APPLICATIONS OF PHYSICALLY BASED
SIMULATIONS OF ELASTIC RODS

A Dissertation
Presented to the Faculty of the Graduate School
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by
Eston Russell Schweickart
December 2019
APPLICATIONS OF PHYSICALLY BASED SIMULATIONS OF ELASTIC RODS

Eston Russell Schweickart, Ph.D.
Cornell University 2019

Physical simulations are widely used to recreate the motions of everyday objects, and elastic rods—long and thin deformable bodies—are an important computer animation primitive. In computer graphics, many simulation models have been developed to produce plausible animations, but many of these methods can also accomplish tasks beyond creating visualizations. This thesis demonstrates how simulation methods for animation of slender rods can be used in other applications.

First, we describe physically based methods for simultaneous generation of animation and sound for deformable rods. We introduce an efficient acoustic radiation method based on dipoles, and show how to tie it to common elastic rod simulation frameworks. We present several examples of our results, including challenging scenes involving thousands of highly coupled frictional contacts.

We then show how elastic rods can improve yarn geometry synthesis techniques. Prior work can generate virtual fiber curves for specific types of yarn, but does not account for how these curves deform as yarns collide. We introduce macro-fibers, elastic rods that represent groups of fibers in a yarn. We split existing yarn curves into macro-fibers and run a short relaxation simulation to allow them to rearrange locally. We show how the resulting curves can be used to predict yarn deformation in the context of knitted fabrics. By following the paths of the macro-fibers, we can generate fiber assemblies for rendering that
more closely resemble the structure of actual cloth than prior work.

Finally, we propose a method for tracking yarn paths through computed tomography (CT) scans of real fabrics. Existing methods are either designed for low-curvature yarn configurations, tend to fail around yarn crossings, or rely on identifying individual fibers within a pattern. Our algorithm finds ridges—chains of high-density voxels that approximate fibers—to guide centerline placement without explicitly identifying fibers, and is robust to arbitrary yarn arrangements. By reconstructing the yarn paths within these volumetric scans, we are more prepared to quantitatively validate and calibrate yarn-level cloth simulations.
BIOGRAPHICAL SKETCH

Eston Schweickart received a B.A. with High Honors in Computer Science with a minor in Mathematics from Oberlin College in 2013. He has worked as a research intern at Mozilla (2013) and Adobe (2015 and 2018).
This thesis is dedicated to Steve and Doug. Thank you for your indispensable advice, guidance, and patience.
ACKNOWLEDGEMENTS

I would like to thank everyone who contributed their time, effort, and expertise to this work: Dae Hyun Kim for early numerical experiments, Rafael Marinheiro for software contributions, Risa Feng for useful discussion, Kim Phoenix for knitted cloth samples, Teresa Porri for operating the CT scanner, Ryan Lefkowitz for debugging rendering artifacts, Sitian Chen for software and image contributions, and Mengqi (Mandy) Xia for high quality yarn renderings. I would also like to express my undying gratitude to all of my mentors and collaborators that helped make this work possible: Steve Marschner, Doug James, Run-dong Wu, Jonathan Leaf, and Timothy Langlois. Many thanks also to the funding sources that supported this work: National Science Foundation (Projects 0905506, 1011919, 1513967, and 1644523), Adobe, Autodesk, and Under Armour.
# TABLE OF CONTENTS

Biographical Sketch .......................................................... iii  
Dedication . ......................................................................... iv  
Acknowledgements ............................................................... v  
Table of Contents ................................................................. vi  
List of Tables ......................................................................... viii  
List of Figures ......................................................................... ix  

1 Introduction ......................................................................... 1  

2 Background .......................................................................... 6  

2.1 Elastics ........................................................................... 6  
2.2 Elastic Rods ..................................................................... 8  
2.2.1 Stretching Forces ......................................................... 8  
2.2.2 Bending Forces ........................................................... 9  
2.2.3 Twisting Forces .......................................................... 11  
2.2.4 Cross-section Shearing ................................................. 12  
2.3 Simulation ........................................................................ 12  

3 Prior Work ........................................................................... 14  

3.1 Elastic Rod Frameworks ..................................................... 14  
3.1.1 General Simulation Frameworks ..................................... 14  
3.1.2 Yarn Simulation Frameworks ......................................... 16  
3.2 Sound Models ................................................................... 17  
3.3 Yarn Geometry and Rendering .......................................... 20  
3.4 Analyzing Volumetric Images ............................................ 22  

4 Animating Elastic Rods with Sound ........................................ 25  

4.1 Rod Dynamics Model ........................................................ 27  
4.1.1 Notation ..................................................................... 27  
4.1.2 Dynamics .................................................................... 29  
4.1.3 Caveats ....................................................................... 30  
4.2 Sound Radiation Model ...................................................... 31  
4.2.1 Dipole Sound Field of a Vibrating Rod Element .............. 31  
4.2.2 Implementation Details ................................................. 35  
4.2.3 Caveats ....................................................................... 38  
4.3 Contact ............................................................................ 39  
4.4 Results ............................................................................. 42  
4.4.1 Examples .................................................................... 43  
4.4.2 Dipole Model Tests ....................................................... 48  
4.5 Discussion and Conclusion ................................................. 49  
4.5.1 Future Work ............................................................... 50
5 Macro-fiber Simulation for High Quality Yarn Geometry 51
  5.1 Yarn Simulation and Geometric Refinement . . . . . . . . . . . . . 54
    5.1.1 Generating Macro-fibers . . . . . . . . . . . . . . . . . . . 55
    5.1.2 Simulation . . . . . . . . . . . . . . . . . . . . . . . . . . 58
    5.1.3 Domain Decomposition . . . . . . . . . . . . . . . . . . . . 61
  5.2 Fiber Instantiation . . . . . . . . . . . . . . . . . . . . . . . . . . 64
    5.2.1 Mapping Macro-fiber Displacement . . . . . . . . . . . . . 65
  5.3 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
    5.3.1 Comparison to Cloth Samples . . . . . . . . . . . . . . . . 68
    5.3.2 Comparison to Tube Mapping . . . . . . . . . . . . . . . . . 70
    5.3.3 Yarn Core Intersections . . . . . . . . . . . . . . . . . . . . 70
  5.4 Discussion and Conclusion . . . . . . . . . . . . . . . . . . . . . 73

6 Towards Tracking Yarn Paths through Volumetric Scans 75
  6.1 Using Ridges to Track Yarn Paths . . . . . . . . . . . . . . . . . . . 76
  6.2 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
  6.3 Discussion and Future Work . . . . . . . . . . . . . . . . . . . . . 81
    6.3.1 Robustness Through Wider Context . . . . . . . . . . . . . 81
    6.3.2 Centerline Smoothing . . . . . . . . . . . . . . . . . . . . . 84
    6.3.3 Applications . . . . . . . . . . . . . . . . . . . . . . . . . . 85
  6.4 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87

7 Conclusion 88

A Proof of Vanishing $D^\omega$ 90

Bibliography 91
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Section</th>
<th>Table Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td><strong>Performance</strong>: For each example, we state the number of control points present in the simulation, the material parameters used (Young’s modulus $E$ and Poisson’s ratio $\nu$), the total wall clock time to generate one second of sound (averaged over the entire simulation), as well as the portion of time spent on rod dynamics simulation, collision detection and resolution, and sound computation. All timing data was measured on an 8-core Intel Xeon E5-2637 v3 CPU.</td>
<td>43</td>
</tr>
<tr>
<td>5.1</td>
<td><strong>Simulation statistics</strong> for generating each of the patterns in Figure 5.9 using macro-fibers. Here we show the amount of wall clock time needed to simulate the macro-fibers for each pattern, the total number of simulation steps, the total number of control points in the pattern, the peak GPU memory usage, the wall clock time to map fiber curves to world space, and the total number of fiber points mapped.</td>
<td>69</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Audiovisual simulation of a spring toy slinking down the stairs: Our method efficiently generates the spatialized sound of this discrete elastic rod (3888 control points) arising from tens of thousands of complex collision events each timestep.</td>
<td>26</td>
</tr>
<tr>
<td>4.2</td>
<td>Notation used within the definition of the rod model.</td>
<td>28</td>
</tr>
<tr>
<td>4.3</td>
<td>(Left) A vibrating rod element depicted as part of a longer extruded rod. (Right) Cross-sectional geometry and notation.</td>
<td>32</td>
</tr>
<tr>
<td>4.4</td>
<td><strong>2D dipole radiation fields for different cross sections</strong>: Acoustic pressure values are calculated in several directions in the $xy$-plane for a few examples of 2D cross sections in response to jerk motion in the vertical direction. Such effects are encoded in the $D$ dipole matrix. The length of each arrow represents the magnitude of the acoustic pressure; orange represents positive values and blue negative.</td>
<td>36</td>
</tr>
<tr>
<td>4.5</td>
<td><strong>Nonrigid damping</strong>: (Top) A 4-ringed Slinky approaching resting contact with the ground plane. (Left) Without damping, we observe buzzing artifacts. (Right) With post-contact nonrigid damping applied, the artifacts are removed, and resting contact is achieved.</td>
<td>40</td>
</tr>
<tr>
<td>4.6</td>
<td><strong>Timestep loop</strong>: This pseudocode describes one iteration of our simulation, taking in the state at time $t_k$ and advancing to time $t_{k+1}$.</td>
<td>42</td>
</tr>
<tr>
<td>4.7</td>
<td><strong>Comparison of frequency spectra</strong> for the Rectangular Rod example. Comparisons are shown between the simulation output (bold line), the recorded data (thin line), and the solutions to the ideal bar equation (dashed lines) for impulses along the long edge (Left) and short edge (Right) of the rod. Note that the ideal bar equation only gives frequencies corresponding to the bending modes, and not their relative intensities. Audio data is obtained from the tail of a short impulse to isolate the bending frequencies.</td>
<td>44</td>
</tr>
<tr>
<td>4.8</td>
<td><strong>Dispersion</strong>: Spectrograms from recorded (Top) and simulated (Bottom) versions of Slinky Dispersion. The pitch glide effects due to dispersion are clearly visible in both.</td>
<td>47</td>
</tr>
<tr>
<td>4.9</td>
<td><strong>Examples</strong>: From left to right: Clock Gong, Rubber Band, Guy Wire, and Chain Link Fence.</td>
<td>48</td>
</tr>
</tbody>
</table>
5.1 Prior work in computer graphics can produce yarn curves that are physically plausible, and generate fiber curve distributions that resemble real yarns. However, when the two are combined naively, the resulting yarn looks overly artificial. We improve on this prior work by splitting the provided yarn curves into macro-fibers, or elastic rod models that represent groups of fibers within a yarn ply. After simulation and inflation, we use these macro-fibers to inform our fiber curve placement. Compared to prior work, this results in yarn geometry that allows fibers to form a denser pattern (blue arrows) while reducing the amount of yarn intersections (red circles).

5.2 Coordinate systems We use three coordinate systems. World space (left), yarn space (center), and ply space (right).

5.3 Macro-fiber Curves in yarn space are the sum of two helices, each with a different radius ($r_i$ and $r_{yarn}$), initial polar angle ($\theta_i$ and $\theta_j$) and twist rate ($\alpha_{ply}$ and $\alpha_{yarn}$). Here the twist rates are exaggerated for visualization purposes.

5.4 Squashed ply When a yarn ply (red) is trapped between two adjacent plies (green), it takes on a cross section shape that is far from circular as seen in the colorized CT scan of a jersey knit (left). Our macro-fiber simulation is able to reproduce this phenomenon. (right).

5.5 Domain decomposition for a $3 \times 3$ grid: (Left) To simulate the center cell, all yarn segments in the green “freeze” boundary are clamped, and all segments outside of this boundary are cropped. (Right) To remove discontinuities over seams, the relaxation is run multiple times over shifted grids (red, green, and blue).

5.6 Domain decomposition comparison The pattern Tumbling Moss Blocks is simulated twice, once as a single domain (left) and once splitting the domain into a $3 \times 3 \times 1$ grid (right). The macro-fiber displacements differ slightly between the two simulations (top), but no major artifacts are noticeable over a larger scale. Once fibers are instantiated along the macro-fiber curves (bottom), the difference between the patterns is negligible.

5.7 In-plane deformation (Left): The cross section plane is tesselated in ply space with the positions of macro-fibers as vertices (blue dots). (Right) The mesh is transformed to a plane in world space based on the corresponding positions of the macro-fibers in world space. The map allows for deformation (notice the elliptical shape) but is locally injective.
5.8 **Vertical bands** (red boxes) appear in jersey knit patterns because of yarn interactions governed by yarn twist rates. Unlike previous work, our method is able to capture these patterns. We compare our method to tube mapping both by rendering photorealistic images to match photographs of knitted patterns (top), as well as rendering CT scans and synthesized fiber curves under the same rendering pipeline (bottom). For full resolution images, please see the supplementary material.

5.9 **Comparison to tube mapping** Here we show a number of complex knitting patterns where fiber curves are instantiated using both tube mapping (top of each cell) and our technique (bottom of each cell). For full resolution images, please see the supplementary material.

6.1 Left: to estimate directions, correlations are calculated three sweeps of the volume, once for each dimension. Right: to calculate off-axis directions, sub-voxel shifts are performed around a center slice. In practice, the volume is resampled to obtain these shifts.

6.2 Top: A slice of a CT scan of a silk weave sample. Bottom: A visualization of the direction estimates using our algorithm; red indicates the direction normal to the slicing plane, green indicates the vertical direction in-plane, and blue indicates the horizontal direction. Note the sharp discontinuities where the warp and weft yarns are in contact.

6.3 Yarn centerlines (solid lines) are traced through CT scans (translucent red). Top: A jersey knit pattern. Bottom: a rib stitch pattern.

6.4 Each ply of a 2-ply yarn (blue and yellow lines) is tracked individually through a scan of a woven fabric (translucent red).
CHAPTER 1
INTRODUCTION

Simulation is a critical tool in many different application domains. Being able to predict the movement and deformation of objects allows *in silico* testing and verification of objects before they are fabricated (resulting in faster and cheaper iteration and more reliable systems), visualization of complex phenomena not visible to the unaided eye, and animations that would be otherwise tedious, if not impossible, to create by hand.

Many different simulation models exist for any particular use case. There are several trade-offs to consider when choosing among them: accuracy, numerical stability, computational efficiency, and generality are all factors that differentiate these methods. For instance, in engineering applications where predictive modeling is frequently used, the need for accuracy justifies relatively high computational costs. Additionally, the context of the simulation is frequently well defined *a priori*, meaning generality is not a dominant concern. On the other hand, in computer graphics contexts such as simulation for movies and games, stability and efficiency are often high priorities, and low accuracy is acceptable as long as visual plausibility is maintained. For interactive applications in particular, it is often important that the simulation functions predictably for arbitrary user-specified geometries and virtual environments.

As a result, researchers in computer graphics have developed frameworks for simulating materials of all kinds, trading off these priorities in ways that are distinct from other fields of research. Very often, the goal is to create an animation or visualization, so the simulation models are tailored for this specific purpose. However, these techniques have properties that often make them
suitable for other uses. This thesis presents two adaptations to simulation models developed within computer graphics to make them suitable for applications other than physically based animation, taking advantage of the generality and computational speed afforded by these models.

We focus on a particular geometric primitive: elastic rods, or deformable bodies that are locally thin in two of their three dimensions. This kind of geometry is common in virtual worlds (e.g., hair, ropes, and strings), and its dynamics have been well studied. Its simple, 1-dimensional nature yields fast and efficient simulation algorithms. Chapter 2 introduces basic simulation and elastic rod concepts as they relate to the field of computer graphics. Chapter 3 explores specific prior related works in computer graphics and other fields in more depth.

The first project in this thesis addresses the problem of syncing physically based animations with sound. Foley is the art of adding recorded sound effects to a film as a post-process. These techniques are traditionally used for both live-action films and animations, and typically require a significant amount of expertise and effort. However, for physically based animations in particular, there exist techniques for synthesizing accompanying audio without any input recordings. For example, modal synthesis is a framework for automatically generating sound to accompany rigid body animations. Unfortunately, the method is ill-suited for elastic rods for a number of reasons: even when made of materials such as aluminum or steel, rods tend to be flexible, which is incompatible with modal synthesis. Furthermore, long rods commonly exhibit dispersive behavior, resulting in characteristic chirping sounds; these are difficult to capture with modal synthesis. In computer music research, specialized acoustic models for rods have been developed that can account for these effects. However, these
techniques are specific to particular scenarios (e.g., a piano string struck by a single hammer), and are not suitable for general virtual environments.

In Chapter 4, we propose a method for augmenting physically based animations of elastic rods with an acoustic model. We choose an existing elastic rod framework that is designed to integrate well into arbitrary virtual environments. For each rod, we add a series of acoustic dipole sources along its length that are modulated by the rod’s behavior. The vibrations that are already resolved by the elastic rod simulator are fed to the dipoles to produce plausible sound. This allows simultaneous generation of synchronized animation and audio for a diverse array of objects and environments, such as a plucked rubber band, a chain link fence struck with a baseball, or a slinky walking down a set of stairs. This method thus expands the range of geometries for which sound synthesis methods are applicable.

We also propose a method that makes use of elastic rod simulation to improve the geometry of synthesized cloth models used in highly detailed renderings of fabric. Modern techniques for creating high quality images of virtual cloth involve modeling the light scattering behavior of the fibers that compose yarns that make up the textile. Given fiber geometry data, renderings can be generated by tracing light paths through the collection of fibers. This implies that we need to know the paths of each of the yarns in the cloth, which are subject to the physical behavior of yarn. Prior work in yarn rendering generally places yarns procedurally without taking yarn mechanics into account, resulting in an unnatural appearance (e.g., yarn bending too sharply). The state of the art is to use elastic rod simulation to “relax” synthesized yarn paths into physically consistent configurations. Individual fiber curves can then be generated
procedurally to follow these yarn curves.

This method works fairly well, but the algorithm for placing the finer-scale fiber paths does not consider the wider context of the cloth’s structure. Real yarns change shape and deform when contacting other yarns, but prior work does not account for these effects. In Chapter 5, we address this problem by making use of efficient elastic rod simulation methods to estimate the shape of yarns and inform the placement of fiber curves. We do this by splitting cylindrical representations of yarns into bundles of parallel elastic rods, which we call macro-fibers. In so doing, we are able to simulate yarn at a finer scale than in previous work, allowing us to account for the way yarns pack together. Using a GPU-based simulator and domain decomposition, we demonstrate how to handle hundreds of thousands of contact points under reasonable time and memory constraints. We then use the simulation results along with an existing fiber curve generation method to synthesize static geometry at the fiber level. We demonstrate the method’s efficacy by showing comparisons of our method to prior yarn synthesis methods, as well as comparisons of synthesized yarn renderings to photographs and volumetric scans of real cloth.

Unfortunately, the evaluation of the results of our macro-fibers technique is not particularly rigorous; the final project in this thesis aims to repair this shortcoming. The comparisons of macro-fibers to prior work and real cloth are qualitative, but quantitative measures would strengthen the method’s assessment. Additionally, simulation parameters are hand-tuned to obtain the best final results—ideally, we could determine simulation parameters by analyzing real yarns. For these tasks, we need to obtain the geometry of real cloth. 3-dimensional structure of a cloth sample can be observed from computed to-
mography (CT) scans. Such a scan provides a 3D grid with a measure of relative density at each voxel. However, extracting the yarn centerlines from these volumetric data is a challenge. Previous work has examined this problem, but these methods either require low-curvature yarn configurations or high resolution input scans.

Chapter 6 proposes a method for extracting these centerline curves from CT scans of knitted fabrics. We describe a filter to estimate fiber directions at every voxel from a density volume. We use this data to track ridges, high density curves that follow fibers, through the volume. By tracking multiple ridges simultaneously, we develop an iterative yarn tracking algorithm that is robust to yarn crossings. Our method makes no assumption on the yarn topology, and can therefore recover yarn curves from a wider array of fabric structures, including knitted patterns with high yarn curvature. Furthermore, tracking ridges allows us to robustly reconstruct yarn paths without fully resolving each individual fiber, permitting larger CT datasets as input. These reconstructed curves pave the way for validation of simulation models, calibration of simulation parameters, and the ability to reverse-engineer a textile.

Finally, Chapter 7 summarizes the insights gained from using elastic rod simulations in these contexts, and discusses directions for future research.
CHAPTER 2
BACKGROUND

2.1 Elastics

We begin by presenting a quick introduction on the theory of elastics, which is the basis on top of which elastic rods are defined. For a more thorough treatment of the subject, see, e.g., [41].

In continuum mechanics, an elastic body is assumed to consist of a continuous substance. This ignores the discrete components of a body, such as molecular and atomic interactions; however, it accurately captures the behavior of these interactions in bulk, as observed at a much larger scale.

In elastic theory, bodies are generally defined to have a “rest” or “reference” state at which the body is relaxed in the absence of external forces. A body can undergo some displacement, which can be decomposed into a sum of a rigid motion (i.e., a translation and a rotation) and a deformation (i.e., any displacement that changes the distance between any two points in the continuum.) The elastic body generates internal forces in response to the latter that push the continuum back towards a rigid transformation of its rest state.

There are two important quantities that underly the theory of elastics: strain and stress. These are multi-valued fields defined at any point inside the body. Strain is a measure of local deformation. There are many ways to quantify strain, but for general 3D solid bodies, it is often linearized and represented as a $3 \times 3$ matrix at each point; that is, it is often derived from the Jacobian of the map from points in the reference state to points in the deformed state.
Stress represents the internal forces that are exerted on the elastic body. Like strain, it is defined everywhere in the continuum, and typically represented with a $3 \times 3$ matrix. Stresses respond to strains induced by deformations to push the body towards a rigid transformation of its reference state. At any single instant in time, stress, momentum, and external forces (such as gravity or contact with another body) are in equilibrium; from these three quantities, the motion and deformation of the object can be obtained.

For simplicity, it is often assumed that a body’s stress depends only on its current strain. Moreover, it is often assumed that the two have a linear relationship; this model is known as (generalized) Hooke’s Law. Other models may use the history of the strain in stress calculations, and in particular, this is necessary to account for plasticity. In addition, other nonlinear models of elasticity have been developed, and these are useful in cases where both strain and stress are large. However, in this thesis, we use the simpler linear formulation of elasticity without plasticity. Furthermore, we assume that any stresses are within the elastic limit of the body; that is, we assume that the body does not tear or fracture in response to the observed deformations.

We define the elastic energy of the body to be a scalar quantity that is half the product of the stress and the strain integrated over the body. This is a non-negative measure of the potential energy induced by elastic deformations. Internal stresses seek to minimize this quantity given external loads.
2.2 Elastic Rods

Elastic rods are a special case of elastic bodies that are locally thin in two of their three dimensions. Given an arbitrary deformation, the theory of solid mechanics defines a strain and stress distribution over such a body, but this special geometry permits some simplification of these quantities. In particular, it is sufficient to define stresses and strains only along a one-dimensional curve that travels through the rod’s longest dimension, known as the rod’s “centerline.” Under certain assumptions about the behavior of the material, the elastic energy obtained from the centerline stresses and strains is equivalent to the elastic energy of the entire body. This greatly simplifies computations without sacrificing significant physical accuracy.

We now introduce the internal forces that appear in elastic rod frameworks. For more thorough discussions, see [4, 63, 53].

We define a curve \( f(s) : \mathbb{R} \to \mathbb{R}^3 \) with endpoints \( f(0) \) and \( f(1) \) which we will use as the centerline of our elastic rod. We restrict \( f \) to be continuous and twice differentiable everywhere. We additionally define \( f_0(s) \) as another curve that represents the rod’s rest state.

2.2.1 Stretching Forces

Stretching, or longitudinal stress, is the simplest kind of force present in an elastic rod. A rod resists longitudinal extension and compression, like a spring. The stretching strain \( \varepsilon_{\text{stretch}} \) defined at point \( f(s) \) is often represented as a single scalar which measures the local tension or compression of the rod relative to its
rest state:

\[ \varepsilon_{\text{stretch}}(s) = \frac{f'(s)}{f_0'(s)} - 1 \]

Note that this formulation is equivalent to the notion of general 3D strain introduced above, but only accounting for deformation along the tangent direction of the rod. Under Hooke’s law, the stress due to tension or compression is proportional to the strain. The scale of this relationship is determined by two factors: the rod’s cross-sectional area \( A(s) \), and the Young’s modulus \( E \), a material-dependent parameter:

\[ \sigma_{\text{stretch}}(s) = EA(s)\varepsilon_{\text{stretch}}(s) \]

This model is simple, but captures the compression and extension behavior of rods quite well. For more interesting deformations, however, we need further theory.

### 2.2.2 Bending Forces

The Euler-Bernoulli model \(^{38}\) is one of the earliest that accounts for the bending behavior of rods in a physically based manner. The model makes the following assumptions:

- The deformation of the rod’s centerline remains in plane. This ensures that any forces due to twisting moments are zero.
- Planar cross sections remain planar after the rod deforms. This implies that no shearing effects occur within the cross section plane.
- There is a linear relationship between the stress and strain, i.e., Hooke’s Law.
In this context, strain due to bending measures the curvature of the centerline, which is related to the second derivative of the curve. Let \( \ell(s) : \mathbb{R} \rightarrow \mathbb{R} \) be the function that maps the range \([0, 1]\) to the arc length parameterization of \( f \), so that \( \|f'(\ell(s))\| = 1 \) for all \( s \in [0, 1] \). Then:

\[
\varepsilon_{\text{bend}}(s) = f''(\ell(s)) - f''(\ell(0))
\]

This assumption comes from considering an infinitesimal extent of the rod, and modeling the bending force as a sum of stretching forces due to the compression and extension over each point in the cross section plane. The total amount of compression or extension that occurs within this infinitesimal extent depends on the shape of the cross section—the further the point is from the centerline, the more stretch or compression it will experience. It can be shown that the total stress is proportional to the second moment of area of the cross section with respect to the axis of curvature (\( I_{\text{curve}} \)). This has a closed analytical form for many common shapes, including circles and polygons. The stress is also proportional to how much the material resists stretching and compression; this is exactly Young’s modulus. Therefore, the stress due to bending can be calculated as:

\[
\sigma_{\text{bend}}(s) = EI_{\text{curve}}(s) \varepsilon_{\text{bend}}(s)
\]

The Euler-Bernoulli model works quite well for beams made of stiff materials under small displacements, but due to its restrictive assumptions, it is not particularly applicable outside analysis of cantilevers or other simple setups. To more accurately model the behavior of rods, we must relax the requirement that the rod’s centerline remains in plane; that is, we must allow the rod to twist.
2.2.3 Twisting Forces

Work by Kirchhoff [60] extended the theory of rods into a broader context. In particular, this model accounts for forces caused by displacements with both bending and twisting components. The two forces interact nonlinearly, though the model maintains the assumption of a linear stress-strain relationship. In order to account for twisting that occurs along the centerline, we must augment our elastic rod representation by assigning an orthonormal frame to each point along the curve. We define an orthonormal frame \([d_1(s), d_2(s), d_3(s)]\) such that \(d_3(s)\) is aligned with \(f'(s)\). This is called the reference frame. There are infinite possible choices of reference frames, but by using a Bishop frame [19], we ensure that this frame is twist-free. Additionally, we can define another orthonormal frame at each point called the material frame, which tracks the orientation of each cross-section of the rod: \([m_1(s), m_2(s), m_3(s)]\). Under the assumption that cross sections remain planar, the material frame differs from the reference frame only by a rotation of \(\theta(s)\) about \(d_3(s) \equiv m_3(s)\). We can then define the twisting strain as:

\[
\varepsilon_{\text{twist}}(s) = \theta'(s) - \theta'_0(s)
\]

The resulting stress is proportional to a material-specific constant called the shear modulus \(G\). It also depends on the cross section; a twist will shear the material in the plane of the cross section to an degree that is dependent on the distance from the centerline. This is calculated as the polar second moment of area \(I_{\text{polar}}\). Altogether, the stress due to twisting is:

\[
\sigma_{\text{twist}}(s) = GI_{\text{polar}}(s)\varepsilon_{\text{twist}}(s)
\]

By accounting for non-planar deformations, the Kirchhoff model allows a wider range of rod displacements to be simulated. However, the assumption
that planar cross sections remain planar is generally not physically correct, as observed by more modern rod models.

### 2.2.4 Cross-section Shearing

The Timoshenko [97] and Cosserat models [4, 77] both introduce cross section shearing. As a rod bends, cross sections do not generally stay perpendicular to the centerline; such shearing occurs especially as the diameter of rods grow larger. These deformations reduce the bending strain along the rod, implying the Euler-Bernoulli and Kirchhoff models overestimate the forces due to bending. There are different ways to measure the amount of shear present, but the most straightforward way is to relax the assumption that the material frame is orthonormal. In the interest of simplicity, we assume the such shearing effects are negligible in this thesis.

### 2.3 Simulation

These forces give us insight into the rod’s physical behavior. For example, given boundary conditions (e.g., constraints on $f$ and/or its derivatives at each endpoint along with external forces), we can solve for $f$ such that stress is minimized. This is done by computing the potential energy induced by the strains:

\[
E_{\text{stretch}} = \frac{1}{2} \int_0^1 EA(s) \varepsilon_{\text{stretch}}^2(s) ds
\]

\[
E_{\text{bend}} = \frac{1}{2} \int_0^1 EI_{\text{curve}}(s) \varepsilon_{\text{bend}}^2(s) ds
\]

\[
E_{\text{twist}} = \frac{1}{2} \int_0^1 GI_{\text{polar}}(s) \varepsilon_{\text{twist}}^2(s) ds
\]
\[ E_{\text{int}} = E_{\text{stretch}} + E_{\text{bend}} + E_{\text{twist}} \]

We can then formulate the problem as a minimization of \( E_{\text{int}} \) over possible curves \( f \) subject to the constraints listed. This is known as the boundary value problem.

Another application is dynamics, which is more common in computer graphics applications. In this framework, we make the centerline \( f \) and the material frame orientation \( \theta \) functions of time \( t \). At each point in time, and at each point along the rod, forces due to momentum balance internal and external forces. More concretely:

\[
\rho A(s) \frac{\partial^2 f(s, t)}{\partial t^2} = -\frac{\partial E_{\text{int}}}{\partial f(s, t)} + b(s, t) \quad \forall s \in [0, 1], t > 0
\]

\[
\rho I_{\text{polar}}(s) \frac{\partial^2 \theta(s, t)}{\partial t^2} = -\frac{\partial E_{\text{int}}}{\partial \theta(s, t)} + \alpha(s, t) \quad \forall s \in [0, 1], t > 0
\]

where \( \rho \) is the density of the material, \( b(s, t) \) encodes the external forces applied to the rod at \( f(s, t) \), and \( \alpha(s, t) \) encodes the angular external forces (torque) applied to \( f(s, t) \) with respect to the axis parallel to \( \frac{\partial f(s, t)}{\partial s} \). Given a rod configuration at time \( t = 0 \) along with initial values for the rod’s velocity and acceleration, the rod’s position may be obtained at an arbitrary time \( t_0 \) by solving the equation above for the acceleration of the elastic rod at \( t_0 \); a second-order ODE can then be solved to find \( f(s, t_0) \).

There are a few special cases where these problems can be solved analytically, but for many scenarios of interest, there is no closed-form solution. Even so, approximations can be found quite easily through techniques such as the Finite Element Method or the Finite Difference Method. The latter is well suited to elastic rods and is commonly used in many computational frameworks, including ours.
CHAPTER 3

PRIOR WORK

3.1 Elastic Rod Frameworks

In the field of computer graphics, a number of frameworks exist for simulating rods (most often ropes, strings, and hair strands) using the Kirchhoff and Cosserat theories with a focus on large displacements and contact. These methods capture the behavior of highly deformable thin structures to varying degrees of physical accuracy. They each define discretized representations of elastic rods, and explain how to estimate strain and stress along their lengths. Integrating these forces with respect to time reveals the rod’s motion through a virtual environment.

3.1.1 General Simulation Frameworks

Pai et al. [77] introduced a framework for simulating Cosserat rods to the computer graphics community. However, this initial formulation was difficult to incorporate into general virtual environments. Super-Helices [12] are primitives for simulating Kirchhoff dynamics for hair animation. In this framework, rods are discretized as $C^1$-continuous assemblies of helix segments. This allows lively simulation with relatively few degrees of freedom, but contact is more difficult to resolve. CORDE [93] is another framework for discretizing Cosserat theory, this time using a finite-element approach with line segments with a quaternion for each edge. This was one of the first frameworks in the community that integrates well with other simulation frameworks. Discrete Elastic Rods [11] is
a popular finite-difference framework for simulating Kirchhoff rod dynamics. Compared to prior work, the degrees of freedom are more compact: a rod is defined as a polyline with an angle (signifying material twist) defined at each edge. Followup work [10] improved on this framework by introducing the concept of parallel transport through time, which yielded a banded stiffness matrix, making stable integration methods more tractable.

More recent work has brought elastic rod theory into the position based dynamics framework [73]. In this setting, elastic energies are used as constraints that are enforced using Jacobi or Gauss-Siedel iterations. Umetani et al. [100] described a way to simulate Cosserat rods by explicitly tracking frames along polyline edges using “ghost points.” Later work [61] proposed an alternative formulation using quaternions to represent edge orientations. A recent publication [3] chose another discretization scheme that allows cross sections to grow and shrink to preserve volume, which is useful for muscle fiber simulation. Compared to force-based methods, these techniques are less physically accurate; however, they are very stable, and highly parallelizable, and suitable for simulation on the GPU. Another method [92] frames Cosserat theory under the projective dynamics framework [20], resulting in highly stable simulations with higher physical accuracy. However, this is done through implicit integration, which cannot be highly parallelized.

These methods focus on visual output and do not address sound synthesis. In Chapter 4, we build upon the Discrete Elastic Rods framework [11, 10] to produce audio from the dynamics of the rod. We choose this model because it fits well into many existing simulation frameworks, and because its compact structure admits efficient simulation algorithms. Other physics-based rod dynam-
ics frameworks could be used with our method of computing sound radiation. However, we caution that force-based schemes are more likely to produce realistic sound than position-based schemes, such as \cite{61}, since our physical model for emitted acoustic pressure uses estimates of the rod’s jerk, i.e., the third time derivative of position.

### 3.1.2 Yarn Simulation Frameworks

Simulating cloth at the level of individual yarns has been studied in a few different works. Generally, yarns have been simulated using elastic rod models similar to those mentioned above. Kaldor et al. \cite{55} described an elastic rod framework for simulating yarn using cubic splines. It is based on the Euler-Bernoulli model, and is not able to encode twisting motions, but it is generally sufficient for simulating garments at the level of individual yarns. Followup work \cite{56} replaced this yarn model with a framework based on Discrete Elastic Rods, which is able encode the plastic behavior of yarn. This framework was later used to relax procedurally-generated yarn patterns into physically plausible configurations using simulation to minimize the elastic energy of each yarn curve while preserving the topology of the yarns (i.e., preventing yarns from “pulling through” one another) \cite{107}. Leaf et al. \cite{64} developed a framework for interactively relaxing user-designed periodic yarn patterns. They were able to quickly simulate yarns by adapting the simple elastic rod simulator proposed by Kaldor et al. \cite{55} to run on the GPU.

An alternate line of work \cite{25,26} efficiently simulated large-scale textiles by explicitly tracking the persistent contacts between yarns as part of the simula-
tion model. This hybrid Eulerian/Lagrangian framework significantly reduced the time needed to detect and resolve contacts between yarns. However, this came at the cost of developing custom internal forces for individual yarn configurations, limiting the domain of applicable patterns.

In Chapter 5, we describe a method for simulating yarn at a fine scale. We use the framework proposed in [64], as it is computationally efficient. Efficiency is a large concern in this work, since we simultaneously simulate several rods in close contact. Though the physical model is relatively simple, it was still shown to be effective in modeling many different knitting patterns.

3.2 Sound Models

In the context of computer music, linear and nonlinear physically based rod models are used, for example, to synthesize sound from stringed instruments, pianos, and pitched percussion; see [39] and [35]. Beyond analytical models for simplified systems, more general simulation frameworks include finite difference methods [14, 15, 6], wave digital filters [13], (banded) waveguide synthesis [96, 37], modal synthesis schemes [15], and Volterra series [43], sometimes in combination; for comparative discussions of these techniques see [89, 101]. In computer music applications, visual output is unnecessary, which justifies certain simplifications; for example, when modeling pitched percussion, it is often only necessary to track displacements in a single dimension. Additionally, waveguides and similar tools, although more efficient than finite difference methods, often do not lend themselves well to integration into 3D environments. Distributed contact between pairs of vibrating structures has been
investigated in this literature, with a focus on preventing numerical instability; see, for example, [18, 23, 17]. However, friction is nearly always ignored in these studies for simplicity. Finally, many of these methods assume the sound is radiating from a naturally straight, cylindrical body (or do not model any form of radiation), while we aim to synthesize sound from more general geometries, e.g., a Slinky.

In the realm of computer animation, modal synthesis is the dominant strategy for synthesizing sound. The method has been studied for decades [95], and been has improved upon many times [102, 76, 52, 65]. In general, the vibration models of an object are calculated by discretizing a volumetric representation of the body and solving a generalized eigenvalue problem. Tetrahedral meshes of long, thin objects tend to result in large and ill-conditioned eigensystems; fortunately, modal synthesis can be performed directly on the rod’s centerline using the theory of elastic rods. However, modal synthesis is inherently linear, meaning that only small displacements are permitted before accuracy degrades. In addition, acoustic transfer calculations for quickly determining how a vibrating object’s sound waves propagate are only valid if the object does not deform significantly. To resolve contact coupling and chattering effects, adaptive modal synthesis and contact solvers have been proposed [112], but they are still inherently limited to small displacements and linear modal sound. These methods are ill-suited to elastic rods, which tend to be highly flexible, even when made of stiff materials such as aluminum and steel.

Accurately and efficiently generating sound from highly deformable bodies remains a difficult challenge. Chadwick et al. [21] investigated the simulation of nonlinear mode coupling for thin shells, and Cirio et al. [27] demonstrated how
to efficiently simulate the turbulent behavior of thin shell vibrations, though the displacements in these schemes are still assumed to be small for the purposes of calculating radiation. Sound produced by crumpling of shells and cloth has been approximated using specialized data-driven techniques [2, 83], possibly with spatially localized use of piecewise rigid modal models [24]. In contrast, our large-displacement dynamic and radiation rod models can be derived from first physical principles. O’Brien et al. [75] presented a method for synthesizing sound from time-domain deformable simulations. Like us, they observe the vibrational signal of elements along the surface of the body in the time domain. A ray-based sound propagation model was used which supports general surface motions, but is not consistent with the acoustic wave equation. In contrast, we are able to derive a consistent radiation model by exploiting acoustic compactness and other properties of slender rods. A recent method [103] describes how to derive sound from vibrating elastic bodies using the wave equation, but it is computationally expensive, and its spatial discretization makes it unsuitable for the scale of rods we would like to simulate (e.g., up to 20 meters long).

Besides what is mentioned above, there is other work investigating sounds generated from slender structures; for instance, Coleman and Dill [28] and McMillen and Goriely [70] describe (planar) wave characteristics of Kirchhoff rods; Bilbao [16] explores modeling reverberation from spring structures; and Dobashi et al. [33] model sound generated by aerodynamic vortices shed by fast-moving rods. In engineering, Akay et al. [1] devised methods for simulating the acoustics of beam structures. Our method is similar to theirs in spirit, as they also time-step vibrations and use a dipole model to estimate the resulting acoustic pressure. However, their radiation model emits the dipole pattern with uniform amplitude in response to any motion in the cross section plane, whereas
we analyze the cross section to calculate the directionally varying acoustic response in this plane. More generally, we have diverging goals: Akay et al. focused on estimating sound from a single bar impulse in a controlled environment, whereas we address audiovisual simulation of large displacements and complex collisions.

3.3 Yarn Geometry and Rendering

Rendering cloth is a common goal in computer graphics, and several works have explored how to generate realistic images of fabrics. Early cloth rendering models focused on 2D sheet representations of textiles with shading models that took yarn geometry into account (e.g., [49, 82]), but these have given way to more detailed cloth representations that can reproduce effects such as the fuzzy silhouettes of fabrics. Many works treat cloth as a 3D volume [50, 109, 110, 108], while others represent cloth as a collection of fiber curves [85, 58, 59]. Fiber-level rendering techniques such as these often place yarns procedurally in a cloth, and do not take into account the physics of the yarn. The simulation methods listed above alleviate this problem by leveraging yarn-scale physics, but they do not account for fiber interaction.

A recent publication [72] describes a method for modeling yarn deformation in cloth. The authors simulate yarn contacts at the fiber level using a computationally expensive material point method. In order to apply the model quickly across larger cloth samples, they use the results of offline, fine-scale simulations to train a neural network, which learns an affine transformation of a yarn’s cross section as a function of yarn-level state. We share a common goal: we would like
to reproduce the geometric changes of yarn in the context of textiles. However, by using a machine learning model and assuming a simplified affine deformation model, this prior work prioritizes speed over accuracy. On the other hand, in our work, we are willing to use more computational resources to obtain results that are more informed by yarn mechanics in the context of cloth. While full-scale fiber-level simulations remain infeasible, our method uses \textit{in situ} simulation to obtain rich, nonlinear yarn deformations.

Researchers in the fields of fiber science and composite structures have explored simulation models for yarn as well. In these contexts, properties such as yarn stress under various external conditions are sought. The Digital Element Method \cite{104} was introduced as a way to represent yarn behavior in woven and braided structures. The authors treated yarns as mass-spring chains and provided a contact model for yarn interactions. Several other works followed up on this idea; one chain of publications in particular, \cite{71, 48, 36, 34} (and others), is the inspiration for our method. In these works, yarns are subdivided into individual digital element models; the yarn cross sections are thus allowed to take on highly eccentric configurations, which is important for simulating 3D woven structures under high loads. The simulation output is then used to build a mesh that is used with Finite Element Method codes. We build on this work by showing how to apply the technique to knitted structures (which are poor fits for the mass-and-spring model used by the Digital Element Method due to regions of high curvature) and multi-ply yarn geometry, and how to use the result to instantiate individual fiber curves in a visually plausible way.

To fit larger-scale simulations into GPU memory, we use a domain decomposition technique. Domain decomposition methods have been applied across
a wide range of areas in graphics. Debunne et al. [30] used domain decomposition methods to support adaptive real time soft body simulations, and Huang et al. [47] used decomposition to handle large deformations of flexible objects. Liang and Lin [66] recently decomposed mesh-based cloth simulations in time, rather than space. Golas et al. [40] decomposed fluid simulations to handle large problem sizes. We demonstrate a domain decomposition tailored to yarn-level cloth simulations to enable large-scale relaxations of highly detailed macro-fiber-level cloth.

3.4 Analyzing Volumetric Images

The problem of extracting geometry from volumetric scans is well studied in the field of medical imaging. Much of this work focuses on reconstructing surfaces in a way that is robust to noise. The techniques most closely related to yarn curve tracking are methods for reconstructing blood vessel curves and topologies. Skeletonization, or finding medial axes of high density areas of the volume, is one of the most common approaches. A number of such algorithms exist, e.g., [68, 67, 78]; see [91] for a comparison. However, the technique admits branching topologies (and sometimes planar surfaces) that are invalid for intertwined yarns: areas where separate yarns come into contact are often treated as a single entity that branches when the yarns separate. While this may be useful for examining yarn topologies, it does not solve the problem of tracking yarn centerlines.

*Tractography* is a line of research focusing on visualizing neural pathways. A prominent method in this literature is diffusion tensor imaging (DTI). Diffu-
sion MRI data is used to track the speed of water diffusion between tissues; the flow gradient at each voxel can be characterized by a tensor whose principal eigenvector corresponds to the direction of principal flow. Integrating spatially along these directions results in flow lines that are used to visualize connections in neural matter [7]. However, this method effectively averages all flow directions seen in the neighborhood of the voxel, which is a poor approximation when multiple nearby pathways travel in different directions. This has led to the development of a number of alternative formulations, where flows at each voxels are modeled as a set of—or as a distribution over—many directions. With this data, flowlines are guided using either deterministic or probabilistic means to choose the most likely direction at each voxel. There is a significant amount of literature in this area, e.g., [99, 9, 105, 98]; Jbabdi and Johansen-Berg [54] summarize many of these approaches and discuss their pitfalls. It is likely that modern tractography techniques would work well for reconstructing yarn pathways in volumetric cloth, since ambiguous directions at yarn crossings is a common pitfall. However, no literature has explored this to our knowledge.

Another class of work focuses on identifying the paths of 1-dimensional strands in volumetric images. Zhao et al. [110] propose a method for tracking yarns through CT scans of woven cloth. The authors iteratively sweep along each yarn’s centerline, alternating between estimating the direction of the yarn, moving forward in space a small amount, and recentering based on the observed densities. However, the algorithm steps through the volume slice-by-slice, and is ineffective at tracking yarns that travel in arbitrary directions, as with knitted structures. Jakob et al. [51] track individual hair fibers using a similar prediction-correction approach, but this scenario is easier to solve than in scans of yarns; hairs are generally well separated, which reduces ambiguity
compared to woven or knitted topologies.

In order to maintain yarn continuity in the presence of contacts, we track the paths of the fibers that make up a yarn. This idea can be attributed to Shinohara et al. [87]. Their algorithm tracks individual fiber strands through the CT volume, and uses them to update the centerline position and direction of the yarn. This method is successful when tested against yarn crossings, non-circular yarn cross sections, and multi-ply yarns. Khungurn et al. [58] discusses another way to track individual fibers through density scans. The algorithm identifies high-density blobs in planar slices of the volume, and then solves a matching problem between consecutive slices to reconstruct fiber paths. However, in these two algorithms, the radius of the fibers must be known \textit{a priori}, and the scan must be of sufficient resolution to distinguish individual fibers in the majority of cases.

Unlike yarn-level trackers, our method is able to track arbitrary yarn configurations, even in regions where yarns are in close contact. In contrast to previous fiber-scale trackers, our method can still successfully track fiber paths even when individual fibers cannot be fully distinguished and are smaller than one voxel in width. This allows us to track yarns through larger regions than prior work and to capture meso-scale structure without depending on higher resolution CT images.
CHAPTER 4

ANIMATING ELASTIC RODS WITH SOUND

Sound is a pervasive part of our world. For virtual environments to be believable, the sounds of virtual objects must be accurately reproduced. In the field of computer graphics, the standard framework to compute the sound generated by solid objects is modal synthesis. These methods involve calculating the frequencies generated by the vibrating surface of the object, and computing the acoustic radiation caused by each frequency band. Modal models are popular because of their computational speed at runtime, their physically based origins, and their plausibility when simulating sound for near-rigid objects.

However, very thin rod-like structures are typically poor candidates for linear modal sound synthesis. Thin objects tend to undergo large displacements and change shape throughout the course of a simulation. This leads to vibration modes that change frequency over time, confounding precomputation, and breaks the assumption of near-rigid motion often used when computing acoustic radiation. For these reasons, simulating acoustic emissions from highly deformable bodies with arbitrary geometry is notoriously expensive. Within the standard linear modal synthesis pipeline, correctly calculating eigenmodes of thin objects requires a fine internal tetrahedral mesh, which is a memory-inefficient representation. Furthermore, dispersion, or vibrational waves traveling through the object at different speeds, can cause chirp-like sounds (like the iconic “blaster” sound) that are characteristic of long, thin objects such as ropes and cables. While it is possible to model this behavior using modal synthesis, it is much more elegantly captured with time-domain methods. Moreover, a good portion of the sound produced by animations of thin objects can be at-
Figure 4.1: **Audiovisual simulation of a spring toy slinking down the stairs:** Our method efficiently generates the spatialized sound of this discrete elastic rod (3888 control points) arising from tens of thousands of complex collision events each timestep.

tributed to acceleration noise [22], which requires additional preprocessing and computation in the modal setting.

All these effects could be computed by a full finite element method simulation that tracks oscillations of the entire body in the time domain using a non-linear internal force model that allows for large displacements. However, this approach also requires an internal mesh representation, is costly at runtime, and still requires a way to calculate sound radiation from the body; taken together, these could potentially increase memory and computation time requirements by an order of magnitude over modal synthesis.

We propose a method based on Kirchhoff rods to generate physically based sound for thin structures, which is memory efficient and captures changing modal frequencies and dispersion effects. Precomputation is orders of magnitude faster than modal synthesis, and in special cases (such as rods with a circular cross section) can be avoided altogether. Acceleration noise arises naturally from our method and does not require additional computation. Most significantly, our approach efficiently produces plausible sound for highly deformable objects that would otherwise be very costly to simulate.
We achieve these properties by making particular assumptions about the geometry of our object that allow us to simplify both the dynamics and the resulting acoustic radiation. By restricting ourselves to thin structures, we may use a finite-difference approach to simulate the object using relatively few degrees of freedom; we then use a dipole array model to simulate sound radiation due to the object’s motion. Our primary contribution is to demonstrate the usefulness of these techniques over a broad range of situations and how they can be integrated into the kinds of 3D environments encountered in many graphics applications.

4.1 Rod Dynamics Model

The core of our simulator is a model for dynamics of rods that is integrated through time to produce motion that drives the acoustic radiation model of §4.2. The degrees of freedom of our body are the positions of points along the centerline and twists that accumulate between them. Dynamics is then modeled based on Kirchhoff theory, specifically the Discrete Elastic Rods model proposed in [11] and extended in [10]. We have chosen this framework because of its efficiency and widespread use in graphics, and because it is a natural fit for sound synthesis.

4.1.1 Notation

We will follow the notation introduced in [10]. A rod is defined by \( n + 2 \) control points connected sequentially by \( n + 1 \) edges; together these define the centerline of the rod. In the following, values associated with control points are
Figure 4.2: Notation used within the definition of the rod model.

indexed using subscripts, and values associated with edges are indexed using superscripts. Let \( x_i \in \mathbb{R}^3 \) be the position of control point \( i \), and let \( \theta^j \) denote the twist of edge \( j \). Let \( q = (x_0^T, \theta^0, \ldots, x_n^T, \theta^n, x_{n+1}^T)^T \in \mathbb{R}^N \) be the generalized coordinates of the rod, where \( N \), the total number of degrees of freedom, is equal to \( 3(n + 2) + (n + 1) \).

For each edge \( e^j = x_{j+1} - x_j \) we define a frame, \([d^j_1, d^j_2, d^j_3]\) \( \in SO(3) \) (where \( d^j_3 = e^j/\|e^j\| \)), called the reference frame. These frames are updated using parallel transport through time (as described in [10] §3) whenever \( q \) is updated. As part of this process, we record the twist \( m_i \) accumulated between each pair of reference frames \( i - 1 \) and \( i \). We then define a material frame \([d^j_1, d^j_2, d^j_3]\) \( \in SO(3) \) at each edge \( e^j \) as the rotation of the reference frame around \( d^j_3 \) by the angle \( \theta^j \). See Figure 4.2.

In addition to these definitions given by prior work, we require an estimate of the material frame at each of the control points in §4.2. Let \( R_i = \)
\([d_{i,1}, d_{i,2}, d_{i,3}] \in SO(3)\) denote this frame at control point \(i\). For the first and last control points, we simply use the material frame of the single neighboring edge; that is, \(R_0 = [d_1^0, d_2^0, d_3^0]\) and \(R_{n+1} = [d_1^n, d_2^n, d_3^n]\). For internal control points, we set 
\[
d_{i,3} = \frac{(d_{i−1}^3 + d_i^3)}{\|d_{i−1} + d_i\|}.
\]
We then parallel transport \(d_{i−1}^1\) and \(d_{i−1}^2\) through space to obtain \(d_{i,1}\) and \(d_{i,2}\), which represent the approximate reference frame at control point \(i\) and are orthogonal to \(d_{i,3}\). We then rotate this frame about \(d_{i,3}\) by the angle \((\theta_{i−1} + \theta_i + m_i)/2\) to obtain \(d_{i,1}\) and \(d_{i,2}\).

### 4.1.2 Dynamics

To introduce dynamics, we make \(q\) a function of time, i.e., \(q = q(t)\). Potential energies due to bending, twisting, and stretching (denoted \(E_b\), \(E_t\), and \(E_s\) respectively) are non-negative scalars defined as functions of these degrees of freedom; see [10] §4 for their definitions. From these values, we derive the internal forces \(f_{\text{int}}(q) \in \mathbb{R}^N\):

\[
f_b(q) = -\frac{\partial E_b}{\partial q} \quad f_t(q) = -\frac{\partial E_t}{\partial q} \quad f_s(q) = -\frac{\partial E_s}{\partial q}
\]

\[
f_{\text{int}}(q) = f_b(q) + f_t(q) + f_s(q)
\]

From this, we can calculate the stiffness matrix of the structure \(K(q) \in \mathbb{R}^{N \times N}\):

\[
K(q) = \frac{\partial f_{\text{int}}(q)}{\partial q}
\]

We also use a diagonally lumped mass matrix \(M \in \mathbb{R}^{N \times N}\) that remains constant throughout the simulation. Furthermore, we introduce a damping matrix \(C \in \mathbb{R}^{N \times N}\) and a velocity-proportional force \(-Cq\) that provides artist-tunable damping to the simulation. As is common in sound synthesis applications, we choose to use a Rayleigh damping matrix [86], i.e., \(C(q) = \alpha M + \beta K(q)\) for
material- and geometry-dependent scalars $\alpha$ and $\beta$. Furthermore, we gather any external forces, such as gravity, into a vector $f_{\text{ext}}(q, \dot{q}) \in \mathbb{R}^N$. This gives rise to the following equations of motion:

$$M\ddot{q} + C(q)\dot{q} - f_{\text{int}}(q) = f_{\text{ext}}(q, \dot{q})$$

Various techniques can be used to integrate the rod through time. We prefer to use an implicit integration scheme for stability when integrating stiff materials such as metals. Backward Euler, the usual algorithm for implicit integration in graphics applications, introduces artificial damping; while this may be acceptable for purely visual simulations, this leads to unacceptably muted sound in our application. For this reason, we use the Newmark-\(\beta\) algorithm with the trapezoidal rule (i.e., $\beta = \frac{1}{4}$ and $\gamma = \frac{1}{2}$) [74], which preserves long-term oscillatory behavior. This requires Newton iteration to solve for the generalized accelerations $\ddot{q}(t_{k+1})$ given the state from the previous step $t_k$, from which we can deduce $\dot{q}(t_{k+1})$ and $q(t_{k+1})$. We assume $C$ is constant throughout the timestep, which allows us to elide taking its derivative with respect to $q(t_{k+1})$, which would otherwise require calculating a third-order tensor multiple times per timestep. We have not observed any audible artifacts or convergence problems attributed to this decision.

### 4.1.3 Caveats

The resulting sound is derived from small-amplitude, high-frequency vibrations arising from the rod model described above, and we can only expect accurate sound if the model adequately simulates the physics of the body. As a result, rods with large or highly eccentric cross sections may not be a good fit for
our method, as they may exhibit audible modes and nonlinear effects that may require a full 3D solid or sheet model to properly resolve. In addition, for Kirchhoff rods, the effects of shearing across the cross section must be negligible—or artificial stiffening (higher frequencies) will occur—which implies geometrically that the cross-sectional diameter be small relative to rod length (which is generally true for our examples). More expensive Cosserat rod models could be used to account for shearing, and there is no particular reason that such a model could not be used within our sound radiation framework.

4.2 Sound Radiation Model

To turn the motions computed using the rod model into sound, we need a model suitable for efficiently estimating the acoustic radiation associated with rapid vibrations due to bending waves, as opposed to typically less audible stretching, twisting, and shearing motions. We now introduce a dipole radiation model associated with a small vibrating rod element, then discuss implementation details and related assumptions of the model.

4.2.1 Dipole Sound Field of a Vibrating Rod Element

Introduction

We approximate the sound field radiating from a rod as a linear superposition of contributions due to the motions of small segments along its length. We begin by considering a small rod element located at the origin $r = 0$, oriented along
the z-axis, with interior $\Omega$ and surface $S = \partial \Omega$ (see Figure 4.3). We assume this segment is moving rigidly, and denote its linear and angular velocities as $v(t)$ and $\omega(t)$ respectively.

![Figure 4.3: (Left) A vibrating rod element depicted as part of a longer extruded rod. (Right) Cross-sectional geometry and notation.](image)

Following the theory of linear acoustics, we seek to approximate the acoustic pressure field $p(r, t)$, which satisfies the wave equation outside the body,

$$
\frac{\partial^2 p(r, t)}{\partial t^2} = c^2 \nabla^2 p(r, t), \quad r \in \mathbb{R}^3 \setminus \Omega,
$$

with suitable boundary conditions, where $c$ is the speed of sound in the surrounding medium. While this can be computationally involved for large bodies, we use the derivation in [44] to approximate the far-field acoustic pressure from a compact, rigidly vibrating object as a dipole source:

$$
p(r, t) = \frac{\rho}{4\pi c r} \hat{r}^\top \frac{\partial^2}{\partial t^2} \int_S (y - \varphi^*(y)) \, v_n(y, \tau) \, dS_y
$$

where $r = ||r||$, $\hat{r} = r/||r||$, and $v_n(y, \tau)$ is the surface normal velocity at point $y \in \partial \Omega$ at retarded time $\tau = t - r/c$. Here $\varphi^*(r) \in \mathbb{R}^3$ is a time-independent harmonic
function called the \textit{velocity potential}, that satisfies

\begin{align*}
\nabla^2 \varphi^\ast (r) &= 0, \quad r \in \mathbb{R}^3 \setminus \Omega, \quad (4.3) \\
\varphi^\ast (r) &\rightarrow 0, \quad \text{as } ||r|| \rightarrow \infty, \quad (4.4) \\
\frac{\partial \varphi^\ast (r)}{\partial n} &= n, \quad r \in S \quad (4.5)
\end{align*}

where \( n(y) \) is the unit normal on \( S \).

The surface normal velocity (assuming the origin and the rod element’s centroid coincide) is

\begin{equation}
v_n(y, \tau) = n(y) \cdot (v(\tau) - y \times \omega(\tau)) \equiv N(y) \begin{bmatrix} v(\tau) \\ \omega(\tau) \end{bmatrix}
\end{equation}

where \( N(y) = (n_{yx,n}) \in \mathbb{R}^6 \). Substituting (4.6) into (4.2), spatial integration can be performed independent of time:

\begin{equation}
p(r, t) = \frac{\rho}{4\pi c} r \cdot \hat{r} \left[ \int_S (y - \varphi^\ast (y)) \cdot N(y) N(y)^\top dS_y \right] \frac{d^2}{dt^2} \begin{bmatrix} v(\tau) \\ \omega(\tau) \end{bmatrix}
\end{equation}

where \( D^v \) and \( D^\omega \) are \( 3 \times 3 \) matrices that depend only on the shape of the element.

Note that we have assumed a slow-moving listener, i.e., \( \dot{\tau} \approx 1 \).

\textbf{Symmetry and rotations}

Due to the properties of \( \varphi^\ast (y) \), it can be shown that for principal rotations about the centroid of the rod element, all entries of \( D^\omega \) vanish for elements with \( xyz \) mirror symmetries.\footnote{See Appendix A for a proof. A nonzero angular contribution can still arise as the element rotates about the \( z \)-axis, but it must be considered at quadrupole order, which for most cross sections is small in comparison to dipole radiation from bending.} Without loss of generality, we will consider rods with sym-
metric cross sections that only change shape smoothly along the rod, and therefore we can use the simpler pure-translational dipole model \[44\],

\[
p(r, t) = \frac{\rho}{4\pi c r} \hat{r}^T D \ddot{\psi}(\tau)
\]  

(4.8)

where \( D \) (previously \( D' \)) characterizes the dipole radiation fields,

\[
D = \oint_S (y - \varphi^i(y)) n(y)^T dS_y.
\]  

(4.9)

Reduction to 2D

If we assume elements are aligned end-to-end along the \( z \)-axis such that “end cap” contributions cancel with their adjacent neighboring elements, we can ignore the parts of \( S \) that are not in contact with the surrounding medium when computing \( D \). Furthermore, since we have assumed that the cross-sectional shape varies slowly along the rod, we note that \( n(y)^T \hat{z} \approx 0 \) on the rest of \( S \), where \( \hat{z} \) is the unit vector along the \( z \)-axis. Consequently \( D \hat{z} = 0 \), which is to say that no dipole source is generated by \( z \)-axis motion for this model. Therefore we can ignore element oscillations in the \( z \) direction and consider only the \( 2 \times 2 \) top-left submatrix of \( D \). Piecewise constant \( z \)-integration can be done for a rod element of length \( \Delta z \), giving

\[
D_{ij} \approx \Delta z \oint_{\Gamma_i} (y_i - \varphi^i_j(y)) n_j(y) d\Gamma_y
\]  

(4.10)

for \( i, j \in \{1, 2\} \), where \( \Gamma \) is the 2D curve defining the element’s cross-sectional boundary. Therefore we only need to determine the boundary values of two \( \varphi^*_i \) velocity potentials corresponding to 2D exterior Laplace problems in the \( xy \)-plane. We now explain in more detail how to compute the \( D \) surface integral in (4.10).
4.2.2 Implementation Details

Boundary Element Method estimation of $D$

To calculate $D$ for a single element, we must first calculate the $x$ and $y$ components of $\varphi^*$. This requires solving a pair of 2D exterior Laplace problems with Neumann boundary conditions on $\Gamma$ (i.e., a 2D version of (4.3), (4.4), and (4.5)). We discretize $\Gamma$ into a set of line segments $\tilde{\Gamma}$, and then use the 2D Boundary Element Method (BEM) to estimate the $x$ and $y$ components of $\varphi^*$ on $\Gamma$; in our implementation we use the direct BEM collocation method (see [79] for formulae). Note that this involves solving a dense, nonsymmetric linear system of equations of size $|\tilde{\Gamma}|$. Matrix entries are computed by integrating over each line segment in $\tilde{\Gamma}$ with respect to the midpoint of each other member of $\tilde{\Gamma}$. The integrals have analytical solutions, but their expressions become singular as the line segment and midpoint approach collinearity. To avoid this instability, we simply approximate each integral using Simpson’s rule with sufficiently many samples. In practice, we define a maximum length for each of our line segments in $\tilde{\Gamma}$; in our framework, this maximum length is 50 µm, which provides a good tradeoff between speed and accuracy. We then calculate the integrand $(y - \varphi^*(y))n(y)^T$ piecewise over $\tilde{\Gamma}$ and sum the results, multiplying by $\Delta z$, to determine the $2 \times 2$ top-left submatrix of $D$.

Rods with circular cross sections are common, and in such cases $D$ is known analytically, eliminating the need for BEM calculations:

$$D = 2\pi a^2 \Delta z \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad (4.11)$$
Figure 4.4: **2D dipole radiation fields for different cross sections:** Acoustic pressure values are calculated in several directions in the \(xy\)-plane for a few examples of 2D cross sections in response to jerk motion in the vertical direction. Such effects are encoded in the \(D\) dipole matrix. The length of each arrow represents the magnitude of the acoustic pressure; orange represents positive values and blue negative.

where \(a\) is the rod’s radius.

**Preprocessing rod elements**

We replicate rod elements for each of our control points along the length of a rod. We define a cross-sectional shape for each control point \(i\), from which we precompute \(D_i\) as described above. (For extruded rods, i.e., rods with cross sections that are constant along their length, we compute \(D\) once and reuse it for every control point.) Because these calculations require an integral along the \(z\) dimension of the element (our \(\Delta z\) length), we assume that the cross section is nearly constant along the element, and multiply the result of the integral over the 2D cross section by the Voronoi length of control point \(i\) to compute \(D_i\).

Bergou et al. [10] define the Voronoi length as \(\bar{l}_i = (\|\bar{e}_i\|^2 + \|\bar{e}_i\|)/2\) where overlines denote quantities computed on the undeformed configuration of the rod; we
extend this definition to the first and last control points by letting $l_0 = \|\vec{e}_i\|$ and $l_{n+2} = \|\vec{e}_i^{n+1}\|$.

Transforming and accumulating element contributions

In practice, the rod element is not always aligned with the z-axis, and is not located at the origin, both during preprocessing and during runtime animation. Therefore, we must transform the equations above to be used within our framework. Let $r_i(t)$ represent the vector from the $i^{th}$ control point, $x_i(t)$, to the position of a listener at a given instant in time $t$. Let $R_i(t)$ approximate the rod’s material frame at control point $i$, as defined in §4.1.1. Taking this transformation into account, the acoustic pressure due to vibrations of element $i$ at time $t$ contributes sound at the listener’s location slightly later:

$$p_i \left(t + \frac{r_i(t)}{c}\right) = \frac{\rho}{4\pi c r_i(t)} \hat{r}_i(t)^\top R_i(t) D_i R_i(t)^\top \ddot{x}_i(t).$$ (4.12)

Using the value of $\ddot{x}_i$ that is computed at each timestep within our integrator, we approximate $\ddot{x}_i$ at each timestep using a simple finite difference calculation: $\ddot{x}_i(t_k + \Delta t/2) \approx (x_i(t_{k+1}) - x_i(t_k))/\Delta t$, where $\Delta t = t_{k+1} - t_k$ is the length of the timestep. (4.12) is computed for each rod element at each timestep and summed into a single array of audio samples. Array entries are associated with discrete timestamps, but (4.12) requires us to record pressure samples at arbitrary times. To accommodate this, acoustic pressure values computed in (4.12) are linearly interpolated between their two nearest entries in time within the sample array.

Note that we require $\Delta t \leq 1/40000$ (sec) to accurately generate audio for the full audible range (up to 20kHz), though larger timesteps may be used at the price of degrading the audio quality. Unless otherwise specified, we choose a sample...
rate and timestep size of 44.1kHz for all of our simulations. For simulations with a different timestep, we record audio samples at the timestep rate, and resample the audio to 44.1kHz after the simulation has completed. Resampling is done using FFmpeg [31], which filters the audio using a Kaiser window with $\beta = 0.9$.

### 4.2.3 Caveats

Several assumptions were made when deriving the dipole-based sound model (4.12), which introduce particular limitations:

- **Acoustic compactness** of the rod cross section was assumed by (4.2) and implies that the rod diameter $d$ is much smaller than the wavelengths of interest, i.e., $d \ll \lambda$. Since $\lambda = c/f$ for frequency $f$ and $c = 340\,\text{m/s}$, for a dominant frequency of 10kHz, this implies that $d \ll 3.4\,\text{cm}$, which is true for our examples. Fatter rods may also require more sophisticated vibrational analyses.

- **Far-field listening positions** were assumed by the dipole radiation model (4.2), and require that the distance to the listener $r$ satisfy $r \gg \lambda$, i.e., be much larger than wavelengths of interest. For typical low-frequency wavelengths of 100Hz–1kHz, this implies minimum distances of 3.4m–0.34m, which is reasonable for computer animation.

- **Cross-sectional symmetry** was assumed for the simplified pure-translational model in (4.8) and is true of most of our examples. More exotic cross sections can be accommodated by including additional angular terms, i.e., $D^\omega$. 

38
• **Smoothness (slowly varying cross section)** was assumed to ignore radiation from z-aligned vibrations. While these contributions are usually negligible, bumpy or more exotic 3D rod shapes may require more detailed analysis to recover these contributions.

### 4.3 Contact

In our framework, we use contact events within the simulation to drive the audible vibrations of the rod. We’ve found that impulse methods are best suited for our application. Many modern sound synthesis frameworks adopt this solution as well; see, for example, [112] and [65]. A number of contact solvers specific to thin structures exist, such as [94] and [57]. In our case, we choose the So-Bogus solver [29], as it is designed to be robust and scalable. Because our algorithm requires resolving collision at the audio sample rate (if not faster), quickly resolving multi-contact problems is critical to the efficiency of our framework. The So-Bogus algorithm solves local problems in parallel using Newton iteration, and solves the global problem using Gauss-Seidel iteration. In the case that the global convergence threshold is not reached, So-Bogus generally still produces a reasonable impulse response.

In many sound synthesis frameworks, contact stiffness is typically modeled using springs, as with Hertz contact theory. However, in the case of thin rods and hard surfaces, these springs become exceptionally stiff; for some of our examples, it is necessary to reduce the timestep size by an order of magnitude to maintain stability. Therefore, we use the So-Bogus solver, a discrete impulse-based method, even though it does not provide parameters for tuning contact
Nonrigid damping: (Top) A 4-ringed Slinky approaching resting contact with the ground plane. (Left) Without damping, we observe buzzing artifacts. (Right) With post-contact nonrigid damping applied, the artifacts are removed, and resting contact is achieved.

Iterative methods such as So-Bogus are prone to buzzing artifacts, as they may not find the exact solution and are not temporally coherent. Such artifacts are perceptible when multiple rods are in resting contact with another. We remove excess energy from bodies coming to rest by applying an impulse counteracting nonrigid motion after contact has been resolved [73]. This damping impulse is weighted higher when the norm of the velocity of the rod is low and vanishes when the norm is higher than a user-specified threshold. The impulse is applied only to the spatial degrees of freedom, except in the special case of straight rods, where it is necessary to account for the twist degrees of freedom to avoid a singular matrix solve when calculating the rigid angular velocity.

During collision detection, we treat each edge of the rod as a rigid body.
We place each edge’s collision detection primitive into a bounding volume hierarchy that is updated each timestep to cull the majority of collision pair candidates. Primitive collision detection is performed using the ODE library [90]. Like [112], we require position-level contact response—that is, we generate contacts whenever we detect interpenetrating geometries, ignoring the relative velocity of the two bodies. This is necessary to achieve resting contact without audible jittering.

Another problem we encounter related to contact resolution is integrator instability. Although Newmark-β has excellent energy conservation properties, it is known to have stability problems in the presence of hard constraints and contacts. In our examples, this resulted in spurious signals at the Nyquist frequency around contact events, and very occasionally some numerical instability from which Newmark-β could not recover. A number of modifications to the method have been proposed that reduce this instability, usually by damping high-frequency oscillations (e.g., [8]). However, we are not aware of an integration method that is approximately as efficient as Newmark-β and doesn’t introduce audible numerical damping. Instead, we apply a sixth-order Butterworth low-pass filter with a cutoff of 19.8kHz to the resulting audio output to manually remove oscillations at the Nyquist frequency. In the case of unrecoverable instabilities, we simply reduce the timestep by half (which results in approximately the same amount of overall computation as Bathe’s integrator using the original timestep). All of our examples are stable when simulated at 88.2kHz.

Finally, we summarize the algorithm in Figure 4.6.
\( q(t_{k+1}), \dot{q}(t_{k+1}), \ddot{q}(t_{k+1}) \leftarrow \text{newmark}(q(t_k), \dot{q}(t_k), \ddot{q}(t_k)) \)

\( C \leftarrow \text{detectContacts}(q(t_{k+1})) \)

\( j \leftarrow \text{calculateContactImpulse}(C, q(t_{k+1}), \dot{q}(t_{k+1})) \)

\( q(t_{k+1}), \dot{q}(t_{k+1}), \ddot{q}(t_{k+1}) \leftarrow \text{applyImpulse}(j) \)

if \( \|\dot{q}(t_{k+1})\| < \text{dampThreshold} \) then

\( j \leftarrow \text{calculateNonrigidDampingImpulse}(\dot{q}(t_{k+1})) \)

\( q(t_{k+1}), \dot{q}(t_{k+1}), \ddot{q}(t_{k+1}) \leftarrow \text{applyImpulse}(j) \)

\( \ddot{q}(t_k + \frac{\Delta t}{2}) \leftarrow (\ddot{q}(t_{k+1}) - \ddot{q}(t_k))/\Delta t \)

for all Listener \( \ell \) do

\( \text{recordSample}(\ell, \ddot{q}(t_k + \frac{\Delta t}{2})) \)

Figure 4.6: **Timestep loop:** This pseudocode describes one iteration of our simulation, taking in the state at time \( t_k \) and advancing to time \( t_{k+1} \).

### 4.4 Results

Our simulator is written in C++. It is multithreaded, and the dynamics of each rod is computed independently. The contact detection step is single-threaded, though we take advantage of OpenMP to solve local contact problems in parallel within the So-Bogus solver. We found that repeated computation of the stiffness matrix becomes the bottleneck in these examples, though contact resolution becomes comparatively expensive in particularly dense cases such as *Slinky Walk*. Precomputation of the \( D \) matrix involves at worst taking an \( LU \) decomposition over a dense matrix with about \( 1000^2 \) entries, and completes in less than a second for all of our examples. See Table 4.1 for performance results.

In examples that include a ground plane, we model sound wave reflection against this plane by recording the acoustic pressure for each pair of rods and listeners twice; once normally, and once with the listener’s position replaced with its reflection across the ground plane. This second signal may be attenuated to simulate absorption; in all of our examples, we multiply the reflected signal by 0.96.
Table 4.1: **Performance**: For each example, we state the number of control points present in the simulation, the material parameters used (Young’s modulus $E$ and Poisson’s ratio $\nu$), the total wall clock time to generate one second of sound (averaged over the entire simulation), as well as the portion of time spent on rod dynamics simulation, collision detection and resolution, and sound computation. All timing data was measured on an 8-core Intel Xeon E5-2637 v3 CPU.

### 4.4.1 Examples

We now enumerate a number of examples that demonstrate the range of sounds we can simulate using our method. Please see the accompanying video [2] to view and hear the full results.

**Rectangular Rod**: An aluminum rod 1 foot in length with a rectangular cross section (1 inch by 0.5 inches) is struck on two of its sides, producing two distinct tones resulting from its anisotropic cross section. In our simulation, we model this impulse using a half-sine force pulse applied to a single control point over a short period. We are able to match the lower tone well, but our simulation overestimates the pitch of the higher tone when struck on its thinner side; the fundamental pitch is raised by about 2% compared to the recording. We attribute this observation to the lack of cross section shearing in our dynamics model, which increases the effective bending stiffness, especially in short and thick rods such as this one. Nevertheless, the metallic character of the recording

---

is well captured by our simulation.

We compare the frequencies of the recording and the simulated audio in Figure 4.7. We also compare these against the solutions to the ideal bar equation, which gives the approximate frequencies of bending modes of a straight rod with uniform cross section. Though we do not match the recorded data perfectly, we approximate it very well for the most perceptually significant frequencies (i.e., between 200Hz and 8000Hz), and it is clear that we predict the modes more accurately than the linear solution given by the ideal bar equation.

Circular Rod: An aluminum rod is dropped on a concrete floor. The 2 foot long rod has a circular cross section with a radius of 0.25 inches. The contact stiffness issue mentioned in §4.3 results in louder high-frequency content than the corresponding recording; however, the frequencies of the bending modes match well. In addition, the rod’s chattering behavior is plausible as it comes to rest.
**Ruler:** An oak ruler, 15 inches in length, is dropped on a concrete floor. Its cross section is nearly rectangular, and is 3.3cm by 0.7cm in its principal dimensions. This example demonstrates our ability to simulate different materials and cross sections. Our simulation generates reasonable sound, though there are audible differences between our result and the recording. We attribute these to the contact stiffness limitations as with the previous example, as well as the relatively high aspect ratio of the object that pushes the limits of our Kirchhoff model.

**Strapping:** A strip of stainless steel strapping is dropped on a concrete floor. It measures 3.5 feet in length, with a rectangular cross section of 1 by \( \frac{1}{32} \) inches. While the sound generated by our method is somewhat plausible, we fail to match a recording of the same object. Due to the high cross section aspect ratio, the body acts more like an elastic sheet than a rod. Although we would need another physics model to accurately reproduce the sound, our radiation model is still effective for this geometry.

**Clock Gong:** A spiral clock gong that is fixed at one end is struck, producing a characteristic sound with a dense frequency spectrum. The gong is steel, has a circular cross section with a radius of 2mm, and is approximately 14cm across its longest dimension. We model the strike as a sum of half-sine pulses. This example shows that our method extends trivially to rods with arbitrary resting shape and with dense modal spectra.

**Rubber Band:** A rubber band is plucked, producing a deep tone. Highly deformable materials such as rubber work well within our framework. At rest, the
band has a length of 18.6cm, and it is stretched between two posts 28.3cm apart. Its cross section measures 1.6mm by 0.8mm.

**Guy Wire (“blaster” sound):** A long wire supporting a tower is struck with a quick impulse, displaying the dispersion effects captured by our framework. This approximates the source of the “blaster” sound well-known in the sound design community. Because of its length, this example would be very costly to simulate with a volumetric mesh representation, but our method is able to simulate it efficiently. The steel rope is approximated as a single steel rod with a circular cross section of radius 0.25in. Its resting length is 20m, but it is stretched to 20.1m between clamped boundary conditions. To accurately simulate bending wave propagation speeds at all audible frequencies, the simulation rate for this example is 88.2kHz.

**Slinky Dispersion:** A Slinky is suspended between a pair of damped springs and is struck with a small metallic bearing. A microphone located close to one end of the Slinky captures the dispersion effects as bending waves travel at different speeds along the Slinky’s length; see Figure 4.8. The steel Slinky consists of 81 loops of a helix 3.25cm in radius. Its cross section is rectangular, measuring 0.95mm by 0.38mm. Again, time integration at 88.2kHz is required to correctly resolve bending wave propagation speeds.

**Chain Link Fence:** A rigid baseball strikes a chain link fence, causing it to rattle. This demonstrates that we can incorporate traditional rigid body dynamics into our framework, and account for large numbers of contacts. The portion of the fence we simulate consists of 35 interlocking aluminum rods, each hav-
Figure 4.8: **Dispersion:** Spectrograms from recorded (Top) and simulated (Bottom) versions of *Slinky Dispersion*. The pitch glide effects due to dispersion are clearly visible in both.

A Slinky walks down a set of stairs. The parameters of this Slinky are the same as in the *Slinky Dispersion* example. The simulation rate for

**Slinky Walk:** A Slinky walks down a set of stairs. The parameters of this Slinky are the same as in the *Slinky Dispersion* example. The simulation rate for
Figure 4.9: **Examples**: From left to right: *Clock Gong, Rubber Band, Guy Wire, and Chain Link Fence.*

this example is 88.2kHz for stability reasons. This example demonstrates the resilience of our method; we can handle tens of thousands of highly coupled frictional contact events per timestep, simulate highly deformable materials, and obtain stable resting contact, all while generating compelling audio.

### 4.4.2 Dipole Model Tests

A number of time-domain sound synthesis frameworks obtain the sound of a vibrating body by sampling the displacement or velocity of a single point (e.g., \[15\] and \[45\]). While this is straightforward to implement for small-displacement or static configurations (like those seen in musical acoustics), it is unclear how to extend this to dynamic 3D environments; any rigid motion of the object needs to be filtered from the audio signal, and it is unclear how to reduce displacement in 3D down to a 1D audio signal. Furthermore, selection of the point can introduce bias in the resulting signal, e.g., a point that lies at a node of a particular vibration mode can lead to the mode’s vibration frequency being underrepresented in the final signal. We compare our radiation model to the velocity signal of a single control point in the static *Rectangular Rod* example above, and present the results in the accompanying video to demonstrate this bias.

The preceding experiment suggests that a full radiation model, with con-
tributions from all elements of the body, is important to the resulting sound quality. However, naively summing acoustic pressure contributions from all elements can result in mode cancellation if the delay due to the speed of sound is not taken into account. The retarded time from (4.2) prevents this. We compare the Rod Drop example with and without the retarded time in the accompanying video; without the delay, the audio signal is prone to clearly audible interference.

Finally, it may seem appealing to eliminate precomputation by approximating the cross section of the rod as a circle. However, this simplification results in perceptually significant changes to the resulting sound if the cross section is far from circular. We demonstrate this effect by comparing our directionally dependent dipole model with a uniform dipole model for the Ruler example. The uniform model artificially exaggerates the high-frequency bending modes that do not radiate well in reality, resulting in increased high-frequency content.

### 4.5 Discussion and Conclusion

We have presented a method for simultaneously generating sound and physically based animation for thin structures. Our model easily fits into existing 3D simulation environments and is able to generate plausible sound even for highly deformable objects.

The radiation model we introduced is valid for any body with a compact cross section, and is independent of dynamics. This means that other dynamics models may be used in place of our rod model. For instance, using Cosserat dynamics may give results that are more physically accurate for rods with thicker
cross sections, or using modal synthesis with our radiation model may provide large gains in efficiency for nearly rigid examples.

For all of our examples above, our method produces plausible sound. However, we are not able to match actual recordings in examples with cross sections with high aspect ratios (Ruler and Strapping). We believe this is due to limitations in our rod model; using other physical models, such as elastic sheets, may provide sound that more accurately matches the recordings.

### 4.5.1 Future Work

We have shown that our method is effective at radiating bending waves; these are generally the vibrations that produce the most sound. However, we do not account for sound radiation from other kinds of modes, such as those arising from longitudinal vibrations. We assume that these are generally insignificant, though some important cases fail to produce correct sound because of their absence. For example, a straight rod dropped directly on one of its ends produces no sound in our framework. Torsional waves are also ignored, though these could plausibly contribute noticeably to the overall sound of rods with highly anisotropic cross sections.

As mentioned in §4.3, one drawback of our method is that there is no way to tune the contact stiffness for collisions between rods and rigid bodies; as a result, many of our examples involving contact exhibit artificially loud high-frequency vibrations excited by very short contact events. We would like to explore ways in which time-domain sound simulations such as ours could take advantage of impulse-based solvers without incurring these artifacts.
Cloth and textiles are ubiquitous in our daily lives, so there is strong demand for high-quality virtual models. Cloth models have been used to predict the geometry of fabrics, to observe their behavior under environmental stresses, and to create realistic renderings and animations. In order to synthesize high-quality images of fabric, previous work has represented cloth with detail resolved down to very small scales, often at the level of individual fibers that compose the yarns. Renderings can then be generated by tracing paths either through fiber geometry or through a volumetric medium. One challenging piece of this process is determining the placement of these fibers given arbitrary cloth geometry.

This has spurred a body of research related to data-driven cloth models. Prior work has explored using computed tomography (CT) scans of cloth samples to provide fiber geometry [110], which is successful but inconvenient: CT scanners are expensive, and it takes significant time and effort to obtain usable volumetric images of such a low-density material. Additionally, one must have physical samples to scan, which makes the method unsuitable for predictive applications such as textile design. Furthermore, it is unclear how to generate volumes for complex knit patterns by assembling CT scans.

As an alternative, synthesis-driven cloth models can overcome these limitations. Schröder et al. [85] demonstrated how to synthesize randomized fiber assemblies which can be tuned with a handful of parameters to resemble a particular type of yarn. This technique allows users to generate an infinite length of virtual yarn, which can then be placed along arbitrary curves that conform
Figure 5.1: Prior work in computer graphics can produce yarn curves that are physically plausible, and generate fiber curve distributions that resemble real yarns. However, when the two are combined naively, the resulting yarn looks overly artificial. We improve on this prior work by splitting the provided yarn curves into macro-fibers, or elastic rod models that represent groups of fibers within a yarn ply. After simulation and inflation, we use these macro-fibers to inform our fiber curve placement. Compared to prior work, this results in yarn geometry that allows fibers to form a denser pattern (blue arrows) while reducing the amount of yarn intersections (red circles).

to the yarn topology of the textile. This promises to reduce the time and effort needed to generate high quality renderings, and can be applied to both woven and knitted fabrics, but it requires tuning fiber parameters and needs good yarn geometry. Zhao et al. [111] improved on this work by using statistics gathered from yarn samples to set parameters. By analyzing a CT scan of a single, straight yarn, they estimated parameter values which could then be used to generate fibers resembling real yarn—but they still used arbitrarily defined procedural yarn curves, limiting realism.

Prior work in computer graphics has shown how to create physically plausible yarn curves via relaxation simulations [107, 64] which, combined with realistic procedural fibers, promises to make artificial fiber models as good as scans. Unfortunately, even when using these methods to generate yarn geometry, the finer-scale fiber paths generated by existing synthesis models do not consider the wider context of the cloth’s structure. Real yarns change shape and deform
when contacting other yarns, but prior work does not account for these effects. These deformations may be small compared to the size of entire textiles, but small changes to yarn geometry, if made globally, can have a visually significant effect on the resulting renderings [111]. Therefore, we argue that the placement of yarn fibers in context must be considered to obtain realistic yarn geometry and appearance.

Simulating each of the tens to hundreds of fibers that constitute a yarn is infeasible; however, simulation at a coarser scale can still provide valuable insight into how yarns take shape. This idea is the keystone for our framework, allowing us to generate fiber assemblies with an awareness of surrounding yarns in a scalable manner. In this work, we make use of efficient elastic rod simulation methods to estimate the shape of yarns and inform the placement of fibers, through two specific contributions:

- **Macro-fiber simulation**: We split yarns into *macro-fibers*, elastic rod models that represent bundles of fibers within the yarn, and demonstrate how to effectively reconstruct yarn behavior through simulation. We show that even though the simulation scale is coarser than the fiber level, we can still reproduce yarn deformation phenomena that are not apparent with yarn-level simulation only, reducing inter-penetration between contacting yarns and improving appearance.

- **Fiber instantiation**: After simulating the macro-fibers, we demonstrate how to adapt previous fiber synthesis methods, using the macro-fiber curves, in a way that accounts for overall yarn deformation.

By generating more plausible fiber assemblies, this method fills in a critical missing piece of the yarn synthesis pipeline, and shrinks the gap in visual quality
between renderings that use CT data and those that use the far more flexible approach of synthesized fiber geometry.

5.1 Yarn Simulation and Geometric Refinement

We begin by giving an overview of our pipeline. To predict how yarn fibers behave in the context of a given cloth pattern, we first consider as input a set of yarn-level curves that describe the topology of the yarn. We use existing simulation frameworks to relax the curves into a physically plausible configuration. We then split each yarn curve into a series of macro-fibers that more accurately captures the yarn’s structure. We relax these macro-fibers while simultaneously adjusting the macro-fiber radius and rest lengths to match the actual bulk of the simulated yarn. Finally, we use the result of this second simulation to map synthesized fiber curves from a straight, undeformed configuration to the yarn curves of the cloth.

We now describe our macro-fiber model, including how to initially place macro-fibers in space, and how to simulate them. This process allows us to predict yarn deformations, which we will later use to guide fiber placement.

Figure 5.2: **Coordinate systems** We use three coordinate systems. World space (left), yarn space (center), and ply space (right).
5.1.1 Generating Macro-fibers

First, we must define the macro-fiber geometry. In this stage of the pipeline, our input is a series of yarn-level curves that describe the overall paths of the yarns, and we output the corresponding macro-fiber curves to be simulated. We define the following three spaces (illustrated in Figure 5.2) which contain different macro-fiber configurations:

- **Ply space**: This is a cylindrical space \((r_{\text{ply}}, \theta_{\text{ply}}, z_{\text{ply}})\) which represents an untwisted ply. In this space, all macro-fibers are straight lines along the \(z_{\text{ply}}\)-axis.

- **Yarn space**: This is a Euclidean space \((x_{\text{yarn}}, y_{\text{yarn}}, z_{\text{yarn}})\) which represents a straight yarn in its relaxed state. In this space multiple plies are twisted around the \(z_{\text{yarn}}\)-axis.

- **World space**: This is the space \((x_{\text{world}}, y_{\text{world}}, z_{\text{world}})\) that contains the input yarn curves and where the simulation will take place.

To generate the macro-fiber geometry in ply space, we choose a fixed number of macro-fibers \(m\) to generate for each ply. This is a tunable parameter which trades off computational cost for the number of degrees of freedom available to represent yarn cross section deformation; for our examples, \(m = 19\). For each macro-fiber \(i\), we define the macro-fiber curve as: \(h^i_{\text{ply}}(z_{\text{ply}}) = (r^i, \theta^i, z_{\text{ply}})\) where \(r^i\) and \(\theta^i\) are constant 2D polar coordinate positions for each macro-fiber. We assume that in this space, the ply is undeformed, so it should be cylindrical in shape. Therefore, we choose an arrangement of macro-fibers that packs into a circle. We have noted that artifacts will occasionally arise in simulation if the distribution of macro-fibers within this circle is rotationally asymmetric, so we
recommend adjusting the number of macro-fibers per ply to obtain an efficient, symmetric packing.

For yarn space macro-fiber geometry (illustrated in Figure 5.3), we use three parameters: the number of plies per yarn \( n \), the yarn twist rate \( \alpha_{\text{yarn}} \), and the ply twist rate \( \alpha_{\text{ply}} \). These parameters are determined by the properties of the yarn to be rendered. We make a copy of ply space for each ply \( j \) within a yarn. Reinterpreting the cartesian space \( (x_{\text{yarn}}, y_{\text{yarn}}) \) in polar coordinates, we offset each ply by \( (r_{\text{yarn}}, \theta^j) \) where \( \theta^j = \frac{2\pi j}{n} \). \( r_{\text{yarn}} \) is chosen based on \( n \) and the radius of the yarn to pack plies together as tightly as possible, and is constant across \( j \). Sweeping along the \( z_{\text{yarn}} \) axis, we rotate each ply around its center at a rate of \( \alpha_{\text{ply}} \), and rotate all plies together around the \( z_{\text{yarn}} \) axis at a rate of \( \alpha_{\text{yarn}} \). Therefore, the path of each macro-fiber takes the form of a sum of two helices:

\[
h_{\text{yarn}}^{i,j}(z_{\text{yarn}}) = \begin{bmatrix}
    r_{\text{yarn}} \cos(\beta_{\text{yarn}} + \theta^j) + r^i \cos(\beta_{\text{ply}} + \theta^j) \\
    r_{\text{yarn}} \sin(\beta_{\text{yarn}} + \theta^j) + r^i \sin(\beta_{\text{ply}} + \theta^j) \\
    z_{\text{yarn}}
\end{bmatrix}
\]  

(5.1)

where \( \beta_{\text{yarn}} = \alpha_{\text{yarn}} z_{\text{yarn}} \) and \( \beta_{\text{ply}} = \alpha_{\text{ply}} z_{\text{yarn}} \).

Our simulator represents rod centerline curves as cubic B-splines, which cannot exactly match the \( h_{\text{yarn}}^{i,j} \) curves specified above. Instead, to generate an approximate B-spline curve, we sample \( h_{\text{yarn}}^{i,j} \) uniformly along its length, and solve a least-squares problem to fit a B-spline to these sampled points \[32\]. To generate uniformly-spaced samples along the curve’s arc length, we must calculate the arc length of \( h_{\text{yarn}}^{i,j} \), which involves solving an integral of the form:

\[
\int_0^z \sqrt{a + b(\cos(\alpha t + d))} \, dt
\]  

(5.2)

We solve this integral numerically by dividing the domain into an integer num-
Figure 5.3: **Macro-fiber Curves** in yarn space are the sum of two helices, each with a different radius ($r^i$ and $r_{yarn}$), initial polar angle ($\theta^i$ and $\theta^j$) and twist rate ($\alpha_{ply}$ and $\alpha_{yarn}$). Here the twist rates are exaggerated for visualization purposes.

We must also calculate the macro-fiber curves in world space that follow the input yarn paths. To do this, we discretize the yarn-level B-splines into polylines, sweep a zero-twist coordinate frame along each edge using parallel transport (see [42]), and use this coordinate system as a map from yarn space to
world space, where $z_{\text{yarn}}$ is aligned with the yarn-level curves’ tangents. This technique is known as tube mapping \cite{42}. The total length of the helix sum along the $z_{\text{yarn}}$ axis corresponds to the resting length of the yarn curve; this ensures that any change in the yarn’s length is reflected in the macro-fibers’ pre-relaxation state in world space.

We assume that the user has chosen a uniform radius per yarn curve. We start by generating all macro-fibers in world space such that the macro-fibers fit inside the volume defined by a circle swept along the yarn curves. However, it is often the case that these volumes intersect in the input; in fact, this is all but certain if yarn-level contact is modeled via penalty forces. This can create topological problems when macro-fibers are instantiated, causing contacting yarns to become entangled when in reality they should merely be in contact. To account for this, we scale down the yarn radius in regions of interpenetration until the intersecting volume vanishes before instantiating macro-fibers. We explain how we make up for this lost yarn volume in the next section.

5.1.2 Simulation

As mentioned in \S1, we treat each macro-fiber as an elastic rod and simulate all macro-fibers together to obtain a lower-energy, relaxed configuration. In doing so, we can capture many of the deformations that are important for large-scale yarn appearance effects. This also allows us to reuse existing yarn-level simulators with few modifications. Several different simulators are available, as listed in \S3; in our framework, we use a simulator implemented according to \cite{64}. We

\footnote{Note that some simulation frameworks such as \cite{56} provide material coordinate frames that track the yarn’s twist along the length of the yarn; in these cases, these frames may be used instead of the parallel transported frame.}
choose this framework because we intend to simulate large systems containing many contacting yarns, and a GPU implementation allows us to conduct these simulations efficiently.

Some yarn simulation models apply twisting forces along the yarns (e.g., [56] and [107]), though others do not (e.g., [55] and [64]). Our simulator does not account for twisting forces on individual rods; however, yarn- and ply-level twisting effects still appear in our simulations, since macro-fibers are wound together within a ply and interact with one another through contact forces.

When converting from yarns to macro-fibers, it is difficult to know the optimal set of rest-length parameters for macro-fibers that will produce a well-packed yarn configuration. Rather than setting these parameters directly, we use a strain balancing algorithm that updates the rest lengths of macro-fiber segments during simulation to match a designated strain value. That is, at each step of the simulation, we calculate the current length of each spline segment, and adjust the rest-length of each segment to obtain a target stretching strain. We require the target strain to be greater than 1 to prevent segments from growing indefinitely; in our examples we use a target strain of 1.03, i.e., 3% elongation of every segment. We also freeze each macro-fiber end point to prevent segments from shrinking indefinitely. Using this approach, the pattern can re-balance to a configuration that normalizes the relative strain per spline segment, which corresponds to a uniform tension on each macro-fiber, producing a well-packed yarn configuration.

In the transition from yarns to macro-fibers, a good deal of the yarn’s volume is lost. Yarn-space macro-fibers take up less volume than the cylindrical volume defined by the yarn-level curves, and the yarns may have shrunk to
avoid macro-fiber collisions as described in §5.1.1. To recover for this lost volume, we gradually increase the radius of the macro-fibers as the simulation progresses, using the radius similarity transformation introduced in [64]. That is, we uniformly increase the radius of all rods in the simulation by a factor of $\gamma$, and modify the force stiffnesses to account for this change, effectively “tightening” the yarn pattern. In practice, we choose a maximum scale factor $\gamma_{\text{max}}$ and gradually increase $\gamma$ from 1 to $\gamma_{\text{max}}$ as the simulation runs. The cross sections of each ply take on a more elliptic shape as the macro-fibers come into contact; see Figure 5.4.

We initially set the contact radius of all macro-fibers to the largest radius that prevents macro-fiber inter-penetration. That is, if the macro-fiber configuration we have chosen permits a packing of macro-fibers with radius $r_{\text{pack}}$, then we set the initial contact radius to $\psi r_{\text{pack}}$ where $\psi$ denotes the smallest scale factor applied to the yarn radius to prevent inter-penetration when instantiating macro-fibers. We then calculate the total scale factor to apply through the radius transformation as a product of two terms:

$$\gamma_{\text{max}} = \frac{1}{\psi} \sqrt{\frac{A_{\text{yarn}}}{A_{\text{macrofibers}}}}$$

where $A_{\text{yarn}}$ is the area of the cross section of the cylindrical yarn, and $A_{\text{macrofibers}}$ is the sum of the cross-sectional areas of all macro-fibers in yarn space. The first term accounts for the yarn scaling as described above. The second accounts for the volume difference between the cylindrical yarn volume and the macro-fiber volume.
Figure 5.4: **Squashed ply** When a yarn ply (red) is trapped between two adjacent plies (green), it takes on a cross section shape that is far from circular as seen in the colorized CT scan of a jersey knit (left). Our macro-fiber simulation is able to reproduce this phenomenon. (right).

5.1.3 Domain Decomposition

The GPU simulator proposed by Leaf et al. [64] is efficient and able to relax large patterns at interactive rates. However, our simulations require significantly more memory than Leaf et al. [64] did for two reasons. The first is the scale of the simulation: there is an order of magnitude more macro-fibers present in a simulation than its yarn-level equivalent, each macro-fiber is sampled in space more finely than its parent yarn, and macro-fibers run parallel to one another, which generates numerous persistent contacts. Furthermore, the simulations of Leaf et al. [64] are periodic, meaning large repeats of knit or woven cloth can be generated from a small patch of simulated data. Periodic boundary conditions require a one-to-one correspondence of yarns across boundaries to ensure a repeatable pattern. When we instantiate macro-fibers, this periodicity is lost unless the rate of yarn and ply twist are both integers with respect to the length of
yarn in a single period. This is rarely true, particularly because this would have to apply to all yarns that cross the domain simultaneously. Therefore, to simulate larger patches of cloth, we must explicitly instantiate all macro-fiber curves, rather than relying on virtual curves that are copies of other curves in the simulation domain. This exacerbates the memory requirements and severely limits the scale of patterns that we can simulate with a fixed GPU memory budget.

To counteract this limitation, we note that it is not necessary to simulate the entire domain simultaneously. We observe that our macro-fibers have fairly small displacements over the course of a simulation, which means the final location of a relaxed spline segment depends only on nearby macro-fiber segments. Therefore, we opt for a domain decomposition approach where pieces of a larger pattern are simulated individually and then stitched together. This allows us to simulate larger patterns even with a limited GPU memory budget.

We now describe the details of our domain decomposition approach. We iteratively subdivide the input domain; at each iteration, we increment the number of domain splits along the current longest axis until each subdomain is small enough to fit into GPU memory. We define a small “freeze” boundary around each cell (see Figure 5.5, left). To simulate a particular cell, we crop the pattern by removing all B-Spline segments that don’t intersect this freeze boundary or the cell. B-spline control points that are within the freeze boundary but outside the cell are given an inverse mass value of 0 to prevent their movement during the cell’s relaxation. We run the simulator on this cropped pattern as normal. When the simulation completes, we update each spline segment’s relaxed position in the original pattern and continue to the next cell.

The algorithm described above creates seams along cell boundaries where
macro-fiber segments are frozen on one side and free to move on the other. To address these, we run the algorithm multiple times, translating the grid at each iteration. In the case that the subdivision occurred along a single axis, we run the algorithm twice with the grid offset of half of a cell length along the other axis; if two axes were subdivided, we run the algorithm 3 times, applying an offset of 1/3 of the cell’s diagonal each time (Figure 5.5 right); finally, in the case that all axes were subdivided at least once, we run the algorithm 4 times, applying an offset of 1/4 of the cell’s diagonal each iteration. This ensures that, within the interior of the pattern, no point is located near a boundary for all runs of the algorithm.

Although the extra simulation time makes the method more computationally expensive, it effectively removes discontinuities. We compare our domain decomposition approach to running an entire model without decomposing the
domain, as shown in Figure 5.6. The macro-fiber displacements differ slightly between these two cases, but once fibers are instantiated along macro-fibers the difference between them is negligible. The simulation time for single tile was 81 minutes, while our 3x3 decomposition took 179 minutes but used nearly an order of magnitude less GPU memory. Domain decomposition allows users to explore this performance trade-off without sacrificing quality of the generated fiber curves.

5.2 Fiber Instantiation

In previous work [85, 111], generating fibers along yarn curves is a two-step process. These methods start by synthesizing fiber curves along a straight yarn, and then deform them to lie along the yarn curves. Yarn configurations are not considered when generating the fibers; instead, each fiber curve within a ply is a helix with procedural irregularities, which is twisted together with other plies to form a yarn. Each ply is deformed into an elliptical shape that is fixed along the length of the yarn to approximate internal yarn deformation.

While this process makes sense for an individual straight yarn, it ignores natural cross section deformation that occurs when yarns interact with one another. In this section, we explain how to use the macro-fiber relaxation to inform the placement of generated fiber curves along yarns.
Figure 5.6: **Domain decomposition comparison** The pattern *Tumbling Moss Blocks* is simulated twice, once as a single domain (left) and once splitting the domain into a 3x3x1 grid (right). The macro-fiber displacements differ slightly between the two simulations (top), but no major artifacts are noticeable over a larger scale. Once fibers are instantiated along the macro-fiber curves (bottom), the difference between the patterns is negligible.

### 5.2.1 Mapping Macro-fiber Displacement

We start by using the framework of Zhao et al. [111] to generate fiber curves in ply space as defined in [5.1.1] that is, we generate untwisted curves that are
defined in a cylindrical space that extends along the $z_{\text{ply}}$-axis. These include migration fibers that fit inside the ply’s “core” as well as flyaway fibers that extend a short distance beyond the core fibers.

We need a way to map points along each fiber curve in ply space to points in world space; this mapping should reflect the results of the macro-fiber simulation, which capture yarn deformation. More concretely, for a particular point along a fiber curve in ply space $p_{\text{ply}}$, we need to know its corresponding position in world space $p_{\text{world}}$, using the macro-fiber displacements as a guide. Ideally this mapping will be smooth to prevent kinks or discontinuities in the world space yarns. Since our macro-fiber guides are continuous (straight lines in ply space and B-splines in world space), every point on a macro-fiber in ply space maps to a single point in world space; we will use these mappings to determine $p_{\text{world}}$.

One feature that keeps tube mapping simple is that points which share their $z_{\text{ply}}$ coordinate map to planes in world space. This is not true in our case—macro-fibers become misaligned through relaxation. However, we have observed that this misalignment is typically very small, and any out-of-plane deformation is typically less in magnitude than the radius of a macro-fiber. This allows us to simplify fiber mapping by approximating it as a 2D problem: given $p_{\text{ply}}$, we find macro-fiber positions in the same cross section (i.e., with the same $z_{\text{ply}}$ coordinate), map these points to world-space, and then fit them to a plane using linear least squares. We assume $p_{\text{world}}$ lies in this plane.

We desire a function that maps cross sections in ply space to cross sections

---

2Due to local stretching, the world space macro-fibers may be different lengths than their ply space counterparts. To account for this, we encode the macro-fiber point in ply space as a spline segment index and an evaluation parameter, and then evaluate this point in world space.
in world space that respects macro-fiber rearrangement. We would like this mapping to be locally injective if possible; otherwise, neighboring fibers will compress into zero volume. However, we’d still like the ply cross section area to change as prescribed by the macro-fibers. For this task, we first discretize the \( x_{\text{ply}} \)-\( y_{\text{ply}} \) plane by creating a 2D mesh with vertices at each of the locations of the macro-fibers. The plane extends out far enough from the ply center to capture all flyaway fibers. We calculate the uniform scale and rotation transformations that best approximate the 2D deformation of macro-fibers in a least squares sense, and then apply them to the mesh. To account for macro-fiber deformation, we balance soft position constraints on the world space macro-fiber positions with the symmetric Dirichlet energy \([88, 84]\). To quickly obtain a minimal energy configuration of the mesh, we use the SLIM framework \([81]\). This results in a mapping that allows nonlinear deformation and changes to the overall scale of the ply, but prevents vanishing volumes. Thus, given a particular fiber point \( p_{\text{ply}} \), we project the point to the 2D cross-sectional plane, find the barycentric coordinates of its enclosing triangle, and use the coordinates to construct \( p_{\text{world}} \) from the corresponding world space triangle.

### 5.3 Results

We wrote our framework in C++ and CUDA. Fiber mapping was parallelized per-ply using OpenMP. Our yarn simulation experiments were performed on an Nvidia GTX Titan X GPU with 12GB of memory, and our yarn mapping was performed with a 3.5GHz Intel Xeon E5-2637 v3 CPU. We show our performance results in Table 5.1. All of our renderings were created with ParaView \([5]\), except for the photorealistic renderings in Figure 5.8, which were rendered with
Figure 5.7: **In-plane deformation** (Left): The cross section plane is tessellated in ply space with the positions of macro-fibers as vertices (blue dots). (Right) The mesh is transformed to a plane in world space based on the corresponding positions of the macro-fibers in world space. The map allows for deformation (notice the elliptical shape) but is locally injective.

PBRT [80].

We now present results generated by our method. We compare the output of our algorithm against two key reference points: CT scans and photographs of two plain-knit fabrics made with known yarns, and results of prior work. We also show that our algorithm results in fewer yarn intersections than prior methods.

### 5.3.1 Comparison to Cloth Samples

We used a knitting machine to create jersey knit (plain knit) cloth samples from two different kinds of 2-ply yarn, called *Puma Stretch* and *Supra Merino*. The interaction between the twist rate of the yarns and knit structure creates vertical
Table 5.1: Simulation statistics for generating each of the patterns in Figure 5.9 using macro-fibers. Here we show the amount of wall clock time needed to simulate the macro-fibers for each pattern, the total number of simulation steps, the total number of control points in the pattern, the peak GPU memory usage, the wall clock time to map fiber curves to world space, and the total number of fiber points mapped.

“barber-pole” bands that are common in jersey knits, and are distinctly more prominent with the more strongly twisted Puma Stretch yarn. Using the tools described in [111], we analyzed CT scans of the yarns to gather parameters for synthesized fiber curves, including the twist rates.

Starting from the same yarn-level simulation of a jersey knit structure, we used the observed twist rates to set up a macro-fiber simulation for each of these fabrics. We then generated fiber curves using the simulation results and the CT-derived parameters for each yarn. We demonstrate the impact of our method by comparing to photographs and CT scans of our knitted samples; see Figure 5.8. In these results, the barber-pole bands can be readily seen, more strongly in the Puma Stretch as in the real fabrics. The tube mapping results have loose plies that do not capture the appearance of these vertical bands well, and they contain many distracting yarn intersections. Our method produces more natural-looking yarns that pack realistically against one another, is able to reproduce the barber-pole effect much better, and predicts some of the differences between these fabrics just from the measured twist rates. The capability of our method to match the effect of sub-yarn-level structure is a promising step...
towards predictive modeling of yarn-level cloth.

### 5.3.2 Comparison to Tube Mapping

Here we present more examples of our method used on more complex knitting patterns. We used a number of patterns in a dataset of yarn curves provided by prior work [64, 107]. For each example, we apply the fiber curves to the yarn path using tube mapping as in [85, 111, 69, 106], and compare that to our method, which uses the macro-fiber fitting technique described above (see Figure 5.9). For each example, we use a 2-ply yarn with the same fiber synthesis parameters as *Puma Stretch*.

### 5.3.3 Yarn Core Intersections

To test the effect of our algorithm on the amount of yarn inter-penetrations, we created a shell mesh around each ply in ply space and mapped it to world space using both tube mapping and our method. We then tetrahedralized the resulting meshes using TetWild [46]; for each tetrahedron, we calculated the winding number of its barycenter with respect to the world space mesh, and summed the volumes of all tetrahedra with winding numbers greater than 1. This represents the amount of volume where “core” fibers (i.e., not flyaways) intersect each other. We performed this experiment on the *Lattice with Seed Stitch* dataset, and found that for tube mapping, approximately 8.0% of the total volume of core yarns was intersecting. On the other hand, for the the macro-fiber mapped meshes, intersecting regions accounted for only 4.8% of the total volume, de-
Figure 5.8: **Vertical bands** (red boxes) appear in jersey knit patterns because of yarn interactions governed by yarn twist rates. Unlike previous work, our method is able to capture these patterns. We compare our method to tube mapping both by rendering photorealistic images to match photographs of knitted patterns (top), as well as rendering CT scans and synthesized fiber curves under the same rendering pipeline (bottom). For full resolution images, please see the [supplementary material](#).
Table 5.9: Comparison to tube mapping Here we show a number of complex knitting patterns where fiber curves are instantiated using both tube mapping (top of each cell) and our technique (bottom of each cell). For full resolution images, please see the supplementary material.

<table>
<thead>
<tr>
<th>Parallelograms</th>
<th>Garter Block</th>
<th>Lattice with Seed Stitch</th>
<th>Montague</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tube mapping</td>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
</tr>
<tr>
<td>Macro-fibers</td>
<td><img src="image4" alt="Image" /></td>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Left Diagonal</th>
<th>Left Diagonal Rib</th>
<th>Tumbling Moss Blocks</th>
<th>Ridged Feather</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tube mapping</td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
<td><img src="image9" alt="Image" /></td>
</tr>
<tr>
<td>Macro-fibers</td>
<td><img src="image10" alt="Image" /></td>
<td><img src="image11" alt="Image" /></td>
<td><img src="image12" alt="Image" /></td>
</tr>
</tbody>
</table>
spite the fact that the total volume was $1.04 \times$ that of the tube mapped meshes. This is unsurprising given that fibers have the ability to deform to avoid collisions in our framework, while tube mapping only permits deformation up to an elliptical ply shape.

5.4 Discussion and Conclusion

We have presented a method for predicting yarn deformation within knitted cloth. This method builds on previous work in yarn relaxation, and makes progress towards bridging the divide between data-driven and synthesis-driven fiber-level cloth rendering. Our method fills a gap in the physically based yarn rendering pipeline where previously there was little methodology reasonably based on physics or observation. Macro-fibers demonstrate many of the key cross-section deformations of a yarn, which influence the yarns’ rendered appearance, without requiring a fiber-level simulation. Furthermore, our method is capable of predicting the effect of yarn twist on the fine-scale fiber geometry and the macroscopic appearance of the fabric. This is a promising step on the road to fully predictive yarn-level cloth modeling.

One clear future direction is to make this method more practical, both in terms of memory and computational efficiency. Luan et al. [69] and Wu and Yuksel [106] both demonstrate how to significantly reduce costs when rendering fiber-level cloth, though both rely on simplified mappings between ply space and world space. A method that could take advantage of these frameworks and simultaneously handle yarn cross section deformations would be desirable.

To obtain a deeper understanding of how yarns behave within knit textiles,
we need better tools for analyzing volumetric images of cloth. Our comparisons to volumetric images of cloth are qualitative, but in order to quantitatively verify that our simulations are producing results that mimic the behavior of real yarns, we would need tools to accurately identify and disambiguate individual yarns through arbitrary paths in the volume.

With this work and future work, we hope to increase the accuracy of virtual cloth models. This will allow for more realistic cloth rendering and simulation, and will broaden the range of applications that fall under predictive cloth modeling, from appearance modeling to fabrication.
CHAPTER 6
TOWARDS TRACKING YARN PATHS THROUGH VOLUMETRIC SCANS

Volumetric scans of fabric can provide rich insight into the construction, appearance, and behavior of cloth. 3D imaging captures the yarn geometry that makes up a fabric, bypassing obscurities and ambiguities due to occlusion. Previous work (e.g., [58, 109, 110]) has successfully used CT scans of textiles for highly detailed appearance models, but this is only one application of this data. For instance, it could also be used to validate and calibrate yarn-level cloth simulation models, adding further realism to physics-based animation of fabrics. Unfortunately, the raw data provided by these scans is in the form of regular voxel grids with density values, which is not directly comparable to the (framed) polyline or curve yarn representations used within computer graphics. Extracting centerlines from this voxel grid is often not straightforward, since inter-yarn contact makes yarn paths ambiguous. Iterative tracking methods may drift from one yarn to another in these regions, especially if the yarns are highly twisted or travel in similar directions. Locally, it is not always obvious which voxels belong to which yarn; it is only through global context that such a distinction becomes clear.

Historically, the some of the most successful approaches to extracting yarn paths from density volumes have used a prediction-correction approach, where yarns are traced iteratively along their length. At each step, the next point on the yarn’s centerline is estimated, and then corrected using the surrounding data [87, 109]. Prior work has shown that tracking individual fibers can help distinguish contacting yarns in a volumetric scan, even when they are traveling in similar directions [87]. However, tracking individual fibers may be difficult
in cases where the scan resolution is not high enough to resolve them.

In this work, we describe our progress in search of an algorithm to robustly obtain centerline curves for scanned yarns. Our main contribution is an algorithm for tracking yarn paths through scans of arbitrary cloth patterns, even if the 3D image’s resolution is not high enough to fully resolve individual fibers. We do this by tracking ridges through the volume, i.e., single-voxel curves that follow the direction of fibers. This provides hints towards yarn continuity, which we use to aid a prediction-correction algorithm. Being able to analyze lower resolution scans reduces memory consumption and permits yarn tracking across larger patches of scanned cloth. We also propose a number of ways the algorithm could be augmented, and how the results can be used to extract insights about the scanned fabric pattern.

6.1 Using Ridges to Track Yarn Paths

Our goal is to find curves that follow fibers through a density volume. For this task, it is useful to first generate a direction volume; that is, for every voxel, we calculate a unit vector that estimates the direction of any nearby yarns or fibers.

For every voxel in the density volume, we aim to find a measure of correlation between nearby density values with respect to a particular axis. We can compute correlations over several axes; the axis with the highest correlation value will be the estimated fiber direction. We use a sum of covariance values as our correlation measure.

We now provide the details of our correlation calculation. Suppose we want
to find the fiber orientation at a voxel \( v_0 = (x_0, y_0, z_0) \in \mathbb{Z}^3 \). For now, let us consider density correlations along the \( x \) axis; let \( v_i = (x_i, y_0, z_0) \). Furthermore, let \( D(v_i) \) be the raw density value stored at a voxel \( v_i \). The covariance of two voxels \( v_i \) and \( v_j \) is defined as \( E \left[ (D(v_i) - E[D(v_i)]) (D(v_j) - E[D(v_j)]) \right] \). We estimate expected values by applying 2D Gaussian blurs to voxel planes that are orthogonal to the correlation axis. For instance, we estimate \( E[D(v_i)] \) by blurring within the plane defined by \( x = i \) and sampling the result at \( v_i \). Given a user-specified integer \( k \), we calculate \( \sum_{-k \leq i \leq k, i \neq 0} (D(v_0) - E[D(v_0)]) (D(v_i) - E[D(v_i)]) \). We then calculate this value for every voxel in the \( x = 0 \) plane, and apply another 2D Gaussian blur to obtain the covariance. Similar calculations can be performed to find correlations along the \( y \) and \( z \) axes respectively. For correlations along directions that do not align with coordinate axes, we perform sub-voxel shifts along consecutive slices of the volume before calculating the correlation. Note that this requires upsampling the volume; we do this using cubic interpolation along each 2D slice. We perform this calculation for each pair of antipodal directions in an cubemap with a resolution of \( 8 \times 8 \) on each face, and use the direction with the highest correlation value as our estimate.

The resulting direction volume has some nice properties. Reasonable direction vectors are found even for voxels with low density, as long as they are in proximity of a yarn. Additionally, by keeping only the direction with the maximum correlation, we do not blend direction vectors, and thus edges between yarns in contact are sharp. Furthermore, the calculations are somewhat insensitive with respect to \( k \); as long as yarns are well resolved and mostly linear in the window chosen, the algorithm returns a reasonable estimate of directions.

Given this direction volume, we can trace ridges through the density volume;
Figure 6.1: Left: to estimate directions, correlations are calculated three sweeps of the volume, once for each dimension. Right: to calculate off-axis directions, sub-voxel shifts are performed around a center slice. In practice, the volume is resampled to obtain these shifts.

Figure 6.2: Top: A slice of a CT scan of a silk weave sample. Bottom: A visualization of the direction estimates using our algorithm; red indicates the direction normal to the slicing plane, green indicates the vertical direction in-plane, and blue indicates the horizontal direction. Note the sharp discontinuities where the warp and weft yarns are in contact.
that is, we can find chains of high-density voxels that follow these direction estimates. Given a seed voxel, we check if its density is beyond a given threshold; if so, we add it to a list and select one of the 28 neighboring voxels as the next candidate to add to the list. Each neighbor is assigned a score that is the product of two factors: (1) the dot product between the unit vector between voxel centers and the direction estimate, and (2) the density of the neighboring voxel. The vectors in the direction volume may be more precise than the 28 directions towards the nearest neighbors; to avoid quantization errors, we define the direction estimate to be the normalized average of the direction vectors of the previous 3 voxels visited. The process then repeats; if the candidate voxel is below the density threshold, the ridge ends. While this procedure draws inspiration from tracing individual fibers, we emphasize that the two are not necessarily equivalent. In particular, ridges can successfully be traced through a yarn even if the density volume does not resolve individual fibers.

We use this as a subroutine in a prediction-correction method to track a single yarn. We begin each iteration with an estimate of the yarn’s centerline and direction (for the first iteration, we rely on the user to provide this data). We maintain a list of “active voxels” that represent points on ridges that follow the yarn being tracked. We first search the plane orthogonal to the yarn’s direction for voxels that have a high density and similar direction estimate, and mark them as active. We then translate this cross-sectional plane along the yarn direction a small amount. We track ridges from the list of active voxels forward until they reach this translated plane (ridges that do not reach the plane after a set number of steps are discarded). The set of locations where each of the ridges intersects the plane becomes the new list of active voxels. We then perform a correction step to refine our estimate of the yarn centerline. First, we update
the yarn’s centerline position by setting it to the average of all active voxels. We then cull active voxels that fall outside the radius of the yarn or ply being tracked. Finally, we update the yarn’s direction by employing a voting scheme over the direction estimates at each active voxel. Since our direction estimates are quantized, we simply choose the most common direction among the active voxels, optionally discarding votes from voxels whose direction estimates are far from the current yarn direction. Given this updated position and direction, we begin the next iteration. We continue iterating until the population of active voxels drops below a given threshold, or if we enter a margin that is a fixed distance away from the border of the scanned volume.

In general, this algorithm appears to be robust to yarns that are in close contact. Occasionally, however, centerlines may drift from one yarn to another, which causes all future tracking to be incorrect. In this case, a user can specify points at which the yarn diverged from the correct path, each with a correct direction vector; rerunning the algorithm while taking this extra information into account can resolve this issue.

6.2 Results

Figure 6.3 shows the results of our algorithm (without any user correction) on CT volumes of two different knitting patterns. Initial yarn positions and directions were manually specified on the left side of the images, and our algorithm tracked the yarn centerlines (indicated by solid lines) towards the right side of the images. In general, our algorithm is successful at tracking yarns through twists and turns. However, centerlines occasionally start to follow incorrect
paths, as shown by the red centerline in the bottom figure. For most of the length of the scan, the red centerline is tracked correctly, but our algorithm incorrectly drifted from one yarn to another at a particular yarn crossing. It then followed the neighboring yarn (also tracked by the white centerline) to the end of the scan. Even so, this type of error could be manually corrected as described above.

Figure 6.4 demonstrates that our method is able to differentiate plies in a 2-ply yarn. This is a particularly challenging case, because fibers from both plies are traveling in very similar directions at all times. In order to prevent centerlines from drifting from one ply to the other, we rely on our ridge tracking algorithm to guide us along the paths of continuous fibers.

6.3 Discussion and Future Work

We now discuss some potential extensions to the presented algorithm to improve the quality of the results, and explore some of the applications that are afforded by this technique.

6.3.1 Robustness Through Wider Context

Manually correcting mistakes made by our tracking algorithm is tedious. The proposed algorithm tracks each yarn or ply individually, but tracking all yarns simultaneously may help to detect inter-yarn drift and automatically correct such errors. For example, in the rib stitch pattern in Figure 6.3, it is visually obvious that the red and white centerlines are tracking the same yarn on the
Figure 6.3: Yarn centerlines (solid lines) are traced through CT scans (translucent red). Top: A jersey knit pattern. Bottom: a rib stitch pattern.
Figure 6.4: Each ply of a 2-ply yarn (blue and yellow lines) is tracked individually through a scan of a woven fabric (translucent red).

right side of the image. This suggests that we may be able to use the proximity of separate centerlines as an indicator that a mistake has been made. Furthermore, it is clear that the four starting locations on the left should have corresponding exit locations on the right side of the image; given that there are only three exit locations, there must be an error. Framing yarn tracking as a max-flow/min-cut problem where only a single centerline is allowed to “flow” from an entrance to an exit could give insight on how to correct tracking mistakes.

Prior knowledge about the structure of yarn may also aid in tracking. For instance, if the scanned yarns are known to have multiple plies, then we should observe the centerlines of the plies staying a small and nearly-fixed distance away from one another, rotating around one another at near-constant speed. Tracking the plies of a yarn simultaneously would help prevent ply centerlines from drifting onto one another. Additionally, if the ply centerlines diverge significantly from their expected paths, then this could indicate a tracking error; backtracking to a known valid state and exploring alternate paths could improve the robustness of the algorithm.

Furthermore, our direction field only records the most likely direction of the fibers; however, it is likely that keeping multiple estimates for each voxel would improve yarn tracking. As results in the tractography literature suggest, maintaining multiple direction estimates could reduce ambiguity in regions of crossing fibers. Choosing the direction that minimizes curvature at each voxel
when constructing ridges could reduce drift between fibers and improve the robustness of our algorithm.

6.3.2 Centerline Smoothing

Due to the noisy nature of CT scans as well as the stochastic behavior of yarn fibers, the reconstructed centerlines are often noisy. Depending on the intended application, this may be undesirable. For example, an analysis of the yarn’s curvature would be misleading, as high-frequency spatial noise would bias the calculations, likely leading to overestimations. To remedy this, smoothing operations could be applied to the tracked centerlines as a post-process. Many curve smoothing algorithms are available; the ideal method would filter out high-frequency noise without moving centerlines far from the actual yarn centers present in the data.

Prior work in fiber tracking [58] performed a smoothing post-processing step. New points along a centerline \( p_1, p_2, \ldots, p_n \) were calculated by minimizing the energy:

\[
E = \sum_i \alpha \| p_i - p_i^{(0)} \|^2 + \| p_{i-1} - 2p_i + p_{i+1} \|^2 
\]

where \( p_i^{(0)} \) is the original position of the \( i \)th vertex before smoothing, and \( \alpha \) balances the two terms and is chosen heuristically. This is analogous to adding a bending force to the centerline and balancing it against springs that pull towards the original data points, and can be solved as a statics problem by using, e.g., nonlinear conjugate gradient. Unfortunately, determining an appropriate value for \( \alpha \) is difficult, especially for highly curved centerlines; values that are too high will fail to remove significant noise, while values that are too low risk
substantially changing the shape of the centerline. Other smoothing algorithms, such as spline fitting, face the same pitfalls.

An alternative approach could use the underlying CT data to guide the smoothing operation. In particular, the term \( \|p_i - p_i^{(0)}\|^2 \) could be replaced with a function that depends on the CT data rather than the unsmoothed centerline data. An example term is \( \sum_{v \in V(p_i)} D(v) \left( \frac{\|p_i - v\|}{r} - 1 \right) \) where \( r \) is the approximate radius of the yarn or ply to be tracked, \( V(p_i) \) is the set of voxels within a distance \( r \) of \( p_i \), and \( D(v) \) is the measured density of voxel \( v \). Minimizing this differentiable quantity pulls \( p_i \) towards areas where the surrounding sphere of radius \( r \) has high density. For yarns or plies with highly noncircular cross sections, choosing a non-spherical domain for \( V \) may be more appropriate.

6.3.3 Applications

Armed with estimated yarn centerlines, we can evaluate yarn simulation models. For many models, the simplest verification method would be to observe the forces that would be applied to these centerlines if they were to be used as the initial state in a simulation. Given reasonable material parameters, imbalanced forces imply that the model is far from its resting state, whereas vanishing forces imply that the simulation is correctly at rest. Sums of residual forces could be used to compare different methods. Another verification method would be starting the simulations with synthesized yarn configurations that have the correct yarn topology, but are perturbed from their resting state, and comparing the converged, relaxed centerlines of each simulation with the reconstructed reference centerlines obtained using our method. Assuming boundary conditions
are correctly accounted for, the relative distance between test and reference centerlines would provide a measure of model accuracy.

Furthermore, we could extract yarn material parameters from tracked centerlines. Let $\alpha$ be a vector of $k$ linear stiffness parameters that we would like to estimate from a given yarn pattern. Suppose we have extracted a yarn’s centerline and discretized it such that it has $n$ degrees of freedom. Let $F(\alpha) : \mathbb{R}^k \rightarrow \mathbb{R}^n$ be the function that maps the input stiffness vector to the resulting forces applied to the observed degrees of freedom according to a particular elastic rod model. Since we have assumed that the stiffnesses are linear, $F(\alpha) = A\alpha + b$ where $A$ is a $n \times k$ matrix and $b$ is a length-$n$ vector that encodes external forces. The input yarn paths are unmoving, so we can assume that the sum of forces present at each degree of freedom sum to zero. Therefore, we should expect $A\alpha = -b$; that is, internal forces should balance out external forces. This system is overconstrained and unlikely to be exactly satisfied given noisy centerline measurements, so we can solve for $\alpha$ using a standard least-squares solver. In order to obtain a reasonable solution, $b$ must be nonzero, which implies a measurable, nontrivial force must be exerted to the pattern as it is scanned. If the reference data is noisy, it might be beneficial to jointly optimize the positions of the control points along with the parameters in an attempt to filter out high-frequency inaccuracies that would artificially lower the stiffness parameters. This could potentially be done with a total least squares approach that optimizes the centerline configuration and the stiffness parameters simultaneously. The effectively adds springs to each control point that allows drifting from the reference data, but penalizes large drifts.
6.4 Conclusion

We have introduced a method for tracking yarn paths through CT scans of cloth. The method is applicable to arbitrary yarn patterns, and performs well in the presence of yarn crossings. Our algorithm is effective even when the fibers are not well resolved, permitting larger input datasets than prior work. With this tool, methods for analyzing 3D images of cloth are more tractable.
CHAPTER 7
CONCLUSION

This thesis has proposed two frameworks that use elastic rod simulation techniques to solve problems beyond physically based animation. This work demonstrates that simulation techniques developed for the purposes of animation are general and fast enough to be useful for other applications, even if some accuracy is sacrificed. In particular, elastic rod simulators that function well in general virtual environments can now be augmented with a lightweight acoustic model that allows simultaneous generation of physically based animation and sound, broadening the range of geometries for which sound synthesis techniques are applicable. Furthermore, yarn geometry models for the purpose of high-quality yarn rendering can now account for deformation of fibers through elastic rod simulation, bringing synthesized yarn a step closer to the quality of yarn geometry obtained through expensive imaging. This was only possible at scale because of the efficiency inherent in the elastic rod models used. As a step towards quantifying the issue of accuracy, this thesis also described preliminary work on a technique for tracking yarn centerlines through CT volumes, which is a crucial prerequisite for calibrating and validating yarn simulation models.

The proposed algorithms have potential applications in a range of scenarios. Foley artists now have another tool at their disposal when creating sound to accompany animations. For textile designers, predictive photorealistic models of textiles are one step closer to becoming a reality. More realistic yarn geometry may also be useful in small-scale simulations of cloth behavior to observe, e.g., fluid-cloth interactions and insulation properties. Finally, the provided insight into extracting fiber bundle paths from volumetric scans could lead to more
accurate yarn simulation models, or methods for reverse engineering fabrics.

Besides these applications, there are a number of other directions for future work. For instance, the main drawback of the proposed sound synthesis model is its speed; further investigation into how to make the method as fast as models such as modal synthesis would make it much more appealing to use in production environments. Coupling this model with existing sound simulation techniques would further expand the range of geometries for which sound synthesis can be used. Finding ways to reduce the memory consumption and computational requirements of macro-fibers would encourage its use in large-scale fabrics. More robust methods for tracking yarn would pave the way towards a better understanding of the physics models used for virtual fabrics. Identifying and classifying the individual fibers within a CT volume would also provide more insight into how yarns deform when they interact with the yarns around them. Finally, schemes for classifying yarn topology would contribute to compact, generalizable descriptions of cloth constructions. Tools to recognize them in collections of yarn curves may speed up simulations. Current yarn-level simulations and relaxations are limited to small timesteps to prevent yarns from passing through one another; by comparing local yarn topology before and after a step, future techniques could detect and respond to yarn pull-through events, allowing larger simulation steps. Additionally, recognizing yarn topologies in CT volumes would aid in reverse engineer cloth patterns.

It is clear that elastic rod models developed in the field of computer graphics have a range of uses. It is hoped that this thesis inspires further exploration of how these techniques can address problems beyond visual animation.
APPENDIX A

PROOF OF VANISHING $D^\omega$

Proof. By definition, $D^\omega = \oint_S (y - \varphi^*(y)) \cdot (y \times n(y))^T dS_y$. On a fully symmetric domain, $n(y)$ is an odd function, i.e., for all $y$ in $S$, $-y$ is also in $S$ and $n(-y) = -n(y)$. $\varphi^*(y)$ is a 3D solution to an external Laplace problem with Neumann boundary conditions (see (4.3), (4.4), and (4.5)). The boundary conditions themselves are determined through the odd function $n(y)$, and the Laplace equations have no symmetry bias; therefore $\varphi^*$ itself is odd in $y$.

The integrand is a product of three functions that are odd in $y$, meaning the integrand itself is also odd in $y$. Thus, $D^\omega$ is determined by an integral of an odd function over a symmetric domain and therefore vanishes. \hfill \Box
BIBLIOGRAPHY


[22] Jeffrey N Chadwick, Changxi Zheng, and Doug L James. Precomputed


Dingzeyu Li, Yun Fei, and Changxi Zheng. Interactive acoustic transfer


[97] Stephen P Timoshenko. Lxvi. on the correction for shear of the differential equation for transverse vibrations of prismatic bars. *The London, Edin-


