time integration); in addition, the velocity of the kinematic virtual bone (the root) should be updated as well.

The updated velocities may be subject to an additional instantaneous impulse of magnitude $I_i \geq 0$ whenever a rope is at its maximal length $l_{max,i}$. Let $\vec{v}_{pre,i}$ and $\vec{v}_{post,i}$ be the

velocity before and after (respectively) this instantaneous exchange of momentum; then, the impulses along the ropes are applied via

$$M_i\vec{v}_{post,i} = M_i\vec{v}_{pre,i} - I_i\hat{l}_i + I_{i+1}\hat{l}_{i+1} \tag{3.6a}$$

$$M_m\vec{v}_{post,m} = M_m\vec{v}_{pre,m} - I_m\hat{l}_m \tag{3.6b}$$

where $i \in \{1, ..., m-1\}$ and $I_i = 0$ whenever $l_i < l_{max,i}$. Note that $\vec{v}_{post,0} = \vec{v}_{pre,0}$, since $M_0 = \infty$. In order to prevent each taut rope from overstretching,

$$(\vec{v}_{post,i} - \vec{v}_{post,i-1}) \cdot \hat{l}_i \leq 0 \tag{3.7}$$

must hold, where $i \in \{1, ..., m\}$. Substituting Equations 3.6a and 3.6b into Equation 3.7 gives

$$-\frac{1}{M_1}I_1 + \frac{\hat{l}_2 \cdot \hat{l}_1}{M_1}I_2 \leq (\vec{v}_{pre,0} - \vec{v}_{pre,1}) \cdot \hat{l}_1 \tag{3.8a}$$

$$\frac{\hat{l}_{i-1} \cdot \hat{l}_i}{M_{i-1}}I_{i-1} - (\frac{1}{M_{i-1}} + \frac{1}{M_i})I_i + \frac{\hat{l}_{i+1} \cdot \hat{l}_i}{M_i}I_{i+1} \leq (\vec{v}_{pre,i-1} - \vec{v}_{pre,i}) \cdot \hat{l}_i \tag{3.8b}$$

$$\frac{\hat{l}_{m-1} \cdot \hat{l}_m}{M_{m-1}}I_{m-1} - (\frac{1}{M_{m-1}} + \frac{1}{M_m})I_m \leq (\vec{v}_{pre,m-1} - \vec{v}_{pre,m}) \cdot \hat{l}_m \tag{3.8c}$$

where $i \in \{2, ..., m-1\}$. This tri-diagonal linear system of inequalities decouples into separate blocks whenever a slack rope has $l_i < l_{max,i}$ or two consecutive ropes are orthogonal. For computational efficiency, we iterate these equations in root-to-tip order (from top to bottom in Equation 3.8) using Gauss-Seidel; for most real-time applications, typically only one root-to-tip sweep is required. In particular, we rewrite 3.8a, 3.8b, and 3.8c as

$$I_1 \geq M_1\left((\vec{v}_{pre,1} - \vec{v}_{pre,0}) \cdot \hat{l}_1 + \frac{\hat{l}_2 \cdot \hat{l}_1}{M_1}I_2\right) \tag{3.9a}$$

$$I_i \geq \frac{M_{i-1}M_i}{M_{i-1} + M_i}\left((\vec{v}_{pre,i} - \vec{v}_{pre,i-1}) \cdot \hat{l}_i + \frac{\hat{l}_{i-1} \cdot \hat{l}_i}{M_{i-1}}I_{i-1} + \frac{\hat{l}_{i+1} \cdot \hat{l}_i}{M_i}I_{i+1}\right) \tag{3.9b}$$

$$I_m \geq \frac{M_{m-1}M_m}{M_{m-1} + M_m}\left((\vec{v}_{pre,m} - \vec{v}_{pre,m-1}) \cdot \hat{l}_m + \frac{\hat{l}_{m-1} \cdot \hat{l}_m}{M_{m-1}}I_{m-1}\right) \tag{3.9c}$$

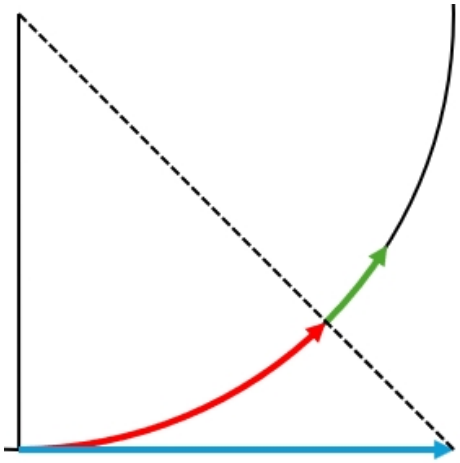
and enforce them sequentially (from top to bottom in Equation 3.9) by choosing each I_i equal to the larger between zero and right hand side.

Note that we chose tip to root for the tension so that each virtual bone feels the weight of all the virtual bones below it even when executing only one iteration. Conversely, we chose root to tip for the impulse since it has been argued (and shown) to work well for contact resolution in prior works (see e.g. [38]). Figure 3.5 illustrates the results one might typically expect with different numbers of iterations.

3.5 Position Update

In contrast to the tension and impulse computations discussed in Section 3.4 which address force and velocity constraints respectively, a stricter approach is desirable for position constraints (especially to avoid errors in the rendered visualizations); thus, we execute one sweep from root to tip on the length of each rope in order to prevent it from exceeding its maximum length. Each virtual bone is updated from its time t^n position to its time t^{n+1} position by considering both its time $t^{n+\frac{1}{2}}$ velocity (computed as described in Section 3.4) and the rope constraint on its position relative to the previously updated virtual bone (in the root to tip sweep). That is, $\vec{x}_i^n$ is updated to $\vec{x}_i^{n+1}$ by considering both $\vec{v}_i^{n+\frac{1}{2}}$ and the rope that connects the virtual bone to $\vec{x}_{i-1}^{n+1}$.

When the rope is slack, the virtual bone's position can evolve freely with no hindrance from the constraint; however, when the rope is taut, the virtual bone is constrained to rotate on the sphere about $\vec{x}_{i-1}^{n+1}$. Approximating this with an evolve-and-project strategy damps the rotation to the linearized rotation. This can be seen by projecting the distance moved in a linearized rotation (colored blue in the figure) back onto a great circle and noting that the arc length thus traversed (colored red in the figure) is smaller than the distance covered in the linearized rotation, which is the arc length distance that should have been traversed (the sum of red and green colored arcs in the figure). Thus, we address the constrained motion with a non-linearized (actual) rotation.



The tautness of the rope is governed by the quadratic function

$$f(s) = |\vec{x}_i^n + s\vec{v}_i^{n+\frac{1}{2}} - \vec{x}_{i-1}^{n+1}|^2 - l_{max,i}^2 \tag{3.10}$$

where $f(s) < 0$ indicates slack and $f(s) > 0$ indicates overstretching. When $f(\Delta t) \leq 0$, the rope is not overstretched at the end of the time step and $\vec{x}_i^{n+1} = \vec{x}_i^n + \Delta t \vec{v}_i^{n+\frac{1}{2}}$ is accepted as the final position. Otherwise, when $f(\Delta t) > 0$, we compute the largest root s_{root} in the interval $[0, \Delta t]$. If there is no root in the interval, then we set $s_{root} = 0$ and project the overstretched $\vec{x}_i^n$ back onto the surface of the sphere via

$$\vec{x}_i^n \leftarrow \vec{x}_{i-1}^{n+1} + \frac{\vec{x}_i^n - \vec{x}_{i-1}^{n+1}}{|\vec{x}_i^n - \vec{x}_{i-1}^{n+1}|} l_{max,i} \tag{3.11}$$

so that it is no longer overstretched. In the interval $[0, s_{root}]$, the virtual bone is allowed to move unhindered by the constraint via $\vec{x}_i^{s_{root}} = \vec{x}_i^n + s_{root} \vec{v}_i^{n+\frac{1}{2}}$. Note that this intentionally ignores any overstretching in $[0, s_{root}]$, since such overstretching is overcome automatically and may only be the (spurious) result of updating the virtual bones one at a time (from root to tip) as opposed to uniformly.

In the interval $[s_{root}, \Delta t]$, the virtual bone is constrained to rotate on the sphere centered at $\vec{x}_{i-1}^{n+1}$ of radius $l_{max,i}$. Given $\vec{l}_i^{s_{root}} = \vec{x}_i^{s_{root}} - \vec{x}_{i-1}^{n+1}$ and $\hat{l}_i^{s_{root}} = \frac{\vec{l}_i^{s_{root}}}{|\vec{l}_i^{s_{root}}|}$, the tangential velocity

$$\vec{v}_T = \vec{v}_i^{n+\frac{1}{2}} - (\vec{v}_i^{n+\frac{1}{2}} \cdot \hat{l}_i^{s_{root}}) \hat{l}_i^{s_{root}} \tag{3.12}$$

is used to determine the distance $d_T = |\vec{v}_T|(\Delta t - s_{root})$ the virtual bone rotates on the great circle specified by the direction $\hat{v}_T = \frac{\vec{v}_T}{|\vec{v}_T|}$. A rotation matrix R is defined to rotate the virtual bone by an amount $\theta = \frac{d_T}{l_{max,i}}$ about the axis $\hat{l}_\theta = \hat{l}_i^{s_{root}} \times \hat{v}_T$ via

$$\vec{l}_i^{n+1} = R(\theta, \hat{l}_\theta) \vec{l}_i^{s_{root}} \tag{3.13}$$

to obtain $\vec{x}_i^{n+1} = \vec{x}_{i-1}^{n+1} + \vec{l}_i^{n+1}$.

After the position update, the velocities may no longer satisfy the constraint preventing overstretching (see Equation 3.7). Thus, new impulses can be computed and applied (following the discussion in the second half of Section 3.4). Alternatively, impulses can instead be computed and applied after resolving collisions, both after the position update and after resolving collisions, or deferred entirely (and left unmodified until the end of second velocity update).

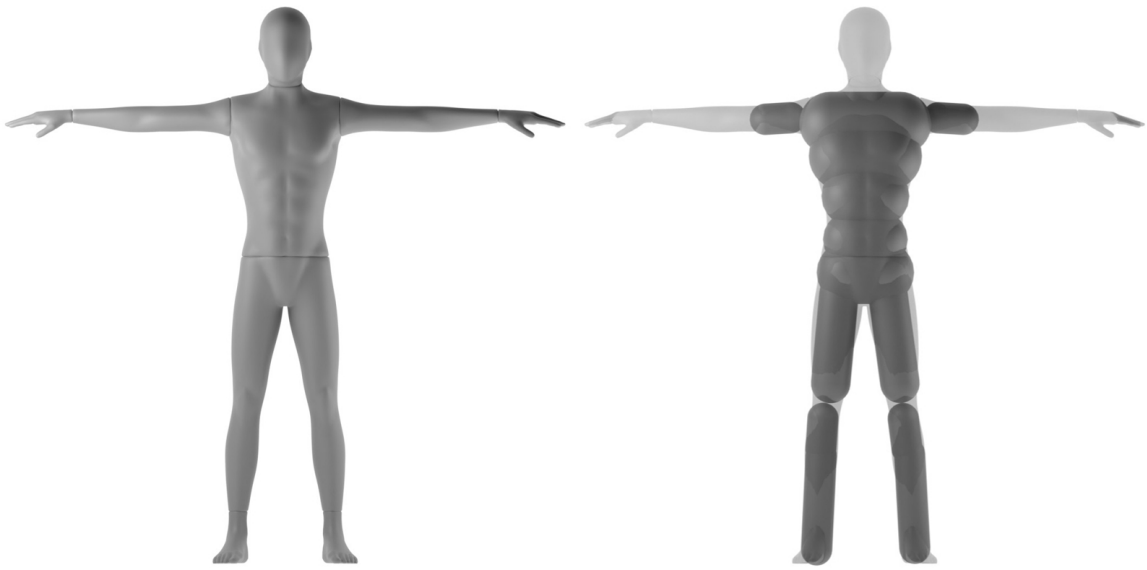


Figure 3.6: For the sake of computational efficiency, we represent the volumetric body (left) with a number of analytically defined SDFs (right). Although other representations (three dimensional grid discretizations, neural representations, etc.) are compatible with our approach, they incur higher computational costs.

3.6 Collisions

Generally speaking, we follow the method in [11] detecting and processing collisions between each non-kinematic virtual bone and each signed distance function (SDF). Although this approach has quite efficient $O(1)$ processing time for each virtual bone, it is computationally expensive to load three dimensional SDF discretizations into memory; in addition, the limited memory availability in most real-time systems makes it infeasible to store the SDFs persistently. Thus, we avoid these costly three dimensional discretizations by utilizing SDFs that can be defined analytically (see e.g. [9, 137, 78]). See Figure 3.6. Alternatively, a number of authors have aimed to represent SDFs via neural networks (see e.g. [83, 92]). If the number of parameters in the neural network can be made to be much smaller than an equivalently accurate three dimensional discretization, then the computational costs associated with high demands on memory could be avoided. Instead of aiming to make each neural SDF utilize less memory than its equivalently accurate three dimensional discretization, one could aim to minimize the additional memory burden incurred by switching from one neural SDF to another (e.g. by using shape descriptors).

Motivated by the cloth-object collision discussion in [103], we propose a modification to

[11] in order to obtain more robust behavior for real-time applications with coarse discretizations and large time steps. In the standard approach, an initially non-interpenetrating particle that ends up in the interior of an SDF is pushed outwards in the $\nabla\phi$ direction until it reaches the surface of the SDF (or a bit further than the surface when aiming for wrinkle preservation as discussed in [11]). In contrast, a real-world particle is unable to penetrate into the interior of an object and instead collides with the surface and responds accordingly. In the limit as the time step goes to zero and the number of points used to represent the surface goes to infinity, the standard numerical approach converges to the correct real-world solution (roughly speaking, ignoring the inability to properly model friction, microscopic structure, etc.). However, the numerical errors are exacerbated by the large time steps and the coarse surface discretizations utilized for real-time applications. Although continuous collision detection (CCD) could be used to increase the accuracy while maintaining a large time step, CCD is too computationally burdensome for most real-time applications (although progress is being made, see e.g. [64] and the references therein).

The standard approach of evolving a particle into an interpenetrating state and subsequently pushing it outwards in the $\nabla\phi$ direction can be thought of as a predictor-corrector method modeling the actual path of the particle (the path that CCD would aim to trace out). Aiming to preserve the computational efficiency of the predictor, we propose modifying the corrector in order to obtain a more accurate final state. This can be done efficiently by choosing a more appropriate direction for push out than $\nabla\phi$. In fact, $\nabla\phi$ can be highly erroneous when objects are thin (causing a particle to be pushed to the wrong side) or have high curvature (causing a particle to be pushed to the wrong direction); moreover, these errors are exacerbated by the larger time steps typically used in real-time applications. Given limited information, our ansatz is that the safest push out direction is the reverse direction along the path the particle traversed as it penetrated into the collision body. At the very least, this aims to return the particle to the point where a CCD collision would have occurred.

Assuming the collision body is stationary, the reverse path out of the collision body back towards the CCD collision point has direction $\vec{r} = \vec{x}^n - \vec{x}^{n+1}$ where $\vec{x}^{n+1}$ is the predicted position of the particle (penetrating into the collision body). The main difficulty associated with using this direction is that it is unclear how far the particle should move. There are several options for addressing this. One could use the local value of $|\phi|$ as usual, but this does not necessarily alleviate interpenetration when $\hat{r} = \frac{\vec{r}}{|\vec{r}|}$ and $\nabla\phi$ point in different

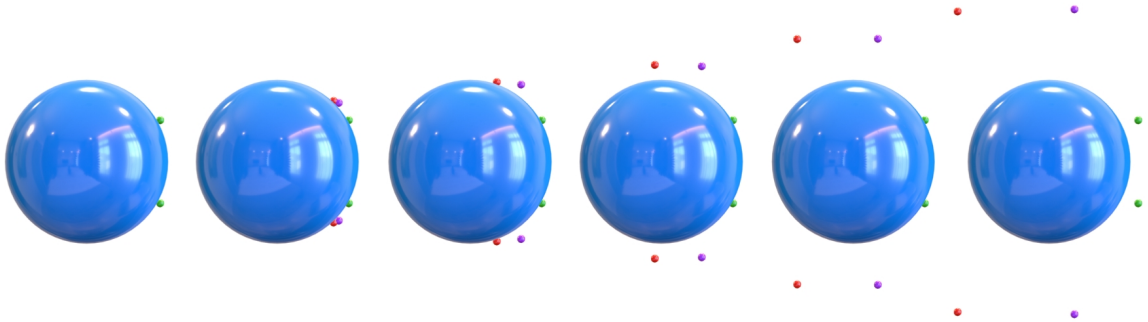


Figure 3.7: Time evolution of a sphere colliding with six particles (6 frames are depicted). The red particles use the $\nabla\phi$ direction for both push out and velocity projection (as is typical), the purple particles use $\hat{r}$ for push out and $\nabla\phi$ for velocity projection, and the green particles use $\hat{r}$ for both push out and velocity projection. Note how replacing $\nabla\phi$ with $\hat{r}$ prevents the particles from quickly working their way around the sphere. In fact, the green particles maintain a persistent contact with the sphere until it stops moving shortly before the last frame (even though friction is not used in this example). The behavior of the green particles is highly preferable to that of the red (and purple) particles when considering collisions between clothing and the human body.

directions. Notably, the local value of $|\phi|$ is always too small, except when $\hat{r} = \nabla\phi$ in which case the particle should exactly reach the surface of the collision body. Therefore, one could iterate the push out a few times in order to better approach the surface. In addition, one could use a small $\epsilon > 0$ to augment the local value of $|\phi|$. This is roughly equivalent to using an SDF thickened by ϵ or to pushing the particle outwards to the $\phi = \epsilon$ isocontour. Finally, note that one could use line search (perhaps via bisection), which is equivalent to CCD for the simple case when the collision body is stationary (CCD is typically much cheaper in this simple case).

When the collision body is moving, $\vec{x}^n$ is not necessarily non-interpenetrating (since the collision body may move to cover it); in such a scenario, $\hat{r}$ is no longer guaranteed to be a suitable replacement for $\nabla\phi$. Fortunately, this is easily remedied by analyzing the problem in the moving frame of the collision body. As the collision body moves and deforms from time t^n to t^{n+1}, we embed the particle to move with it. This guarantees that the new particle location, denoted $\vec{x}_B^n$, is non-interpenetrating. For example, when the collision body moves rigidly, that same rigid motion is applied to $\vec{x}^n$ to obtain $\vec{x}_B^n$. For deforming and/or skinned collision bodies, the deformation/skinning needs to be extended to $\vec{x}^n$ in order to obtain $\vec{x}_B^n$. This allows $\vec{r} = \vec{x}_B^n - \vec{x}^{n+1}$ to be used as a valid push out direction.

After the position $\vec{x}^{n+1}$ is modified to be collision-free, the velocity $\vec{v}^{n+\frac{1}{2}}$ is modified to ensure that the relative normal velocity $(\vec{v}^{n+\frac{1}{2}} - \vec{v}_\phi) \cdot \hat{N}$ does not point into the SDF via

$$v_N^{new} = \max(\vec{v}^{n+\frac{1}{2}} \cdot \hat{N}, \vec{v}_\phi \cdot \hat{N}) \tag{3.14}$$

where $\vec{v}_\phi$ is the velocity of the (extended, if necessary) collision body at $\vec{x}^{n+1}$, and $\hat{N}$ may be chosen as either $\nabla\phi$ or $\hat{r}$. The relative tangential velocity

$$\vec{v}_{rel,T} = \vec{v}^{n+\frac{1}{2}} - \vec{v}^{n+\frac{1}{2}} \cdot \hat{N} - \vec{v}_{\phi,T} \tag{3.15}$$

is defined using the tangential velocity $\vec{v}_{\phi,T} = \vec{v}_\phi - \vec{v}_\phi \cdot \hat{N}$ of the (extended, if necessary) collision body at $\vec{x}^{n+1}$. When the friction coefficient μ is non-zero, the relative tangential velocity is modified via

$$\vec{v}_T^{new} = \vec{v}_{\phi,T} + \max(0, 1 - \mu \frac{v_N^{new} - \vec{v}^{n+\frac{1}{2}} \cdot \hat{N}}{|\vec{v}_{rel,T}|}) \vec{v}_{rel,T} \tag{3.16}$$

as suggested by [10]. The final post collision velocity is given by $v_N^{new}\hat{N} + \vec{v}_T^{new}$. See Figure 3.7.

The collisions are processed sequentially from root to tip. Since collisions alter the positions of the virtual bones, length constraints are enforced in this step as well. This is accomplished by adjusting the position of a virtual bone via $\vec{x}_i^{new} = \vec{x}_{i-1} + l_{max,i}\hat{l}_i$ whenever $l_i > l_{max,i}$. Of course, this can create new collisions, so back-and-forth iteration is desirable. The root to tip sweep is done only once, and any back-and-forth iteration between the collision and the length constraint happens only once for each virtual bone. We recommend starting this iteration with the length constraint, since it may remove the need for collisions. We also stress the importance of finishing this iteration with the collision check in order to preserve a non-interpenetrating state.

3.7 Neural Skinning

After each frame of rope chain simulation, a full cloth mesh needs to be reconstructed for rendering. After extensive experimentation, we were not able to obtain reasonable results via any of the standard skinning methods (such as LBS [71] or Dual Quaternion Skinning [51]). We were also unable to remedy these issues with a corrective quasistatic neural

network (see Section 3.8). The difficulties are likely due to the rope chain simulation's inability to produce good rotational information for the virtual bones, even with various procedural modifications. Thus, we took an alternative approach that utilizes a neural network to infer PCA coefficients for the cloth mesh from the virtual bone translational degrees of freedom (only). This resulted in a mesh suitable enough for a corrective QNN to operate on (see Section 3.8).

For each cloth mesh under consideration, we utilize on the order of 5000 frames of simulated data (any offline simulation system will do) in order to construct a standard PCA model based on non-rigid displacements from the cloth rest state. The rest state of the cloth mesh is defined by its steady-state draped position in the rest pose of the body. Given an animated body pose, the rigid component of the cloth displacement is calculated as the rigid displacement of a key body part (the neck for the cape and the pelvis for the skirt) and removed from the cloth displacement in order to obtain its non-rigid displacement. About 100 PCA basis functions are retained for neural skinning. When the cloth meshes were converted to rope chains (see Figure 3.2), not all virtual bones were simulated (see the non-interconnected virtual bones in Figure 3.2); however, these virtual bones are needed in order to reconstruct the full cloth mesh and as such are also used as input into the neural skinning network. Given non-rigid displacements of the virtual bones from their positions in the cloth mesh rest state (their rigid component is identical to that used for the cloth mesh), the neural skinning network is trained to infer the approximately 100 PCA coefficients used to reconstruct the full cloth mesh.

We utilize a lightweight 2-layer MLP with 500 neurons per layer. In order to train the network, the same 5000 frames of simulation data (previously described) is used for supervision. The virtual bones are embedded in the cloth rest state and barycentrically enslaved to move with the simulation data yielding the inputs for the network. For each frame utilized in the data term of the loss function, the network-inferred PCA coefficients are used to reconstruct a cloth mesh that is compared to the simulation ground truth vertex-by-vertex via a standard L2 norm. Importantly, each frame is treated as non-sequential in order to reduce the need for a higher capacity neural network, minimize the burden associated with collecting more training data, and avoid unnecessary overfitting. We did not find the need to add any additional terms (such as Laplacians) that would regularize the cloth mesh, since the use of a PCA model already provides sufficient regularization.

Even when the ground truth cloth meshes used in training do not interpenetrate the

body, the inferred results will typically contain interpenetrations. This is caused by the regularization used to combat overfitting and to increase the efficacy of generalization to unseen data. In order to alleviate interpenetration without adversely affecting desirable regularization, we include a PINN-style collision loss during training. When a cloth vertex is found to be interpenetrating, we calculate a suitable non-interpenetrating position for that vertex (using push out) and include the difference between the vertex and its non-interpenetrating state in the PINN-style collision loss. The non-interpenetrating state can be calculated using $\nabla\phi$ as the push out direction along with the ϕ value (with or without iteration) or CCD to calculate the push out distance. Given the potentially large inaccuracy of the cloth state during training, $\nabla\phi$ often points in an unhelpful direction; thus, we instead chose an alternative push out direction $\hat{r}$ pointing from the current position to the ground truth position. Note that it can be desirable for the non-interpenetrating state to be well-separated from the collision body (not just on the zero-isocontour) in order to create a buffer on the vertices that helps to alleviate edge interpenetrations (and/or to preserve wrinkling, see e.g. [11]). Since we detach the non-interpenetrating state from the automatic differentiation graph, an interpenetrating vertex's contribution to the gradient from the PINN-style collision loss is parallel to its contribution from the data term. This can be interpreted as increasing the importance of matching the ground truth for vertices that are interpenetrating as compared to vertices that are non-interpenetrating; in fact, we use a weight of 1000 on the PINN-style collision loss and a weight of only 0.1 for the data term.

It is important to note that there is a large discrepancy between the virtual bone configurations that arise from rope chain simulations and the configurations in the training data, which are obtained by barycentrically embedding virtual bones in a mass-spring cloth simulation mesh. That is, we are aiming to make the network generalize well to (sometimes significantly) out-of-distribution data. The usual process of utilizing holdout data does help to increase the ability of a network to generalize to unseen data; however, the errors can still be quite large when the training data and holdout data come from one distribution and the unseen data comes from another. This is more akin to a domain gap. Both the PINN collision loss and the regularization obtained via the PCA model help to alleviate these issues with out-of-distribution unseen data. Even though the network-inferenced cloth may not follow the dynamic trajectory of the virtual bones as closely as one might prefer, the cloth tends to have an aesthetically pleasing shape and be interpenetration-free.

Figure 3.8: Top row: first five principle components of the neural skinning PCA model for the cape. Bottom row: first five principle components of the neural shape PCA model for the cape. All of the images depict the augmentation of the rest shape cloth mesh by the PCA displacements, even though the displacements are added to the skinned mesh (not the rest state mesh) during neural shape inference. The PCA model used for skinning tends to capture low-frequency deformations, while the PCA model used for shape inference is better suited for capturing higher-frequency deformations.

3.8 Neural Shape Inference

Similar to the neural skinning in Section 3.7, we utilize a 2-layer MLP with 500 neurons per layer in order to inference about 100 PCA coefficients from the non-rigid displacements of the virtual bones from their positions in the cloth rest state. The same 5000 frames of simulated data is used for training; however, the PCA model is calculated based on the non-rigid displacement between the output of the neural skinning network and the ground truth (see [129] for interesting discussion on multi-stage approaches). While the neural skinning is responsible for constructing a coarse approximation to the cloth mesh, the neural shape inference focuses on capturing more of the high-frequency spatial detail. This is obvious when comparing the two PCA models (See Figure 3.8). Similar to Section 3.7, a PINN-style collision loss is also included.

3.9 Results and Discussion

For both the neural skinning and the neural shape networks, we used a weight of 0.1 on the data term and weight of 1000 on the PINN-style collision loss. The initial learning rate for Adam [56] was set to 10^{-4} to train the neural skinning network and 10^{-5} to train the neural shape network. A smaller learning rate was specified for the neural shape network, since its PCA coefficients represent smaller displacements. A cosine annealing schedule was used to decay the learning rate over epochs. The collision body SDFs were expanded by a small amount, and only one iteration was used for push out (for computational efficiency).

We obtained our 5000 frames of training data (for both the cape and the skirt) using Houdini Vellum. 80 percent of the frames were used in the loss function to train the neural network, and 10 percent of the frames (unseen in training) were used in a validation loss in order to choose parameters that might generalize well. Figures 3.9 and 3.10 demonstrate the efficacy of the neural skinning network and the neural shape network respectively. Another 10 percent of the frames were kept as true holdout data, in order to predict the ability of the networks to generalize to unseen (but still in-distribution) data. See Figure 3.11. Finally, Figure 3.12 showcases the efficacy of the PINN-style collision loss.

In order to demonstrate how our networks generalize to unseen out-of distribution data from rope chain simulations, we considered two types of loose-fitting garments: capes and skirts. See Figures 3.15 and 3.16. Weak lateral springs were added in order to connect virtual bones from different rope chains (for both the skirt and the cape). Wind drag was added to the cape virtual bones. Damping, relative to the parent virtual bone in the chain, was added to the skirt virtual bones. The rope chains were collided against the analytic SDFs depicted in Figure 3.6. Given the virtual bone positions obtained via rope chain simulation, we reconstructed the full cloth mesh using the neural skinning and neural shape networks. Even though the virtual bone positions generated from the rope chain simulations are out of distribution as compared to the training, validation, and holdout data, Figures 3.15 and 3.16 demonstrate that the networks generalized well and obtained good results.

Finally, we consider an RNN approach. Following the low frequency module in [82], we trained a Gated Recurrent Unit (GRU) [16]. The inputs are the current body pose and the latent vector from the prior state, and the outputs are the current virtual bone positions and the current latent vector. For the sake of a fair comparison, we used the same 5000 frames of (cape) simulation as training data; however, the RNN would obviously benefit

Figure 3.9: The first row shows the results of the neural skinning network, which can be compared to the ground truth training data in the second row. In order to demonstrate that the network does have the ability to match the ground truth data, the third row shows the overfit results obtained via overtraining; of course, an overfit network will not generalize well to unseen data.

Figure 3.10: The first row shows the results of the neural skinning network (identical to the first row in Figure 3.9). The second row shows the results of the neural shape network applied on top of the neural skinning result. The third row shows the ground truth training data. The fourth row shows how overtraining the neural shape network leads to results that well match (albeit overfit, similar to the last row in Figure 3.9) the ground truth training data.

Figure 3.11: The first row shows the results of the neural skinning network, and the second row shows the results of the neural shape network applied on top of the neural skinning result. These holdout frames from the training set give some indication of how the network will perform on unseen data. Note that the network needs to generalize to out-of-distribution data (from rope chain simulations, as is discussed in the last paragraph in Section 3.7), not just to holdout frames from the training set.

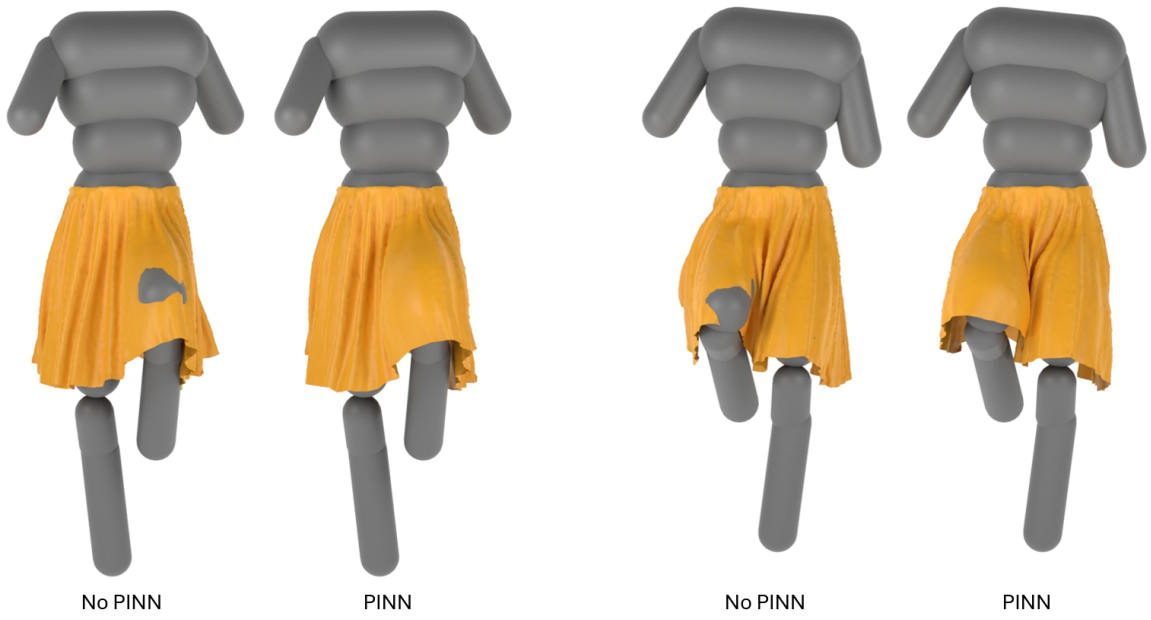


Figure 3.12: Comparing the result of a neural skinning network trained without and with the PINN-style collision loss.

from having access to an increased amount of training data. A separate RNN was trained for each rope chain. See Figures 3.13 and 3.14. Alternatively, the RNNs could be trained to match rope chain simulations; however, there is no reason to believe that predicting out-of-distribution data would fair any better than predicting in-distribution data.

3.10 Conclusion and Future Work

We presented a novel method for the real-time simulation of loose-fitting garments leveraging neural networks for both skinning and shape inference. We demonstrated that only a small number of degrees of freedom is necessary in order to capture the ballistic motions. In order to overcome the locking artifacts typically incurred by using only a small number of degrees of freedom, we proposed a rope chain simulation method that maintains inextensibilily while still allowing swinging, rotating, and buckling without resistance. A new history-based approach to collisions was introduced in order to accommodate fast-moving collision bodies and large time steps. A neural skinning solution was utilized in order to create a cloth mesh from the rope chain simulation degrees of freedom, and a quasistatic neural shape network was subsequently used in order to add additional details. These networks

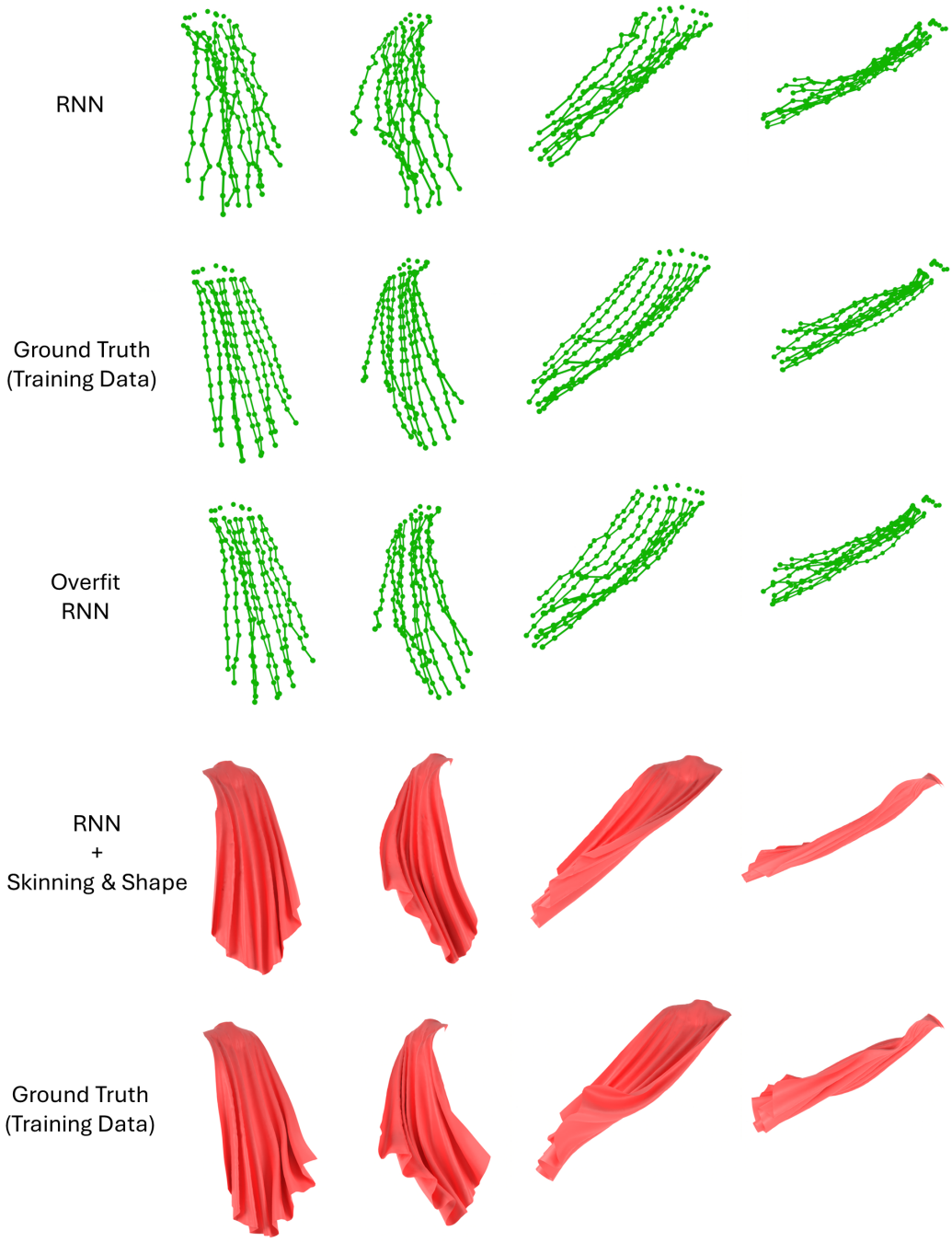


Figure 3.13: We trained a separate RNN for each rope chain (10 RNNs in total). The RNN results (first row) are quite noisy compared to the ground truth training data (second row). The thrid row shows the overfit results obtained via overtraining. The fourth row shows the result of applying our neural skinning and neural shape inference to the RNN-inferred virtual bone positions (from the first row). Comparing these results to the ground truth training data (fifth row) illustrates that our networks generalize well to this noisy RNN input.

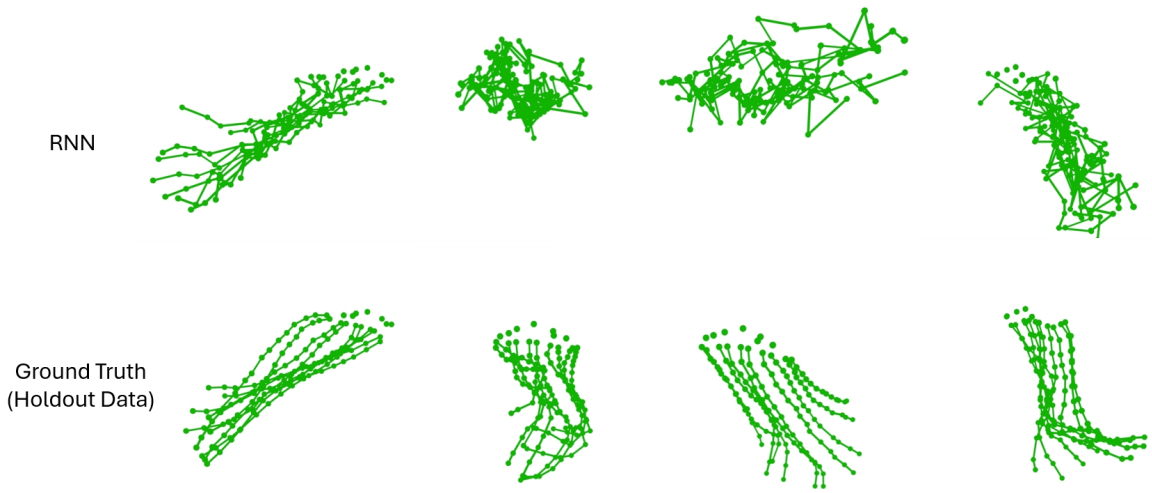


Figure 3.14: The main issue with the RNN is that it generalizes very poorly to the holdout data. Presumably, this issue could be fixed by collecting more and more training data, but that excessively increases the cost incurred in both data collection (via numerical simulation) and training (via optimization). Even though our skinning and shape networks can smooth the noise in this RNN output (as they did in Figure 3.13), the dynamics will still be completely wrong.

were trained with the aid of a PINN-style collision loss.

Since they are based on the PCA coefficients, the neural skinning and shape networks produce reasonable mesh output even with out-of-distribution input; however, out-of-distrubution input does adversely affect their ability to maintain an interpenetration-free state. Training the networks with a PINN-style collision loss gives reasonably penetration-free results for in-distribution data, but not necessarily for out-of-distribution data. Improving the interpenetration- freeness for out-of-distribution input is perhaps the most important area for the future work. Although the mesh could be post-processed to be interpenetration-free, doing so significantly decreases performance and thus the applicability for real-time applications. It is relatively cheap to process collisions for a small number of virtual bones, but far more expensive to process collisions for the entire mesh. Other interesting directions for future work include: making the rope chain simulation differentiable in order to automatically find constitutive parameters that best match the ground truth, investigating different network architectures for the neural skinning, generalizing a single network so that it can be used across different garment types and/or body shapes, etc.

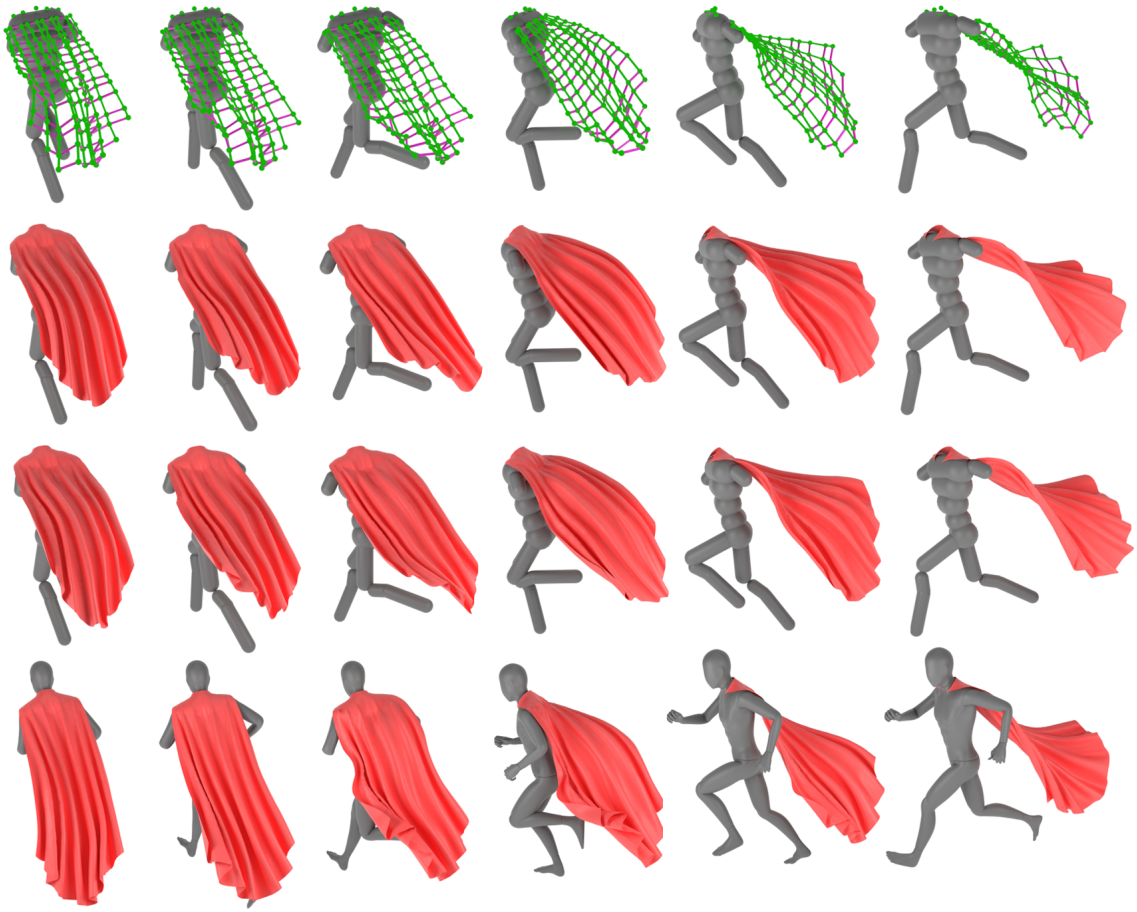


Figure 3.15: Simulation of a cape on an animation of a running person. First row: rope chain simulation. The rope chains are shaded green, and the weak lateral springs are visualized as purple edges. Second row: neural skinning, based on the first row. Third row: neural shape inference, based on the second row. Fourth row: Houdini cloth simulation with approximately 12K vertices (as a reference). The Houdini simulation and our rope chain approach both produce good dynamics, although the Houdini simulation does exhibit visually displeasing erroneous over-stretching artifacts; in addition, the Houdini simulation is an order of magnitude slower than our currently unoptimized code.

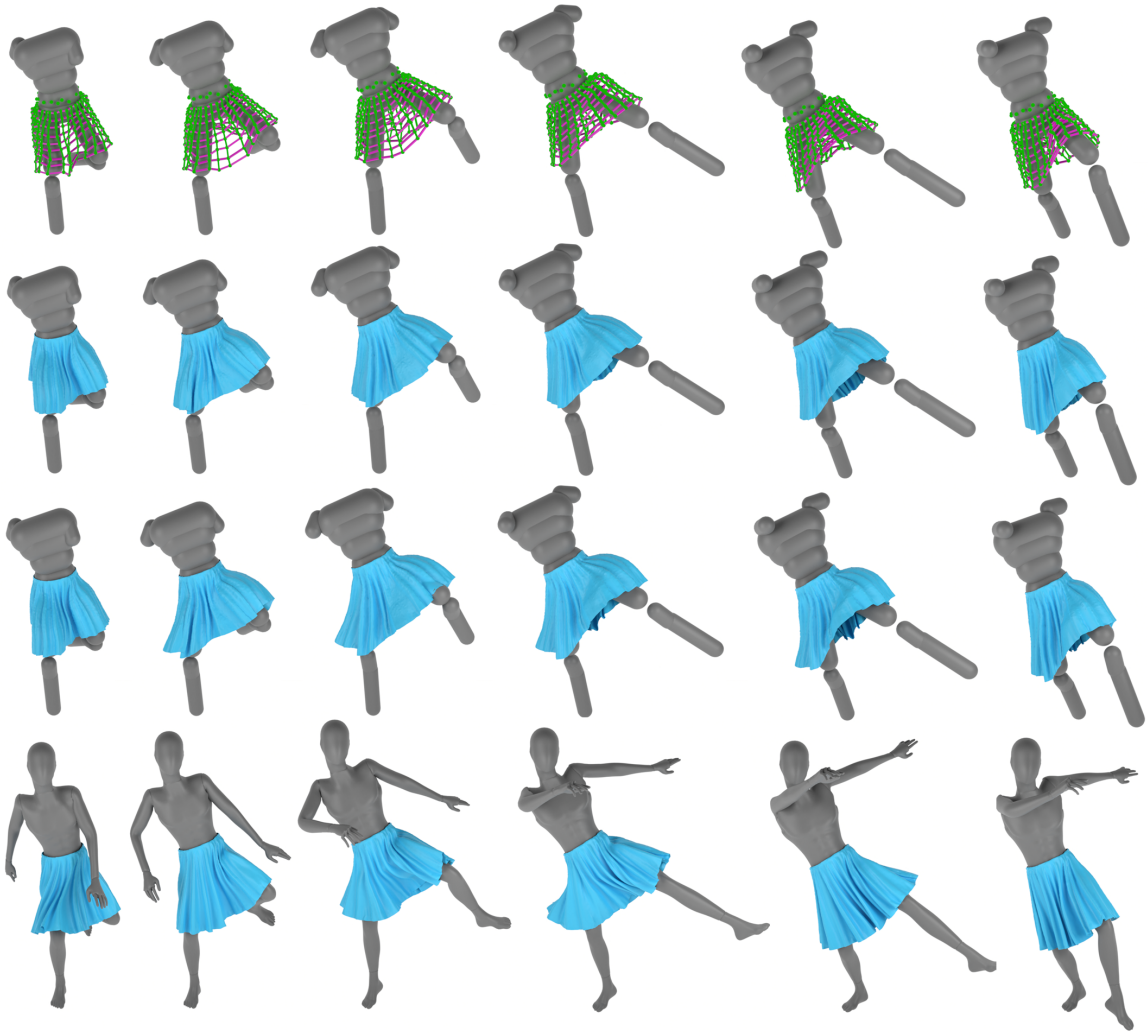


Figure 3.16: Simulation of a skirt on an animation of a dancing person. First row: rope chain simulation. The rope chains are shaded green, and the weak lateral springs are visualized as purple edges. Second row: neural skinning, based on the first row. Third row: neural shape inference, based on the second row. Fourth row: Houdini cloth simulation with approximately 18K vertices (as a reference). The skirt is rendered with a blue color in order to differentiate it from the yellow skirt figures, which use barycentrically embedded virtual bone positions instead of rope chain simulations as the network input.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

This dissertation presents a general paradigm for achieving high-resolution, high-fidelity physics simulation in real-time. The main takeaway of this thesis is rethinking and redesigning real-time physics models given the significant progress of deep neural networks for quasistatics. Inspired by the recent advancement of deep neural networks in approximating quasistatic simulations, we employ a lightweight quasistatic neural network (QNN), either a fully connected network or a convolutional neural network (CNN), for both flesh and cloth simulations to efficiently handle quasistatic aspects. Given that QNNs have excelled at capturing quasistatics in recent years, we rethought the dynamic component and propose physics models mainly for capturing ballistic motion that are simple but very effective with QNN-based enhancement. We carefully designed these physics models so that they are not only fast and stable but also free from the artifacts associated with low-resolution mass-spring simulations. In Chapter 2, we introduce a zero-restlength spring physics model to capture the secondary jiggling motion of flesh. This model can be integrated analytically, ensuring unconditional stability and allowing for the use of arbitrarily large time steps. In Chapter 3, we propose a 1D rope chain physics model to capture the dynamics of loose-fitting garments with large-scale ballistic motion. Despite using inequality constraints on each rope chain, this model can be efficiently solved with only a few Gauss-Seidel iterations. By leveraging QNNs and redesigning physics models for them, our paradigm can be practically applied in real-world applications, showcasing a promising direction for future research.

4.2 Future Work

Looking ahead, I am interested in exploring the following research directions:

4.2.1 Collisions

In Chapter 2, we found that a simple zero-restlength spring model can generate visually collision-free flesh simulations. However, adding a collision step in the analytical solution could improve robustness. In Chapter 3, although we efficiently handle collisions between virtual bones and the human body, the QNN-inferred cloth mesh might still suffer from interpenetrations. We, along with other recent works [101, 7, 8, 39, 100], address this issue by adding a collision loss during training. Despite its effectiveness on training data, the network might still output interpenetrated cloth mesh when given data outside the training set, especially out-of-distribution input. Therefore, a promising future direction is to develop methods for robustly generating collision-free meshes under all conditions.

4.2.2 Differentiable Simulation

Making our physics model differentiable is an intriguing avenue because it enables the automatic learning of constitutive parameters from ground truth simulation data. In Chapter 2, we made our zero-restlength spring model differentiable by carefully deriving the partial derivatives and implementing them robustly in code. The next step is to make our 1D rope chain simulation from Chapter 3 differentiable, allowing us to learn rope constitutive parameters from ground truth cloth simulations as well.

4.2.3 Network Architecture

For both our flesh and cloth simulation projects, we use simple fully connected networks or convolutional neural networks as our QNN architecture for efficiency. Although effective, it would be interesting to explore other network architectures, such as graph neural networks (GNNs) [136] and mesh convolutional neural networks (MeshCNNs) [41], which might offer further improvements in performance and generalization.

4.2.4 Generalization

To be practically used in real applications, our QNNs and physics models are currently dedicated to one body type/shape. A compelling research direction is to explore how to generalize our method for different body shapes. Potential approaches include adding body shape coefficients as network inputs [83] and automating the virtual bone setup process for different body shapes using techniques like skin decomposition [60].

Bibliography

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

[2] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*, pages 408–416. 2005.

[3] 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.

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

[5] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, 1998.

[6] Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. Discrete elastic rods. In *ACM SIGGRAPH 2008 papers*, pages 1–12. 2008.

[7] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. PBNS: physically based neural simulation for unsupervised garment pose space deformation. *ACM Trans. Graph.*, 40(6):198:1–198:14, 2021.

[8] Hugo Bertiche, Meysam Madadi, Emilio Tylson, and Sergio Escalera. Deepsd: Automatic deep skinning and pose space deformation for 3d garment animation. In

Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 5471–5480, 2021.

[9] James F Blinn. A generalization of algebraic surface drawing. *ACM transactions on graphics (TOG)*, 1(3):235–256, 1982.

[10] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 594–603, 2002.

[11] Robert Bridson, Sebastian Marino, and Ronald Fedkiw. Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, page 28–36, Goslar, DEU, 2003. Eurographics Association.

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

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

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

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

[16] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[17] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. *ACM Trans. Graph.*, 21(3):604–611, July 2002.

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

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

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

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

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

[23] Junqi Diao, Jun Xiao, Yihong He, and Haiyong Jiang. Combating spurious correlations in loose-fitting garment animation through joint-specific feature learning. In *Computer Graphics Forum*, volume 42, page e14939. Wiley Online Library, 2023.

[24] Epic Games. Metahuman creator. https://metahuman.unrealengine.com.

[25] Epic Games. Pushing next-gen real-time technology in marvel 1943: Rise of hydra. Game Developers Conference 2024.

[26] Epic Games. Unreal engine 5. https://www.unrealengine.com.

[27] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22, 2001.

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

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

[30] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 23–30, 2001.

[31] Lawson Fulton, Vismay Modi, David Duvenaud, David IW Levin, and Alec Jacobson. Latent-space dynamics for reduced deformable simulation. In *Computer graphics forum*, volume 38, pages 379–391. Wiley Online Library, 2019.

[32] Willi Geiger, Nick Rasmussen, Samir Hoon, and Ron Fedkiw. Space battle pyromania. In *SIGGRAPH Sketches*, page 88, 2005.

[33] Zhenglin Geng, Daniel Johnson, and Ronald Fedkiw. Coercing machine learning to output physically accurate results. *Journal of Computational Physics*, 406:109099, 2020.

[34] Naga K Govindaraju, David Knott, Nitin Jain, Ilknur Kabul, Rasmus Tamstorf, Russell Gayle, Ming C Lin, and Dinesh Manocha. Interactive collision detection between deformable models using chromatic decomposition. *ACM Transactions on graphics (TOG)*, 24(3):991–999, 2005.

[35] Artur Grigorev, Michael J Black, and Otmar Hilliges. Hood: Hierarchical graphs for generalized modelling of clothing dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16965–16974, 2023.

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

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

[38] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. *ACM transactions on graphics (TOG)*, 22(3):871–878, 2003.

[39] Erhan Gundogdu, Victor Constantin, Amrollah Seifoddini, Minh Dang, Mathieu Salzmann, and Pascal Fua. Garnet: A two-stream network for fast and accurate 3d cloth draping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8739–8748, 2019.

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

[41] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (ToG)*, 38(4):1–12, 2019.

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

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

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

[45] Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. Tetrahedral meshing in the wild. *ACM Trans. Graph.*, 2018.

[46] Ning Jin, Wenlong Lu, Zhenglin Geng, and Ronald P Fedkiw. Inequality cloth. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 1–10, 2017.

[47] Ning Jin, Yilin Zhu, Zhenglin Geng, and Ronald Fedkiw. A pixel-based framework for data-driven clothing. In *Computer Graphics Forum*, volume 39, pages 135–144. Wiley Online Library, 2020.

[48] Yongxu Jin, Yushan Han, Zhenglin Geng, Joseph Teran, and Ronald Fedkiw. Analytically integratable zero-restlength springs for capturing dynamic modes unrepresented by quasistatic neural networks. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022.

[49] Yongxu Jin, Dalton Omens, Zhenglin Geng, Joseph Teran, Abishek Kumar, Kenji Tashiro, and Ronald Fedkiw. A neural-network-based approach for loose-fitting clothing. *arXiv preprint arXiv:2404.16896*, 2024.

[50] Daniel Johnson and Ronald Fedkiw. Addressing discontinuous root-finding for subsequent differentiability in machine learning, inverse problems, and control. *Journal of Computational Physics*, 497:112624, 2024.

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

[52] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)*, 27(4):1–23, 2008.

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

[54] Doyub Kim, Woojong Koh, Rahul Narain, Kayvon Fatahalian, Adrien Treuille, and James F O'Brien. Near-exhaustive precomputation of secondary cloth effects. *ACM Transactions on Graphics (TOG)*, 32(4):1–8, 2013.

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

[56] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[57] Tsuneya Kurihara and Natsuki Miyata. Modeling deformable human hands from medical images. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 355–363, 2004.

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

[59] Zorah Lahner, Daniel Cremers, and Tony Tung. Deepwrinkles: Accurate and realistic clothing modeling. In *Proceedings of the European conference on computer vision (ECCV)*, pages 667–684, 2018.

[60] Binh Huy Le and Zhigang Deng. Smooth skinning decomposition with rigid bones. *ACM Transactions on Graphics (TOG)*, 31(6):1–10, 2012.

[61] Dohae Lee, Hyun Kang, and In-Kwon Lee. Clothcombo: Modeling inter-cloth interaction for draping multi-layered clothes. *ACM Transactions on Graphics (TOG)*, 42(6):1–13, 2023.

[62] Christopher Lewin. Swish: Neural network cloth simulation on madden nfl 21. In *ACM SIGGRAPH 2021 Talks*, pages 1–2. 2021.

[63] John P Lewis, Matt Cordner, and Nickso Fong. 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 2000*, pages 165–172. ACM Press/Addison-Wesley Publishing Co., 2000.

[64] Cheng Li, Min Tang, Ruofeng Tong, Ming Cai, Jieyi Zhao, and Dinesh Manocha. P-cloth: interactive complex cloth simulation on multi-gpu systems using dynamic matrix assembly and pipelined implicit integrators. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

[65] Ren Li, Benoît Guillard, and Pascal Fua. Isp: Multi-layered garment draping with implicit sewing patterns. *Advances in Neural Information Processing Systems*, 36, 2024.

[66] Yu Di Li, Min Tang, Xiao Rui Chen, Yun Yang, Ruo Feng Tong, Bai Lin An, Shuang Cai Yang, Yao Li, and Qi Long Kou. D-cloth: Skinning-based cloth dynamic

prediction with a three-stage network. In *Computer Graphics Forum*, volume 42, page e14937. Wiley Online Library, 2023.

[67] Yu Di Li, Min Tang, Yun Yang, Zi Huang, Ruo Feng Tong, Shuang Cai Yang, Yao Li, and Dinesh Manocha. N-cloth: Predicting 3d cloth deformation with mesh-based networks. In *Computer Graphics Forum*, volume 41, pages 547–558. Wiley Online Library, 2022.

[68] Haolin Liu, Ye Han, Daniel Emerson, Houriyeh Majditehran, Qi Wang, Yoed Rabin, and Levent Burak Kara. Real-time prediction of soft tissue deformations using data-driven nonlinear presurgical simulations. *arXiv preprint arXiv:2010.13823*, 2020.

[69] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: a skinned multi-person linear model. *ACM Trans. Graph.*, 34(6), oct 2015.

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

[71] Thalmann Magnenat, Richard Laperrière, and Daniel Thalmann. Joint-dependent local deformations for hand animation and object grasping. Technical report, Canadian Inf. Process. Soc, 1988.

[72] Ishit Mehta, Manmohan Chandraker, and Ravi Ramamoorthi. A level set theory for neural implicit evolution under explicit flows. In *European Conference on Computer Vision*, pages 711–729. Springer, 2022.

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

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

[75] Luke Metz, C Daniel Freeman, Samuel S Schoenholz, and Tal Kachman. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2021.

[76] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159. Citeseer, 2003.

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

[78] Hitoshi Nishimura, Makoto Hirai, Toshiyuki Kawai, Toru Kawata, Isao Shirakawa, and Koichi Omura. Object modeling by distribution function and a method of image generation. *Journal of papers IEICE Japan'85*, 68(4), 1985.

[79] NVIDIA. Rtx ray tracing. https://developer.nvidia.com/rtx/ray-tracing.

[80] Young Jin Oh, Tae Min Lee, and In-Kwon Lee. Hierarchical cloth simulation using deep neural networks. In *Proceedings of Computer Graphics International 2018*, pages 139–146. 2018.

[81] Dinesh K Pai. Strands: Interactive simulation of thin solids using cosserat models. In *Computer graphics forum*, volume 21, pages 347–352. Wiley Online Library, 2002.

[82] Xiaoyu Pan, Jiaming Mai, Xinwei Jiang, Dongxue Tang, Jingxiang Li, Tianjia Shao, Kun Zhou, Xiaogang Jin, and Dinesh Manocha. Predicting loose-fitting garment deformations using bone-driven motion networks. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

[83] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[84] Chaitanya Patel, Zhouyingcheng Liao, and Gerard Pons-Moll. 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, pages 7365–7375, 2020.

[85] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2020.

[86] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

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

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

[89] Xavier Provot. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation'97: Proceedings of the Eurographics Workshop in Budapest, Hungary, September 2–3, 1997*, pages 177–189. Springer, 1997.

[90] 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.

[91] Alan Marquez Razon, Yizhou Chen, Yushan Han, Steven Gagniere, Michael Tupek, and Joseph Teran. A linear and angular momentum conserving hybrid particle/grid iteration for volumetric elastic contact. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(3):1–25, 2023.

[92] Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Meshsdf: Differentiable iso-surface extraction. *Advances in Neural Information Processing Systems*, 33:22468–22478, 2020.

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

[94] Nadine Abu Rumman and Marco Fratarcangeli. Skin deformation methods for interactive character animation. In *Computer Vision, Imaging and Computer Graphics Theory and Applications: 11th International Joint Conference, VISIGRAPP 2016, Rome, Italy, February 27–29, 2016, Revised Selected Papers 11*, pages 153–174. Springer, 2017.

[95] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2304–2314, 2019.

[96] Yasmin Salehi and Dennis Giannacopoulos. Physgnn: A physics–driven graph neural network based model for predicting soft tissue deformation in image–guided neurosurgery. *Advances in Neural Information Processing Systems*, 35:37282–37296, 2022.

[97] 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.

[98] Igor Santesteban, Elena Garces, Miguel A Otaduy, and Dan Casas. Softsmpl: Data-driven modeling of nonlinear soft-tissue dynamics for parametric humans. In *Computer Graphics Forum*, volume 39, pages 65–75. Wiley Online Library, 2020.

[99] Igor Santesteban, Miguel A Otaduy, and Dan Casas. Learning-based animation of clothing for virtual try-on. In *Computer Graphics Forum*, volume 38, pages 355–366. Wiley Online Library, 2019.

[100] Igor Santesteban, Miguel A Otaduy, and Dan Casas. SNUG: Self-Supervised Neural Dynamic Garments. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[101] Igor Santesteban, Nils Thuerey, Miguel A Otaduy, and Dan Casas. Self-supervised collision handling via generative 3d garment models for virtual try-on. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11763–11773, 2021.

[102] Andrew Selle, Michael Lentine, and Ronald Fedkiw. A mass spring model for hair simulation. In *ACM SIGGRAPH 2008 papers*, pages 1–11. 2008.

[103] Andrew Selle, Jonathan Su, Geoffrey Irving, and Ronald Fedkiw. Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. *IEEE transactions on visualization and computer graphics*, 15(2):339–350, 2008.

[104] Hyewon Seo, Kaifeng Zou, and Frederic Cordier. Dsnet: Dynamic skin deformation prediction by recurrent neural network. In *Computer Graphics International Conference*, pages 365–377. Springer, 2021.

[105] Yidi Shao, Chen Change Loy, and Bo Dai. Towards multi-layered 3d garments animation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14361–14370, 2023.

[106] Seung Heon Sheen, Egor Larionov, and Dinesh K Pai. Volume preserving simulation of soft tissue with skin. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(3):1–23, 2021.

[107] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021.

[108] Jonathan Richard Shewchuk. Tetrahedral mesh generation by delaunay refinement. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, page 86–95, 1998.

[109] Hang Si. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)*, 41(2):1–36, 2015.

[110] Eftychios Sifakis, Igor Neverov, and Ronald Fedkiw. Automatic determination of facial muscle activations from sparse motion capture marker data. In *ACM SIGGRAPH 2005 Papers*, pages 417–425. 2005.

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

[112] Avneesh Sud, Naga Govindaraju, Russell Gayle, Ilknur Kabul, and Dinesh Manocha. Fast proximity computation among deformable models using discrete voronoi diagrams. In *ACM SIGGRAPH 2006 Papers*, pages 1144–1153. 2006.

[113] Nigel Sumner, Samir Hoon, Willi Geiger, Sebastian Marino, Nick Rasmussen, and Ron Fedkiw. Melting a terminatrix. In *SIGGRAPH 2003 Sketches & Applications*, 2003.

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

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

[116] Qingyang Tan, Yi Zhou, Tuanfeng Wang, Duygu Ceylan, Xin Sun, and Dinesh Manocha. A repulsive force unit for garment collision handling in neural networks. In *European Conference on Computer Vision*, pages 451–467. Springer, 2022.

[117] Min Tang, Ruofeng Tong, Zhendong Wang, and Dinesh Manocha. Fast and exact continuous collision detection with bernstein sign classification. *ACM Transactions on Graphics (TOG)*, 33(6):1–8, 2014.

[118] Min Tang, Tongtong Wang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. I-cloth: Incremental collision handling for gpu-based interactive cloth simulation. *ACM Transactions on Graphics (TOG)*, 37(6):1–10, 2018.

[119] Joseph Teran, Sylvia Blemker, V Ng Thow Hing, and Ronald Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 68–74. Citeseer, 2003.

[120] Joseph Teran, Eftychios Sifakis, Silvia S Blemker, Victor Ng-Thow-Hing, Cynthia Lau, and Ronald Fedkiw. Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):317–328, 2005.

[121] Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, page 181–190, 2005.

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

[123] 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.

[124] Bohan Wang, Mianlun Zheng, and Jernej Barbič. Adjustable constrained soft-tissue dynamics. In *Computer Graphics Forum*, volume 39, pages 69–79. Wiley Online Library, 2020.

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

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

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

[128] Tuanfeng Y Wang, Tianjia Shao, Kai Fu, and Niloy J Mitra. Learning an intrinsic garment space for interactive authoring of garment animation. *ACM Transactions on Graphics (TOG)*, 38(6):1–12, 2019.

[129] Yongji Wang and Ching-Yao Lai. Multi-stage neural networks: Function approximator of machine precision. *Journal of Computational Physics*, page 112865, 2024.

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

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

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

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

[134] Jane Wu, Diego Thomas, and Ronald Fedkiw. Weakly-supervised 3d reconstruction of clothed humans via normal maps. *arXiv preprint arXiv:2311.16042*, 2023.

[135] Nannan Wu, Qianwen Chao, Yanzhen Chen, Weiwei Xu, Chen Liu, Dinesh Manocha, Wenxin Sun, Yi Han, Xinran Yao, and Xiaogang Jin. Agentdress: Realtime clothing synthesis for virtual agents using plausible deformations. *IEEE Transactions on Visualization and Computer Graphics*, 27(11):4107–4118, 2021.

[136] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[137] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Soft objects. In *Advanced Computer Graphics: Proceedings of Computer Graphics Tokyo'86*, pages 113–128. Springer, 1986.

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

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

[140] Jiayi Eris Zhang, Seungbae Bang, David I.W. Levin, and Alec Jacobson. Complementary dynamics. *ACM Transactions on Graphics*, 2020.

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

[142] Fang Zhao, Zekun Li, Shaoli Huang, Junwu Weng, Tianfei Zhou, Guo-Sen Xie, Jue Wang, and Ying Shan. Learning anchor transformations for 3d garment animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 491–500, 2023.

[143] Mianlun Zheng, Yi Zhou, Duygu Ceylan, and Jernej Barbic. A deep emulator for secondary motion of 3d characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5932–5940, 2021.

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